Low-rank Tensor Decompositions in Kernel-based Machine Learning

Dissertation

zur Erlangung des akademischen Grades

doctor rerum naturalium
(Dr. rer. nat.)

von **M.Sc. Kirandeep Kour**

geb. am **03.11.1994** in Kaman, Rajasthan, India

genehmigt durch die Fakultät für Mathematik

der Otto-von-Guericke-Universität Magdeburg

Gutachter: **Prof. Dr. Peter Benner**

Prof. Dr. Martin Stoll

Dr. Sergey Dolgov

eingereicht am: **16.08.2023**

Verteidigung am: **06.12.2023**

PUBLICATIONS

The material presented in this thesis has either been published, accepted, or submitted for publication.

Chapter 4 concludes with a discussion of the following published work,

[98]: K. Kour, S. Dolgov, M. Stoll, and P. Benner (2023), Efficient Structure-preserving Support Tensor Train Machine, *Journal of Machine Learning Research*, 24 (4), pp. 1–22, http://jmlr.org/papers/v24/20-1310.html.

The content of Chapter 5 is published in the following preprint,

[97]: K. Kour, S. Dolgov, P. Benner, M. Stoll, and M. Pfeffer (2023). A weighted subspace exponential kernel for support tensor machines, *ArXiv, abs/2302.08134*, https://doi.org/10.48550/arXiv.2302.08134.

In Chapter 7, a short description of the following published work is mentioned. As this work takes a different route and breaks the flow of the thesis,

[84]: D. S. Karachalios, I. V. Gosea, K. Kour, A. C. Antoulas (2024), Bilinear realization from input-output data with neural networks, *Scientific Computing in Electrical Engineering*, pp. 184–192, Springer Nature Switzerland, https://doi.org/10.1007/978-3-031-54517-7_21.

ABSTRACT

An increasing amount of collected data are high-dimensional multi-way arrays (tensors), and it is crucial for efficient learning algorithms to exploit this tensorial structure as much as possible. The ever present curse of dimensionality for high dimensional data and the loss of structure when vectorizing the data motivates the use of tailored low-rank tensor classification methods. In the presence of small amounts of training data, kernel methods offer an attractive choice as they provide the possibility for a nonlinear decision boundary. This thesis focuses on developing low-rank decomposition based kernel methods. One of the first proposed approach is the Tensor Train Multi-way Multi-level Kernel (TT-MMK), which combines the simplicity of the Canonical Polyadic (CP) decomposition, the classification power of the Dual Structure-preserving Support Vector Machine (SVM), and the reliability of the Tensor Train (TT) approximation. This proposed algorithm is based-on computing a CP decomposition by avoiding the NP-hard issue of finding the best CP rank by computing first a TT decomposition and call it TT-CP factorization. Along with the experiments it is shown that the TT-MMK method is usually more reliable computationally, less sensitive to tuning parameters, and gives higher prediction accuracy in the SVM classification when benchmarked against other state-of-the-art techniques. Additionally, the classification model that includes low-rank tensor decomposition as a crucial initial step reduces the computational complexity and extracts informative features. However, what decisive features of the tensors are exploited by these kernels is often unclear. Therefore, this work also proposes a novel kernel that is based on the Tucker decomposition. For this kernel the Tucker factors are computed based on re-weighting of the Tucker matrices with tuneable powers of singular values from the Higher Order Singular Value decomposition (HOSVD). This provides a mechanism to balance the contribution of the Tucker core and factors of the data. The support tensor machines with this new kernel benchmark on several datasets. First experiment is using generated synthetic data where two classes differ in either Tucker factors or core, and compare the novel and previously existing kernels. This shows robustness of the new kernel with respect to both classification scenarios. Further, testing the new method on real-world datasets is also included. The proposed kernel has demonstrated a similar test accuracy than the state-of-the-art TT-MMK, and a significantly lower computational time. Finally, this thesis defines a regularized form of Support Tensor Machine (STM) as a classification machine learning model. The algorithm builds a full Gradient Descent Primal (GDP) optimization problem that takes initialized variables from the partial GDP model, which is a standard STM, optimized with TT-CP low-rank approximation. The full GDP is a tensor decomposition method tailored to the classification difficulty and a classification method that exploits the low-rank model of the data. The proposed optimization regime and the relationship between primal-dual for TT-CP decomposition has been theoretically proven. This proposed work shows some numerical challenges and detailed discussion on the further plausible advancements.

ZUSAMMENFASSUNG

Ein zunehmender Teil der gesammelten Daten sind hochdimensionale Mehrweg-Arrays (Tensoren), und es ist für effiziente Lernalgorithmen entscheidend, diese Tensorstruktur so weit wie möglich auszunutzen. Der allgegenwärtige Fluch der Dimensionalität für hochdimensionale Daten und der Strukturverlust bei der Vektorisierung der Daten motiviert den Einsatz maßgeschneiderter Tensor-Klassifizierungsverfahren mit niedrigem Rang. Bei kleinen Mengen von Trainingsdaten sind Kernel-Methoden eine attraktive Wahl, da sie die Möglichkeit für eine nichtlineare Entscheidungsgrenze bieten. Diese Arbeit konzentriert sich auf die Entwicklung von Kernel-Methoden, die auf Low-Rank-Zerlegung basieren. Einer der ersten vorgeschlagenen Ansätze ist der Tensor Train Multi-way Multi-level Kernel (TT-MMK), der die Einfachheit der kanonischen polyadischen Dekomposition, die Klassifizierungsleistung der Dual Structure-preserving Support Vector Machine und die Zuverlässigkeit der Tensor Train (TT) Approximation kombiniert. Der vorgeschlagene Algorithmus basiert auf der Berechnung einer kanonischen polyadischen (CP) Zerlegung, indem er das NP-schwere Problem, den besten CP-Rang zu finden, vermeidet, indem er zuerst eine Tensor-Train (TT)-Zerlegung berechnet und ihn TT-CP-Faktorisierung nennt. Zusammen mit den Experimenten wird gezeigt, dass die TT-MMK-Methode in der Regel rechnerisch zuverlässiger ist, weniger empfindlich auf Tuning-Parameter reagiert und eine höhere Vorhersagegenauigkeit bei der SVM-Klassifizierung bietet, wenn sie mit anderen modernen Techniken verglichen wird. Darüber hinaus reduziert das Klassifizierungsmodell, das die Tensorzerlegung mit niedrigem Rang als entscheidenden ersten Schritt beinhaltet, die Rechenkomplexität und extrahiert informative Merkmale. Es ist jedoch oft unklar, welche entscheidenden Merkmale der Tensoren von diesen Kerneln genutzt werden. Daher wird in dieser Arbeit auch ein neuartiger Kernel vorgeschlagen, der auf der Tucker-Zerlegung basiert. Für diesen Kernel werden die Tucker-Faktoren auf der Grundlage einer Neugewichtung der Tucker-Matrizen mit einstellbaren Potenzen der Singulärwerte aus der HOSVD-Zerlegung berechnet. Dies bietet einen Mechanismus, um den Beitrag des Tucker-Kerns und der Faktoren der Daten auszugleichen. Die Support-Tensor-Maschinen mit diesem neuen Kernel werden an verschiedenen Datensätzen getestet. Das erste Experiment verwendet synthetische Daten, bei denen sich zwei Klassen entweder in den

Tucker-Faktoren oder im Kern unterscheiden, und vergleicht den neuen mit den bereits vorhandenen Kerneln. Dies zeigt die Robustheit des neuen Kernels in Bezug auf beide Klassifizierungsszenarien. Weitere Tests der neuen Methode auf realen Datensätzen sind ebenfalls enthalten. Der vorgeschlagene Kernel hat eine ähnliche Testgenauigkeit wie der Stand der Technik und eine deutlich geringere Rechenzeit gezeigt. Schließlich wird in dieser Arbeit eine regularisierte Form der Support Tensor Machine (STM) als maschinelles Klassifikationslernmodell definiert. Der Algorithmus erstellt ein vollständiges Gradient Descent Primal (GDP)-Optimierungsproblem, das initialisierte Variablen aus dem partiellen GDP-Modell übernimmt, das eine Standard-STM ist und mit TT-CP-Niedrigrang-Approximation optimiert wurde. Das vollständige BIP ist eine auf das Klassifizierungsproblem zugeschnittene Tensor-Zerlegungsmethode und eine Klassifizierungsmethode, die das Low-Rank-Modell der Daten ausnutzt. Das vorgeschlagene Optimierungsregime und die Beziehung zwischen primär-dual für die TTCP-Zerlegung wurden theoretisch nachgewiesen. Die vorgeschlagene Arbeit zeigt einige numerische Herausforderungen und eine detaillierte Diskussion weiterer plausibler Weiterentwicklung.

LIBRARIES

Following mentioned libraries and toolboxes are used for the work proposed in this thesis.

Remark:
Mostly, the implementations are done on the Max Planck Institute's server (Mechthild) in a virtual environment.

- Tensor Toolbox: Computation of low-rank aproximations and working with N-way array or tensors includes use of Tensor Toolbox for MATLAB by Brett W. Bader, Tamara G. Kolda and others.
- TT Toolbox: This toolbox is available in both languages MATLAB and Python,, which is useful to directly apply TT decomposition for particular given problem. This was developed by Sergey Dolgov and Ivan Oseledets.
- LIBSVM: This MATLAB based library works for Support Vector Machine and related models. This was developed by Chih-Chung Chang and Chih-Jen Lin.
- Tensorflow: This is a software library for high performance numerical computation and based on machine learning approach with Python script. **Tensorboard** is used for visualizing neural networks. Tensorflow uses models from Scikit-learn, Keras, generation of plots with Matplotlib, data frame work Pandas, and scientific computation Numpy.
- t3f : This is a software library implementation of tensor train decomposition and related functions on top of tensorflow. Parallel processing capabilities are built in and taken care in the backend by tensorflow.
- NARX : This is a MATLAB toolbox for nonlinear model identification basics. One more evolved version of this is NARMAX [7, 33].

CONTENTS

List of Figures xv

List of Tables xvii

List of Algorithms xix

List of Symbols xxi

List of Terminology xxv

Abbreviations xxvii

1 Introduction 1
- 1.1 The Synopsis and Significance of A Journey Towards Conflation 1
- 1.2 Outline of This Thesis 6

2 Mathematical Foundations of Supervised Machine Learning 7
- 2.1 Mathematical Introduction to Supervised Learning Models 7
- 2.2 Linear Classification 10
  - 2.2.1 Support Vector Machine 10
  - 2.2.2 Primal-dual Relationship 11
- 2.3 Nonlinear Classification 13
  - 2.3.1 Reproducing Kernel Hilbert Spaces 15
  - 2.3.2 Feature Map and Kernel Trick 17
  - 2.3.3 Kernelised Support Vector Machine 19
- 2.4 Optimizing a Supervised Learning Model 21
  - 2.4.1 Cross Validation 22
- 2.5 Chapter Summary 24

3 Low-rank Tensor Methods in Supervised Learning 27
- 3.1 Introduction 27
- 3.2 Tensor Algebra 28

3.3 Low-rank Tensor Factorization . . . 32
3.3.1 Canonical Polyadic Decomposition . . . 33
3.3.2 Tucker Decomposition . . . 36
3.3.3 Tensor Train Decomposition . . . 38
3.3.4 Existing Interrelationship of Tensor Decompositions . . . 40
3.3.4.1 CP to TT . . . 40
3.3.4.2 Qunatized Tensor Train (QTT) . . . 41
3.3.4.3 QTT-Tucker . . . 42
3.4 Tensor Classification . . . 42
3.4.1 Support Tensor Machine . . . 42
3.4.2 Tensor-product Reproducing Kernel Hilbert Space . . . 44
3.4.3 Kernelised Support Tensor Machine . . . 45
3.4.4 Reproducing Tensor Kernels . . . 47
3.5 Low-rank Tensor Decomposition in Kernel Models . . . 48
3.6 Chapter summary . . . 49

4 Efficient Structure-preserving Support Tensor Train Machine 51
4.1 Introduction . . . 51
4.1.1 Problem Formulation . . . 52
4.1.2 Related Work . . . 53
4.2 The Proposed Algorithm . . . 53
4.2.1 Uniqueness Enforcing TT- SVD . . . 54
4.2.2 TT-CP Expansion . . . 57
4.2.3 Norm Equilibration . . . 57
4.2.4 Noise-aware Threshold and Rank Selection . . . 58
4.2.5 Reproducing Tensor Kernels (§3.4.4) . . . 58
4.2.6 Nonlinear Mapping . . . 59
4.3 Numerical Tests . . . 61
4.3.1 Data Collection . . . 62
4.3.2 Influence of Individual Algorithmic Steps . . . 63
4.3.3 Comparison to Other Methods . . . 65
4.4 Chapter Summary . . . 68

5 A Weighted Subspace Exponential Kernel for Support Tensor Machines 71
5.1 Introduction . . . 71
5.1.1 Related Work . . . 72
5.1.2 Problem Formulation . . . 73
5.1.3 Converting TT into CP . . . 74
5.1.4 Converting Tucker to CP . . . 74
5.2 Reproducing Tensor Kernels . . . 75
5.2.1 The Subspace Kernel . . . 76

5.2.2 The Weighted Subspace Exponential Kernel 77
5.2.3 Computational Complexity of the Different Kernels 78
5.3 A Numerical Study on Synthetic Data 80
5.3.1 Interpreting CP and Tucker 80
5.3.2 A Synthetic Experiment 81
5.3.3 Results and Interpretation 82
5.4 Classification of Real Datasets 84
5.4.1 Numerical results 85
5.5 Chapter Summary 89

6 Simultaneously-updated Support Tensor Machine 91
6.1 Introduction 91
6.2 Regularized Tensor Machine Learning Model 93
6.2.1 Regularized Support Tensor Machine 93
6.3 Optimizing Regularized Models 99
6.3.1 Primal Dual SVM Relation 100
6.3.2 Partial Simultaneous Update using Gradient Descent 103
6.3.3 Full Simultaneous Update using Gradient Descent 104
6.3.4 Gradient Descent Enhancement 106
6.3.5 Full Simultaneous update TT-MMK 108
6.4 Numerical Assessment 109
6.4.1 Convergence Analysis 110
6.5 Chapter Summary 111

7 Summary and Outlook 113
7.1 Conclusions 113
7.2 Future Research Directions 115
7.2.1 Direction 1: Neural Support Tensor Machine (Multiple Kernel Learning) 115
7.2.2 Direction 3: Gene Expression Recovery 119
7.3 Outlook 122

Bibliography 123

Index 141

Theses 141

LIST OF FIGURES

1.1 Main Focus of Thesis. 5

2.1 Machine Learning type distribution. 9
2.2 Non-linear(kernel) and linear SVM. 13
2.3 Creation of reproducing kernel in Hilbert space 17
2.4 Nonlinear mapping using kernel trick: (a) Nonlinear classification of data in $\mathbb{R}^2$, (b) Linear classification in higher dimension ($\mathbb{R}^3$). 18
2.5 Kernel trick with RKHS embedding. 19
2.6 SVM as a Neural Network. 20

3.1 Unfolding of a 3-dimensional tensor. 29
3.2 Kronecker Product of two matrices **A** and **B**. 31
3.3 canonical polyadic decomposition of a 3D tensor. 34
3.4 Tucker decomposition of a 3-way tensor. 36
3.5 Tensor train decomposition of an $M^{th}-$order tensor. 38
3.6 Relation representation of an $M^{th}-$order tensor $\mathcal{X}$ between CP format and TT format. 41
3.7 Kernel trick with TP-RKHS embedding. 45

4.1 Test classification accuracy of different versions of the TT-MMK algorithm: "TT" vs "uTT" (left top), "TT" vs "TTCP" (right top), "TTCP" vs "TTCPe" (left bottom), and "TTCPe" vs the final algorithm "uTTCPe" (right bottom) for the ADNI dataset. Lines denote averages, and shaded areas denote 95% confidence intervals over 20 runs. 64
4.2 CPU time vs classification accuracy for ADNI data with truncation rank from 1 to 10. 66
4.3 Classification accuracy, average (lines) $\pm$ one standard deviation (shaded areas) over 20 runs. Left: ADNI dataset. Right: ADHD dataset. 67
4.4 Hyperspectral images with different labels 68
a GroundTruth of Indian Pines dataset 68
b GroundTruth of Salinas dataset 68

5.1 Classification Accuracy with 95% confidence interval for different noise levels in the *core*-scenario of the generated synthetic data with different rank truncation $r_{approx} = 1, 3, 5, 10$ (subspace kernel doesn't have any rank truncation). 82
5.2 Classification Accuracy with 95% confidence interval for different noise levels in the *leaf*-scenario of the generated synthetic data with different rank truncation $r_{approx} = 1, 3, 5, 10$ (subspace kernel doesn't have any rank truncation). 83
5.3 The plot shows the CPU time vs classification accuracy for synthetic data (*leaf*-case) with truncation rank from 1 to 10 between. The comparison is done between proposed WSEK (5.2.2) and subspace kernel ([167]). The subspace is referred as subspace in the plot and has no rank truncation. Therefore, there is only one point for each noise level. 84
5.4 The plot shows the CPU time vs classification accuracy for synthetic data (*core*-case) with truncation rank from 1 to 10 between. The comparison is done between proposed WSEK (5.2.2) and subspace kernel ([167]). The subspace is referred as subspace in the plot and has no rank truncation. Therefore, there is only one point for each noise level. 85
5.5 Classification Accuracy with 95% confidence interval for different p values in WSEK for ADNI (left) and ADHD (right) dataset with rank truncation $R \in [1, 10]$ 86
5.6 (left) Comparison of mean classification accuracy with variance for the different kernels with different rank truncation for ADNI dataset, (right) Comparison of mean classification accuracy with variance for the different kernels with different rank truncation for ADHD dataset. 87
5.7 Comaprison of 95% confidence interval for DuSK with CP, TT, and Tucker decompositions: (left) ADNI dataset (right) ADHD dataset. 88

7.1 Classification accuracy of ADHD data for 20 epochs. 117
7.2 TT-MMK Kernel Approximation. 118
7.3 Neural Support Tensor Network. 119

LIST OF TABLES

3.1 Storage Complexity. 41

4.1 Average classification accuracy in percentage for different methods and data sets 69

5.1 Memory Storage. 75

5.2 Theoretical complexity of computing a single entry of the different kernel matrices from data given in different formats. Note that some kernel-format combinations are not defined. 80

5.3 Maximum average classification accuracy in percentage ± standard deviation for different methods, data sets, and rank $R \in [1, 10]$. The values for TTCP-DuSK are taken from [98] for comparison. 87

5.4 Comparison of CPU time for $R \in [1, 10]$ for ADNI dataset. The values for TTCP-DuSK are taken from [98] for comparison. The time comparison is computed based on the originally presented work [71, 167]. 89

5.5 Comparison of CPU time for $R \in [1, 10]$ for ADHD dataset. The values for TTCP-DuSK are taken from [98] for comparison. The time comparison is computed based on the originally presented work [71, 167]. 89

LIST OF ALGORITHMS

1.1 Perceptron Learning Algorithm. 2

3.1 CP decomposition of a tensor using Basic ALS [68, 95]. 35
3.2 Sequential Truncated HOSVD [183] 38
3.3 TT-SVD Decomposition using truncated SVD (tSVD) [133] 39
3.4 TT Rounding (Recompression or Truncation) [133]. 40

4.1 Uniqueness Enforcing TT-SVD 55
4.2 TT-CP approximation of the STM Kernel 61

5.1 Tucker to CP. 75
5.2 Weighted HOSVD. 79

6.1 Projected Gradient Descent 105
6.2 Projected Gradient Descent for KSTM. 106
6.3 Simultaneously-update for TT-MMK (SimUpTT-MMK). 108

LIST OF SYMBOLS

Notation

Symbol	Description
$\mathbb{R}$, $\mathbb{C}$	fields of real and complex numbers
$\mathbb{C}_+$, $\mathbb{C}_-$	open right/open left complex half plane
$\mathbb{R}_+$, $\mathbb{R}_-$	strictly positive/negative real line
$\mathbb{R}^n$, $\mathbb{C}^n$	vector space of real/complex n-tuples
$\mathbb{R}^{m\times n}$, $\mathbb{C}^{m\times n}$	real/complex $m \times n$ matrices
$\lvert\xi\rvert$	absolute value of real or complex scalar
$\arg \xi$	argument of complex scalar
$\mathrm{Re}(\mathbf{A})$, $\mathrm{Im}(\mathbf{A})$	real and imaginary part of a complex
	quantity $\mathbf{A} = \mathrm{Re}(\mathbf{A}) + \imath\,\mathrm{Im}(\mathbf{A}) \in \mathbb{C}^{n\times m}$
$\overline{\mathbf{A}}$ a_{ij}	the (i, j)-th entry of $\mathbf{A}$
$\mathbf{A}(i : j, :)$, $\mathbf{A}(:, k : \ell)$	rows $i, \ldots, j$ of $\mathbf{A}$ and columns $k, \ldots, \ell$ of $\mathbf{A}$
$\mathbf{A}(i : j, k : \ell)$	rows $i, \ldots, j$ of columns $k, \ldots, \ell$ of $\mathbf{A}$
$\mathbf{A}^{\mathrm{T}}$	the transpose of $\mathbf{A}$
$\mathbf{A}^*$	$:= (\overline{\mathbf{A}})^{\mathrm{T}}$, the complex conjugate transpose
$\mathbf{A}^{-1}$	the inverse of nonsingular $\mathbf{A}$
$\mathbf{A}^{-\mathrm{T}}$, $\mathbf{A}^{-*}$	the inverse of $\mathbf{A}^{\mathrm{T}}$, $\mathbf{A}^*$
$\mathbf{I}_n$, $\mathbf{I}_{n,r}$	identity matrix of dimension n, first r columns of I_n
$\mathbb{1}_r$	$:= (1, \ldots, 1)^{\mathrm{T}} \in \mathbb{R}^r$
$\sigma_{\max}(\mathbf{A})$	the largest singular value of $\mathbf{A}$
$\mathrm{tr}(\mathbf{A})$	$:= \sum_{i=1}^{n} a_{ii}$, trace of $\mathbf{A}$

$\Vert u\Vert_p$	$:= (\sum_{i=1}^n \lvert u_i\rvert^p)^{1/p}$ for $u \in \mathbb{C}^n$ and $1 \leqslant p < \infty$
$\Vert u\Vert_\infty$	$:= \max_i \lvert u_i\rvert$, the maximum norm of u
$\Vert \mathbf{A}\Vert_p$	$:= \sup\{\Vert \mathbf{A}u\Vert_p : \Vert u\Vert_p = 1\}$, subordinate matrix p-norm, $1 \leqslant p \leqslant \infty$
$\Vert \mathbf{A}\Vert_\mathrm{F}$	$:= \sqrt{\sum_{i,j}\lvert a_{ij}\rvert^2} = \sqrt{\mathrm{tr}(\mathbf{A}^*\mathbf{A})}$, the Frobenius norm of matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$
$\Vert u\Vert$, $\Vert \mathbf{A}\Vert$	Euclidean vector or subordinate matrix norm $\Vert\cdot\Vert_2$
$\kappa_p(\mathbf{A})$	$:= \Vert\mathbf{A}\Vert_p \left\Vert\mathbf{A}^{-1}\right\Vert_p$, the p-norm condition number for nonsingular $\mathbf{A}$
$\kappa(\mathbf{A})$	the 2-norm condition number for regular $\mathbf{A}$
$\mathbf{A} \succ (\succeq)0$, $\mathbf{A} \prec (\preceq)0$	short form for $\mathbf{A}$ is self-adjoint positive/negative (semi)definite, also abbreviated by *s(s)pd* and *s(s)nd*
$\mathbf{A} > (\geqslant)\mathbf{B}$	$:\Leftrightarrow \mathbf{A} - \mathbf{B} > (\geqslant)0$, i.e., element-wise partial ordering: $(a_{ij} - b_{ij} > (\geqslant)0,\ \forall ij)$

$\overline{i_1,\dots,i_M}$	multi-index ($\overline{i_1,\dots,i_M} = 1 + \sum_{k=1}^{M}(i_k - 1)\prod_{m=1}^{k-1} I_m$)
x	scalar value ($x \in \mathbb{R}$)
$\mathbf{x}$	vector ($\mathbf{x} \in \mathbb{R}^I$)
$\mathbf{X}$	matrix ($\mathbf{X} \in \mathbb{R}^{I\times J}$)
$\boldsymbol{\mathcal{X}}$	tensor ($\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_M}$)
$x_{i_1 i_2\ldots i_M}$	$(i_1, i_2, \ldots, i_M)$-entry of a tensor
$\boldsymbol{\mathcal{X}}_{(m)}$	m-mode matricization $((x_{(m)})_{i_m,\overline{i_1,\ldots,i_{m-1},i_{m+1},\ldots,i_M}} = x_{i_1,\ldots,i_M})$
$\boldsymbol{\mathcal{X}}_{\langle m\rangle}$	mode-$\{m\}$ canonical matricization $((x_{\langle m\rangle})_{\overline{i_1,i_2,\ldots,i_m},\overline{i_{m+1},\ldots,i_M}} = x_{i_1,i_2,\ldots,i_M})$
$\boldsymbol{\mathcal{Z}} = \boldsymbol{\mathcal{X}} \odot_m \boldsymbol{\mathcal{Y}}$	mode-m Khatri–Rao product $(\boldsymbol{\mathcal{Z}}(:,\ldots:,i_m,:,\ldots,:)$ $= \boldsymbol{\mathcal{X}}(:,\ldots:,i_m,:,\ldots,:) \otimes \boldsymbol{\mathcal{Y}}(:,\ldots:,i_m,:,\ldots,:))$
$\boldsymbol{\mathcal{Z}} = \boldsymbol{\mathcal{X}} \times_m^n \boldsymbol{\mathcal{Y}}$	mode-(m,n) contracted product
$\boldsymbol{\mathcal{X}} \times_M^1 \boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} \times^1 \boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} \bullet \boldsymbol{\mathcal{Y}}$	mode-$(M,1)$ contracted product
$\boldsymbol{\mathcal{X}} \times_m \mathbf{A}$	m-mode product ($(\boldsymbol{\mathcal{X}} \times_m \mathbf{A})_{(m)} = \mathbf{A}\boldsymbol{\mathcal{X}}_{(m)},\ \ \mathbf{A} \in \mathbb{R}^{J\times I_m}$)
$\boldsymbol{\mathcal{Z}} = \boldsymbol{\mathcal{X}} \circ \boldsymbol{\mathcal{Y}}$	outer product ($z_{i_1,\ldots,i_M,j_1,\ldots,j_N} = x_{i_1,\ldots,i_M} y_{j_1,\ldots,j_N}$.)

$\mathbf{A} \otimes \mathbf{B}$	Kronecker product ($\begin{bmatrix} a_{i,1}\mathbf{B} & \cdots & a_{i,J}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{IK \times JL}$) $\mathbf{A} \in \mathbb{R}^{I \times J}, \mathbf{B} \in \mathbb{R}^{K \times L}$
$\mathcal{Z} = \mathcal{X} \circledast \mathcal{Y}$	Hadamard (component wise) product ($z_{i_1,i_2,\cdots,i_M} = x_{i_1,i_2,\cdots,i_M} y_{i_1,i_2,\cdots,i_M}$)
$\langle M \rangle$	integer values from 1 to M ($\{1, 2, \cdots, M\}$)
$\langle \mathcal{X}, \mathcal{Y} \rangle$	inner product of tensors ($\sum_{i_1}^{I_1} \sum_{i_2}^{I_2} \ldots \sum_{i_m}^{I_M} x_{i_1 i_2 \ldots i_m} y_{i_1 i_2 \ldots i_m}$)
$\|\mathcal{X}\|$	Frobenius norm of the tensor $\mathcal{X}$ ($\sqrt{\langle \mathcal{X}, \mathcal{X} \rangle}$)

LIST OF TERMINOLOGY

Mathematics	Machine learning & Statistics	Quantum Physics
$M-$th order tensor		rank-M tensor
Alternating Least Squares (ALS)	Forward-Backward Algorithms	One-site Density Matrix Renormalization Group (DMRG) / DMRG1
Modified Alternating Least Squares (MALS)	Block Nonlinear Gauss-Seidel Methods	Two-site Density Matrix Renormalization Group (DMRG) / DMRG2
Tensor Train decomposition (TT)	Hidden Markov Models (HMM)	Matrix Product Space (MPS) with Open Boundary Condition (OBC)
Tensor Networks	Neural Networks and Graphical Models	Tensor Networks

ABBREVIATIONS

Abbreviations	Description
3D or nD	3-dimensional, n-dimensional
ADNI	Alzheimer's Disease Neuroimaging Initiative
ADHD	Attention Deficit Hyperactivity Disorder
AI	Artificial Intelligence
ALS	Alternating Least Squares
BTF	Bayesian Tensor Factorization
CANDECOMP	CANonical DECOMPosition
CNN	Convolution Neural Network
CPU	Central Processing Unit
CP	Canonical Polyadic
CT	Computed Tomography
CV	Cross Validation
DL	Deep Learning
DMRG	Density Matrix Renormalization Group
DuSK	Dual Structure-preserving Kernel
EEG	Electroencephalogram
fMRI	Functional Magnetic Resonance Imaging
GA	Genetic Algorithm
GPU	Graphics Processing Unit
gPGD	Generalised Projected Gradient Descent
GQCP	Generalised Quantised Canonical Polyadic
HS	Hilbert Space

HSI	Hyper Spectral Imaging
HOSVD	Higher Order Singular Value Decomposition
ICA	Independent Component Analysis
KCP	Kernelised CP
KPCA	Kernel Principal Component Analysis
KKT	Karush Kuhn Tucker
KRR	Kernel Ridge Regression
KSTM	Kernelised Support Tensor Machine
K-STTM	Kernelised Support Tensor Train Machine
LOOCV	Leave One Out Cross Validation
LRMF	Low-rank Matrix Factorization
LRTF	Low-rank Tensor Factorization
MALS	Modified ALS
ML	Machine Learning
MLE	Maximum Likelihood Estimation
MLSVD	Multi Linear Singular Value Decomposition
MMK	Multi-way Multi-level Kernel
MKL	Multiple Kernel Learning
MSE	Mean Square Error
MPS	Matrix Product State
NMF	Non-negative Matrix Factorization
NLMA	Numerical Linear and Multilinear Algebra
NN	Neural Network
PARAFAC	PARAllel FACtorization
PCA	Principal Component Analysis
PGD	Projected Gradient Descent
QCP	Quantised Canonical Polyadic
QP	Quadratic Programming
QTT	Quantized Tensor Train

RBF	Radial Basis Function
RKHS	Reproducing Kernel Hilbert Space
SimUpTT-MMK	Simultaneously Updated TT-MMK
SMM	Support Matrix Machine
SMO	Sequential Minimal Optimization
STM	Support Tensor Machine
STuM	Support Tucker Machine
STTM	Support Tensor Train Machine
STL	Supervised Tensor Learning
SVM	Support Vector Machine
SVD	Singular Value Decomposition
TL	Transfer Learning
TP-RKHS	Tensor Product Reproducing Kernel Hilbert Space
TPU	Tensor Processing Unit
TT	Tensor Train
TTV	Tensor Time Vector
TTM	Tensor Time Matrix
TT-MMK	Tensor Train Multi-way Multi-level Kernel
uTT	Uniqueness enforced Tensor Train
uTTCPe	Uniqueness-enforced Tensor Train Canonical Polyadic with Equlibration
WSEK	Weighted Subspace Exponential Kernel

CHAPTER 1 INTRODUCTION

1.1 The Synopsis and Significance of A Journey Towards Conflation 1
1.2 Outline of This Thesis 6

Contents

1.1 The Synopsis and Significance of A Journey Towards Conflation

Psychological curiosity seeks to comprehend complex human behaviour by exploring the human mind, while neuroscientists attempt to uncover similar insights by studying the intricate human brain using a more practical data-based approach. In both cases, computer scientists/ mathematicians strive to develop tools that can analyse data obtained from case studies in psychology or experiments in neuroscience. Through these efforts, the main aim remains to understand how the brain engages in activities such as processing, analysing, and transforming information. The brain's neurodynamics thus form a natural foundation for learning how information is stored and processed. In the 1960s, while working at Cornell University, psychologist Rosenblatt made a connection the between brain's neurons and Perceptrons, which are simple machines that can simulate neurons. The correlation is straightforward (Algorithm 1.1): the inputs that the brain receives are like instructions, the prioritization of which instructions to execute is assigned weights ($\mathbf{w}$), executing instructions is accomplished through linear functions, and avoiding over/underdoing is achieved through a threshold. Perceptrons are genotypic models with a memory mechanism that enables them to learn responses to various stimuli in experiments. Initially, Rosenblatt and his team employed the Mark I Perceptron to classify images. However, complications arise when dealing with more complex situations, such as noise in data or other external factors. Just as the human brain

Algorithm 1.1: Perceptron Learning Algorithm.

```
$P$ : positive label input
$Q$ : negative label input
Initialize $\mathbf{w}$ randomly
while convergence do
  Pick random $\mathbf{x} \in P \cup Q$
  if $\mathbf{x} \in P$ and $\mathbf{w}^T\mathbf{x} < 0$ then
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$
  end if
  if $\mathbf{x} \in Q$ and $\mathbf{w}^T\mathbf{x} \geqslant 0$ then
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$
  end if
end while
```

struggles to cope with uncertain and complex situations, it is challenging to construct a machine's brain that can handle such circumstances.

Artificial Intelligence

In the 1960s, the field of artificial intelligence (AI) emerged to develop efficient computer models that could contain the intelligence of an artificially built brain learning a model capable of recognizing complex patterns with less memory storage. One such model is the Mark I Perceptron [148]. This was considered a far better artificially built brain model than the available computer. Rosenblatt's approach was based on the construction of decision rules associated with the finding of linear hyperplanes in a space (see Algorithm 1.1). However, Minsky and Papert's book, *Perceptrons: An Introduction to computational Geometry* [118], pointed out that the Perceptron has severe shortcomings with complex pattern recognition tasks, specifically, that it is unable to classify non-linear patterns. Its classificatory capacities are limited to linearly separable patterns, and that it has wider limitations of having a single-layer network. As a result, a mathematical framework for the Perceptron was needed for better scientific explanation of the feasibility of a task and also for a generalized understanding of the Perceptron. Before long, the much-needed early successes of the multi-layer perceptron back-propagation swung back the pendulum towards machine learning. Afterward, more complicated systems could be carried over from the perceptron. In this way, the perceptron became a theoretical archetype of machine learning today. However, Minsky and Papert's book was often interpreted as implying that, even with multi-layered perceptrons *back-propagation* [153, 151, 104, 135], the problem beyond *linear separability* could not be solved.

In 1995, Corinna Cortes and Vladimir Vapnik [47] presented a supervised learning model, “Support Vector Network,” which conceptually implemented the idea of a *non-linear embedding* of the input vectors to a high-dimensional feature space. In this feature space, a linear decision surface is constructed. The special properties of the decision surface ensure the high generalization ability of the learning machine. This provided a benchmark to go beyond linear separability that wasn’t possible before. It is noteworthy that VM is a learning system that can answer two important questions about learning machines. The first question is whether a learning machine can accurately represent the intended information. The second question is whether the learning algorithm can get closer to the desired information as it learns from more examples. For instance, a Support Vector Network that employs a universal kernel can not only represent any predicate with arbitrary precision but also approach it with increasing accuracy as the number of training examples grows. This guarantees an exceptional level of performance, exceeding that of the heuristic specification approach.

Machine learning has made great progress in practical applications, but our understanding of its underlying theory is still limited. However, computers are excellent at tasks that have clear mathematical instructions because programming is essentially a mathematical exercise. Unfortunately, many important tasks, like recognizing visual patterns, don’t have clear mathematical instructions.

There are two common ways to deal with this problem. One is to come up with approximate instructions based on rules of thumb, which is called heuristic construction. The other approach is to use a large set of training images and let a learning machine figure out the appropriate instructions. This second approach has been successful in cases where we have access to large datasets of images and have the computational power, such as with GPU computing, to process such large datasets. Even though the theoretical understanding behind learning machines is lacking, they have still achieved practical successes in various domains.

Neuroimaging

Functional magnetic resonance imaging (fMRI) [57, 124, 141] is a powerful tool used in various fields, including neuroscience, psychology, and medical diagnosis. These fMRI datasets are four-dimensional, comprising space and time, and contain voxels (3D pixels). Recently, there has been growing interest in using fMRI images to develop artificial intelligence (AI) models [184, 158]. One potential application of fMRI images in AI development is to train models to recognize patterns in brain activity that correspond to specific mental states or behaviours [160]. Another potential application is to use fMRI images to improve speech recognition algorithms by analysing brain activity during speech production and perception [175]. However, the collection of such data that is easily accessible, simple, or inexpensive may not be possible due to data protection laws, higher cost, and lack of resources. This is one major issue that has been overlooked by the machine learning community. One particular problem that is of great interest is the development of a model that can distinguish between healthy individuals and those with a specific disease based on resting-state functional magnetic resonance images. This is essentially a binary classification problem, but the challenge lies in the small sample size of the dataset.

Moreover, neural network-based techniques often attempt to learn patterns through least square curve-fitting issues, which can easily overlook rare information in medical data that could be crucial for detecting diseases. For instance, there exist NN-based models that correctly predicted the occurrence of cancer in a patient, not based on the tumour in the brain, but rather on some nearby information in the data [88]. This is a glaring flaw in terms of medical research.

Consequently, to analyse the complex and multidimensional data generated by fMRI scans, numerical multilinear algebra is used. This involves decomposing the data into simpler components to better understand the underlying patterns and structures in brain activity. This can lead to new insights into brain function and the development of more advanced AI algorithms. However, using fMRI images for DL/ML development comes with challenges, such as the need for large datasets and the difficulty of interpreting complex patterns of brain activity. Overall, the combination of fMRI images and numerical multilinear algebra has the potential to advance our understanding of the brain and develop more advanced AI algorithms [4, 5].

Numerical Multi-linear Algebra

As technology continues to advance, it has become increasingly clear that data is one of the most valuable resources for the future. In light of this, the big data paradigm has given rise to a pressing need for multidimensional machine learning techniques that provide tools for modelling and analysing complex data structures with multiple dimensions. Numerical multilinear algebra is a branch of linear algebra that deals with higher-order tensors, which are generalizations of matrices and vectors. The classical NLMA has been enthusiastically embraced by the machine learning community as a powerful tool for addressing big data issues [191, 26, 4, 207, 43, 5].

Meanwhile, in the mathematical community, researchers have been grappling with the challenges posed by multidimensional data. Thus, the concept of tensor (multidimensional arrays) decomposition was born [76, 75]. As the curse of dimensionality is a well-known problem in this field, researchers have found more effective ways of representing tensorial data [52, 99, 201, 134, 86]. This has led to a renewed interest in classical numerical linear algebra approaches and the generalization of low-rank methods for matricizing tensorial data. However, this process can be laborious and requires a significant amount of storage memory and high computational complexity. Moreover, the conversion into matrix or vector format may not preserve the underlying structure of the tensorial data. As a result, there has been a growing need for the direct implementation of tensor-based approaches.

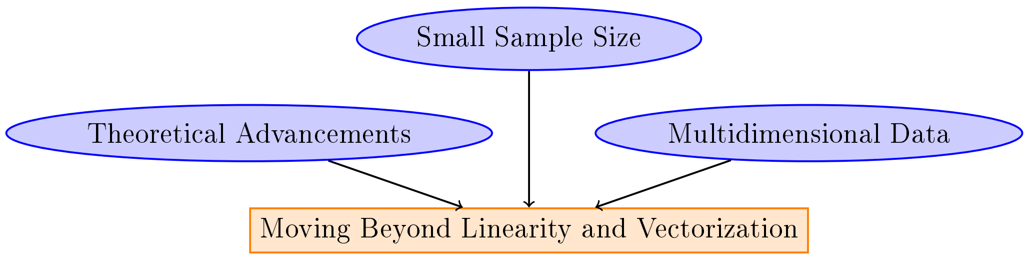


Figure 1.1: Main Focus of Thesis.

Hence, researchers across various fields, such as mathematics, physics, quantum chemistry, and chemometry, have tackled these concerns independently. In the late twentieth century, the first tensor decomposition was introduced by two different groups [28, 68]. Since then, many more efficient tensor decomposition formats have been introduced to solve scientific issues. Some of these well-known low-rank formats are highly stable, easily tunable to specific problems, and require approximately linear storage in terms of data dimension. These are some of the qualities that make low-rank tensor formats an attractive choice to explore further.

Moving Beyond Linearity and Vectorization (Significance of conflation)

Theoretical and algorithmic advancements in terms of the explainability or interpretability of AI models are crucial in several fields, especially in situations where the consequences of a model's predictions can have a significant impact on people's lives, such as healthcare, finance, and criminal justice. Improved algorithms can help address increasingly complex issues and unlock AI's full potential for new innovations. Therefore, the need for robust and more interpretable models is at its peak to evolve and meet the ethics of AI while considering its impact on society. The presented research work aims to promote tiny discoveries and build new classification ML models, acting as a catalyst for building the theoretical foundation of n dimensional (nD)-ML models. As shown in Figure 1.1, the main contribution breaks down into three main categories. Firstly, the model goes beyond linearity to non-linear machine learning that can be more accurate than linear methods in certain scenarios, such as highly non-linear data or complex interactions between features. Secondly, to incorporate multidimensionality, moving beyond vectorization provides a solution to the issue of lost structural information. Additionally, the curse of dimensionality is addressed with the important techniques of numerical multilinear algebra in machine learning based on tensor decompositions for extracting useful information from high-dimensional data. Tensor decomposition involves breaking down a tensor into a sum of simpler tensors, which can reveal hidden structure or patterns in the data. The tools borrowed from the NLMA community help combine and analyse data from different views by enabling the development of more accurate and robust models. Models that rely on such techniques have applications in fields such as image processing [122, 190, 187, 159, 95], signal processing [165, 168, 173, 41], and several other fields [40, 42, 37, 4, 21]. Beyond building nonlinear tensor-based models, the pinnacle of the presented research is to build well-explained, state-of-the-art, robust, and computationally cheap models.

1.2 Outline of This Thesis

The thesis has been divided into three main parts: an introduction to the topic in the first part, contributed work in the second part, and a conclusion of the research in the third and final part. The main goal of this monograph is to combine research projects on low-rank tensor decompositions in kernel-based methods.

Chapter 2 in the first part of the monograph introduces classical machine learning classification models, with a focus on the maximum margin model, also known as "Support Vector Machine (SVM)." This chapter provides a detailed description of the SVM model for linear and non-linear boundary value problems, along with some terminologies and definitions that are commonly used in the machine learning community. In Chapter 3, the vector-based classification model is elaborated to a tensor-based model. As the multidimensional (tensor) based model has a long research literature, Chapter 3 covers a short review of the state-of-the-art models of the respective period that hinge on different low-rank tensor factorizations. Hence, the chronicle of the developed tensor method and corresponding algorithms are well-explained.

The contributed work is distributed among three chapters. Chapter 4 discusses the development of a new low-rank tensor decomposition algorithm that optimizes the non-linear boundary classification SVM model. The work presented in this chapter was originally published in the "Journal of Machine Learning Research [98]." Chapter 5 is available on arXiv and is currently in the final stage of preparation [97]. The primary task was the development of another new low-rank tensor decomposition algorithm along with a new function to define non-linearity among tensor data. Chapter 6 is an auxiliary chapter and has not been published or submitted anywhere. The work done in this chapter is an extension of Chapter 4. This chapter highlights certain theoretical aspects of the non-convex, non-linear regularized SVM model for tensor input data.

The last part of this thesis includes Chapter 7. The §7.1 is a conclusive part of the thesis that demonstrates the strengths, weaknesses, and an analysed overview of the proposed contributed work. Meanwhile, §7.2 is designed to leave readers intrigued by compelling them to ponder upon plausible open questions and their interconnection with other fields, as well as applications.

CHAPTER 2

MATHEMATICAL FOUNDATIONS OF SUPERVISED MACHINE LEARNING

2.1 Mathematical Introduction to Supervised Learning Models 7
2.2 Linear Classification 10
2.2.1 Support Vector Machine 10
2.2.2 Primal-dual Relationship 11
2.3 Nonlinear Classification 13
2.3.1 Reproducing Kernel Hilbert Spaces 15
2.3.2 Feature Map and Kernel Trick 17
2.3.3 Kernelised Support Vector Machine 19
2.4 Optimizing a Supervised Learning Model 21
2.4.1 Cross Validation 22
2.5 Chapter Summary 24

Contents

2.1 Mathematical Introduction to Supervised Learning Models

Linear regression [70], the earliest form of what is now known as regression analysis, was developed using the method of the least squares in the 19th century. In the 1940s, logistic regression was introduced as an alternative approach. The term “generalized linear model” was coined in the early 1970s to describe a class of statistical learning methods that included both linear and logistic regression [48]. However, these methods were primarily linear, as fitting non-linear relationships was computationally challenging at the time. By the 1980s, advancements in computing technology made non-linear methods more accessible [82]. This led to the development of classification and regression trees, generalized additive models, and neural networks [116, 59]. In the 1990s, support vector machines emerged as a popular non-linear method.

The idea of building these mentioned linear and nonlinear model throughout history was to find an interpolation of input data points corresponding to output data points. Suppose input data is $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, $\mathbf{x}_i \in \mathbb{R}^m$ and response is $\mathcal{Y}$ then general formulation of interpolation function would be written as,

$$\mathcal{Y} = f(\mathcal{X}) + \epsilon,$$

where f is some *unknown function* and ϵ is an *error term*. In all the cases, those models approximate the target variable $\mathcal{Y}$ by a weighted linear combination of input variables, $\mathbf{w}^{\mathrm{T}}\mathbf{x} + b$, where $\mathbf{w} \in \mathbb{R}^m$. For developing the model, we would need to estimate the function f by a function $\hat{f}$ which is a prediction function at a set of predictors $\mathcal{X}$ such that $\mathcal{Y} \approx \hat{f}(\mathcal{X})$ for any observation $(\mathcal{X}, \mathcal{Y})$. For minimizing the error or finding the best estimation $\hat{f}$, we need to measure the quality of the fit via calculation *mean square error* (MSE), given by,

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}(\mathbf{x}_i))^2, \tag{2.1}$$

the MSE represents the *training error*, this means it observe the fit for a seen dataset that is used during the training of model. While, we are interested in the accuracy of the predictions of our method to previously unseen test data. Therefore, the training MSE is not a good metric to build a generalized model. Mathematically, we want an estimated function $\hat{f}$ such that $\hat{f}(\mathbf{x}_{i0}) \approx y_{i0}$, where $(\mathbf{x}_{i0}, y_{i0})$ are previously unseen data point. This implies that we want to minimize the *test* MSE, given by,

$$test\ MSE = \frac{1}{N_s}\sum_{i=1}^{N_s}(y_{i0} - \hat{f}(\mathbf{x}_{i0}))^2, \quad N_s \rightarrow \text{number of test data} \tag{2.2}$$

since the training MSE and the test MSE appear to be closely related. Unfortunately, there is a fundamental problem with this strategy: there is no guarantee that the method with the lowest training MSE will also have the lowest test MSE. As the flexibility of the supervised learning method increases, there is a monotone decrease in the training MSE and a U-shape in the test MSE [70, 80]. This is a fundamental property of statistical learning that holds regardless of the particular data set at hand and regardless of the supervised model being used.

Overfitting [80]: Increasing model flexibility reduces training MSE, but it may not improve test MSE. Overfitting occurs when a method produces a small training MSE but a large test MSE because it has found patterns in the training data that do not exist in the test data. Overfitting happens when a more flexible model results in a larger test MSE than a less flexible model would have.
Underfitting [80]: It occurs when a model is not flexible enough to capture the underlying patterns in the data. This can result in both high training and high test errors. In other words, the model is too simple to capture the complexity of the data, and as a result, it performs poorly on both the training and test data.

In practice, one can usually compute the training MSE with relative ease, but estimating test MSE is considerably more difficult because usually no test data are available. One important method to estimate this minimum is *cross-validation* (§2.4.1), which is a cross-validation method for estimating test MSE using the training data.

Classification Model

The above-mentioned concepts for regression setting are directly encountered in classification setting with only some changes due to the fact that $\mathcal{Y} \notin \mathbb{R}$ any more rather $y \in \{-1, 1\}^N$ for the given predictors $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, $\mathcal{X} \in \mathbb{R}^{N,m}$ with N training measurements. The response variable in case of *classification* is *qualitative*, that are often referred to as *categorical*. Further we discuss a special approach for predicting qualitative responses, a process that is known as classification. Classification problems, both binary and multiclass, are encountered in numerous applications in daily life and scientific research. These applications include healthcare (disease diagnosis), marketing, retail, image and speech recognition, sentiment analysis, natural language processing (document classification), and more. Thus, the significance of classification models cannot be underestimated in these fields. The distribution of classification models is shown in Figure 2.1.

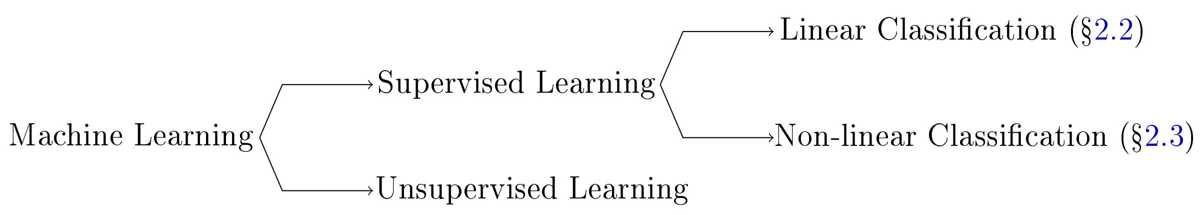

Linear Classification (§2.2)
Supervised Learning
Machine Learning
Non-linear Classification (§2.3)
Unsupervised Learning

Figure 2.1: Machine Learning type distribution.

2.2 Linear Classification

Early machine learning models were based on linear classification of data. This means, the data is linearly separable by a hyperplane that needs to be optimized (see Figure 2.2). As explained in introductory Chapter 1, the Perceptron was a first linear classification model. This section focuses mainly on one of the most popular methods that are for classifying data points Support Vector Machines (SVM) [185, 186]. These are based on margin maximization and the computation of the corresponding weights via an optimization framework, typically the SMO algorithm [140].

2.2.1 Support Vector Machine

For constructing Support Vector Machine (SVM) for linear classification model a training data set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N \in (\mathcal{X}, \mathcal{Y})$, with *input data* $\mathbf{x}_i \in \mathbb{R}^m$ with $y_i = f(\mathbf{x}_i), \quad \forall i \in \langle N \rangle$ is used. A hypotheses H, a binary classifier like SVMs, arranges now every new observation into the target space $\mathcal{Y}$. For SVM there can be many such hypotheses the ones with a small complexity are favoured. A linear classifier has hypotheses with small complexity, which can be defined as follows:

$$H = \{x \mapsto sign(\mathbf{w}^T\mathbf{x} + b) : \mathbf{w} \in \mathbb{R}^m, b \in \mathbb{R}\}.$$

The SVM hypothesis H could be defined as follows,

$$f(x) = \begin{cases} \mathbf{w}^T\mathbf{x}_i + b > 0, & y = 1 \\ \mathbf{w}^T\mathbf{x}_i + b < 0, & y = -1 \end{cases}$$

The goal here is to find an optimization regime for above-mentioned hypothesis and for finding optimized hyperparameters $\mathbf{w}$ and b.

Separable and non-separable case

In the separable case, the training sample set $\mathcal{X}$ can be separated perfectly by a linear hyperplane into positively and negatively labelled points. Generally, it is not possible to clearly separate the training data set $(\mathcal{X}, \mathcal{Y})$. That means it is possible that there are points which are located between the marginal hyperplanes or on the wrong side of the hyperplane. Then there exists $\mathbf{x}_i \in S$ such that

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1.$$

Therefore the slack variables ξ_i exists to get a relaxed version where points lying in the margin or on the wrong side are allowed. In this relaxed version, for each $i \in \langle N \rangle$, there exist $\xi_i \geqslant 0$ such that

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1 - \xi_i.$$

If a point is classified incorrectly or correctly but with a margin less than 1 it has a slack variable $\xi_i > 0$. This point is defined as an outlier. It is now desired to reduce the sum of the slack variables $\sum_{i=1}^{N} \xi_i$, or, more generally $\sum_{i=1}^{N} \xi_i^k$ for some $k \geqslant 1$. A more detailed description is given in [70].

Remark 2.1:
According to the statistical learning theory [186], SVM-based learning performs well when the number of training measurements is larger than the complexity of the model. Moreover, the complexity of the model and the number of parameters to describe the model are always in a direct proportion. ◊

In other words, having more data to train on can lead to better results, especially when dealing with more complex models. The complexity of a machine learning model, such as an SVM, is directly related to the number of parameters required to describe the model. In essence, as the model becomes more complex, it tends to have more parameters that need to be adjusted during the learning process.

2.2.2 Primal-dual Relationship

This subsection focuses on finding the smallest complexity SVM's hypotheses. To define an optimization problem for SVM we need to maximize the margin $\rho = \frac{1}{\|\mathbf{w}\|}$ that means minimize the distance $\|\mathbf{w}\|$ or $\frac{1}{2}\|\mathbf{w}\|^2$. The SVMs non-separable [70] case not only includes the maximizing margin but also needs a trade-off parameter $C \geqslant 0$ with slack variable to keep an account of non-separable data points. These parameters manage to retain a trade-off between maximizing the margin and minimizing the sum of slack variables. The problem formulation is the following optimization problem,

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i^k \tag{2.3}$$
$$\text{subject to} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1 - \xi_i \wedge \xi_i \geqslant 0, i \in \langle N \rangle.$$

The tendency of this primal optimization problem is having a unique solution. The concern is to find the best learning algorithm or an optimization method. The optimization problem is convex since the constraints are affine and the objective function is convex for any $k \geqslant 1$. The norm $\|\cdot\|_k$ is convex so that the objective function $\boldsymbol{\xi} \mapsto \sum_{i=1}^{N} \xi_i^k = \|\boldsymbol{\xi}\|_k^k$ is also convex. The choice of k leads to more or fewer penalizations of the slack terms. For $k = 1$ (Hinge loss) or $k = 2$ (quadratic loss) we get the most straightforward solution. Here we consider $k = 1$ for the following analysis. Considering a quadratic objective function and affine constraints, a quadratic programming (QP) optimization problem is achieved. As long as, concerns are related to such convex QP, there are a lot of solvers [199, 63, 120, 55], but the computational cost is high. As for the dual optimization, there exist optimization procedures that are speeding up the training of the SVM because the dual optimization problem is not depending on ξ_i and therefore also not depending on the Lagrange variables $\mu_i \geqslant 0, i \in \langle N \rangle$. The Lagrangian can then be defined for all $\mathbf{w} \in \mathbb{R}^m, b \in \mathbb{R}$, and $\boldsymbol{\alpha} \in \mathbb{R}_+^N$, by

$$\mathcal{L}_P(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\boldsymbol{\alpha}_i[y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i] - \sum_{i=1}^{N}\mu_i\xi_i.$$

The optimal solution needs to satisfy the KKT conditions:

1. Primal Feasibility:

$$(1 - \xi_i - y_i(\mathbf{w}^T\mathbf{x}_i + b)) \leqslant 0 \tag{2.4}$$
$$-\xi_i \leqslant 0 \implies \xi_i \geqslant 0 \tag{2.5}$$

2. Dual feasibility:

$$\boldsymbol{\alpha}_i \geqslant 0, \quad \mu_i \geqslant 0 \tag{2.6}$$

3. Complementary slackness:

$$\forall i, \boldsymbol{\alpha}_i(1 - \xi_i - y_i(\mathbf{w}^T\mathbf{x}_i + b)) = 0 \implies \boldsymbol{\alpha}_i = 0 \vee y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1 - \xi_i \tag{2.7}$$
$$\forall i, \mu_i\xi_i = 0 \implies \mu_i = 0 \vee \xi_i = 0. \tag{2.8}$$

4. Gradient:

$$\nabla_{\mathbf{w}}\mathcal{L} = \mathbf{w} - \sum_{i=1}^{N}\boldsymbol{\alpha}_i y_i \mathbf{x}_i = 0 \implies \mathbf{w} = \sum_{i=1}^{N}\boldsymbol{\alpha}_i y_i \mathbf{x}_i \tag{2.9}$$
$$\nabla_b\mathcal{L} = -\sum_{i=1}^{m}\boldsymbol{\alpha}_i y_i = 0 \implies \sum_{i=1}^{N}\boldsymbol{\alpha}_i y_i = 0 \tag{2.10}$$
$$\nabla_{\xi_i}\mathcal{L} = C - \boldsymbol{\alpha}_i - \mu_i = 0 \implies \boldsymbol{\alpha}_i + \mu_i = C \tag{2.11}$$

From these KKT conditions the following dual form of Lagrangian is obtained, where b is *bias*, $\boldsymbol{\alpha}$ are the dual optimization parameter, both $\boldsymbol{\alpha}$ and b depends on *support vectors*. The *support vectors* are the points that lies on the marginal hyperplanes. Consequently, the maximum-margin hyperplane is defined by the support vectors. We get the objective function:

$$\mathcal{L}_D = \min_{\boldsymbol{\alpha}} \sum_{i=1}^{N}\boldsymbol{\alpha}_i - \frac{1}{2}\sum_{i=1}^{N}\boldsymbol{\alpha}_i\boldsymbol{\alpha}_j y_i y_j (\mathbf{x}_i^T\mathbf{x}_j)$$

$$\text{subject to (Linear Constraints):} \quad 0 \leqslant \boldsymbol{\alpha}_i \leqslant C \wedge \sum_{i=1}^{N}\boldsymbol{\alpha}_i y_i = 0. \tag{2.12}$$

Note that (2.12) is a quadratic function of $\boldsymbol{\alpha}$, therefore this optimization is a QP problem. The so far best known algorithm is Sequential minimal optimization (SMO, used as optimization method in `LIBSVM`[1]; mentioned in §4.3) method [140]. This optimization algorithm speeds up the training of SVM. The benefit of this algorithm is that it uses several small optimizations with only two Lagrange multipliers instead of carrying out one large optimization. This $\boldsymbol{\alpha}$ can be used to determine the hypothesis or decision function for a point x via:

$$f(x) = \operatorname{sign}(\mathbf{w}^T\mathbf{x} + b) = \operatorname{sign}\left(\sum_{i=1}^{N} \boldsymbol{\alpha}_i y_i {\mathbf{x}_i}^T \mathbf{x}_i + \sum_{i=1}^{N}\left(y_i - \sum_{j=1}^{N} \boldsymbol{\alpha}_j y_j {\mathbf{x}_i}^T \mathbf{x}_j\right)\right) \tag{2.13}$$

The training of linear SVMs is straight forward and simple. However, the training data is not usually linearly separable. If that is the case we can try to find a linear hyperplane in a higher dimension. This process is mentioned in the next section.

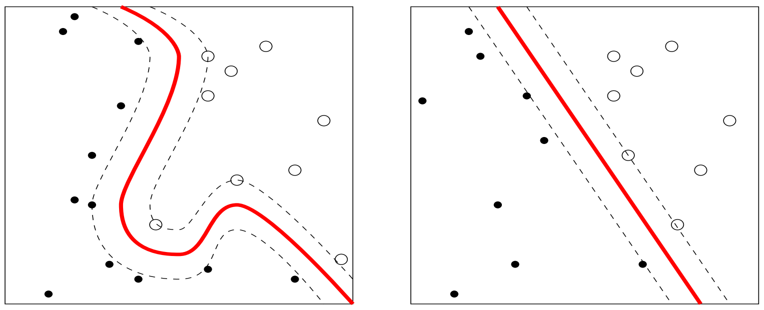


Figure 2.2: Non-linear(kernel) and linear SVM.

2.3 Nonlinear Classification

For nonlinear classification (see Figure 2.2), the data get transformed by a nonlinear function $\Phi(x)$ into a Feature space with higher dimensionality.

Similar to linear classification case the nonlinear SVM classifier [70, 80] for non-separable with hinge loss ($k = 1$) case can be given as follows,

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$
$$\text{subject to} \quad y_i(\langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle + b) \geqslant 1 - \xi_i, \quad \xi_i \geqslant 0, \quad \forall i \tag{2.14}$$

[1]https://www.csie.ntu.edu.tw/~cjlin/libsvm/

and the Lagrangian becomes,

$$\mathcal{L}_P(\mathbf{w}, b, \boldsymbol{\alpha}, \mu) = \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{N}\xi_i + \sum_{i=1}^{N}\boldsymbol{\alpha}_i(1 - \xi_i - y_i(\langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle + b)) - \sum_{i=1}^{N}\mu_i\xi_i. \tag{2.15}$$

The KKT conditions would immediately work as linear case and the above-mentioned parameters $\mathbf{w}$ and b are computed accordingly,

$$\text{weight:} \quad \mathbf{w} = \sum_{i=1}^{N}\boldsymbol{\alpha}_i y_i \Phi(\mathbf{x}_i) \tag{2.16}$$

$$\begin{aligned} \text{bias:} \quad b &= \sum_{i=1}^{N} y_i - \mathbf{w}^T\Phi(\mathbf{x}_i) \\ &= \sum_{i=1}^{N}\Big(y_i - \langle \sum_{j=1}^{N}\boldsymbol{\alpha}_j y_j \Phi(\mathbf{x}_j), \Phi(\mathbf{x}_i)\rangle\Big) \end{aligned} \tag{2.17}$$

If we take $\mathbf{w}$ from the primal form $\mathbf{w} = \sum_{i=1}^{N}\boldsymbol{\alpha}_i y_i \Phi(\mathbf{x}_i)$ and put this into above equation along with other constraint, then we end up with the dual formulation and corresponding relation between weight parameters and data can be given in the following way,

$$\begin{aligned} &\max_{\boldsymbol{\alpha}_1,\dots,\boldsymbol{\alpha}_N} \quad \sum_{i=1}^{N}\boldsymbol{\alpha}_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\boldsymbol{\alpha}_i\boldsymbol{\alpha}_j y_i y_j \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j)\rangle \\ &\text{subject to} \quad 0 \leqslant \boldsymbol{\alpha}_i \leqslant C \wedge \sum_{i=1}^{N}\boldsymbol{\alpha}_i y_i = 0, \end{aligned} \tag{2.18}$$

or the well known form of the *nonlinear* binary classification given by,

$$\begin{aligned} &\underset{\boldsymbol{\alpha}}{min} \quad \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{Q}\boldsymbol{\alpha} - \boldsymbol{e}^T\boldsymbol{\alpha} \\ &\text{subject to} \quad 0 \leqslant \boldsymbol{\alpha}_i \leqslant C, \quad i = 1, \dots, N \wedge \sum_{i=1}^{N}\boldsymbol{\alpha}_i y_i = 0 \end{aligned} \tag{2.19}$$

where $\mathbf{Q}_{ij} = y_i y_j \Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_j), \boldsymbol{e} = [1, 1, \cdots, 1]^T$ and $\mathbf{w} = \sum_{i=1}^{N}\boldsymbol{\alpha}_i y_i \Phi(\mathbf{x}_i)$. With this the decision boundary is as follows:

$$\begin{aligned} \langle \mathbf{w}, \mathbf{x}\rangle + b \text{ (linearity)} &\rightarrow \langle \mathbf{w}, \Phi(\mathbf{x})\rangle + b \text{ (nonlinearity)} \\ f(\mathbf{x}) &= \sum_{i=1}^{N}\boldsymbol{\alpha}_i y_i \Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}) + b, \end{aligned} \tag{2.20}$$

where a tuning function Φ defines the nonlinear decision boundary with $\Phi\colon \mathbf{x}_i \to \Phi\left(\mathbf{x}_i\right)$. In practice, explicitly knowing the right Φ function is not obvious. Also, the computation of $\left\langle \Phi\left(\mathbf{x}_i\right), \Phi\left(\mathbf{x}_j\right)\right\rangle$ is expensive when it is mentioned explicitly. Therefore, finding an implicit way to this issue is required. Typically, it is done by using the so-called *Kernel Trick* [155, 154].

Remark 2.2:
The given constraint in Lemma 2.11 for input training data $\mathbf{x}_i \neq \mathbf{x}_j, \forall i \neq j$ can easily be fulfilled for any classification problem, just by adding a small perturbations to each data point. $\diamondsuit$

2.3.1 Reproducing Kernel Hilbert Spaces

Some important definitions and theorems are presented here to understand kernel trick [155, 154]. The mathematical setting for it to hold

Definition 2.3:
A Hilbert space $(\mathcal{H})$ is a complete inner product space. $\diamondsuit$

Definition 2.4:
A functional f over $\mathcal{H}$ is Linear functional such that $L_x : \mathcal{H} \to \mathbb{R}$ with $L_x(f) = f(x)$, $\forall f \in \mathcal{H}$ $\diamondsuit$

Theorem 2.5 (Riesz Representation Theorem [11, 150]):
In a Hilbert space $\mathcal{H}$, all bounded linear functionals are of the form $\langle \cdot, f\rangle_{\mathcal{H}}$ for some $f \in \mathcal{H}$. This means f on $\mathcal{H}$ represents as, $f(x) = \langle x, z\rangle$, where f is uniquely defined with $\|f\| = \|z\|$. $\diamondsuit$

Definition 2.6:
[8, 27] $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a *kernel* if

(a) symmetry: $k(x, y) = k(y, x)$

(b) positive semi-definite $K_{ij} = k(x_i, x_j), \forall i, j \in \langle m\rangle$, & $x_i, x_j \in \mathcal{X}$, where $\forall \mathbf{a} \in \mathbb{R}^I$ Gram Matrix K, s.t. $\mathbf{a}^{'} K \mathbf{a} \geqslant 0$ $\diamondsuit$

In order to compute $\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j)\rangle$ we are trying to learn it explicitly. This is done by using the following definition of *Reproducing Kernels.*

Definition 2.7 (Reproducing Kernel):
[8] Let $\left(\mathcal{H}, \langle \cdot, \cdot\rangle\right)$ be a Hilbert space of real valued functions Φ on some set $\mathcal{X}$. A function $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is said to be a *Reproducing Kernel* of $\mathcal{H}$ iff :

(a) $k(\cdot, x) \in \mathcal{H}, \forall x \in \mathcal{X}$

(b) $\left\langle k(\cdot, x), \Phi \right\rangle = \Phi(x), \forall x_1, x_2, \ldots, x_N \in \mathcal{X}, \forall \Phi \in \mathcal{H}$.

By using reproducing kernel, a Hilbert space can be defined on such kernels.

Definition 2.8:
A Reproducing Kernel Hilbert Space (RKHS) [16] $\left(\mathcal{H}, \langle\cdot,\cdot\rangle, k\right)$, is a Hilbert space $\mathcal{H}$ of functions $\left(\mathcal{H}, \langle\cdot,\cdot\rangle\right)$ that possesses a reproducing kernel k. $\Diamond$

One of the most important theorem that have been used for solving a nonlinear statistical models [70] is as follows,

Theorem 2.9 (Mercer's Theorem [81, 51, 27]):
Consider k is a continuous positive semi-definite kernel on a compact set $\mathcal{X}$, and $T_k \colon L_2(\mathcal{X}) \to L_2(\mathcal{X})$

$$(T_k f)(\cdot) = \int_{\mathcal{X}} k(\cdot, x) f(x) dx$$

is positive semi-definite integral operator, this means, $\forall f \in L_2(\mathcal{X})$,

$$\int_{\mathcal{X}} k(s,t) f(s) f(t) ds dt \geqslant 0.$$

Then there exists an *orthonormal basis* $\{\phi_i\}$ of $L_2(\mathcal{X})$ space that consist of eigenfunctions of T_k such that the corresponding eigenvalues λ_i are non-negative. Also, the eigenfunctions that correspond to non-zero eigenvalues are continuous in $\mathcal{X}$ and the kernel $k(s,t)$ is

$$k(s,t) = \sum_{\infty}^{i=1} \lambda_i \phi_i(s) \phi_i(t)$$

for which the convergence is absolute and uniform, that is,

$$\lim_{m\to\infty} \sup_{s,t} |k(s,t) - \sum_{m}^{i=1} \lambda_i \phi_i(s) \phi_i(t)| = 0.$$

Theorem 2.10 (Moore-Aronszajn Theorem [172]):
If $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is positive definite then there is a unique RKHS $\mathcal{H} \subset \mathbb{R}^{\mathcal{X}}$ with reproducing kernel k. $\Diamond$

Figure 2.3 clarify connections among above mentioned definitions, that helps to further elaborate the connection between nonlinear function Φ and a reproducing kernel [8].

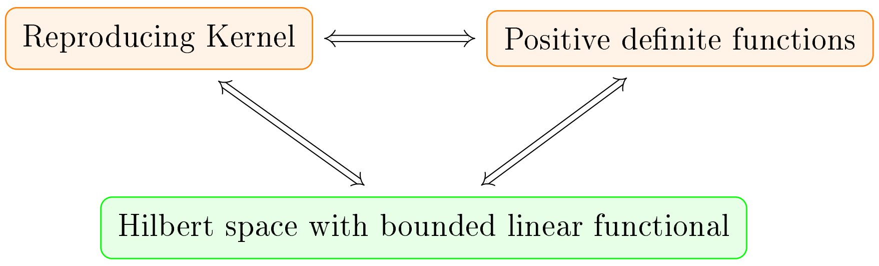

Reproducing Kernel
Positive definite functions
Hilbert space with bounded linear functional

Figure 2.3: Creation of reproducing kernel in Hilbert space

2.3.2 Feature Map and Kernel Trick

Hence, there exists a well-defined function $\Phi\colon \mathbb{R}^m \rightarrow \mathbb{F}$ such that $\Phi(x) = k_x$ (reproducing kernel), $k : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$,

$$k_{i,j} = k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle_{\mathbb{F}}, \tag{2.21}$$

the typically unknown function (Φ) is called **feature map**, and the *feature space* $\mathbb{F}$ is a Hilbert Space [156]. The evaluation of the inner product between two feature vectors is carried out in (2.21), is known as the *kernel trick*, that not only is computationally tractable but also avoids the function Φ. The kernel matrix ($K = \left[k(\mathbf{x}_i, \mathbf{x}_j)\right]_{ij} \in \mathbb{R}^{m \times m}$) that results from the continued evaluation of the kernel function on the set of data points is then positive semi-definite. It is used to get a linear learning algorithm to learn a *nonlinear boundary*, without explicitly knowing the nonlinear function Φ. The only task needed for the SVM is thus to choose a legitimate kernel function. That is how we work with the input data in the high-dimensional space while doing all the computation in the dual low dimensional space. Figure 2.4 illustrates the linear separation in a higher dimensional space. The inner product is computed using kernel function.

This is called the *kernel trick*. Some of the most popular reproducing kernels are the Fisher Kernel, Radial Basis Function Kernel (RBF), and Polynomial Kernel. The Fisher Kernel analyses and measures the similarity of two objects while the values of the RBF Kernel only depend on the distance from the center. The Polynomial Kernel uses the polynomials of the original variables to embody the similarity of vectors in the higher

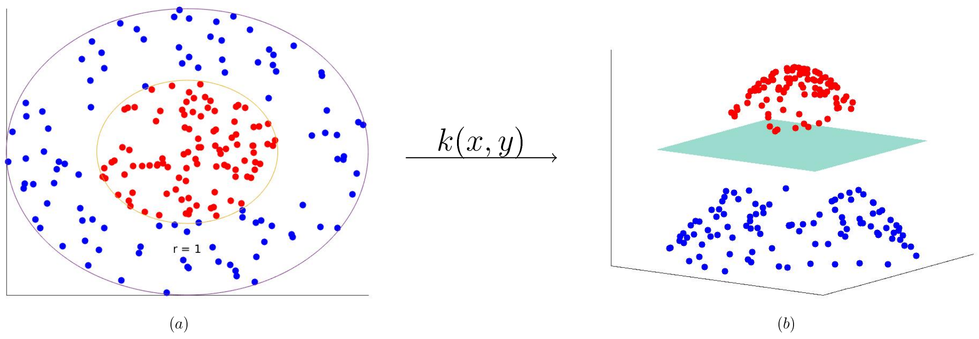


Figure 2.4: Nonlinear mapping using kernel trick: (a) Nonlinear classification of data in $\mathbb{R}^2$, (b) Linear classification in higher dimension ($\mathbb{R}^3$).

dimension. Some of these popular choices for k are given as,

$$\begin{aligned}
\text{Linear}: k(\mathbf{x},\mathbf{x}^{'}) &= \langle \mathbf{x},\mathbf{x}^{'} \rangle,\\
d^{th}\text{degree polynomial}: k(\mathbf{x},\mathbf{x}^{'}) &= (\theta + \langle \mathbf{x},\mathbf{x}^{'} \rangle)^d,\\
\text{Radial Basis or Gaussian}: k(\mathbf{x},\mathbf{x}^{'}) &= \exp\Big(- \frac{\|\mathbf{x}-\mathbf{x}^{'}\|^2}{2g^2}\Big),\\
\text{Fisher Kernel}: k(\mathbf{x},\mathbf{x}^{'}) &= \mathbf{U}_{\mathbf{x}}^T \mathcal{I}^{-1} \mathbf{U}_{\mathbf{x}'},\\
&\text{Fisher score [106]} \quad \mathbf{U}_{\mathbf{x}} = \nabla_\theta log P(\mathbf{x}|\theta), \quad \mathcal{I} \rightarrow \text{Fisher information}\\
\text{Inverse Multiquadratic}: k(\mathbf{x},\mathbf{x}^{'}) &= \frac{1}{\sqrt{\|\mathbf{x}-\mathbf{x}'\|^2 + c^2}}.
\end{aligned}$$

These kernels allow non-linear classification for the (2.19) while computing $\mathbf{Q}_{ij} = y_i y_j k\left(\mathbf{x}_i, \mathbf{x}_j\right)$. This way the SVM model performs well in higher dimensions. The solution of the convex optimization problem (2.19) is defined as the global minimum so that an optimal hyperplane is guaranteed. But finding the right kernel can cause a high computational cost. The dimensionality of the RKHS in Gaussian is infinite while in polynomial it is finite. As by above given Moore-Aronszajn theorem, it is shown that a symmetric positive definite kernel is associated to a unique RKHS [172]. So by exhibiting an infinite-dimensional RKHS corresponding to the Gaussian kernel, that is done in [172] and similarly for polynomial kernel as well.

The nonlinear function $\Phi(x)$ transform data into a Hilbert Space while preserving linearity of data (that means data can be linearly separable in higher dimension) according to Lemma 2.11. Although, dimensionality of feature space induced by $\Phi(x)$ can be much larger than original input space.

Lemma 2.11:
For any given (C, g^2) and $\mathbf{x}_i \neq \mathbf{x}_j, \forall i \neq j$, the solution $(\boldsymbol{\alpha})$ of equation (2.12) is unique. Also, for every $g^2, \{\Phi(\mathbf{x}_i)|y_i = 1\}$ and $\{\Phi(\mathbf{x}_i)|y_i = -1\}$ are linearly separable. [23, 142]$\Diamond$

Mercer's theorem provides conditions under which a kernel function is valid, ensuring that the associated feature space is a Hilbert space. Therefore the mapping of $\Phi(x)$ ((2.21)) into a Hilbert Space makes theorem guarantees that the SVM optimization problem remains convex.

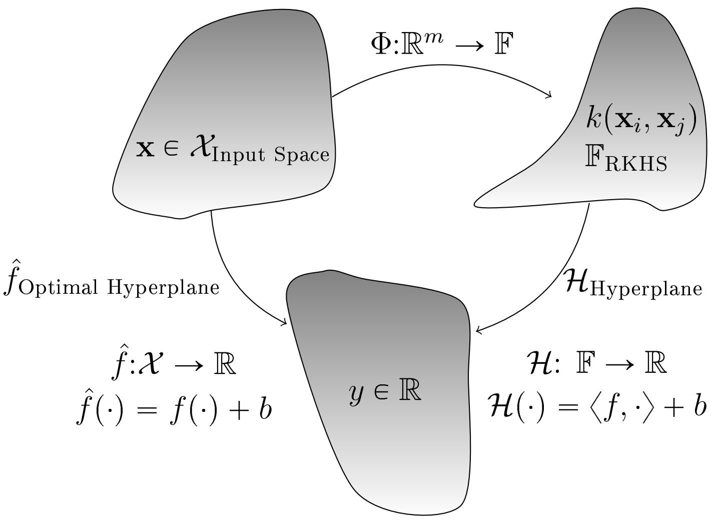

Φ:$\mathbb{R}^m \rightarrow \mathbb{F}$
$\mathbf{x} \in \mathcal{X}_{\text{Input Space}}$
$k(\mathbf{x}_i, \mathbf{x}_j)$
$\mathbb{F}_{\text{RKHS}}$
$\hat{f}_{\text{Optimal Hyperplane}}$
$\mathcal{H}_{\text{Hyperplane}}$
$\hat{f}$:$\mathcal{X} \rightarrow \mathbb{R}$
$\hat{f}(\cdot) = f(\cdot) + b$
$y \in \mathbb{R}$
$\mathcal{H}$: $\mathbb{F} \rightarrow \mathbb{R}$
$\mathcal{H}(\cdot) = \langle f, \cdot \rangle + b$

Figure 2.5: Kernel trick with RKHS embedding.

For a given input dataset ($\mathcal{X}_{\text{Input Space}}$), finding realtion between nonlinear mapping Φ on $\mathcal{X}_{\text{Input Space}}$ as depicted in Figure 2.5, is similar to computing kernel in $\mathbb{F}_{\text{RKHS}}$. This leads to optimal hyperplane for the given output y, that lies in Hilbert space for data in RKHS feature space.

2.3.3 Kernelised Support Vector Machine

As explained earlier, the Figure 2.5 depict the correlations mentioned in the sections §2.3.1, and §2.3.2. which has explained the kernel trick mentioned in §2.3.2. The use of kernel trick and computation of $\langle \Phi(x), \Phi(x^{'}) \rangle$ as $k(x, x^{'})$ leads to a new form of the dual nonlinear boundary objective function (2.18), that can be rewritten as follows,

$$\max_{\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_N} \sum_{i=1}^{N} \boldsymbol{\alpha}_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$
$$\text{subject to} \quad 0 \leqslant \boldsymbol{\alpha}_i \leqslant C \wedge \sum_{i=1}^{N} \boldsymbol{\alpha}_i y_i = 0, \tag{2.22}$$

and the decision function can also be derived directly from (2.20) as,

$$f(\mathbf{x}) = \sum_{i=1}^{N} \boldsymbol{\alpha}_i y_i k(\mathbf{x}_i, \mathbf{x}) + b, \tag{2.23}$$

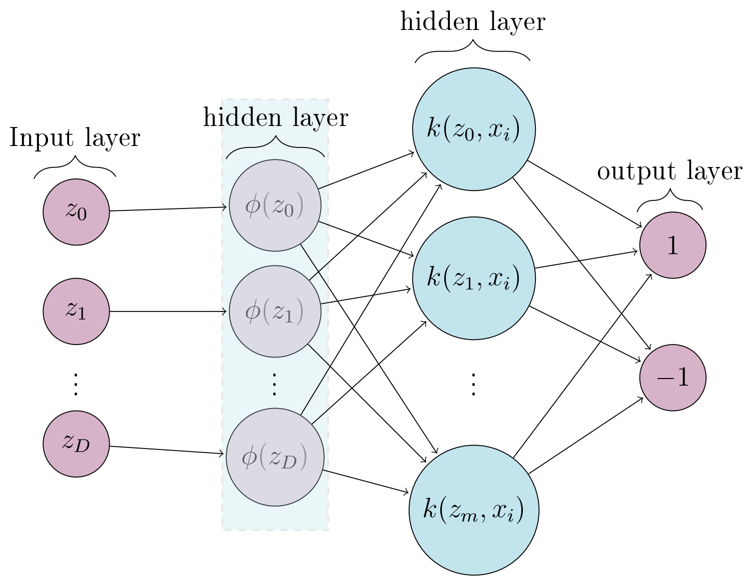


Figure 2.6: SVM as a Neural Network.

The SVM are considered to be earlier or smaller version of present DL models. Every neural network (NN) has a input-output structure with some hidden layers in between. These hidden layers are some nonlinear function. Mathematically, the NN is a composition of nonlinear functions that eventually minimize the hinge loss or in some other cases different loss functions can be defined. Hence, as depicted in Figure 2.6, the SVM model can be seen as a neural network. In the SVM model the hidden layer is equivalent to computation of the kernel matrix (Gram matrix), the $\boldsymbol{\alpha}$ parameter corresponds to weight parameter between hidden and input layers. Therefore, SVMs are also called Support Vector Network [47].

SVM with 2-dimensional Input

The general SVM problem mentioned in (2.3) deals with data expressed in a vector form, $\mathbf{x}$. If, on the other hand, the collected data is 2-dimensional (Matrix) $\mathbf{X}$, the matrix-SVM is known as Support Matrix Machine (SMM). The first typical idea would be to vectorize it by computing $\text{vec}(\mathbf{X})$, and then feed it to the standard SVM. However, in some scientific applications, the natural expression of data comes in the form of matrices, where the structural information could be beneficial for classification. Hence, the intutive soft margin SMM from (2.3) [115] can be written as follows,

$$\begin{aligned} &\min_{\mathbf{W},b,\boldsymbol{\xi}} \frac{1}{2}\langle \mathbf{W}, \mathbf{W} \rangle + C\sum_{i=1}^{N} \xi_i \\ &\text{subject to} \quad y_i(\langle \mathbf{W}, \mathbf{X}_i \rangle + b) \geqslant 1 - \xi_i,\ \boldsymbol{\xi} \geqslant 0, \quad i \in \langle N \rangle, \end{aligned} \tag{2.24}$$

with $\langle \mathbf{W}, \mathbf{W} \rangle = \text{tr}(\mathbf{W}^{\mathrm{T}}\mathbf{W})$ equation (2.24) becomes,

$$\begin{aligned} &\min_{\mathbf{W},b,\boldsymbol{\xi}} \frac{1}{2}\text{tr}(\mathbf{W}^{\mathrm{T}}\mathbf{W}) + C\sum_{i=1}^{N} \xi_i \\ &\text{subject to} \quad y_i(\text{tr}(\mathbf{W}^{\mathrm{T}}\mathbf{X}_i) + b) \geqslant 1 - \xi_i,\ \boldsymbol{\xi} \geqslant 0, \quad i \in \langle N \rangle \end{aligned} \tag{2.25}$$

If we observe closely, (2.25) is essentially equivalent to the standard SVM (2.3) when defining $\mathbf{w} = \text{vec}(\mathbf{W}^{\mathrm{T}})$, which fails to exploit the correlation between the matrix structured data, since

$$\text{tr}(\mathbf{W}^{\mathrm{T}}\mathbf{X}_i) = \text{vec}(\mathbf{W}^{\mathrm{T}})^{\mathrm{T}}\text{vec}(\mathbf{X}_i^{\mathrm{T}}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}_i, \tag{2.26}$$

$$\text{tr}(\mathbf{W}^{\mathrm{T}}\mathbf{W}) = \text{vec}(\mathbf{W}^{\mathrm{T}})^{\mathrm{T}}\text{vec}(\mathbf{W}^{\mathrm{T}}) = \mathbf{w}^{\mathrm{T}}\mathbf{w}, \tag{2.27}$$

The lost of information while exploitation of correlation between data and data or data and weights persist in multi-dimensional input data. In multidimensional input (tensors), the standard numerical linear algebra approaches immediately do not hold. Therefore, discovering an optimization regime of SVM and corresponding solution ideas are fascinating challenges. This is a topic for the next section.

2.4 Optimizing a Supervised Learning Model

The optimal solution of objective function (2.19) is usually build upon some known methods for tuning hyperparameters and choosing loss functions. This sections provides an overview of well-known method for optimizing hyperparameters in ML, along with some well-known loss function for classification model.

2.4.1 Cross Validation

Cross-validation (CV) [145] is a resampling method commonly used in statistical model learning. Resampling involves drawing data from a training set $\mathcal{X}$ and refitting a model to obtain additional information. However, this approach can be computationally expensive due to multiple iterations of data fitting. Nonetheless, research has made significant advancements in addressing issues such as the curse of dimensionality. CV is used to estimate the test mean squared error (MSE) or test error in general when data is limited. This allows us to achieve a certain level of expected performance by selecting an appropriate level of flexibility or degree of freedom. Model assessment involves evaluating a model's performance, while model selection involves selecting the proper level of flexibility for a model. In CV, we estimate the test error rate by holding out a subset of the training observations from the fitting process and applying the statistical learning method to these held-out observations. This process is repeated k times, with different subsets used as the validation set each time. By estimating the test error using CV, we can avoid overfitting and choose the appropriate level of flexibility for the model. Although CV can be computationally intensive, it is a powerful technique for model assessment and selection, applicable to various statistical learning methods such as regression, classification, and clustering [70, 80, 156].

Validation Set Approach: This is a simple method for estimating the test error rate in statistical model learning. It involves dividing the complete sample data, also known as input data, into two subsets: the training set and the validation or holding-out set. The model is trained on the separated training set, and the fitted model is used to predict the response on the validation set. The validation mean squared error (MSE) is then considered the test error rate. This approach is widely used in practice, but it has some limitations, such as the potential for high variability due to the particular way in which the data is split. Other methods, such as k-fold cross-validation, can be used to overcome these limitations and obtain a more reliable estimate of the test error rate.

The validation set approach has a limitation in that it does not account for previously unknown data when estimating the test error rate. This is because the same validation set is used to estimate the error during the training procedure. The reasons for this limitation are explained by the following points:

1. The validation estimate of the test error rate can be highly variable due to the sensitivity of the estimated error rate to the particular way in which the data is split between the training and validation sets. This can result in overfitting or underfitting of the model and an unreliable estimate of the test error rate.
2. The validation approach uses a subset of observations for training, which may cause

the validation set error rate to overestimate the test error rate for the entire data set due to statistical methods performing worse when trained on fewer observations.

k-fold Cross Validation

In the k-fold CV, the set of observations is divided randomly into k groups of approximately equal size. Each fold is used as a validation set once, and the remaining $k-1$ folds are used for training. The method's mean squared error (MSE_k) is then calculated on the held-out fold, and the process is repeated k times, with each fold used as a validation set. Hence, the resulted k error estimation is used to get average error as follows,

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\mathrm{MSE}_i, \tag{2.28}$$

where $\mathrm{CV}_{(k)}$ is $k-$fold CV estimate. The CV brings some advantages in ML models and here are some points outlining the advantages of k-fold cross-validation,

1. **Efficient use of data:** k-fold CV allows us to use most of the available data for training and testing without the need for a separate validation set. This can be particularly useful when data is limited or expensive to collect.
2. **Low bias:** Because k-fold CV averages the test error over k folds, it can provide a more accurate estimate of the true test error than a single validation set approach. This can be particularly useful when the dataset is small or when the model has high variance.
3. **Better model selection:** k-fold CV can help to choose the best model among several alternatives. By evaluating the performance of different models on the same data, it can help to identify the model that has the lowest test error.
4. **Robustness:** k-fold CV is less sensitive to the partitioning of the data than other methods such as leave-one-out CV (LOOCV). Because k-fold cross-validation averages over k different partitions, it is less likely to be affected by outliers or anomalies in the data.
5. **Flexibility:** k-fold CV can be easily adapted to different datasets and models. The value of k can be adjusted to provide more or less emphasis on training or testing, depending on the needs of the analysis.

Loss functions [70]: Based on how the loss functions are defined, such as squared loss, absolute loss, Huber loss, exponential loss, logistic loss, hinge loss, cross entropy loss, and multi-class hinge loss, each of these leads to the solution of different problems. Cross entropy loss is used when the data is categorical, while logistic loss is used when building a classification model out of a real output function, and so on. The main concern when choosing a loss function is that it is preferably smooth, robust, and sparse.

The classification model are based on mainly two types of loss function. These are as follows,

- **Hinge loss/Multi class SVM Loss [149, 147]:** The primary objective in classification is to ensure that the score assigned to the correct category surpasses the combined scores of all incorrect categories by a certain safety margin, typically one. This is why hinge loss is commonly employed in maximum-margin classification tasks, particularly in SVM. Despite being non-differentiable, hinge loss exhibits convexity, which facilitates its utilization with standard convex optimization methods prevalent in the field of machine learning.

$$\text{Hinge loss} = \big[1 - y_i f(x_i)\big]_+ \tag{2.29}$$

$$\text{Huberised square Hinge loss [149]} = \begin{cases} -4y_i f(x_i) & y_i f(x_i) < -1, \\ \big[1 - y_i f(x_i)\big]^2 & \text{otherwise.} \end{cases} \tag{2.30}$$

  Where y_i are the true labels and $f(x_i)$ predicted value from SVM objective function at datapoint x_i.

- **Cross entropy loss/Negative log likelihood :** Cross entropy loss meausres a probability value between 0 and 1. It promotes confident predictions for the correct class and penalizes deviations between predicted and true probabilities. Minimizing this loss through techniques like gradient descent improves classification accuracy. Regularization terms like l_1 or l_2 regularization are often combined with cross entropy loss to prevent overfitting and enhance model generalization.

$$\text{Cross entropy loss} = -(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)), \tag{2.31}$$

  where y_i are the true labels and $\hat{y}_i$ are the predicted labels. More detailed discussion over loss fucntions is mentioned in [70].

2.5 Chapter Summary

The chapter starts out with a short historical background of supervised learning while explaining *training-MSE* and *test-MSE* for regression problem (§2.1) as a reference model. The idea is broadened to classification (§2.1) in next two sections for linear (§2.2) and nonlinear (§2.3) cases, respectively. The §2.2.1 fancy mathematical construction of objective function for optimizing classification problem based on maximum-margin approach. The §2.2.2 includes Lagrangian frame to achieve dual form, that is easy to handle for nonlinear case as per the kernel trick mentioned in §2.3.2. From §2.1 to §2.3.2 introduce the building a dual SVM model while the last section §2.4 talk about how to deal with concern like bias-variance trade off by prefacing detailed description of Cross Validation approach, and some definitions relate to ML community.

CHAPTER 3 LOW-RANK TENSOR METHODS IN SUPERVISED LEARNING

3.1 Introduction 27
3.2 Tensor Algebra 28
3.3 Low-rank Tensor Factorization 32
3.3.1 Canonical Polyadic Decomposition 33
3.3.2 Tucker Decomposition 36
3.3.3 Tensor Train Decomposition 38
3.3.4 Existing Interrelationship of Tensor Decompositions 40
3.3.4.1 CP to TT 40
3.3.4.2 Qunatized Tensor Train (QTT) 41
3.3.4.3 QTT-Tucker 42
3.4 Tensor Classification 42
3.4.1 Support Tensor Machine 42
3.4.2 Tensor-product Reproducing Kernel Hilbert Space 44
3.4.3 Kernelised Support Tensor Machine 45
3.4.4 Reproducing Tensor Kernels 47
3.5 Low-rank Tensor Decomposition in Kernel Models 48
3.6 Chapter summary 49

Contents

3.1 Introduction

Tensor representations are often very useful in mitigating the small sample size problem in discriminating subspace selection, because the information about the structure of objects is inherent in tensors and is a natural constraint which helps reduce the number of unknown parameters in the description of a learning model. In other words, when the number of training measurements is limited, tensor-based learning machines are expected to perform better than the corresponding vector-based learning machines, as vector representations are associated with several concerns, such as loss of information for structured data and over-fitting for high-dimensional data.

3.2 Tensor Algebra

This section introduces terminology and definitions used throughout the thesis. Multi-dimensional tensor structure has much more power and is richer than linear algebra. A tensor is a multidimensional array [95] which is a higher order generalization of vectors and matrices. The M^{th}-order tensor ($M \geqslant 3$) is denoted by a calligraphic letter $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$, its entries by $x_{i_1 i_2 \ldots i_M}$, a matrix by a boldface upper case letter $\mathbf{X} \in \mathbb{R}^{I \times J}$, and a vector by a boldface lower case letter $\mathbf{x} \in \mathbb{R}^{I}$. Matrix and vector elements are denoted by $x_{ij} = \mathbf{X}(i, j)$ and $x_i = \mathbf{x}(i)$, respectively. The order of a tensor is the number of its *dimensions*, *ways* or *modes*. The *size* of a tensor stands for the maximum index value in each mode. For example, $\mathcal{X}$ is of order M and the size in each mode is I_m, where $m \in \langle M \rangle$. A *fiber* or *tube* of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ is a vector attained by fixing all but one index, *e.g.* $\mathbf{x}(:, i_2, i_3, \ldots, i_M)$ is a fiber when $i_m,\ m = 2, 3, \ldots, M$ are fixed. A *slice* of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ is a matrix attained by fixing all but two indices, *e.g.* $\mathbf{X}(:, :, i_3, \ldots, i_M)$ is a slice when $i_m,\ m = 3, \ldots, M$ are fixed, using [95, 105].

For simplicity, throughout this thesis all tensors are assumed to be real valued.

Definition 3.1:
An m-mode matricization $\mathcal{X}_{(m)} \in \mathbb{R}^{I_m \times I_1 \ldots I_{m-1} I_{m+1} \ldots I_M}$ for $m \in \langle M \rangle$ is the unfolding (or flattening) of an M^{th}-order tensor into a matrix in the appropriate order of elements, *i.e.* a tensor element $(i_1, i_2, \ldots i_M)$ maps to an element (i_m, j) of a matrix as follows [95]:

$$j = 1 + \sum_{k=1, k \neq m}^{M} (i_k - 1) J_k \quad \text{with} \quad J_k = \prod_{\ell=1, \ell \neq m}^{k-1} I_\ell .$$

This is known as the *classical* [212] or *Kolda* [95] unfolding. This multi-indices is little-endian convention (reverse lexicographic ordering) mentioned in [53]. $\diamondsuit$

Definition 3.2:
An m-mode product $\mathcal{X} \times_m \mathbf{A} \in \mathbb{R}^{I_1 \times \ldots \times I_{m-1} \times J \times I_{m+1}, \times \ldots \times I_M}$, given $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ and $\mathbf{A} \in \mathbb{R}^{J \times I_m}$, is defined as a tensor-matrix product in m^{th} way:

$$\mathcal{Y}_{(m)} = (\mathcal{X} \times_m \mathbf{A})_{(m)} = \mathbf{A} \mathcal{X}_{(m)} .$$

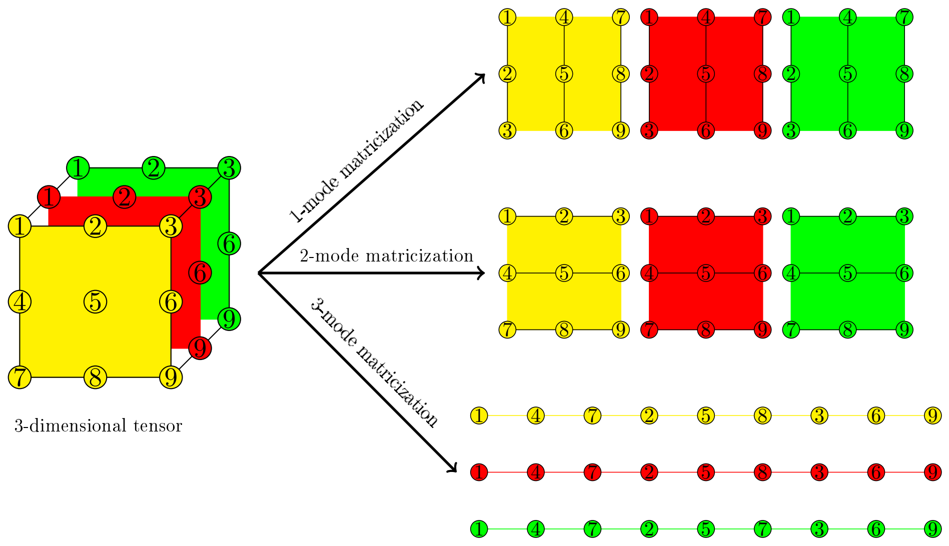


Figure 3.1: Unfolding of a 3-dimensional tensor.

This is also known as tensor-times-matrix (TTM) product. Some efficient algorithms for TTM are mentioned in [107, 13, 14]. Similarly, tensor-times-vector (TTV) product for a vector $\mathbf{a} \in \mathbb{R}^{I}_{m}$ is defined. TTV results in a tensor $\boldsymbol{\mathcal{Z}} = \boldsymbol{\mathcal{X}}_{(m)} \bar{\times}_m \mathbf{a} \in \mathbb{R}^{I_1 \times \cdots \times I_{m-1} \times I_{m+1} \times \cdots \times I_M}$ with the entries as,

$$z_{i_1,\ldots,i_{m-1},i_{m+1},\ldots,i_M} = \sum_{i_m=1}^{I_m} x_{i_1,\ldots,i_{m-1},i_m,i_{m+1},\ldots,i_M}\, a_{i_m}.$$

Thw product of a tensor with a matrix doesn't change the tensor order while product with vector reduces the order by one.

Definition 3.3:
A mode-$\{m\}$ canonical matricization is also the mode-$(1, 2, \ldots, m)$ matricization for a fixed index $m \in \langle M \rangle$, of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$. The element wise entry of the matrix is defined as follows, [60]

$$\boldsymbol{\mathcal{X}}_{\langle m \rangle} \in \mathbb{R}^{I_1 I_2 \cdots I_m \times I_{m+1} \cdots I_M}, \quad (\boldsymbol{\mathcal{X}}_{\langle m \rangle})_{i,j} = x_{i_1,i_2,\ldots,i_N}, \quad i \in I_1 I_2 \cdots I_m, j \in I_{m+1} \cdots I_M.$$

The matricization operator in the MATLAB notation (reverse lexicographic) is given by

$$\boldsymbol{\mathcal{X}}_{\langle m \rangle} = \text{reshape}\left(\boldsymbol{\mathcal{X}}, \left[\prod_{p=1}^{m} I_p, \prod_{p=m+1}^{M} I_p\right]\right).$$

Some properties holds between mode-$\{m\}$ and other matricization/vectorization, such as,

$$\mathcal{X}_{\langle 1\rangle} = \mathcal{X}_{(1)}, \quad \mathcal{X}_{\langle M-1\rangle} = \mathcal{X}^{\mathrm{T}}_{(M)}, \quad \mathcal{X}_{\langle M\rangle} = \mathrm{vec}(\mathcal{X}). \qquad \Diamond$$

$$\begin{aligned} \text{Tensorization} &:= (\text{matrix/vector} \rightarrow \text{tensor}) \\ \text{Matricization/Vectorization} &:= (\text{tensor} \rightarrow \text{matrix/vector}) \end{aligned}$$

Definition 3.4:
The inner product of given tensors $\mathcal{X}$, $\mathcal{Y} \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_M}$ is defined as

$$\langle \mathcal{X}, \mathcal{Y}\rangle = \sum_{i_1}^{I_1}\sum_{i_2}^{I_2}\cdots\sum_{i_m}^{I_M} x_{i_1 i_2\ldots i_m} y_{i_1 i_2\ldots i_m}.$$

Definition 3.5:
The Kronecker Product of matrices $\mathbf{A} \in \mathbb{R}^{I\times J}, \mathbf{B} \in \mathbb{R}^{K\times L}$ is defined as usual by

$$\mathbf{A}\otimes\mathbf{B} = \begin{bmatrix} a_{1,1}\mathbf{B} & \cdots & a_{1,J}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{I,1}\mathbf{B} & \cdots & a_{I,J}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{IK\times JL}.$$

Similarly, the Kronecker product of two tensors $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_M}, \mathcal{Y} \in \mathbb{R}^{J_1\times J_2\times\ldots\times J_M}$ returns a tensor $\mathcal{Z} = \mathcal{X}\otimes\mathcal{Y} \in \mathbb{R}^{I_1J_1\times I_2J_2\times\ldots\times I_MJ_M}$. $\Diamond$

Definition 3.6:
the Khatri-Rao product for matrices is a column-wise Kronecker product,

$$\mathbf{A}\odot\mathbf{B} = [\mathbf{a}_1\otimes\mathbf{b}_1, \mathbf{a}_2\otimes\mathbf{b}_2, \cdots, \mathbf{a}_R\otimes\mathbf{b}_R] \in \mathbb{R}^{IK\times R}.$$

Moreover, the mode-m Khatri–Rao product of two Mth-order tensors, $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_M}$, and $\mathcal{Y} \in \mathbb{R}^{J_1\times J_2\times\ldots\times J_M}$ returns a tensor $\mathcal{Z} = \mathcal{X}\odot_m\mathcal{Y} \in \mathbb{R}^{I_1J_1\times\cdots\times I_{m-1}J_{m-1}\times I_m\times I_{m+1}J_{m+1}\times\cdots\times I_MJ_M}$, with $I_m = J_m$, and in terms of Kroneker product,

$$\mathcal{Z}(:,\ldots:, i_m, :,\ldots,:) = \mathcal{X}(:,\ldots:, i_m, :,\ldots,:)\otimes\mathcal{Y}(:,\ldots:, i_m, :,\ldots,:)$$

. $\Diamond$

Definition 3.7:
The outer product is the primary operate to multiply two different tensors. It is known as *outer* or *tensor* product. For two given tensors $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_M}$ and $\mathcal{Y} \in \mathbb{R}^{J_1\times J_2\times\ldots\times J_N}$ in outer product results in a tensor $\mathcal{Z} = \mathcal{X}\circ\mathcal{Y} \in \mathbb{R}^{I_1\times\cdots\times I_M\times J_1\times\cdots\times J_N}$ where each element is computed as,

$$z_{i_1,\ldots,i_M,j_1,\ldots,j_N} = x_{i_1,\ldots,i_M}\, y_{j_1,\ldots,j_N}.$$

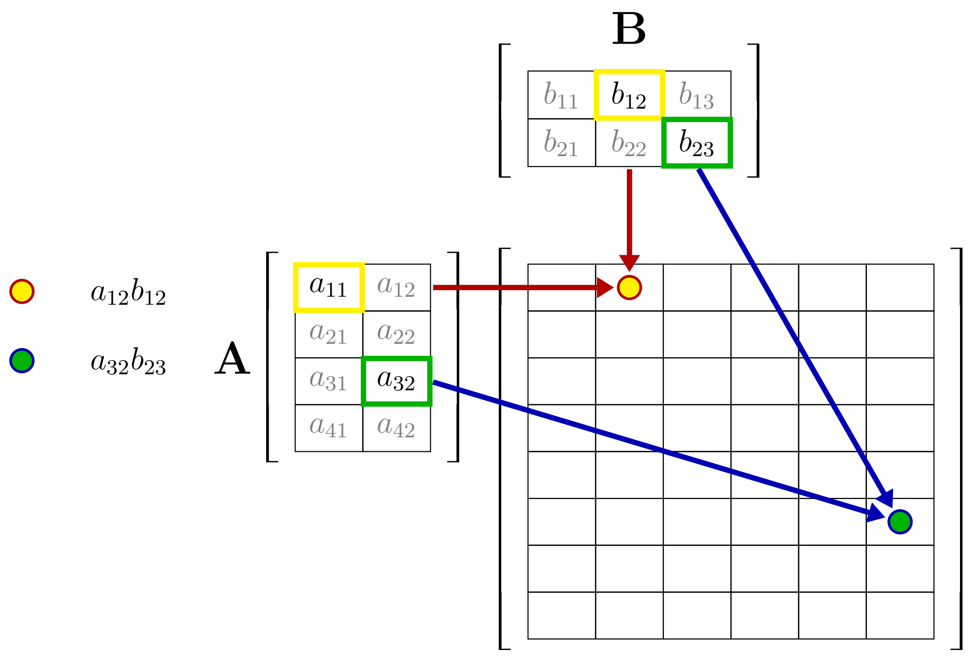


Figure 3.2: Kronecker Product of two matrices **A** and **B**.

When three non-zero vectors $\mathbf{a} \in \mathbb{R}^I$, $\mathbf{b} \in \mathbb{R}^J$ and $\mathbf{c} \in \mathbb{R}^K$ are multiplied with using outer product then it results in a 3rd-order tensor $\mathcal{X} = \mathbf{a} \circ \mathbf{b} \circ \mathbf{c} \in \mathbb{R}^{I \times J \times K}$ with each entry computed as $x_{ijk} = a_i\, b_j\, c_k$.

Definition 3.8:
A Rank-1 tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ is which can be expressed exactly as the outer product, $\mathcal{X} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \cdots \circ \mathbf{a}^{(M)}$ of non-zero factor vectors, $\mathbf{a}^{(m)} \in \mathbb{R}^{I_m}$, and the tensor entries of rank-1 is given as,

$$x_{i_1,i_2,\ldots,i_N} = a^{(1)}_{i_1} a^{(2)}_{i_2} \cdots a^{(N)}_{i_N}.$$

Definition 3.9:
Hadamard Product (element-wise product/ entry-wise product/ Schur product is defined for two same dimensional tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ as follows,

$$(\mathcal{X} \circledast \mathcal{Y})_{i_1,i_2,\cdots,i_M} = x_{i_1,i_2,\cdots,i_M} y_{i_1,i_2,\cdots,i_M}$$

Definition 3.10:
A mode-(m,n) contracted product $\mathcal{Z} = \mathcal{X} \times^n_m \mathcal{Y} \in \mathbb{R}^{I_1 \times \ldots \times I_{m-1} \times I_{m+1} \times \ldots \times I_M \times J_1 \times \ldots \times J_{n-1} \times J_{n+1} \times \ldots \times J_N}$, for given tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ and $\mathcal{Y} \in \mathbb{R}^{J_1 \times J_2 \times \ldots \times J_N}$, with $I_m = J_n$, yields a tensor $\mathcal{Z}$ of order $(M + N - 2)$ with entries

$$z_{i_1,\ldots,i_{m-1},i_{m+1},\ldots,i_M,j_1,\ldots,j_{n-1},j_{n+1},\ldots,j_N} = \sum_{i_m=1}^{I_M} x_{i_1,\ldots,i_{m-1},i_m,i_{m+1},\ldots,i_M} y_{j_1,\ldots,j_{n-1},i_m,j_{n+1},\ldots,j_N}.$$

This is a fundamental and the most important operation which can be considered a higher-dimensional analogue of matrix multiplication, inner product, and outer product. Also, known as tensor contraction. In such manner tensors can be contracted in several dimensions or sometimes in all.

Definition 3.11:
A mode-(M,1) contracted product $\mathcal{Z} = \mathcal{X} \times_M^1 \mathcal{Y} = \mathcal{X} \times^1 \mathcal{Y} \in \mathbb{R}^{I_1 \times \ldots \times I_{M-1} \times J_2 \times \ldots \times J_M}$, for given tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ and $\mathcal{Y} \in \mathbb{R}^{J_1 \times J_2 \times \ldots \times J_M}$, with $I_M = J_1$, yields a tensor $\mathcal{Z}$ that can be written as,

$$\mathcal{Z} = \mathcal{X} \ \times_M^1 \ \mathcal{Y} = \mathcal{X} \times^1 \mathcal{Y} = \mathcal{X} \bullet \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{M-1} \times J_2 \times \cdots \times J_N},$$

with entries

$$z_{i_1,\ldots,i_{M-1},j_2,\ldots,j_M} = \sum_{i_M=1}^{I_M} x_{i_1,\ldots,i_M} y_{i_M,j_2,\ldots,j_M}.$$

In comparison to the matrix-matrix product, the available efficient algorithms for tensor contraction are limited. Therefore, dealing with related high computational cost is done by executing above-mentioned tensor operations approximately.

Curse of dimensionality: The number of entries in tensor grows exponentially with dimension M. *e.g.*for a given tensor of order 40 with size of each dimension being $I_1 = I_2 = \ldots = I_M = 2$ will need 2^{20} (terabyte) storage! Hence, storing tensor explicitly is not affordable. Also, increment of dimensionality can lead to overfitting, poor generalization, and unreliable predictions. This way, high-dimensional data can pose challenges in terms of analysis and modelling of data while having extremely large number of degrees of freedom.
Blessing of dimensionality: On the other hand, blessing of dimensionality refers to the advantages that arise when dealing with high-dimensional data. In particular, high-dimensional spaces can offer more flexibility and more room for variation, which can be useful in capturing complex patterns and relationships in the data. For example, in natural language processing, high-dimensional embedding of words can capture subtle nuances in meaning and context, which can lead to more accurate language models. Similarly, in computer vision, high-dimensional feature representations can capture fine-grained details in images, which can improve object recognition and classification accuracy. It can also offer opportunities for capturing complex patterns and relationships that may not be apparent in lower-dimensional spaces.

3.3 Low-rank Tensor Factorization

Tensor decomposition methods have been significantly enhanced during the last two decades [40, 42], and applied to solve problems of varying computational complexity.

The main goal is the linear (or at most polynomial) scaling of the computational complexity in the dimension of a tensor. The key ingredient is the separation of variables via approximate low-rank factorizations. The past research in machine learning have proposed methodologies to extracted these low-rank factorization that include some prior statistical data information such as uniqueness, well-posedness, sparsity, non-negativity. These low-rank factorizations help to optimize hyperparameters and provide desired features in the developed model. These desired features eventually help in building the optimized and more efficient machine learning model. The basic approach is to these low-rank tensor factorization (LRTF) is similar to low-rank matrix factorization (LRMF), in which a matrix is approximated by factor matrices and an error term as follows,

$$\mathbf{X} = \mathbf{\Lambda} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} + \mathbf{E} = \sum_{r=1}^{R} \lambda_r \, \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} + \mathbf{E} = \sum_{r=1}^{R} \lambda_r \, \mathbf{a}_r^{(1)} \, \mathbf{a}_r^{(2)^{\mathrm{T}}} + \mathbf{E}, \qquad (3.1)$$

the factor matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ fulfil the predetermined constraints with a diagonal scaling matrix $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \dots, \lambda_R)$. This formulation gives an intuitive multi-way approach for a multi-way data array (tensor), this is explained in §3.3.1. Additionally, some other well established LRMF are the Singular Value Decomposition (SVD), Principal Component Analysis (PCA), Independent Component Analysis (ICA), Non-negative Matrix Factorization (NMF), etc. The SVD expression (3.2) gives as an intuition to Higher Order SVD §3.3.2 or Tucker decomposition §3.3.2 that we discuss later.

$$\mathbf{X} = \mathbf{S} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} = \sum_{r=1}^{R} \sigma_r \, \mathbf{u}_r^{(1)} \circ \mathbf{u}_r^{(2)} = \sum_{r=1}^{R} \sigma_r \, \mathbf{u}_r^{(1)} \mathbf{u}_r^{(2)^{\mathrm{T}}}, \qquad (3.2)$$

where $\mathbf{U}^{(1)} \in \mathbb{R}^{I_1 \times R}$ and $\mathbf{U}^{(2)} \in \mathbb{R}^{I_2 \times R}$ are column-wise orthogonal matrices and $\mathbf{S} \in \mathbb{R}^{R \times R}$ is a diagonal matrix with $\sigma_r > 0$ in a monotonically non-increasing order. Some widely used tensor factorizations are mentioned more.

3.3.1 Canonical Polyadic Decomposition

Introducing a tensor in polyadic form was firstly done in 1927 [75, 76], where a tensor was presented as a finite sum of rank-1 tensors. In 1970, in the psychometric community, the same decomposition was reintroduced under the names of CANDECOMP (CANonical DECOMPosition) [28] as well as PARAFAC (PARallel FACtors) [68]. Simultaneously, Möcks [119] discovered the same decomposition as *topographic component model*. This model was built to deal with brain images. Eventually it found its standard and now most commonly used name, the CP (Canonical Polyadic) decomposition after Kiers [91].

The CP decomposition of an $M^{th}-$order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ is a factorization into a sum of rank-1 components, which is given element-wise as

$$\begin{aligned} x_{i_1 i_2 \ldots i_M} &\cong \sum_{r=1}^{R} \mathbf{a}^{(1)}_{i_1,r} \mathbf{a}^{(2)}_{i_2,r} \cdots \mathbf{a}^{(M)}_{i_M,r}, \\ &= \sum_{r=1}^{R} \mathbf{a}^{(1)}_{r} \circ \mathbf{a}^{(2)}_{r} \circ \cdots \circ \mathbf{a}^{(M)}_{r} \\ \text{or shortly,} \qquad \mathcal{X} &\cong [\![\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \cdots, \boldsymbol{A}^{(M)}]\!], \end{aligned} \tag{3.3}$$

where $\mathbf{A}^{(m)} = \left[\mathbf{a}^{(m)}_{i_m,r}\right] \in \mathbb{R}^{I_m \times R}$, $m = 1, \ldots, M$, are called *factor matrices* of the CP

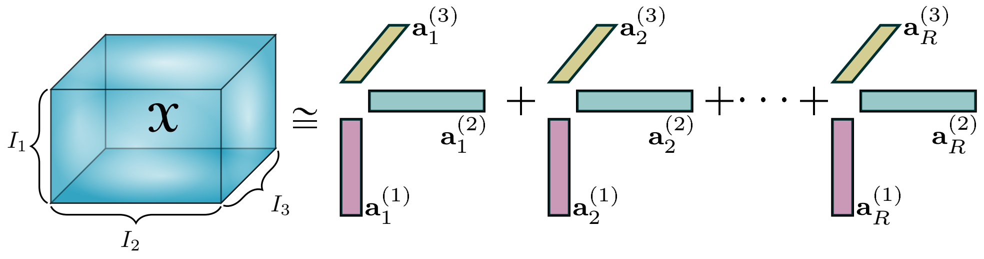


Figure 3.3: canonical polyadic decomposition of a 3D tensor.

decomposition, see Figure 3.3, and R (smallest) is called the CP-rank for which the CP decomposition holds exactly. The notation with $[\![\cdot]\!]$ is also called the Kruskal [100] representation of the CP factorization. The $m-$mode matricization version of CP decomposition can be written as following,

$$\mathbf{X}_{(m)} \approx \mathbf{A}^{(m)} \left(\mathbf{A}^{(M)} \odot \cdots \odot \mathbf{A}^{(m+1)} \odot \mathbf{A}^{(m-1)} \odot \cdots \odot \mathbf{A}^{(1)} \right) \tag{3.4}$$

Despite the simplicity of the CP format, the problem of the best CP approximation is often ill-posed [50]. This implies that the best rank-R CP approximation of a given data tensor may not exist. However, a rank-R tensor can be approximated arbitrarily well by a sequence of tensors for which the CP ranks are strictly less than R. A practical CP approximation can be computed via the Alternating Least Squares (ALS) method [125], but the convergence may be slow. It may also be difficult to choose the rank R.

Algorithm 3.1: CP decomposition of a tensor using Basic ALS [68, 95].

```
1: Input: $M$-dimensional tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$, rank $R$, iter.
2: Ensure: Cores $\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \cdots, \boldsymbol{A}^{(M)}$ of the CP decomposition $\mathcal{X}$ in the CP-format with
   scaling vector $\lambda \in \mathbb{R}^R$
3: Initialize $\mathbf{A}^{(m)}$.
4: while not converged or iteration $\leqslant$ iter do
5:    for $m = 1, 2, \ldots, M$ do
6:       $\mathbf{A}^{(m)} \leftarrow \mathcal{X}_{(m)} \left(\bigodot_{\tilde{m} \neq m} \mathbf{A}^{(\tilde{m})}\right) \left(\circledast_{\tilde{m} \neq m} (\mathbf{A}^{(\tilde{m})^T} \mathbf{A}^{(\tilde{m})})\right)^{\dagger}$
7:       Unit length normalize columns of $\mathbf{A}^{(m)}$: $\left(\mathbf{A}^{(m)}(i, j) \leftarrow \frac{\|\mathbf{A}^{(m)}(i,j)\|}{\|\mathbf{A}^{(m)}(:,j)\|}\right)$
8:       Store the norm in $\lambda$
9:    end for
10: end while
11: return $\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \cdots, \boldsymbol{A}^{(M)}$ and $\lambda$
```

Advantage of CP-ALS: The Alternating Least Squares (ALS) method for computation of CP approximation is mentioned in Algorithm 3.1. The ALS approach computate optimized CP factors by finxing all but one factor at a time. It is known for its elegant simplicity, and has proven to be effective for well-defined problems. To tackle the challenge of processing tensors on a large scale, parallel ALS algorithms implemented over distributed memory have been proposed in previous studies [36, 85].

Uniqueness of CP: The CP factorization is a powerful technique because of its uniqueness under mild conditions without having any additional constraints on the factor matrices [100, 164]. Naturally, if the constituents in one or more modes are understood to exhibit certain characteristics, for example, positivity, orthogonality, statistical independence, or sparseness, this pre-existing knowledge can be assimilated into the CPD algorithms to simultaneously ease uniqueness conditions and heighten precision and consistency. Moreover, such limitations can promote improved physical comprehensibility of the extracted components [162, 92, 171, 93, 213, 108].

Applications: The CP decomposition has already been established as an advanced tool for blind signal separation in vastly diverse branches of signal processing and machine learning [5, 95, 121, 9, 193, 181, 165]. It is also routinely used in exploratory data analysis, where the rank-1 terms capture essential properties of dynamically complex datasets, while in wireless communication systems, signals transmitted by different users correspond to rank-1 terms in the case of line-of-sight propagation and therefore admit analysis in the CP format. Another potential application is in harmonic retrieval and direction of arrival problems, where real or complex exponential have rank-1 structures, for which the use of CP decomposition is quite natural [163, 161, 173]

.

3.3.2 Tucker Decomposition

The Tucker decomposition [182] is a more general decomposition compared to CP, along with having Tucker matrix factors, it also has core tensor with smaller dimensions compared to the original tensor. The difference in CP is that, the cube core tensor has non-zero elements only on the main diagonal. For a given tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ the Tucker decomposition or Tucker-M (*not unique*) model is expressed as follows [105],

$$\begin{aligned}\mathcal{X} &\cong \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_M=1}^{R_M} g_{r_1 r_2 \ldots r_M} \left(\mathbf{u}_{r_1}^{(1)} \circ \mathbf{u}_{r_2}^{(2)} \circ \ldots \circ \mathbf{u}_{r_M}^{(M)} \right) \\ &= [\![\mathcal{G}; \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(M)}]\!], \end{aligned} \tag{3.5}$$

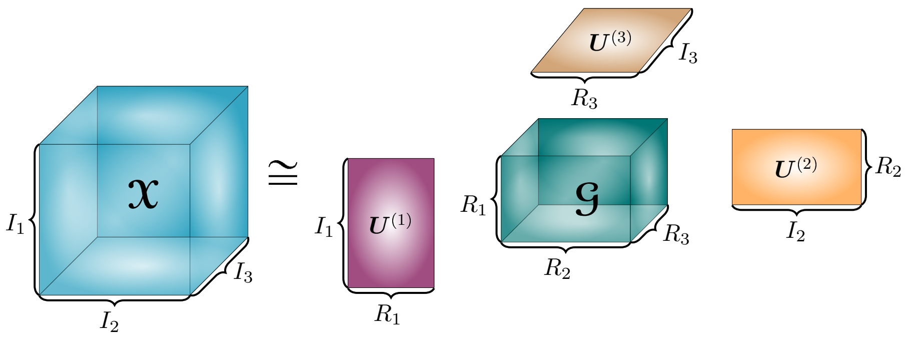


Figure 3.4: Tucker decomposition of a 3-way tensor.

where $\mathbf{U}^{(m)} = \left[\mathbf{u}_{i_m,r}^{(m)} \right] \in \mathbb{R}^{I_m \times R_m}$, $m = 1, \ldots, M$, are called mode-m *factor matrices* of the Tucker decomposition (see Figure 3.4) and typically $R_m << I_m, \quad m = 1, 2, \ldots, M$, are called the Tucker ranks. The core tensor is denoted by $\mathcal{G} \in \mathbb{R}^{R_1 \times R_2 \times \ldots \times R_M}$. If $\mathbf{U}^{(m)}$ are full rank, the Tucker decomposition is called *independent Tucker format*, while with orthogonal factor matrices $\mathbf{U}^{(m)}$ it is called, *orthonormal format*. The standard format of the Tucker-M model is orthogonal. Tucker decompositions have a long history, recent surveys and more detailed information [95, 60, 44, 41, 165]. The next tensor decomposition §3.3.3 is equivalent to the Tucker-2 model of a 3^{rd} order tensor. The Tucker-2 model has the third core as the identity.

Higher Order SVD: Special form of Tucker

The Higher Order SVD known as HOSVD is also called Multi Linear Singular Value Decomposition or MLSVD [102, 49]. The HOSVD is considered to be special case of Tucker model, where HOSVD algorithm is one way of computing *orthonormal format* of Tucker-M model. Along with having orthogonal factor matrices $\mathbf{A}^{(m)}$, the core tensor is *all-orthogonal*.

Computational Cost and Various approaches: The HOSVD or Tucker decomposition applies the matrix-SVD on tensor tensor unfolding $\mathcal{X}_{(m)} \in \mathbb{R}^{I_m \times I_1 \ldots I_{m-1} I_{m+1} \ldots I_M}$. For the large-scale tensor input standard computer memory easily exceeds. There are two standard solutions to this problem, first is using a divide-and-conquer approach on a partition of $\mathcal{X}_{(m)}$ and eventually on orthogonal matrices $\mathbf{U}^{(m)}$, and the second approach is useful when the matrix has low-rank. The randomized SVD algorithm deals with it efficiently. Some literature in this direction are [34, 65, 45, 137].

The Algorithm 3.2 has been proposed to deal with computational cost of computing HOSVD and known as *Sequential Truncated HOSVD* [183].

Applications and Advantages: the Tucker decomposition is a versatile technique that finds applications in various signal processing tasks, including blind source separation, feature extraction, classification, and subspace-based harmonic retrieval. It can also be used for signal compression and enhancement. By identifying and classifying relevant data, it simplifies feature extraction and helps reveal desired information. Furthermore, it is useful in anomaly detection [54], which involves identifying patterns, signals, outliers, or features that do not conform to expected behaviours. The projected data onto a lower-dimensional subspace using Tucker decomposition helps to easily identify anomalies. This way it provides a natural framework with an assumption that normal and abnormal patterns will appear significantly different in the projected subspace.

Algorithm 3.2: Sequential Truncated HOSVD [183]

```
Input: $M$-dimensional tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ with truncation accuracy $\epsilon$
Output: HOSVD of $\mathcal{X}$ in the Tucker-format $\mathcal{X} = [\![\mathcal{G}; \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(M)}]\!]$,
Initialize $\mathcal{Y} \leftarrow \mathcal{X}$
for $m = 1$ to $M$ do
    Compute $m-$mode matricization $\mathcal{Y}_{(m)}$ of tensor $\mathcal{Y}$
    Compute truncated SVD: $\mathcal{Y}_{(m)} = \mathbf{U}^{(m)}\mathbf{SV} + \mathbf{E}_m, \quad \|\mathbf{E}_m\|_F \leqslant \frac{\varepsilon}{\sqrt{M}}$
    Core computation: $\mathcal{G} \leftarrow \mathbf{VS}$
end for
$\mathcal{G} \leftarrow \text{reshape}(\mathcal{G}, [R_1, \ldots, R_M])$
return  Core tensor $\mathcal{G}$ and orthogonal factor matrices $\mathbf{U}^{(m)} \in \mathbb{R}^{I_m \times R_m}$.
```

3.3.3 Tensor Train Decomposition

To alleviate the difficulties of the CP decomposition mentioned above, the new form of tensor decomposition was introduced as the Tensor Train (TT) [133, 131] decomposition. The TT approximation of an $M^{th}-$order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ is defined element-wise as

$$x_{i_1 i_2 \ldots i_M} \cong \sum_{r_0, \ldots, r_M} \mathcal{C}^{(1)}_{r_0, i_1, r_1} \mathcal{C}^{(2)}_{r_1, i_2, r_2} \cdots \mathcal{C}^{(M)}_{r_{M-1}, i_M, r_M},$$
$$\mathcal{X} \cong \langle\!\langle \mathcal{C}^{(1)}, \mathcal{C}^{(2)}, \ldots, \mathcal{C}^{(M)} \rangle\!\rangle, \tag{3.6}$$

where $\mathcal{C}^{(m)} \in \mathbb{R}^{R_{m-1} \times I_m \times R_m}$, $m = 1, \ldots, M$, are 3rd-order tensors called *TT-cores* (see Figure 3.5), and $R_0, \ldots, R_M$ with $R_0 = R_M = 1$ are called *TT-ranks*. The TT

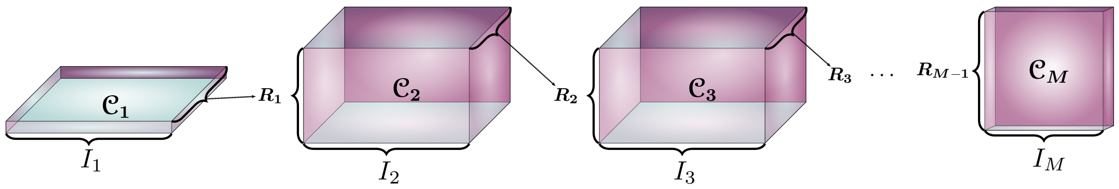


Figure 3.5: Tensor train decomposition of an $M^{th}-$order tensor.

decomposition was introduced as the Matrix Product State (MPS) [136, 6, 157] representation. It was subsequently extended by many researchers. In fact, the TT was rediscovered several times under different names: MPS, valence bond states, and density matrix renormalization group (DMRG) [195]. Also, in quantum physics the ALS algorithm is called the one-site DMRG, while the Modified ALS (MALS) is known as the two-site DMRG [195, 189, 188, 157, 77, 129].

Advantage: An important advantage of the TT/MPS format its simpler practical implementation. The alluring capability of the TT format is its ability to perform algebraic operations directly on TT-cores avoiding full tensors. These operations only requires TT-cores to be stored and processed, which makes the number of parameters to scale linearly in the tensor order, M, of a data tensor and all mathematical operations are then performed only on the low-order and relatively small size core tensors. Moreover, a stable quasi-optimal rank reduction for TT approximation of any given tensor using the tSVD (Algorithm 3.3) can be computed. This builds on the fact that the TT decomposition constitutes a recursive matrix factorization, where each TT-rank is the matrix rank of the appropriate unfolding of the tensor, and hence the TT approximation problem is well-posed [133].

Drawback: A major drawback of TT format is while applying basic mathematical operations on TT-cores, the produced tensor that is also in TT format, generally exhibit increasing TT-ranks. Also, the rank of TT format is not invariant to the permutation of the modes or sizes.

Algorithm 3.3: TT-SVD Decomposition using truncated SVD (tSVD) [133]

```
1: Input: $M$-dimensional tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$, relative error threshold $\epsilon$.
2: Ensure: Cores $\mathcal{C}^{(\boldsymbol{1})}, \mathcal{C}^{(\boldsymbol{2})}, \cdots, \mathcal{C}^{(\boldsymbol{M})}$ of the TT-approximation $\mathcal{X}'$ to $\mathcal{X}$ in the TT-format
   with TT-rounding ranks $r_m$ equal to the $\delta$-ranks of the unfolding $\mathcal{X}_{(m)}$ of $\mathcal{X}$, where
   $\delta = \sqrt{\frac{\epsilon}{M-1}} \|\mathcal{X}\|_F$.
3: Initialize $\hat{\mathbf{Z}}_1 = \mathcal{X}_{(1)}, R_0 = 1$.
4: for  $m = 1$ to $M - 1$ do
5:     $\mathbf{Z}_m :=$ reshape$\left(\hat{\mathbf{Z}}_m, [R_{m-1} I_m,\ I_{m+1} \cdots I_M]\right)$
6:     Compute $\delta$-truncated SVD: $\mathbf{Z}_m = \mathbf{U}_m \mathbf{S}_m \mathbf{V}_m^T + \mathbf{E}_m,\ \|\mathbf{E}_m\|_F \leqslant \delta$, where
       $\mathbf{U}_m = [u_1^{(m)}, u_2^{(m)}, \ldots, u_{R_m}^{(m)}],\ \mathbf{S}_m = \mathrm{diag}(\sigma_1^{(m)}, \sigma_2^{(m)}, \ldots, \sigma_{R_m}^{(m)}),\ \mathbf{V}_m =$
       $[v_1^{(m)}, v_2^{(m)}, \ldots, v_{R_m}^{(m)}]$
7:     $\mathcal{C}^{(m)}_{r_{m-1}, i_m, r_m} = u^{(m)}_{r_{m-1} + (I_m - 1) R_{m-1},\ r_m}$,
8:     $\mathcal{C}^{(m)} :=$ reshape$(\mathbf{U}_m, [R_{m-1}, I_m, R_m])$
9:     $\hat{\mathbf{Z}}_{m+1} := \mathbf{S}_m \mathbf{V}_m^T$
10: end for
11: $\mathcal{C}^{(M)} = \hat{\mathbf{Z}}_M$
12: return $\mathcal{X}' = \langle\langle \mathcal{C}^{(\boldsymbol{1})}, \mathcal{C}^{(\boldsymbol{2})}, \cdots, \mathcal{C}^{(\boldsymbol{M})} \rangle\rangle$
```

Algorithm 3.4: TT Rounding (Recompression or Truncation) [133].

```
1: Input: $M$th-order tensor $\mathcal{X}' = \langle\langle \mathcal{C}^{(\mathbf{1})}, \mathcal{C}^{(\mathbf{2})}, \cdots, \mathcal{C}^{(\boldsymbol{M})} \rangle\rangle \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ is in the TT
   format, relative error threshold $\varepsilon$ with an overestimated TT ranks,
   $\mathbf{r}_{TT} = \{R_1, R_2, \ldots, R_{M-1}\}$ and ranks bounded by $R_{\max}$
2: Output: An $M$th-order tensor $\hat{\mathcal{X}}$ with a reduced TT rank; such that
   $\|\mathcal{X}' - \hat{\mathcal{X}}\|_F \leqslant \varepsilon \, \|\mathcal{X}'\|_F$
3: Initialize by computing truncation parameter $\delta = \frac{\varepsilon}{\sqrt{M-1}} \|\mathcal{X}'\|_F$.
4: — Left-to-right Orthogonalization —
5: for $m = 1$ to $M - 1$ do
6:     $\mathcal{C}^{(m)}_{<2>} := \mathbf{Q}_m \mathbf{R}$, where $\mathcal{C}^{(m)}_{<2>} \in \mathbb{R}^{R_{m-1} I_m \times R_m}$
7:     $\mathcal{C}^{(m)}_{<2>} \leftarrow \mathbf{Q}_m$ and $\mathcal{C}^{(m+1)}_{<1>} \leftarrow \mathbf{R}\mathcal{C}^{(m+1)}_{<1>}$, where $\mathcal{C}^{(m+1)}_{<1>} \in \mathbb{R}^{R_m \times I_{m+1} R_{m+1}}$
8: end for
9: — Compression of the orthogonalized representation —
10: for $m = M$ to 2 step $-1$ do
11:     Compute $\delta$-truncated SVD: $\mathcal{C}^{(m)}_{<1>} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}$,
12:     Govern minimum rank $\hat{R}_{m-1} \leftarrow \sum_{r > R_{m-1}} \sigma_r^2 \leqslant \delta^2 \|\mathbf{S}\|^2$
13:     New cores $\hat{\mathcal{C}}^{(m-1)}_{<2>} \leftarrow \hat{\mathcal{C}}^{(m-1)}_{<2>} \hat{\mathbf{U}}\hat{\mathbf{S}}$ and $\hat{\mathcal{C}}^{(m)}_{<1>} = \hat{\mathbf{V}}^{\mathrm{T}}$
14: end for
15: return $M$th-order TT tensor $\hat{\mathcal{X}} = \langle\langle \hat{\mathcal{C}}^{(1)}, \hat{\mathcal{C}}^{(2)}, \ldots, \hat{\mathcal{C}}^{(M)} \rangle\rangle \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_M}$, with
    reduced cores $\hat{\mathcal{C}}^{(m)} \in \mathbb{R}^{\hat{R}_{m-1} \times I_m \times \hat{R}_m}$
```

3.3.4 Existing Interrelationship of Tensor Decompositions

3.3.4.1 CP to TT

A tensor in the CP format, given by

$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(N)}, \tag{3.7}$$

can be straightforwardly converted into the TT format as follows. Since each of the R rank-1 tensors can be represented in the TT format of TT rank $(1, 1, \ldots, 1)$, taking into account that the CP decomposition is a sum of R rank-1 tensors, therefore it is possible that,

$$\mathcal{X} = \sum_{r=1}^{R} \langle\langle \mathbf{a}_r^{(1)\mathrm{T}}, \mathbf{a}_r^{(2)\mathrm{T}}, \ldots, \mathbf{a}_r^{(M)\mathrm{T}} \rangle\rangle \tag{3.8}$$

$$= \langle\langle \mathcal{C}^{(\mathbf{1})}, \mathcal{C}^{(\mathbf{2})}, \cdots, \mathcal{C}^{(\boldsymbol{M})} \rangle\rangle, \tag{3.9}$$

where the TT-cores $\mathcal{C}^{(m)} \in \mathbb{R}^{R \times I_m \times R}$ have diagonal lateral slices $\mathbf{C}^{(m)}(:, i_m, :) = \mathbf{C}^{(m)}_{i_m} = \text{diag}\left(a_{i_m,1}, a_{i_m,2}, \ldots, a_{i_m,R}\right) \in \mathbb{R}^{R \times R}$ for $m = 2, 3, \ldots, M-1$ and $\mathbf{C}^{(1)} = \mathbf{A}^{(1)} \in \mathbb{R}^{I_1 \times R}$ and $\mathbf{C}^{(M)} = \mathbf{A}^{(M)\,\mathrm{T}} \in \mathbb{R}^{R \times I_M}$ (see Figure 3.6).

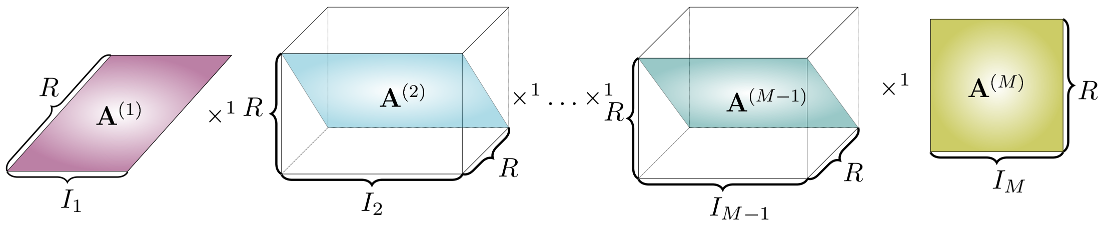


Figure 3.6: Relation representation of an $M^{th}-$order tensor $\mathcal{X}$ between CP format and TT format.

Table 3.1: Storage Complexity.

1. Full tensor format (ktensor)	$\mathcal{O}(I^M)$
2. CP	$\mathcal{O}(MIR_{CP})$
3. Tucker	$\mathcal{O}(MIR_{Tucker} + R^M_{Tucker})$
4. TT/MPS	$\mathcal{O}(MIR^2_{TT})$
5. CP-TT	
6. QTT [52]	$\mathcal{O}(Mlog(I)R^2_{QTT}R^4_{TT})$
7. QTT-Tucker [52]	$\mathcal{O}(Mlog(I)R^2_{QTT}R^2_{Tucker} + MR^2_{TT}R_{Tucker})$

3.3.4.2 Qunatized Tensor Train (QTT)

The QTT (Quantized Tensor Train) is considered an extension of the TT (Tensor Train) decomposition, with factorization into quantized sizes (e.g., 2, 3, 4). The QTT represents more deeper structure of the TT decomposition by introducing virtual modes. The main advantage of QTT is analogous to the 'blessing in disguise' concept, but in a tensorial context, referred to as the 'blessing of dimensionality' [40]. The QTT was introduced in [90], and [132] as *Quantized tensor network (QTN)*. In whcih the small 3rd-order cores are connected through tensor contractions while providing an efficient, compact, and low-rank representation of a data tensor, reducing the problem of curse of dimensionality.

to understand concept of QTT, if we take a huge size vector $\mathbf{x} \in R^I, I = 2^p$ then an example tensor $\mathcal{X}$ as $2 \times 2 \times \cdots \times 2$ with dimension p can be created. There often exists a well compressed tensor factorization of such large vector $\mathbf{x}$ by imposing possible LRTF.

Although, the ranks of the QTT format often increase significantly with data size, however with a linear improvement in accuracy. To address this, the QTT-Tucker format [52] was invented, and is explained in next subsection. This format applies the

TT approximation to both the Tucker core tensor and the factor matrices, supporting distributed computation and often resulting in bounded ranks, thus mitigating the curse of dimensionality.

3.3.4.3 QTT-Tucker

The QTT-Tucker format combines the benefits of the Tucker, TT, and QTT formats. In many cases, the ranks of the Tucker and TT formats are similar. This means that using the QTT approximation of the Tucker factors instead of the TT ones can often result in smaller ranks for the same level of accuracy and avoid high rank peaks in the middle of a TT. The QTT-Tucker format is a closed manifold, similar to the Tucker format. Work has been done on converting the "TT-to-Tucker" format and its extended version, where TT is first converted into its quantic form (QTT), to the Tucker format. This approach requires only M blocks with cubic storage in rank, whereas $Mlog(I)$ blocks have a quadratic dependence, as in the linear QTT format. The QTT approximation of the Tucker factors is useful for approximating tensors generated by smooth functions. The reason why this last step was not explicitly written is that, when computing a Tucker decomposition, it is more efficient to keep the Tucker factors separate from the format used for the core. The QTT-Tucker format has a worse asymptotic storage estimate compared to other formats, but it performs better than the "standard" linear-structured QTT in some cases. Additionally, the rank distribution is more uniform compared to the linear QTT.

The storage complexities for different low-rank tensor format has for their Interrelationship conversion is summarized in Table 3.1.

3.4 Tensor Classification

Consider now a typical problem in computer vision, where the objects are represented by data tensors and the number of the training measurements are limited. This leads to the need of using a classical machine learning model that deals with tensor data avoiding the *curse of dimensionality* by using low-rank tensor methods. Further, the mathematical foundation for such model is now explained.

3.4.1 Support Tensor Machine

The previous chapter established a foundation for SVMs (§2.2.1), and consideration of tensors naturally leads to a tensor-based extension of SVM, called the *support tensor machine* (STM). For this, consider a general supervised learning scenario with N training measurements, $\{\mathcal{X}_i, y_i\}, i \in \langle N \rangle$, represented by Mth-order tensors, $\mathcal{X}_i \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_M}$, which are associated with the scalar variable y_i. As here the focus is on tensor classification model, $y_i \in \{+1, -1\}$ that is, it takes categorical values, which is a standard *classification problem.* The linear boundary STM [176, 177, 67] can be formulated through the composition of M, QP sub-problems with inequality constraints in non-separable case,

$$\begin{aligned} &\min_{\mathbf{w}_m, b, \boldsymbol{\xi}} \frac{1}{2} \|\mathbf{w}_m\|^2 \prod_{1 \leqslant \tilde{m} \leqslant M}^{\tilde{m} \neq m} (\|\mathbf{W}_{\tilde{m}}\|^2) + C \sum_{i=1}^{N} \xi_i \\ &\text{subject to} \quad y_i \left(\mathbf{w}_m^{\mathrm{T}} (\mathcal{X}_i \bar{\times}_{\tilde{m} \neq m} \mathbf{w}_{\tilde{m}}) + b \right) \geqslant 1 - \xi_i, \quad \boldsymbol{\xi} \geqslant 0, \quad i \in \langle N \rangle. \end{aligned} \tag{3.10}$$

Once the STM solution has been obtained, the class label of a test example, $\mathcal{X}_t$, can be predicted by a nonlinear transformation

$$y(\mathcal{X}) = \operatorname{sign}\left(\mathcal{X}_* \bar{\times}_1 \mathbf{w}_1 \cdots \bar{\times}_M \mathbf{w}_M + b\right). \tag{3.11}$$

As it is discussed in previous chapter that in practice, it is often more convenient to solve the optimization problem (3.10) by considering the *dual problem* (2.12), given by

$$\begin{aligned} &\max_{\alpha_1, \dots, \alpha_N} \sum_{i=1}^{N} \boldsymbol{\alpha}_i - \frac{1}{2} \sum_{i,j=1}^{N} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y_i y_j \langle \mathcal{X}_i, \mathcal{X}_j \rangle, \\ &\text{subject to} \quad \sum_{i=1}^{N} \boldsymbol{\alpha}_i y_i = 0, \quad 0 \leqslant \boldsymbol{\alpha}_i \leqslant C, \quad i \in \langle N \rangle, \end{aligned} \tag{3.12}$$

where $\boldsymbol{\alpha}$ are the Lagrange multipliers. It is straightforward that when the input samples, $\mathcal{X}_i$ are vectorized, this model converts into standard vector SVM (2.3). As mentioned in SMM models (§2.3.3) the lost structured issue persist with tensors as well. In case of tensors, the inner product of a tensor has been looked through a different perspective, such as the computation of $\langle \mathcal{X}_i, \mathcal{X}_j \rangle$, is equivalent to the inner product of the rank one decomposition $\left(\mathcal{X}_i \approx \sum_{r=1}^{R} \mathbf{x}_{ir}^{(1)} \circ \mathbf{x}_{ir}^{(2)} \circ \cdots \circ \mathbf{x}_{ir}^{(N)} \text{ and } \mathcal{X}_j \approx \sum_{r=1}^{R} \mathbf{x}_{jr}^{(1)} \circ \mathbf{x}_{jr}^{(2)} \circ \cdots \circ \mathbf{x}_{jr}^{(M)} \right)$ as follows [67],

$$\langle \mathcal{X}_i, \mathcal{X}_j \rangle \approx \sum_{p=1}^{R} \sum_{q=1}^{R} \left\langle \mathbf{x}_{ip}^{(1)}, \mathbf{x}_{jq}^{(1)} \right\rangle \left\langle \mathbf{x}_{ip}^{(2)}, \mathbf{x}_{jq}^{(2)} \right\rangle \cdots \left\langle \mathbf{x}_{ip}^{(N)}, \mathbf{x}_{jq}^{(N)} \right\rangle, \tag{3.13}$$

and (3.12) can be solved by a sequential QP optimization algorithm. The prediction function for a test example $\mathcal{X}_t$ is given as,

$$y(\mathcal{X}_t) = \operatorname{sign}\left(\sum_{i=1}^{N} \sum_{p=1}^{R} \sum_{q=1}^{R} \boldsymbol{\alpha}_i y_i \prod_{m=1}^{M} \left\langle \mathbf{x}_{ip}^{(m)}, \mathbf{x}_{tq}^{(m)} \right\rangle + b \right). \tag{3.14}$$

Considering this approach for the non-linear, non-separable STM can be an intuitive choice. This is further discussed.

3.4.2 Tensor-product Reproducing Kernel Hilbert Space

The approach to tensor-product RKHS is based on standard RKHS (§2.3.1). In machine learning, tensor products have been used to build kernels for a long time [56, 79]. The structure of tensor product functions has been exploited for the existing regularization in [166]. Here kernel functions can be considered as a mean for defining a new topology which implies *a priori* knowledge about the invariance in the input space [156]. This means, that the kernel function's predictions or behavior should remain unchanged even when the input undergoes certain transformations or perturbations. From §2.3.1 assume that $(\mathcal{H}_{\tilde{m}}, \langle\cdot,\cdot\rangle_{\tilde{m}}, k^{(\tilde{m})})$ is a RKHS on a domain $\mathcal{X}_{\tilde{m}}$, $\tilde{m} \in \langle M\rangle$ such that [166],

$$\mathbf{x} = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(M)}) \in \mathcal{X}, \quad \text{s.t.} \quad \mathcal{X} := \times_{\tilde{m}}^{M} \mathcal{X}_{\tilde{m}}.$$

If a vector space is formed by the linear combinations of the functions $\otimes_{\tilde{m}=1}^{M} f^{(\tilde{m})}$ that simply gives the tensor-product HS denoted by $(\boldsymbol{\mathcal{H}}, \langle\cdot,\cdot\rangle)$, without loss of generality, defined by:

$$\otimes_{\tilde{m}\in\langle M\rangle} f^{(\tilde{m})} \mapsto \prod_{\tilde{m}\in\langle M\rangle} f^{(\tilde{m})}(\mathbf{x}^{(\tilde{m})}), \qquad f^{(\tilde{m})} \in \mathcal{H}_m, \ \forall\ \tilde{m} \in \langle M\rangle, \tag{3.15}$$

where $\otimes_{\tilde{m}\in\langle M\rangle}$ is a *rank-1 tensor*, then the HS related to tensor product is $\boldsymbol{\mathcal{H}}$ given by $(\mathbf{f}, \mathbf{g}, \ldots)$ in a way that,

$$\mathbf{f} = \sum_{\mathbf{i}} \boldsymbol{\alpha}_{\mathbf{i}} f_{i_1}^{(1)} \otimes f_{i_2}^{(2)} \otimes \ldots \otimes f_{i_M}^{(M)}, \tag{3.16}$$

with $\mathbf{i} = (i_1, i_2, \ldots, i_M)$ and

$$\boldsymbol{f}(x) = \sum_{\boldsymbol{i}} \boldsymbol{\alpha_i} (f_{i_1}^{(1)} \otimes f_{i_2}^{(2)} \otimes \ldots \otimes f_{i_M}^{(M)})(x) = \sum_{\boldsymbol{i}} \boldsymbol{\alpha_i} \prod_{\tilde{m}=1}^{M} f_{i_{\tilde{m}}}^{(\tilde{m})}(\mathbf{x}^{(\tilde{m})}), \tag{3.17}$$

taking a symmetric kernel function as,

$$\boldsymbol{k} : (\mathbf{x}_i, \mathbf{x}_j) \mapsto k^{(1)}(\mathbf{x}_i^{(1)}, \mathbf{x}_j^{(1)})\ k^{(2)}(\mathbf{x}_i^{(2)}, \mathbf{x}_j^{(2)})\ \ldots\ k^{(M)}(\mathbf{x}_i^{(M)}, \mathbf{x}_j^{(M)}), \tag{3.18}$$

from (3.17) there is,

$$\boldsymbol{f} = \sum_{\boldsymbol{i}} \boldsymbol{\alpha_i} \prod_{\tilde{m}=1}^{M} \langle f_{i_{\tilde{m}}}^{(\tilde{m})}, k^{(\tilde{m})}(\cdot, \mathbf{x}^{(\tilde{m})}) \rangle_{\tilde{m}} = \langle \boldsymbol{f}, \boldsymbol{k}_x \rangle. \tag{3.19}$$

Here, $\boldsymbol{k}_x$ is corresponding *reproducing kernel* for tensor product HS. Hence, the *tensor product Reproducing Kernel Hilbert Space* is denoted by $(\boldsymbol{\mathcal{H}}, \langle \cdot, \cdot \rangle, \boldsymbol{k})$ with $k^{(\tilde{m})}$ a *factor kernel* for $\mathcal{H}_{\tilde{m}}$ the *factor space.*

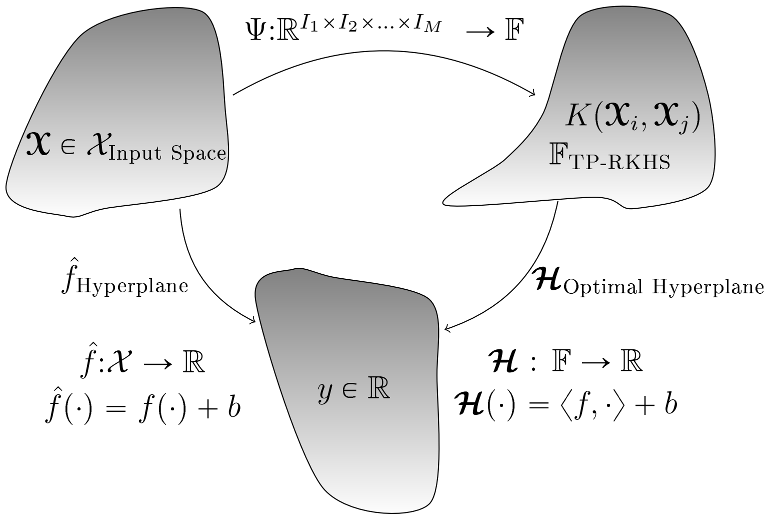

$\Psi{:}\mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M} \rightarrow \mathbb{F}$
$\boldsymbol{\mathcal{X}} \in \mathcal{X}_{\text{Input Space}}$
$K(\boldsymbol{\mathcal{X}}_i, \boldsymbol{\mathcal{X}}_j)$
$\mathbb{F}_{\text{TP-RKHS}}$
$\hat{f}_{\text{Hyperplane}}$
$\boldsymbol{\mathcal{H}}_{\text{Optimal Hyperplane}}$
$\hat{f}{:}\mathcal{X} \rightarrow \mathbb{R}$
$\hat{f}(\cdot) = f(\cdot) + b$
$y \in \mathbb{R}$
$\boldsymbol{\mathcal{H}} : \mathbb{F} \rightarrow \mathbb{R}$
$\boldsymbol{\mathcal{H}}(\cdot) = \langle f, \cdot \rangle + b$

Figure 3.7: Kernel trick with TP-RKHS embedding.

Figure 3.7 illustrate a nonlinear mapping from tensor input space ($\boldsymbol{\mathcal{X}} \in \mathcal{X}_{\text{Input Space}}$) to feature TP-RKHS ($\mathbb{F}_{\text{TP-RKHS}}$). Both spaces leads to the computation of the optimal hyperplane for output y where one is explicit and another is implict (kernel) way.

3.4.3 Kernelised Support Tensor Machine

The presented model here uses a data set $\{(\boldsymbol{\mathcal{X}}_i, y_i)\}_{i=1}^{N}$ with input data in the form of a tensor $\boldsymbol{\mathcal{X}}_i \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$. The maximum margin approach to get the separation hyperplane is taken. The non-linear boundary STM can be formulated from (3.10) the composition of M, QP sub-problems with inequality constraints in non-separable case,

$$\begin{aligned} &\min_{\mathbf{w}_m, b, \boldsymbol{\xi}} \ \frac{1}{2} \|\mathbf{w}_m\|^2 \prod_{1 \leqslant \tilde{m} \leqslant M}^{\tilde{m} \neq m} (\|\mathbf{w}_{\tilde{m}}\|^2) + C \sum_{i=1}^{N} \xi_i \\ &\text{subject to} \quad y_i \Big(\mathbf{w}_m^{\mathrm{T}} \left(\Psi(\boldsymbol{\mathcal{X}}_i) \bar{\times}_{\tilde{m} \neq m} \mathbf{w}_{\tilde{m}} \right) + b \Big) \geqslant 1 - \xi_i, \quad \boldsymbol{\xi} \geqslant 0, \quad i \in \langle N \rangle. \end{aligned} \tag{3.20}$$

Hence, the mentioned objective function for a nonlinear boundary in the tensor space can be further written in much simpler way, as follows [25]:

$$\begin{aligned} \min_{\mathcal{W},b} \quad & \langle \mathcal{W}, \mathcal{W} \rangle + C \sum_{i=1}^{N} \xi_i \\ \text{subject to} \quad & y_i(\langle \mathcal{W}, \Psi(\mathcal{X}_i) \rangle + b) \geqslant 1 - \xi_i \qquad \xi_i \ \geqslant 0 \quad \forall i. \end{aligned} \tag{3.21}$$

The classification setup given in eq. (3.21) is known as Support Tensor Machine (STM) [176]. The dual formulation of the corresponding primal problem can be given as follows:

$$\begin{aligned} \max_{\alpha_1,\ldots,\alpha_N} \quad & \sum_{i=1}^{N} \boldsymbol{\alpha}_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y_i y_j \langle \Psi(\mathcal{X}_i), \Psi(\mathcal{X}_j) \rangle \\ \text{subject to} \quad & 0 \leqslant \boldsymbol{\alpha}_i \leqslant C, \quad \sum_{i=1}^{N} \boldsymbol{\alpha}_i y_i = 0 \ \ \forall i. \end{aligned} \tag{3.22}$$

The nonlinear feature mapping $\Psi \colon \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M} \to \mathbb{F}$ takes tensorial input data to a higher dimensional space similarly to the vector case. Hence, by extending the kernel trick, explained in §2.3.2, we have reproducing kernel $K : \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M} \times \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M} \to \mathbb{R}$ (Figure 3.7), such that,

$$K_{i,j} = K\left(\mathcal{X}_i, \mathcal{X}_j\right) = \langle \Psi(\mathcal{X}_i), \Psi(\mathcal{X}_j) \rangle_{\mathbb{F}}, \tag{3.23}$$

now, the STM can be defined as follows:

$$\begin{aligned} \max_{\alpha_1,\ldots,\alpha_N} \quad & \sum_{i=1}^{N} \boldsymbol{\alpha}_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y_i y_j K(\mathcal{X}_i, \mathcal{X}_j) \\ \text{subject to} \quad & 0 \leqslant \boldsymbol{\alpha}_i \leqslant C, \quad \sum_{i=1}^{N} \boldsymbol{\alpha}_i y_i = 0 \ \ \forall i. \end{aligned} \tag{3.24}$$

The STM classifier for predicting correct labels of test tensor data is given by,

$$G(\mathcal{X}) = \operatorname{sign}\left(\sum_{i=1}^{N} \boldsymbol{\alpha}_i y_i \langle \Psi(\mathcal{X}_i), \Psi(\mathcal{X}) \rangle + b_0 \right). \tag{3.25}$$

By using the *kernel trick* [155], this becomes,

$$G(\mathcal{X}) = \operatorname{sign}\left(\sum_{i=1}^{N} \boldsymbol{\alpha}_i y_i K(\mathcal{X}_i, \mathcal{X}) + b_0\right). \tag{3.26}$$

The value of b_0 is given as follows,

$$\begin{aligned} b_0 &= \frac{1}{N_0} \sum_{i:\boldsymbol{\alpha}_i \in (0,C)} \left(y_i - \sum_{j=1}^{N} \boldsymbol{\alpha}_j \langle \Psi(\mathcal{X}_j), \Psi(\mathcal{X}_i) \rangle \right), \\ &= \frac{1}{N_0} \sum_{i:\boldsymbol{\alpha}_i \in (0,C)} \left(y_i - \sum_{j=1}^{N} \boldsymbol{\alpha}_j K(\mathcal{X}_j, \mathcal{X}_i) \right), \quad \text{with} \quad N_0 = \sum_{i:\alpha_i \in (0,C)} 1. \end{aligned} \tag{3.27}$$

This setup is called the *Kernelised STM (KSTM)*. Once the real-valued function (kernel) value for each pair of tensors is available, state-of-the-art methods, such as LIBSVM [29] can be used, which relies on the *Sequential Minimal Optimization* algorithm to optimize the weights α_i and provides optimal parameter values $\boldsymbol{\alpha}_i$ and b_0. Hence, the pre-eminent part is the kernel function $K(\mathcal{X}_i, \mathcal{X}_j)$. However, the direct treatment of large tensors can be both numerically expensive and inaccurate due to over-fitting. Therefore, it is needed to choose a kernel that exploits the tensor decomposition. The next section is a brief overview of research work in the direction of optimizing and finding state-of-the-art for Kernelised Support Tensor Machine (3.24).

3.4.4 Reproducing Tensor Kernels

At the hand of tensor-product RKHS (§3.4.2), discussion about the kernels for tensor-valued inputs, that exploit multi-way structures while maintaining the notion of similarity measures, comes forward. Most straightforward tensor-valued *reproducing kernels* would be generalized vector-valued into Mth-order tensors. The most common tensor-valued kernels, $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, are given by,

$$\text{Linear kernel:} \quad K(\mathcal{X}, \mathcal{X}') = \langle \operatorname{vec}(\mathcal{X}), \operatorname{vec}(\mathcal{X}') \rangle. \tag{3.28}$$

There are many possibilities to define kernel functions for tensors. As the focus of this thesis is on LRTF methods (§3.3) with nonlinear boundary, therefore here are some ways to define *factor kernel* functions,

Vectorization :

$$K(\mathcal{X},\mathcal{Y}) = \Big\langle \Phi(\mathrm{vec}(\mathcal{X})), \Phi(\mathrm{vec}(\mathcal{Y}))\Big\rangle, \tag{3.29}$$

Matricization :

$$K(\mathcal{X},\mathcal{Y}) = \Big\langle \Phi(\mathbf{X}_{(1)},\dots,\mathbf{X}_{(M)}), \Phi(\mathbf{Y}_{(1)},\dots,\mathbf{Y}_{(M)})\Big\rangle, \tag{3.30}$$

CP Decomposition :

$$K(\mathcal{X},\mathcal{Y}) = \Big\langle \Phi\big(\mathbf{A}_X^{(1)},\dots,\mathbf{A}_X^{(M)}\big), \Phi\big(\mathbf{B}_Y^{(1)},\dots,\mathbf{B}_Y^{(M)}\big)\Big\rangle, \tag{3.31}$$

TT Decomposition :

$$K(\mathcal{X},\mathcal{Y}) = \Big\langle \Phi\big(\mathcal{C}_X^{(1)},\dots,\mathcal{C}_X^{(M)}\big), \Phi\big(\mathcal{C}_Y^{(1)},\dots,\mathcal{C}_Y^{(M)}\big)\Big\rangle. \tag{3.32}$$

In order to define a similarity measure that directly exploits the multilinear algebraic structure of the input tensors, [167, 168] proposed a tensorial kernel which both exploits the algebraic geometry of tensors spaces and provides a similarity measure between the different subspaces spanned by higher-order tensors. Another approach is a balanced compromise between (3.13) and the *factor kernel* defined on $m-$ mode matricization of a pair of tensors as follows,

$$K(\mathcal{X},\mathcal{X}') = \prod_{m=1}^{M} k^{(m)}\left(\mathcal{X}_{(m)},\mathcal{X}'_{(m)}\right). \tag{3.33}$$

In case of higher dimensions as known from earlier sections, matricization is not efficient as this similarity measure does not generalize well. Therefore, different kind of kernels were introduced in literature, *e.g.* [209, 38, 39] suggested a new group of probabilistic product kernels based on generative models. The advantage of probabilistic tensor kernels is that they provide a way to model one tensor to a lower-dimensional vector spaces, this makes it possible for multiway relations to be captured within a similarity measure. This kernel can then effectively capture the statistical properties of tensors, which promises to be a powerful tool for multidimensional structured data analysis. The following section gives an overview of development of such tools over decades along with it advantages and drawbacks.

3.5 Low-rank Tensor Decomposition in Kernel Models

When tensor objects are reshaped into vectors, a substantial amount of inherent information present in the tensorial data is lost, as discussed earlier. This phenomenon is evident in domains like medical images, such as fMRI data, where adjacent voxel values tend to exhibit proximity [71]. Consequently, to mitigate this loss of information, the application of STM (Section 2.3) is recommended, as emphasized by various authors [177, 214, 62].

In a notable departure from the classical maximum-margin criterion, Wolf et al. [198] introduced the idea of minimizing the rank of the weight parameter alongside orthogonal constraints on its columns. A subsequent enhancement by Pirisiavash [139] relaxed these orthogonality constraints to refine the Wolf method.

Addressing the tensor factorization strategies, Hao et al. [67] delved into an R-sum rank-one tensor factorization for each input tensor. In contrast, Kotsia et al. [96] opted for the Tucker decomposition of the weight parameter to retain a richer structural understanding. Further extending this paradigm, Zeng et al. [204] incorporated a Genetic Algorithm (GA) as a precursor to the Support Tucker Machine (STuM), enhancing the contraction of the input feature tensor.

Notably, these strategies, including the R-sum rank-one tensor, Tucker, and TT approximations [31], predominantly center on linear data representations. However, the limitation of a linear decision boundary's suitability for complex real-world data separation is well-recognized [70].

The subsequent chapters will pivot towards addressing the challenge of nonlinearity in tandem with low-rank approximation.

3.6 Chapter summary

This chapter begins with a systematic transition from LRMF to LRTF in Section 3.3 and a basic guide to tensor notations and definitions in Section 3.2. Section 3.3 demonstrates linear and multilinear dimensionality reduction approaches for analysing extreme-scale multidimensional data. The main goal is to illustrate that tensor decompositions are a natural phenomenon to study the inherent compression ability of data and corresponding process information. The numerical multilinear algebra serves as the mathematical backbone to build strong and stable models for classification in large-scale datasets. Section 3.3.4 showcases the ease of multiple tensor decomposition into each other with no extra cost, leading to a win-win situation in selecting the best features from each of these decompositions while working better with constraints on factor matrices. While research on tensor methods for dimensionality reduction in many modern applications of machine learning is emerging, Section 3.4 provides an explicit overview of the classification problem with tensor as input data. Section 3.4.1 serves as a transitional mathematical theory from vector to tensor space, such as RKHS (Section 2.3.1) to tensor-product RKHS (Section 3.4.2) and kernelised SVMs to Kernelised STMs (Section 3.4.3). The field of machine learning is growing exponentially, with multiple application problems and the theoretical need for a mathematical foundation in machine learning to estimate this curse of dimensionality. Section 3.5 furnishes this need with the work done in the advancement of the field, showcasing a small historical story from the beginning of LRTFs in Kernel methods that is clearly conveyed by the section's title.

CHAPTER 4

EFFICIENT STRUCTURE-PRESERVING SUPPORT TENSOR TRAIN MACHINE

4.1 Introduction 51
  4.1.1 Problem Formulation 52
  4.1.2 Related Work 53
4.2 The Proposed Algorithm 53
  4.2.1 Uniqueness Enforcing TT- SVD 54
  4.2.2 TT-CP Expansion 57
  4.2.3 Norm Equilibration 57
  4.2.4 Noise-aware Threshold and Rank Selection 58
  4.2.5 Reproducing Tensor Kernels (§3.4.4) 58
  4.2.6 Nonlinear Mapping 59
4.3 Numerical Tests 61
  4.3.1 Data Collection 62
  4.3.2 Influence of Individual Algorithmic Steps 63
  4.3.3 Comparison to Other Methods 65
4.4 Chapter Summary 68

Contents

4.1 Introduction

The previous chapters have provided in detail the main problem Kernelized Support Tensor Machine (§3.4.3) but do not include any solution approaches. The existing kernel for tensor data gives a small pavement to the long way of finding solution to KSTM (3.24). The depth literature review in §3.5 provide a route through history for such advancements. This chapter is also a contribution work in similar direction. Recently, kernel approximations in the TT format have been introduced in [32]. Initially, a similar idea for fMRI data sets had pursued, but it was observed that the nonlinear SVM classification using the TT factors directly leads to poor accuracy, since different TT factors have different dimensions and scales, making the feature space more complicated. Hence, this chapter provides a better exploitation of the data structure.

Main Novelty: This chapter explains the development of an efficient structure-preserving nonlinear kernel function for SVM classification of tensorial data (3.24), by computing a reliable CP approximation for DuSK (4.6). The starting point is computation of TT approximation (3.6) of the data points, which can be computed reliably by the TT-SVD algorithm (Algorithm 3.3). Moreover, the uniqueness enforcement on the SVD factors, such that "close" tensors yield "close" TT factors, is included. Along with, performing an exact expansion of the TT decomposition into the CP format. This unifies the dimensions of the data used in classification. Finally, the redistribution of the norms of the CP factors to equilateral the actual scales of the data elements, is done. This yields a CP decomposition that is free from scaling indeterminacy, while being a reliable approximation of the original data. It is observed that using this decomposition in DuSK significantly increases the classification accuracy and stability of the STL (§3.4.1).

The chapter is structured as follows. The Section §4.2, explains the entire proposed algorithm step by step. In particular, the introduction to the uniqueness enforced TT-SVD algorithm (§4.2.1), the TT-CP expansion (§4.2.2) and the norm equilibration (§4.2.3). The Section §4.3 benchmarks the different steps of the proposed algorithm and propose comparison to a variety of competing methods using two data sets each from two different fields with a limited amount of training data, which are known to be challenging for classification.

4.1.1 Problem Formulation

This chapter provides a method for solving the following problem,

$$\begin{aligned} \max_{\alpha_1,\ldots,\alpha_N} \quad & \sum_{i=1}^{N} \boldsymbol{\alpha}_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \boldsymbol{\alpha}_i\boldsymbol{\alpha}_j y_i y_j K(\boldsymbol{\mathcal{X}}_i, \boldsymbol{\mathcal{X}}_j) \\ \text{subject to} \quad & 0 \leqslant \boldsymbol{\alpha}_i \leqslant C, \quad \sum_{i=1}^{N} \boldsymbol{\alpha}_i y_i = 0 \quad \forall i. \end{aligned} \tag{4.1}$$

As mentioned earlier in the literature review of KSTM methods (§3.5), using low-rank approximation for kernel computation is one way of dealing with computation cost as well as producing state-of-the-art.

4.1.2 Related Work

Motivation : This chapter draws motivation from the works by DuSK and MMK[71, 72], which are based on the CP decomposition. CP-based approaches are widely used in signal processing due to their convenience for working with factor kernels. The Dual Structure-preserving Kernel (DuSK) was introduced for STL, tailored specifically to SVM and tensor data, using the CP format [71]. Subsequently, the concept of Kernelised-CP (KCP) factorization was introduced, leading to the Multi-way Multi-level Kernel (MMK) technique [72, 73], which further extends the kernelization in factors. Detailed descriptions of DuSK can be found in §4.2.5. DuSK and MMK, when coupled with an accurate CP approximation, offer effective and efficient classification solutions. However, the numerical instability and computational difficulty associated with CP approximation for arbitrary data can be limiting [50]. Optimization methods such as Newton, Steepest Descent, or Alternating Least Squares used to obtain the CP decomposition may only yield locally optimal solutions, posing challenges in distinguishing between local and global optima.

Utilizing Low-rank Approximations and Kernel Methods: The limitations of CP don't hinder the use of low-rank approximation in kernel methods. Tensor decompositions and kernel-based techniques have become invaluable tools in various learning tasks. For instance, the TT decomposition is employed in both input tensors and corresponding weight parameters in generalized linear models, as shown in [127]. Kernel Principal Component Analysis (KPCA), a nonlinear feature extraction technique, was proposed in [200]. In [103], a strategy to accelerate Convolutional Neural Networks (CNN) involves applying a low-rank CP decomposition on kernel projection tensors. The Tensor Train (TT) decomposition, offering a stable approximation similar to Tucker format while scaling well to higher dimensions like CP, has led to a natural generalization of DuSK (MMK) to the TT format, as outlined in [31].

Furthermore, the applicability of the incentives from DuSK and MMK [71, 72] in the context of TT and enhanced methods is elaborated upon in the subsequent sections.

4.2 The Proposed Algorithm

The first essential step towards using tensors is to approximate them in a low-parametric representation. To achieve a stable learning model, our method starts with computing the TT approximations of all data tensors. The second most expensive part is the computation of $K\left(\mathcal{X}_i, \mathcal{X}_j\right)$ for each pair of tensors. Therefore, an approximation of the kernel is required. Moreover, the kernel has the property of exploiting the factorized tensor representation. These issues are resolved in the rest of this section.

4.2.1 Uniqueness Enforcing TT- SVD

Since the TT decomposition is computed using the SVD [133] (Algorithm 3.3), the particular factors $\mathcal{C}^{(1)}, \mathcal{C}^{(2)}, \ldots, \mathcal{C}^{(M)}$ are defined only up to a sign indeterminacy. For example, the first step is, to compute the SVD of the 1-mode matricization,

$$\mathcal{X}_{(1)} = g_1 u_1 v_1^\top + \cdots + g_{I_1} u_{I_1} v_{I_1}^\top,$$

followed by truncating the expansion at rank R_1 or according to the accuracy threshold ε, choosing R_1 such that $g_{R_1+1} < \varepsilon$. However, any pair of vectors $\{u_{r_1}, v_{r_1}\}$ can be replaced by $\{-u_{r_1}, -v_{r_1}\}$ without changing the whole expansion. While this is not an issue for data compression, classification using TT factors can be affected significantly by this indeterminacy. For example, tensors that are close to each other should likely produce the same label. In contrast, even a small difference in the original data may lead to a different sign of the singular vectors, and consequently, significantly different values in the kernel matrix $K(\mathcal{X}_j, \mathcal{X}_i)$ and the predicted label (3.26). As it will be explained in §4.2.6, the kernels are functions of the left singular vectors u_i only (Algorithm 4.2).

The signs of the singular vectors are fixed as follows. For each $r_1 = 1, \ldots, R_1$, find the position of the maximum in modulus element in the left singular vector, $i^*_{r_1} = \arg\max_{i=1,\ldots,I_1} |u_{i,r_1}|$, and make this element positive,

$$\bar{u}_{r_1} := u_{r_1}/\text{sign}(u_{i^*_{r_1},r_1}), \quad \bar{v}_{r_1} := v_{r_1} \cdot \text{sign}(u_{i^*_{r_1},r_1}).$$

Finally, collection of $\bar{u}_{r_1}$ is done into the first TT core, $\mathcal{C}^{(1)}_{r_0,i_1,r_1} = \bar{u}_{i_1,r_1}$, and continue with the TT-SVD algorithm using $\bar{v}_{r_1}$ as the right singular vectors. In contrast to the sign, the whole dominant singular terms $u_{r_1} v_{r_1}^\top$ depend continuously on the input data, and so do the maximum absolute elements. The procedure is summarized in Algorithm 4.1.

Lemma 4.1:
Assume that the singular values $g_1^{(m)}, \ldots, g_{R_m}^{(m)}$ are simple for each $m = 1, \ldots, M-1$. Then Algorithm 4.1 produces the unique TT decomposition. $\Diamond$

Proof. The m-th TT core produced in TT-SVD is a reshape of the left singular vectors of the Gram matrix of the current unfolding, $\mathbf{A}_m := \mathbf{Z}_m \mathbf{Z}_m^\top$. To set up an induction, notice that $\mathbf{Z}_1 = \hat{\mathbf{Z}}_1 = \mathcal{X}_{(1)}$ is unique, and assume that $\mathbf{Z}_m$ is unique too. Consider the eigenvalue decomposition $\mathbf{A}_m \mathbf{U}_m = \mathbf{U}_m \mathbf{\Lambda}_m$, $\mathbf{\Lambda}_m = \text{diag}(\lambda_1^{(m)}, \ldots, \lambda_{R_{m-1}I_m}^{(m)})$. Since the eigenvalues $\lambda_i^{(m)} = (g_i^{(m)})^2$ are simple, each of them corresponds to an eigenspace of dimension 1, spanned by the corresponding column of $\mathbf{U}_m$. This means that each eigenvector is unique up to a scalar factor, and, if the eigenvector is real and has Euclidean norm 1, the scalar factor can only be 1 or -1. The latter is unique if it is chosen as the sign of the largest in modulus element of the eigenvector (which is always nonzero), with ties broken to take the first of identical elements. It remains to establish the uniqueness

Algorithm 4.1: Uniqueness Enforcing TT-SVD

```
1: Input: $M$-dimensional tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$, relative error threshold $\epsilon$.
2: Ensure: Cores $\mathcal{C}^{(1)}, \mathcal{C}^{(2)}, \cdots, \mathcal{C}^{(M)}$ of the TT-approximation $\mathcal{X}'$ to $\mathcal{X}$ in the TT-format
   with TT-rounding (Alg. 3.4) ranks $r_m$ equal to the $\delta$-ranks of the unfoldings $\mathcal{X}_{(m)}$ of $\mathcal{X}$,
   where $\delta = \sqrt{\frac{\epsilon}{M-1}}\|\mathbf{A}\|_F$.
3: Initialize $\hat{\mathbf{Z}}_1 = \mathcal{X}_{(1)}, R_0 = 1$.
4: for $m = 1$ to $M-1$ do
5:     $\mathbf{Z}_m :=$ reshape $\left(\hat{\mathbf{Z}}_m, [R_{m-1}I_m,\ I_{m+1} \cdots I_M]\right)$
6:     Compute $\delta$-truncated SVD: $\mathbf{Z}_m = \mathbf{U}_m\mathbf{S}_m\mathbf{V}_m^T + \mathbf{E}_m$, $\|\mathbf{E}_m\|_F \leqslant \delta$, where
       $\mathbf{U}_m = [u_1^{(m)}, u_2^{(m)}, \ldots, u_{R_m}^{(m)}], \mathbf{S}_m = \text{diag}(g_1^{(m)}, g_2^{(m)}, \ldots, g_{R_m}^{(m)}), \mathbf{V}_m =$
       $[v_1^{(m)}, v_2^{(m)}, \ldots, v_{R_m}^{(m)}]$
7:     for $r_m = 1$ to $R_m$ do
8:         $i^*_{r_m} = \arg\max_{i=1,\ldots,R_{m-1}I_m} |u_{i,r_m}^{(m)}|$ (with ties broken to first element)
9:         $\bar{u}_{r_m}^{(m)} := u_{r_m}^{(m)}/\text{sign}(u_{i^*_{r_m},r_m}^{(m)}), \quad \bar{v}_{r_m}^{(m)} := v_{r_m}^{(m)} \cdot \text{sign}(u_{i^*_{r_m},r_m}^{(m)})$
10:        $\mathcal{C}_{r_{m-1},i_m,r_m}^{(m)} = \bar{u}_{r_{m-1}+(I_m-1)R_{m-1},\ r_m}^{(m)}$
11:    end for $\bar{\mathbf{V}}_m = [\bar{v}_1^{(m)}, \bar{v}_2^{(m)}, \ldots, \bar{v}_{R_m}^{(m)}]$
12:    $\hat{\mathbf{Z}}_{m+1} := \mathbf{S}_m\bar{\mathbf{V}}_m^T$
13: end for
14: $\mathcal{C}^{(M)} = \hat{\mathbf{Z}}_M$
```

of $\mathbf{Z}_{m+1}$ to complete the induction. By the orthogonality of $\bar{\mathbf{U}}_m = [\bar{u}_1^{(m)}, \ldots, \bar{u}_{R_m}^{(m)}]$, we get $\hat{\mathbf{Z}}_{m+1} = \bar{\mathbf{U}}_m^T\mathbf{Z}_m$, and since the reshape is unique, so is $\mathbf{Z}_{m+1}$. □

Remark 4.2:
Most of the data featuring in machine learning are noisy. Therefore, the singular values of the corresponding matricizations are simple almost surely, and hence the TT decomposition delivered by Algorithm 4.1 is unique almost surely. ◇

It is reasonable to say that if the sign of the singular values is undetermined, then an SVD algorithm [133] would be unstable in selecting such signs when the training data is slightly changed. Also, these signs would affect the classifier (3.26), hence to kernel. Therefore, to make it more understandable at this point how the sign of the singular values affects the kernels and thus the classifier. Here let consider two classes of 4×4 matrices of the following form:

$$\begin{aligned}
&\text{label } 1: \quad \mathbf{X}^{(1)} = \left[\exp(-ij)\right]_{i,j=1}^4 \qquad && \mathbf{X}^{(2)} = \mathbf{X}^{(1)} + 10^{-3} \cdot \mathbf{Z}^{(x)}, \quad \mathbf{Z}_{i,j}^{(x)} \sim \mathcal{N}(0,1),\\
&\text{label } -1: \quad \mathbf{Y}^{(1)} = \left[\exp(-2.5 \cdot ij)\right]_{i,j=1}^4 \qquad && \mathbf{Y}^{(2)} = \mathbf{Y}^{(1)} + 10^{-3} \cdot \mathbf{Z}^{(y)}, \quad \mathbf{Z}_{i,j}^{(y)} \sim \mathcal{N}(0,1),
\end{aligned}$$

where the elements of $\mathbf{Z}^{(x)}$ and $\mathbf{Z}^{(y)}$ are independent samples from the standard normal distribution. That is, $\mathbf{X}^{(2)}$ and $\mathbf{Y}^{(2)}$ are small perturbations of $\mathbf{X}^{(1)}$ and $\mathbf{Y}^{(1)}$, and belong to the same classes. Performing the Singular Value Decomposition of all matrices,

$$\begin{aligned} U^{(1)}S^{(1)}V^{(1)} &= \mathbf{X}^{(1)}, & U^{(2)}S^{(2)}V^{(2)} &= \mathbf{X}^{(2)}, \\ U^{(3)}S^{(3)}V^{(3)} &= \mathbf{Y}^{(1)}, & U^{(4)}S^{(4)}V^{(4)} &= \mathbf{Y}^{(2)}, \end{aligned}$$

and compute the elements of the kernel matrix as the Gaussian kernel of the first three left singular vectors,

$$K_{p,q} = \exp\left(- \sum_{k=1}^{3} \frac{||U_k^{(p)} - U_k^{(q)}||_F^2}{0.5^2} \right), \qquad p, q = 1, \ldots, 4.$$

Using the default economic `svd` function in MATLAB gives the following first elements of the singular vectors (up to 4 decimal digits):

k	1	2	3
$U_{1,k}^{(1)}$	-0.9406	0.3369	-0.0420
$U_{1,k}^{(2)}$	-0.9411	0.3361	0.0264

Note the different signs in the elements of the third singular vectors, leading to the following kernel matrix

$$K = \begin{bmatrix} 1.0000 & 0.0000 & 0.0856 & 0.0014 \\ 0.0000 & 1.0000 & 0.0000 & 0.0000 \\ 0.0856 & 0.0000 & 1.0000 & 0.0006 \\ 0.0014 & 0.0000 & 0.0006 & 1.0000 \end{bmatrix}.$$

In particular, $|K_{1,3}| > |K_{1,2}|$, indicating that $\mathbf{X}^{(1)}$ has more similarity to $\mathbf{Y}^{(1)}$ than $\mathbf{X}^{(2)}$, despite the fact that $\mathbf{Y}^{(1)}$ is a matrix of a different class. In contrast, fixing the signs of the singular vectors as described in Section 4.2.1 gives

$$K = \begin{bmatrix} 1.0000 & \mathbf{0.1110} & 0.0856 & 0.0014 \\ \mathbf{0.1110} & 1.0000 & \mathbf{0.1365} & \mathbf{0.0001} \\ 0.0856 & \mathbf{0.1365} & 1.0000 & 0.0006 \\ 0.0014 & \mathbf{0.0001} & 0.0006 & 1.0000 \end{bmatrix}$$

with the correct relation $|K_{1,2}| > |K_{1,3}|$, putting $\mathbf{X}^{(2)}$ into the same class with $\mathbf{X}^{(1)}$.

4.2.2 TT-CP Expansion

Despite the difficulties in *computing* a CP approximation, its simplicity makes the CP format a convenient and powerful tool for revealing hidden classification features in the input data. However, as long as the TT decomposition is available, it can be converted into the CP format suitable for the kernelized classification.

Proposition 4.3:
For a given TT decomposition (3.6),a CP decomposition can be obtained

$$\sum_{r_0,\dots,r_M} \mathfrak{C}^{(1)}_{r_0,i_1,r_1} \mathfrak{C}^{(2)}_{r_1,i_2,r_2} \cdots \mathfrak{C}^{(M)}_{r_{M-1},i_M,r_M} = \sum_{r=1}^{R} \hat{H}^{(1)}_{i_1,r} \hat{H}^{(2)}_{i_2,r} \cdots \hat{H}^{(M)}_{i_M,r}, \tag{4.2}$$

$\diamondsuit$

by merging the ranks $r_1, r_2, \dots r_M$ into one index $r = r_1 + (r_2 - 1)R_1 + \dots + (r_M - 1)\prod_{\ell=1}^{M-1} R_\ell$, $r = 1, \dots, R$, $R = R_1 \cdots R_M$, and introducing the CP factors

$$\hat{H}^{(m)}_{i_m,r} = \mathfrak{C}^{(m)}_{r_{m-1},i_m,r_m}, \quad m = 1, \dots, M.$$

The transformation to obtain the TT-CP expansion is free from any new computations and simply requires rearranging and replicating the original TT cores. Although this expansion is valid for arbitrary dimensions, the number of terms may increase massively for higher dimensions. However, since many experimental datasets are typically three or four-dimensional tensors, the TT-CP expansion is feasible.

It should be noted that the number of terms R in the CP decomposition equation (4.2) can be larger than the minimal CP rank of the exact CP decomposition of the given tensor. However, it is observed in numerical tests that the nonlinear kernel function is more sensitive to the features of the data rather than the number of CP terms per se, and therefore, the expansion (4.2) leads to better classification accuracy than attempting to compute an optimal CP approximation using an ALS method.

The main reason for using the TT-CP expansion is twofold. On the one hand, the DuSK kernel is better formulated for CP-like structures, and on the other hand, approximating low rank via CP-ALS is challenging. The TT, on the other hand, is more stable in an algorithmic way since the estimation of the best low-rank for TT depends on matrix SVD ranks. Therefore, combining the simplicity of CP with its compatibility to DuSK and the stability of TT leads to improved performance.

4.2.3 Norm Equilibration

The researchers tried using the TT-CP expansion with the CP kernel from [72] directly in preliminary experiments, but it did not result in better classification results. Later, they tried using the DuSK kernel [72], which introduces the same width parameter for all CP factors. However, this requires all CP factors to have identical (or at least close) magnitudes, which is not the case in the plain TT-SVD algorithm [133], where different TT cores have different norms. To address this issue, the TT-CP expansion is rescaled to ensure that the columns of the CP factors have equal norms. This ensures that the same kernel features are produced with the same width parameter, which is found to be a crucial ingredient for the successful TT-SVM classification.

Given a rank-r TT-CP decomposition $[\![\hat{H}^{(1)}, \hat{H}^{(2)}, \cdots, \hat{H}^{(M)}]\!]$, compute the total norm of each of the rank-1 tensors

$$n_r = \left\| \hat{H}_r^{(1)} \right\| \cdots \left\| \hat{H}_r^{(M)} \right\|, \tag{4.3}$$

and distribute this norm equally among the factors,

$$H_r^{(m)} := \frac{\hat{H}_r^{(m)}}{\left\| \hat{H}_r^{(m)} \right\|} \cdot n_r^{1/M}, \qquad m = 1, 2, \cdots, M. \tag{4.4}$$

4.2.4 Noise-aware Threshold and Rank Selection

In general, measurement or preprocessing noise can affect both computational and modeling aspects of data coming from real world applications. The TT ranks may increase due to the lack of any meaningful TT decomposition in a tensor of noise, and the classification may be spoiled if the noise is too large. However, the SVD can be used as a de-noising algorithm automatically, as the dominant singular vectors are often "smooth" and represent a useful signal, while the latter singular vectors capture primarily the noise. Therefore, computing the TT approximation with deliberately low TT ranks / large truncation threshold is actually beneficial. However, the TT rank should not be too low to ensure sufficient accuracy in approximating the features of the tensor. To evaluate the effectiveness of the model, cross-validation is used, which involves re-sampling the data into training-testing data sets. Since the precise magnitude of the noise is unknown, a k-fold cross-validation test is carried out (where $k = 5$) to find the optimal TT rank.

4.2.5 Reproducing Tensor Kernels (§3.4.4)

In this section, possible choices for tensor kernels are discussed, as an extension to the part 1 of *reproducing kernels* for tensors (§3.4.4). Firstly, brief recapitulation of some existing tensor kernels and then a new approach to using these kernels on the proposed low-rank TTCP approximation.

The Gaussian kernel

The natural idea of defining a kernel for tensorial data would be to extend the classical Gaussian kernel directly from vector to tensor format. That is, the computation can be given directly as follows,

$$K(\mathcal{X}, \mathcal{Y}) = \exp\left(\frac{-\|\mathcal{X} - \mathcal{Y}\|_F^2}{2g^2}\right), \tag{4.5}$$

The distance between the two input tensors is computed using the Frobenius norm, with g being the length scale of the kernel. Efficient computation of this norm is possible in each of the tensor formats introduced above and the leading terms of the complexity estimates are shown in Table 5.2. However, valuable information about the different tensor modes is lost by treating the tensor as a simple vector. It has been observed (e.g. in [71, 72]) that this straightforward idea yields suboptimal classification results and that introducing more sophisticated tensor kernels can lead to further improvement.

Dual Structure-preserving Kernel

The Dual Structure-preserving Kernel (**DuSK**) was introduced first in [71] for a rank-one tensor factorization and was later extended for the Kernelized CP decomposition in [72]. For given tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ and $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ and their corresponding CP decomposition given by $[\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(M)}]\!]$ and $[\![\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(M)}]\!]$, the formulation of the kernel approximation by DuSK is given as follows:

$$\begin{aligned} \langle \Psi(\mathcal{X}), \Psi(\mathcal{Y}) \rangle &= K(\mathcal{X}, \mathcal{Y}) \\ &= K\left(\sum_{i=1}^{R} \mathbf{a}_i^{(1)} \otimes \mathbf{a}_i^{(2)} \otimes \cdots \otimes \mathbf{a}_i^{(M)}, \sum_{j=1}^{R} \mathbf{b}_j^{(1)} \otimes \mathbf{b}_j^{(2)} \otimes \cdots \otimes \mathbf{b}_j^{(M)}\right) \\ &= \sum_{i,j=1}^{R} k(\mathbf{a}_i^{(1)}, \mathbf{b}_j^{(1)}) k(\mathbf{a}_i^{(2)}, \mathbf{b}_j^{(2)}) \cdots k(\mathbf{a}_i^{(M)}, \mathbf{b}_j^{(M)}), \end{aligned} \tag{4.6}$$

where,

$$k(\mathbf{a}, \mathbf{b}) = \exp\left(\frac{-\|\mathbf{a} - \mathbf{b}\|^2}{2g^2}\right). \tag{4.7}$$

In short, the individual factors of the CP decomposition are used to evaluate the kernel function $k(\cdot, \cdot)$. The motivation behind DuSK is simple: direct comparison of feature vectors in each mode is likely to improve classification, given that the CP decomposition is often unique (up to norm equilibration). However, since this kernel is specifically designed for CP tensors, converting tensors to the CP format first (see §4.2) is necessary.

4.2.6 Nonlinear Mapping

Now, with the homogenized TT-CP decompositions of the input tensors equipped, a nonlinear kernel function can be defined. The rationale behind DuSK proposed in [71, 72] is closely followed, and its generalized form for tensors of arbitrary dimension is expressed. It is assumed that the feature map function from the space of tensors to a *tensor product Reproducing Kernel Hilbert Space* [169] $\Psi : \mathbb{R}^{I_1} \times \cdots \times \mathbb{R}^{I_M} \mapsto \mathbb{F}$ consists of separate feature maps acting on different CP factors,

$$\Psi : \sum_{r=1}^{R} \mathbf{h}_r^{(1)} \otimes \mathbf{h}_r^{(2)} \otimes \cdots \otimes \mathbf{h}_r^{(M)} \mapsto \sum_{r=1}^{R} \phi(\mathbf{h}_r^{(1)}) \otimes \phi(\mathbf{h}_r^{(2)}) \otimes \cdots \otimes \phi(\mathbf{h}_r^{(M)}). \tag{4.8}$$

The fact that the data is given in the CP format allows for the exploitation of this format to aid in classification. However, the feature function $\phi(\mathbf{a})$ is to be implicitly defined through a kernel function. Similar to the standard SVM, the kernel trick is applied to (4.8) to obtain a practically computable kernel. Given CP approximations of two tensors $\mathcal{X} = [x_{i_1,\ldots,i_M}]$ and $\mathcal{Y} = [y_{i_1,\ldots,i_M}]$, the kernel can be computed,

$$x_{i_1,\ldots,i_M} \approx \sum_{r=1}^{R} h_{i_1,r}^{(1)} h_{i_2,r}^{(2)} \cdots h_{i_M,r}^{(M)}, \qquad y_{i_1,\ldots,i_M} \approx \sum_{r=1}^{R} p_{i_1,r}^{(1)} p_{i_2,r}^{(2)} \cdots p_{i_M,r}^{(M)},$$

we compute

$$\begin{aligned}
\langle \Psi(\mathcal{X}), \Psi(\mathcal{Y}) \rangle &= K(\mathcal{X}, \mathcal{Y}) \\
&= K\left(\sum_{r=1}^{R} \mathbf{h}_r^{(1)} \otimes \mathbf{h}_r^{(2)} \otimes \cdots \otimes \mathbf{h}_r^{(M)}, \sum_{r=1}^{R} \mathbf{p}_r^{(1)} \otimes \mathbf{p}_r^{(2)} \otimes \cdots \otimes \mathbf{p}_r^{(M)} \right), \\
&= \langle \Psi(\sum_{r=1}^{R} \mathbf{h}_r^{(1)} \otimes \mathbf{h}_r^{(2)} \otimes \cdots \otimes \mathbf{h}_r^{(M)}), \Psi(\sum_{r=1}^{R} \mathbf{p}_r^{(1)} \otimes \mathbf{p}_r^{(2)} \otimes \cdots \otimes \mathbf{p}_r^{(M)}) \rangle, \\
&= \sum_{i,j=1}^{R} \langle \phi(\mathbf{h}_i^{(1)}), \phi(\mathbf{p}_j^{(1)}) \rangle \langle \phi(\mathbf{h}_i^{(2)}), \phi(\mathbf{p}_j^{(2)}) \rangle \cdots \langle \phi(\mathbf{h}_i^{(M)}), \phi(\mathbf{p}_j^{(M)}) \rangle, \\
&= \sum_{i,j=1}^{R} k(\mathbf{h}_i^{(1)}, \mathbf{p}_j^{(1)}) k(\mathbf{h}_i^{(2)}, \mathbf{p}_j^{(2)}) \cdots k(\mathbf{h}_i^{(M)}, \mathbf{p}_j^{(M)}).
\end{aligned} \tag{4.9}$$

Each pair of the tensor input data, represented by its CP factors, computes this kernel approximation. Accurate learning requires choosing the width parameter $g > 0$ judiciously. The proposed model, named the Tensor Train Multi-way Multi-level Kernel (TT-MMK), is derived from the TT decomposition and aims to extract optimal low-rank features while constructing a more accurate and efficient classification model. The kernel values (4.9) are plugged into the STM optimizer (3.24) to complete the algorithm. Algorithm 4.2 summarizes the overall idea.

Algorithm 4.2: TT-CP approximation of the STM Kernel

```
    **Input:** data $\{\mathcal{X}_n\}_{n=1}^{N}\mathbb{R}^{I_1\times I_2\times\ldots\times I_M}$, TT-rank $R$.
    **Output:** Kernel matrix approximation $\left[K(\mathcal{X}_u,\mathcal{X}_v)\right]\in\mathbb{R}^{N\times N}$
    for $n=1$ to $N$ do
        Compute TT approximation $\mathcal{X}_n \cong \langle\langle \mathcal{C}^{(1,n)},\mathcal{C}^{(2,n)},\cdots,\mathcal{C}^{(M,n)}\rangle\rangle$ by Algorithm 4.1.
        Compute TT-CP expansion
        $[\![\mathbf{H}^{(1,n)},\mathbf{H}^{(2,n)},\cdots,\mathbf{H}^{(M,n)}]\!] = \langle\langle \mathcal{C}^{(1,n)},\mathcal{C}^{(2,n)},\cdots,\mathcal{C}^{(M,n)}\rangle\rangle$ as (4.2) with equilibrated norms as (4.4).
    end for
    for $u,v=1$ to $N$ do
        $K(\mathcal{X}_u,\mathcal{X}_v)\approx\sum_{i,j=1}^{R}k(\mathbf{h}_i^{(1,u)},\mathbf{h}_j^{(1,v)})k(\mathbf{h}_i^{(2,u)},\mathbf{h}_j^{(2,v)})\cdots k(\mathbf{h}_i^{(M,u)},\mathbf{h}_j^{(M,v)})$ as (4.9).
    end for
```

4.3 Numerical Tests

- **Experimental Settings**
  All numerical experiments have been done in `MATLAB 2016b`. In the first step, the computation in the TT format of an input tensor uses the `TT-Toolbox`[1], where the function `@tt_tensor/round.m` has modified to enforce the uniqueness enforcing TT-SVD (§4.2.1). Moreover, the implementation of the TT-CP conversion, together with the norm equilibration is concluded. For the training of the TT-MMK model, the `svmtrain` function available in the `LIBSVM`[2] library is accustomed. All the experiments run on a machine equipped with Ubuntu release 16.04.6 LTS 64-bit, 7.7 GiB of memory, and an Intel Core i5-6600 CPU @ 3.30GHz×4 CPU. The codes are available publicly on `GitHub`[3].

- **Parameter Tuning**
  The entire TT-SVM model depends on three parameters. First, to simplify the selection of TT ranks, take all TT ranks equal to the same value $R\in\{1,2,\ldots 10\}$. Another parameter is the width of the Gaussian Kernel g. Finally, the third parameter is a trade-off constant C for the KSTM optimization technique (3.24). Both g and C are chosen from $\{2^{-8},2^{-7},\ldots,2^{7},2^{8}\}$. For tuning R,g and C to the best classification accuracy, use the *k-fold cross validation* with $k=5$. Along with this, all computations are reapeated 20 times and computation of statistics such as average, standard deviation, and numerical quantiles have been done over these runs. This ensures a confident and reproducible comparison of different techniques.

[1] https://github.com/oseledets/TT-Toolbox
[2] https://www.csie.ntu.edu.tw/~cjlin/libsvm/
[3] https://github.com/mpimd-csc/Structure-preserving_STTM

4.3.1 Data Collection

1. Resting-state fMRI Datasets
    - **Alzheimer Disease (ADNI):** The ADNI[4] stands for Alzheimer Disease Neuroimaging Initiative. It contains the resting state fMRI images of 33 subjects. The data set was collected from the authors of the paper [72]. The images belong to either Mild Cognitive Impairment (MCI) with Alzheimer Disease (AD), or normal controls. Each image is a tensor of size $61 \times 73 \times 61$, containing 271633 elements in total. The AD+MCI images are labeled with -1, and the normal control images are labeled with 1. Preprocessing of the data sets is explained in [71].
    - **Attention Deficit Hyperactivity Disorder (ADHD):** The ADHD data set is collected from the ADHD-200 global competition data set[5]. It is a publicly available preprocessed fMRI data set from eight different institutes, collected at one place. The original data set is unbalanced, so 200 subjects have chosen randomly, ensuring that 100 of them are ADHD patients (assigned the classification label -1) and the 100 other subjects are healthy (denoted with label 1). Each of the 200 resting state fMRI samples contains $49 \times 58 \times 47 = 133574$ voxels.

    Note: It is mentioned in the MMK paper [72] that the exact indices of the collected data are not mentioned, which means that the collected dataset might not be exactly the same. As a result, accuracy percentages cannot be directly compared to the MMK paper.

2. Hyperspectral Image (HSI) Datasets: The `mat` file for both the datasets and their corresponding labels[6] have taken. The following datasets have three dimensional tensor structure of different sizes, where each tensor data point represents a pixel value. Therefore, for the presented experiment, a patch of size 5×5 for two different pixel values, in order to get a binary classification dataset have taken.
    - **Indian Pines:** The HSI images were collected via the Aviris Sensor[7] over the Indian Pines test site. The size of the dataset is 145×145 pixels over 224 spectral values. Hence, the size of the tensor data is $145 \times 145 \times 224$. The collected `mat` for presented experiment has reduced band size 200. This excludes bands covering the region of water absorption: [104-108], [150-163]. The original dataset contains 16 different labels to identify different corps and

[4]http://adni.loni.usc.edu/
[5]http://neurobureau.projects.nitrc.org/ADHD200/Data.html
[6]http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes
[7]https://aviris.jpl.nasa.gov/

living areas. Only 50 datapoints for each of the two labels 11 (Soybean-mintill) and 7 (Grass-pasture-mowed) have taken.

- **Salinas:** This HSI images data was collected by 224 band Aviris Sensor over Salinas valley, California. Similar to Indian Pines, in this case, samples for two GroundTruths also have collected, namely 9 (Soil-vinyard-develop) and 15 (Vinyard-untrained) each with 50 datapoints. The size of the dataset is 512×217 pixels over 224 spectral values. Hence, the size of the tensor data is $512 \times 217 \times 224$.

Note: After the publication of this work, the same experiments were also conducted on a Computed Tomography (CT) dataset. The `.mat` file for the dataset was obtained from **MosMedData**[a]. This dataset consists of anonymised human lung CT scans with COVID-19-related findings, as well as scans without such findings. It comprises 1000 data points corresponding to four different conditions related to COVID-19. The CT scans were acquired between March 1, 2020, and April 25, 2020, and were provided by municipal hospitals in Moscow, Russia. The classification accuracy achieved for this data is nearly on par with the state-of-the-art. This also serves as a benchmark for the model since existing state-of-the-art models are computationally expensive, especially considering the optimization of DL-based techniques which tends to be costly in general.

[a] https://mosmed.ai/en/

4.3.2 Influence of Individual Algorithmic Steps

In the first test investigation of the impact of each individual transformation of the TT decomposition, outlined in §4.2.1–§4.2.3. Firstly, a counterpart of the DuSK kernel (4.9) directly to the initial TT approximation of the data tensors can be applied. Given TT decompositions

$$x_{i_1,i_2,i_3} = \sum_{r_1,r_2=1}^{R_1,R_2} \mathcal{C}^{(1)}_{i_1,r_1}\mathcal{C}^{(2)}_{r_1,i_2,r_2}\mathcal{C}^{(3)}_{r_2,i_3} \quad \text{and} \quad y_{i_1,i_2,i_3} = \sum_{t_1,t_2=1}^{R_1,R_2} \mathcal{S}^{(1)}_{i_1,t_1}\mathcal{S}^{(2)}_{t_1,i_2,t_2}\mathcal{S}^{(3)}_{t_2,i_3},$$

computation of a separable kernel similarly to (4.9) via

$$k(\mathcal{X},\mathcal{Y}) = \sum_{r_1,t_1=1}^{R_1}\sum_{r_2,t_2=1}^{R_2} k(\mathcal{C}^{(1)}_{r_1},\mathcal{S}^{(1)}_{t_1})k(\mathcal{C}^{(2)}_{r_1,r_2},\mathcal{S}^{(2)}_{t_1,t_2})k(\mathcal{C}^{(3)}_{r_2},\mathcal{S}^{(3)}_{t_2}), \tag{4.10}$$

has performed and a similar approach was also proposed recently in [32]. Comparison of two versions of this TT-DuSK kernel: “uTT” and “TT”, which correspond to the TT-SVD algorithm with and without uniqueness enforcement (Algorithm 4.1), respectively, are executed.

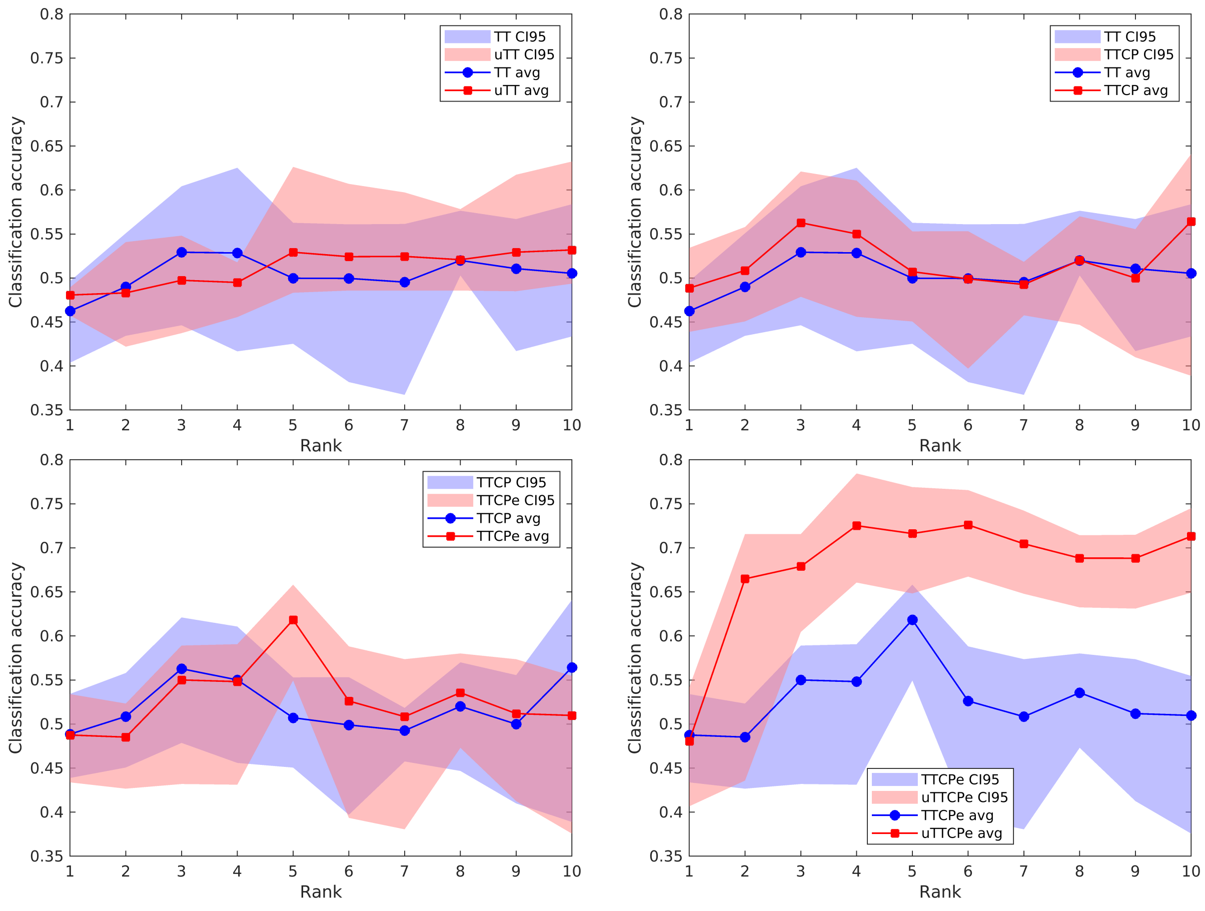


Figure 4.1: Test classification accuracy of different versions of the TT-MMK algorithm: “TT” vs “uTT” (left top), “TT” vs “TTCP” (right top), “TTCP” vs “TTCPe” (left bottom), and “TTCPe” vs the final algorithm “uTTCPe” (right bottom) for the ADNI dataset. Lines denote averages, and shaded areas denote 95% confidence intervals over 20 runs.

Next, the expansion of the TT format without uniqueness enforcement into the CP format as described in Section 4.2.2 and Eq. (4.2) is done, but without equilibrating the norms, and apply the DuSK kernel (4.9). The corresponding classifier is called “TTCP”. Note that for a *given* TT decomposition and its exact TT-CP expansion the values of the kernels (4.10) and (4.9) coincide. However, different runs of the classification algorithm may produce different signs of the singular vectors in the TT-SVD algorithm, different initial guesses in the SVM, and different splitting of the data into training and test sets during the cross validation.

Finally, construct the norms of the CP factors equilibrated as described in Section 4.2.3 and Eq. (4.4), followed by the DuSK kernel (4.9). Depending on using or not using the uniqueness enforcement during the initial TT computation, the corresponding classifiers are called "uTTCPe" and "TTCPe", respectively.

In Figure 4.1 pairwise comparision of these versions of the algorithm to ensure clarity of overlapping confidence intervals has executed. Top left plot of Figure 4.1 shows that the direct TT counterpart of the DuSK kernel (4.10) gives a poor test accuracy, although the uniqueness enforcement can improve it slightly for higher ranks.

Next, comparison of the TT and TTCP DuSK kernels (top right of Figure 4.1) is shown. This is merely a sanity check, since deterministic algorithms would give the same results. Correspondingly, randomized algorithms give overlapping realisations of the test accuracy.

In the bottom left plot of Figure 4.1 compares TTCP DuSK kernels with and without norm equilibration, but without uniqueness of the underlying TT decomposition. We see that the norm equilibration provides a noticeably higher test accuracy at rank 5. Nevertheless, the mean accuracy is still below 65%.

Finally, upon plugging in both the uniqueness-enforced TT format and its norm-equilibrated TTCP expansion (Figure 4.1, bottom right), the test accuracy of the TT-MMK classifier exceeds 70%, with the best average accuracy of 73% achieved for rank 4. This demonstrates the significance of all steps in the TT-MMK classifier.

Note: the DuSK kernel of the TTCP **with** norm equilibration is **not** equivalent to the TT-DuSK kernel though, since the norm of a TTCP factor,

$$n_r = \left\|\hat{H}_r^{(1)}\right\|\left\|\hat{H}_r^{(2)}\right\|\left\|\hat{H}_r^{(3)}\right\| = \left\|\mathfrak{C}_{r_1}^{(1)}\right\|\left\|\mathfrak{C}_{r_1,r_2}^{(2)}\right\|\left\|\mathfrak{C}_{r_2}^{(3)}\right\|, \quad r = r_1 + (r_2-1)R_1,$$

depends on **both** TT rank indices r_1 and r_2, and the norm-equilibrated TTCP factors, for example,

$$H_r^{(1)} := \frac{\mathfrak{C}_{r_1}^{(1)}}{\left\|\mathfrak{C}_{r_1}^{(1)}\right\|} \cdot n_r^{1/3},$$

depend on both r_1 and r_2 too, and are not reducible back to the TT format. However, since this information concerns a competing method TT-DuSK (and not the main contribution), it is not included at this stage of research.

4.3.3 Comparison to Other Methods

Next, the comparison of the classification accuracy of the final proposed TT-MMK method ("uTTCPe") with the accuracy of the following existing approaches is given.

SVM: the standard SVM with Gaussian Kernel. This is the most used optimization method for vector input based on the maximum margin technique. The

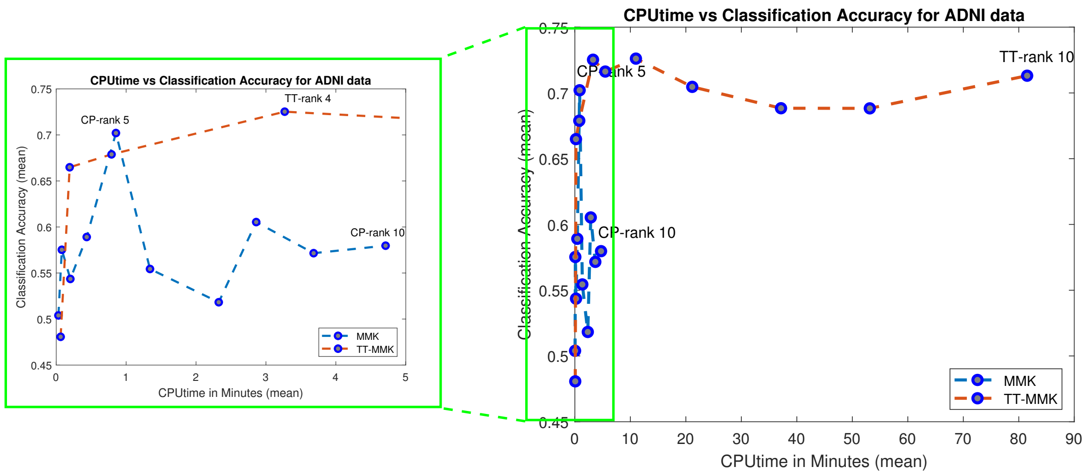


Figure 4.2: CPU time vs classification accuracy for ADNI data with truncation rank from 1 to 10.

objective function mentioned in (2.18) has been optimized using LIBSVM using the *kernel trick* [155].

STuM: The Support Tucker Machine (STuM) [96] uses the Tucker decomposition. The weight parameters of the SVM are computed for optimization into Tucker factorization form.

DuSK: The idea of DuSK [71] is based on defining the kernel approximation for the rank-one decomposition. This is one of the first methods in this direction. [71] solves the STM (3.24), with kernel approximation using the DuSK format similar to (4.9).

MMK: This method is an extension of DuSK to the KCP input. The latter is the CP format with factor matrices (3.3) projected onto a covariance or random matrix [72]. The proposed work framework used the original DuSK and MMK codes provided by the authors of the research article [72].

Improved MMK: This is actually a simplified MMK, where the projection of the CP onto the KCP is omitted (the covariance/random matrices are replaced by the identity matrices).

KSTTM: This method is applied directly on the TT-cores with two different types of kernel computations, namely K-STTM prod and K-STTM sum [32]. In the experiments, this method ran out of memory for the ADHD dataset during the computation of the kernel matrix.

TT-MMK: This is the proposed method.

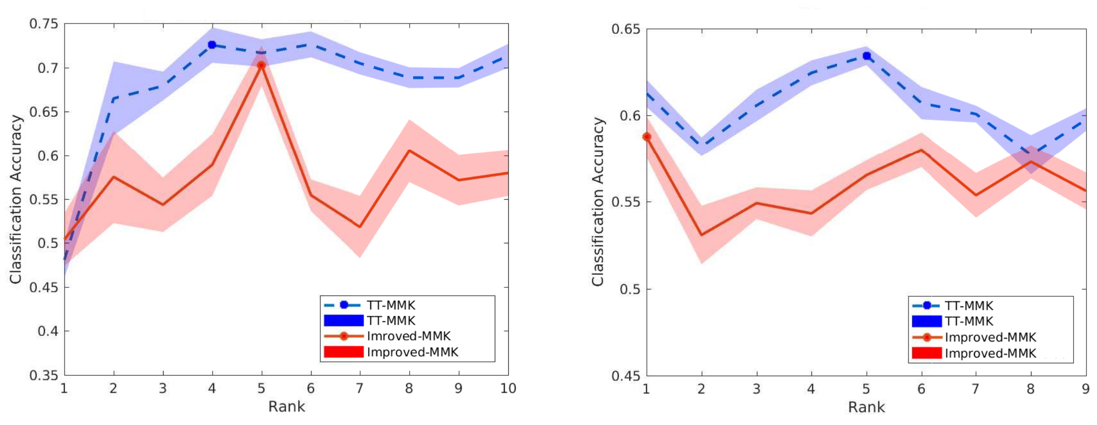


Figure 4.3: Classification accuracy, average (lines) $\pm$ one standard deviation (shaded areas) over 20 runs. Left: ADNI dataset. Right: ADHD dataset.

The key observations from the results shown in Table 4.1 and Figure 4.3 are as follows.

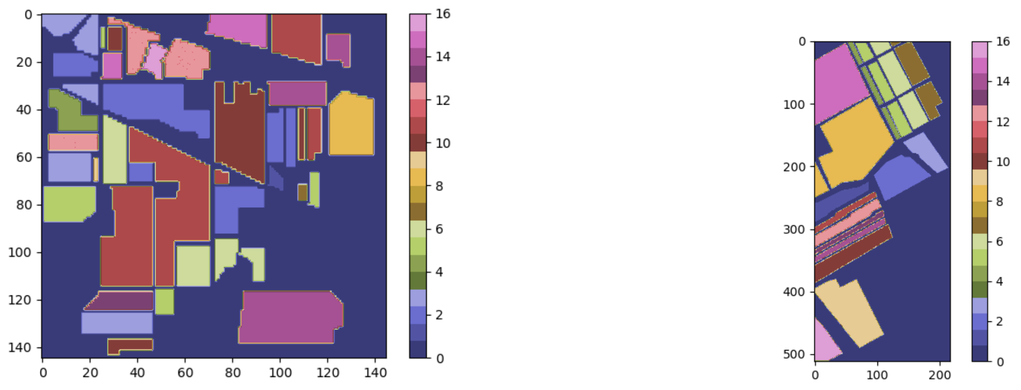
(In)sensitivity to the TT Rank Selection: Figures 4.3 and 4.1 (bottom right) show that the proposed method gives almost the same accuracy for different TT ranks. For some samples, even the TT rank of 2 gives a good classification. Note that this is not the case for MMK, which requires a careful selection of the CP rank.

Computational Robustness: while the CP decomposition can be computed using only iterative methods in general, all steps of the kernel computation in TT-MMK are "direct" in a sense that they require a fixed number of linear algebra operations, such as the SVD and matrix products.

Computational Complexity: approximating the full tensor in the TT format has the same $\mathcal{O}(n^{M+1})$ cost as the Tucker and CP decompositions. All remaining operations with the factors scale linearly in the dimension M and mode sizes, and polynomially in the ranks.

Classification Accuracy: the proposed method gives the best average classification accuracy compared to five other state of the art techniques.

Running Time: The time taken by the MMK and TT-MMK experiments for the ADNI data with $C, g \in \left[2^{-8}, 2^{-7}, \ldots, 2^{7}, 2^{8}\right]$ are $\approx$ 17 minutes and $\approx$ 3.5 hours, respectively, for the entire range of $R \in \{1, 2, \ldots 10\}$. However, if Figure 4.2 looked closely at, the TT-MMK achieves nearly the best accuracy for any rank starting from 2. This means that even though the TT-MMK process takes more time than MMK for the same TT ranks due to the higher CP rank induced by (4.2), the higher test accuracy is a reasonable reward for the larger CPU time. In particular,



(a) GroundTruth of Indian Pines dataset (b) GroundTruth of Salinas dataset

Figure 4.4: Hyperspectral images with different labels

if the range of R to $\{1,\ldots,5\}$ is reduced, which is sufficient to discover the best classifiers for both methods, the timings are closer: MMK needs about 1 minute for its best variant (CP rank 5), while the TT rank 4 solution of a better accuracy is computed in about 3 minutes. This slightly higher runtime is acceptable for a better classification accuracy.

Reproducibility: Figure 4.3 shows that the MMK method has a higher empirical standard deviation (0.05 for the ADNI dataset, 0.02 for the ADHD dataset) compared to the TT-MMK method with standard deviations of 0.03 and 0.01 for the ADNI and ADHD datasets, respectively. This shows that TT-MMK is more predictable.

Generalization: Top accuracy (see Table 4.1) in datasets from two different areas (fMRI and HSI) shows that the method is suitable for a wide range of binary tensor classification problems.

4.4 Chapter Summary

The STM model presented in this chapter, proposes a new low-rank approximation method, called the TT-CP decomposition, which is described in section §4.2. The approach involves three main steps. Firstly, the singular vectors are enforced with uniqueness during TT-SVD computation (§4.2.1). Secondly, the TT format is converted exactly to the CP format. Finally, to ensure the robustness of the model with respect to rank-truncation, norm equilibration is performed (§4.2.3).

In addition, section §4.2.4 explains the necessity and importance of applying TT-

Table 4.1: Average classification accuracy in percentage for different methods and data sets

Methods	ADNI	ADHD	Indian Pines	Salinas
SVM	49	52	46	47
STuM	51	54	57	74
DuSK	55	57	60	92
MMK	69	58	93	98
Improved MMK	70	58	94	98
K-STTM prod	60	-	76	**100**
K-STTM sum	60	-	73	**100**
TT-MMK	**73**	**63**	**99**	99

rounding (3.4) when computing the TT decomposition with TT-SVD (3.3). Furthermore, section §4.2.5 is a sequel to the previous chapter's §3.4.4, which introduces two new reproducing kernels and a way of computing factor kernels over the CP format. The use of these kernels for the low-rank TT-CP decomposition is explained in section §4.2.6.

The newly proposed TT-CP approximation and corresponding kernel matrix computation lead to better solutions for the KSTM model. Numerical tests on real-world data, such as fMRI and hyperspectral images, are presented in section §4.3. A crucial subsection of this part is the influence of individual algorithmic steps on the presented datasets. Furthermore, to compare two consecutive steps of the final algorithm at a time, including the confidence intervals over the randomization of signs in the singular vectors (for the TT approximation without uniqueness enforcement), initial guesses in the SVM, and data partitioning in the cross-validation, statistical importance of the results can be assessed. Section §4.3.2 authenticates the proposed research work and positions it as a state-of-the-art model at the time.

CHAPTER 5

A WEIGHTED SUBSPACE EXPONENTIAL KERNEL FOR SUPPORT TENSOR MACHINES

5.1 Introduction . . . 71
5.1.1 Related Work . . . 72
5.1.2 Problem Formulation . . . 73
5.1.3 Converting TT into CP . . . 74
5.1.4 Converting Tucker to CP . . . 74
5.2 Reproducing Tensor Kernels . . . 75
5.2.1 The Subspace Kernel . . . 76
5.2.2 The Weighted Subspace Exponential Kernel . . . 77
5.2.3 Computational Complexity of the Different Kernels . . . 78
5.3 A Numerical Study on Synthetic Data . . . 80
5.3.1 Interpreting CP and Tucker . . . 80
5.3.2 A Synthetic Experiment . . . 81
5.3.3 Results and Interpretation . . . 82
5.4 Classification of Real Datasets . . . 84
5.4.1 Numerical results . . . 85
5.5 Chapter Summary . . . 89

Contents

5.1 Introduction

This chapter builds upon the problem discussed in the previous Chapter 4 and focuses on solving the problem concerned in §4.1.1. As mentioned in §3.5, classification methods that rely on low-rank approximations, such as CP (§3.3.1), Tucker (§3.3.2), or TT (§3.3.3), tend to perform better since they preserve influential information of the tensor input data, resulting in better features. However, it is not entirely clear why kernels based on low-rank decompositions are effective for classification. This chapter dives into both the concerns, first the exploration of better suited LRTF and affiliated kernel.

5.1.1 Related Work

Motivation : The main goal of this work is to create a new way of transforming data that captures nonlinear relationships. Recently, approximating tensors using low-rank decompositions has gained a lot of attention in scientific computing [40, 95, 37, 111]. Early attempts on the use of HOSVD/MLSVD [167, 168, 209] applied kernel methods to tensor data, which involves flattening the tensor data. However, this approach leads to high-dimensional vectors and matrices, making them prone to overfitting and potentially losing important tensor structure, which calls for alternative methods.

The well-defined problem of Tucker approximation enables us to compute an almost optimal Tucker approximation using singular value decompositions (SVD) [96] or relaxed orthogonality has been used for enhancement [139]. Efforts to extract key aspects of tensorial data and design corresponding kernels include the Tucker *subspace kernel* [167]. The factor match score, discussed in [4], provides a consistent way to compare tensor decomposition feature vectors. Further understanding of the KCP approach [73] comes through a kernelized Tucker model, influenced by [169]. The Tensor Train (TT) decomposition, similar to Tucker but scalable to higher dimensions like the CP format, was extended into the TT format of DuSK (MMK) [31].

However, the exact reason for the success of low-rank decomposition-based kernels in classification remains unclear. Moreover, since tensor decompositions introduce nonlinear parameters, their representation might lack uniqueness. For example, Tucker and TT decompositions are not affected by factor rotation and scaling, and they can be interconverted with changes in rank. In Chapter 4, a method is introduced that significantly improves classification accuracy by removing redundancy in rotation, scaling, and TT to CP conversion in the TT-MMK method [98]. As the TT-decomposition is also a tree-based tensor format, the advantages of the Tucker format extend to the TT format as well. However, the complexity of the TT format increases quadratically with the ranks, while the Tucker format's complexity grows exponentially with the Tucker ranks. Yet, this reduction in complexity comes at a trade-off with interpretability: the meaning of the TT components is not as straightforward.

Utilizing Low-rank Approximation and Kernel Methods : Although most of the proposed works based on HOSVD focus mainly on the factors of Tucker/HOSVD, the data could contain crucial features not only in the Tucker subspaces but also in the Tucker core. Addressing this challenge is the focus of Chapter 5, which aims to fill this gap by introducing a novel kernel that demonstrates high robustness in identifying where classification information resides within the tensor. Before delving into the details of the contributions in the subsequent chapters, a following section provides a summary and explanation of some important kernels introduced in the previous works mentioned.

This chapter underscores the importance of the TT to CP decomposition for DuSK, taking a slightly different approach in selecting the type of low-rank approximation. While examining existing methods, this chapter addresses challenges and issues that arise and presents effective solutions to tackle them.

Main Novelty: This chapter predominantly centers on the innovative kernel, showcasing remarkable robustness in dealing with the positioning of classification information within the tensor. The primary focal point of this chapter involves introducing a novel approach to express the Tucker (HOSVD) decomposition, featuring weighted factors. These weighted factors play a pivotal role in constructing a fresh *kernel* tailored for support tensor machines. This newly devised kernel not only enables swift computations but also incorporates a weighting mechanism that considers both Tucker factors and the core, facilitating the creation of a nonlinear decision boundary.

Through comprehensive numerical experiments conducted on synthetic data, where class assignment is based on either Tucker factors (*leaf*) or the core, it has been empirically verified that the novel kernel yields accurate classification outcomes in nearly all scenarios, akin to the pre-existing Tucker-based kernels. Additionally, while maintaining comparable or even improved levels of classification accuracy, the proposed kernel exhibits a higher level of versatility concerning rank truncation of LRTF, all achieved with significantly reduced CPU time.

In summary, the method described in this chapter shows that the new kernel consistently performs better than state-of-the-art methods in two important aspects: it almost always achieves higher classification accuracy, and it is more efficient in terms of CPU time in all scenarios. This superiority holds true even when dealing with real-world datasets, highlighting the practical benefits of this approach.

5.1.2 Problem Formulation

This chapter provides a method for solving the following problem,

$$\begin{aligned} \max_{\alpha_1,\ldots,\alpha_N} \quad & \sum_{i=1}^{N} \boldsymbol{\alpha}_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \boldsymbol{\alpha}_i\boldsymbol{\alpha}_j y_i y_j K(\boldsymbol{\mathcal{X}}_i, \boldsymbol{\mathcal{X}}_j) \\ \text{subject to} \quad & 0 \leqslant \boldsymbol{\alpha}_i \leqslant C, \quad \sum_{i=1}^{N} \boldsymbol{\alpha}_i y_i = 0 \;\; \forall i. \end{aligned} \tag{5.1}$$

The core problem formulation remains consistent, with the divergence being influenced by both the selection of Low-Rank Tensor Factorization (LRTF) and the chosen Kernel. The specific gaps highlighted in the preceding section, as elaborated in §5.1, are effectively addressed and resolved in the subsequent content of this chapter.

The mathematical formulation of the two issues at hand is elaborated as follows:

- Choosing the LRTF of $\mathcal{X}_i, \quad \forall i \in < N >$
- Choosing the Kernel $K(\mathcal{X}_i, \mathcal{X}_j)$

The chapter chapter 4 dig into the TT-CP format as LRTF (§4.2). The direct implication of the TT to CP conversion for HOSVD/Tucker would be Tucker-CP format. Further, these two conversions are discussed, and their use as LRTF in (5.1). In addition, the selection of kernel and further exploration of new development is also part of this chapter.

5.1.3 Converting TT into CP

Evident from (3.3), (3.5), and (3.6), is the ability to transform a tensor from Tucker or TT format into the CP format through summation over all Tucker ranks $R_1, \ldots, R_M$, or all TT-ranks $r_0, \ldots, r_M$, respectively. However, it's important to note that this conversion doesn't yield a minimal CP decomposition; rather, it results in a CP representation of the tensor. Notably, none of these decomposition formats is inherently unique, as CP permits the rescaling of factor matrix columns, and Tucker and TT can incorporate identity matrices $I = QQ^{-1}$ between modes without altering the tensor's essence.

In a prior study (Chapter 4), the challenge of converting TT to CP in a meaningful manner was resolved by imposing uniqueness in the TT-SVD, followed by conversion to CP. Column norm equilibration was also employed to eliminate ambiguity in the CP representation. This method can also be extended to the conversion of Tucker into CP. The Higher Order SVD (HOSVD) achieves uniqueness by fixing the sign of each singular vector. This entails identifying the element with the greatest absolute value and rendering it positive. Analogous to Chapter 4, one can show that under the assumption of simple singular values, this results in unique Tucker factors. These factors do not vary too much if the noise is moderate. In the case of the Tucker tensor, conversion to CP involves summing across all ranks $R_1, \ldots, R_M$, and subsequent norm equilibration ensures the even distribution of the scalar $g_{r_1 r_2 \ldots r_M}$ in (3.5) among all factors.

5.1.4 Converting Tucker to CP

Viewing the CP decomposition of a tensor through a different lens, it can be perceived as a specialized instance of the Tucker decomposition featuring a diagonal core tensor (Algorithm 5.1). Specifically, if the matrix ranks of the factor matrices $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(M)}$ are constrained to be less than a given threshold R, it's feasible to construct a Tucker decomposition with Tucker ranks $R_m < R$ for each $m = 1, \ldots, M$. In such scenarios, the core tensor might not necessarily be diagonal; however, it could potentially include numerous zero entries when $R < R_1 + \cdots + R_M$.

Irrespective of the chosen decomposition method, whether CP or Tucker, the factor matrix $\mathbf{A}^{(m)}$ in CP format or the leaf $\mathbf{U}^{(m)}$ in Tucker format, serves to span the subspace of $\mathbb{R}^{I_m}$, encompassing the data pertaining to the m-th mode. This subspace is inherently defined by the columns of the m-mode matricization of the entire tensor.

The memory need for storage for individual and conversionsn(§5.1.3, §5.1.4) among LRTF are mentioned in the Table 5.1.

Algorithm 5.1: Tucker to CP.

```
Input: $M$th-order tensor $\mathcal{X}' = [\![\mathcal{G}; \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \dots, \mathbf{U}^{(M)}]\!]$ is in the Tucker format with
Tucker ranks $[R_1, R_2, \cdots, R_M] = R_{Tucker}$
Output: CP format $\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(M)}$
— Normalization —
$\mathcal{X}$= hosvd(full($\mathcal{X}$), 0, $R_{Tucker}$)
for $m = 1$ to $M$ do
  for $r_m = 1$ to $R_m$ do
    $\mathbf{A}^{(m)}(:, sub2ind(R_{Tucker}, i, j, k)) = \mathbf{U}^{(m)}(:, r_m)$
    $\mathbf{\Lambda}(:, sub2ind(R_{Tucker}, i, j, k)) = \mathcal{G}(r_1, r_2, \cdots, r_m)$
  end for
end for
return  CP decomposition with $\mathbf{A}^{(m)}$ factor matrices
```

Table 5.1: Memory Storage.

1. Full tensor format (ktensor)	$\mathcal{O}(I^M)$
2. CP	$\mathcal{O}(MIR_{CP})$
3. Tucker	$\mathcal{O}(MIR_{Tucker} + R_{Tucker}^M)$
4. TT/MPS	$\mathcal{O}(MIR_{TT}^2)$
5. TT-to-Tucker	$\mathcal{O}(MR^2(I) + MR^3)$
6. Tucker-to-CP	$\mathcal{O}(MR_{Tucker}(I) + MR_{Tucker}R_{CP})$ [52]

5.2 Reproducing Tensor Kernels

This section is third sequel of mentioned §3.4.4, and §4.2.5 from preceding chapters. As the focus is on tucker decomposition, therefore, an existing kernel based on tucker factors is introduced further. Additionally, significant research work proposed in this chapter is encompassed in this section.

5.2.1 The Subspace Kernel

So far, we have understood that the naive way of defining the distance between two tensors ($\|\mathcal{X} - \mathcal{Y}\|$), using the so-called Euclidean metric, is suboptimal for capturing the topology of the input patterns [98]. Therefore, we explaine a factor-based *Subspace kernel* [167] that depends on the chordal distance on the Grassmannian manifolds of matrix unfolding. This kernel relies on MLSVD and possesses an invariance property.

Instead of comparing the feature vectors in the CP decomposition, one can use a similar approach for the Tucker format. Here, the feature vectors are stored in the leaf-components $\mathbf{U}^{(m)}$ and they span the column spaces of the m-th matricizations. However, using only these components for the classification would mean that all information in the core component $\mathcal{G}$ would be ignored, as discussed in detail in Sec. 5.3.2. For this reason, the authors of [167] have chosen to compare the *subspaces spanned by singular vectors of tensor unfoldings* instead: Let $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ denote an M^{th}-order tensor, when the SVD is applied on the mode-m unfolding as $\mathcal{X}_{(m)} = \mathbf{U}_{\mathcal{X}}^{(m)} \Sigma_{\mathcal{X}}^{(m)} \mathbf{V}_{\mathcal{X}}^{(m)\mathsf{T}}$ and similarly for $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$, $\mathcal{Y}_{(m)} = \mathbf{U}_{\mathcal{Y}}^{(m)} \Sigma_{\mathcal{Y}}^{(m)} \mathbf{V}_{\mathcal{Y}}^{(m)\mathsf{T}}$, then the Chordal distance-based kernel is defined as,

$$K(\mathcal{X}, \mathcal{Y}) = \prod_{m=1}^{M} \exp\Big(- \frac{1}{2g^2} \Big\| \mathbf{V}_{\mathcal{X}}^{(m)} \mathbf{V}_{\mathcal{X}}^{(m)+} - \mathbf{V}_{\mathcal{Y}}^{(m)} \mathbf{V}_{\mathcal{Y}}^{(m)+} \Big\|_F^2 \Big) \tag{5.2}$$

Note that here, we use the *projections* onto these subspaces, given by the Moore-Penrose-Pseudoinverse $\mathbf{V}_{\mathcal{X}}^{(m)+}$. Since the components $\mathbf{V}_{\mathcal{X}}^{(m)}$ are (usually) orthogonal, this can simply be replaced by the transpose. Furthermore, for computational reasons, it makes sense to only compute the mixed term $\mathbf{V}_{\mathcal{X}}^{(m)} \mathbf{V}_{\mathcal{Y}}^{(m)\mathsf{T}}$ in each summand. This kernel provides us with rotation and reflection invariance for elements on the Grassmann manifold.

To understand the motivation behind the invention of this kernel, as explained in [167], a glimpse of the proposed work is provided here to continue the flow of reading. Firstly, we consider the distance between the unfoldings of two tensors. This approach treats tensors as collections of linear subspaces obtained from their respective matricizations. Hence, the factor kernel for tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$, and any g is defined as,

$$k^m(\mathcal{X}, \mathcal{Y}) := \exp\left(- \frac{1}{2g^2} d(\mathcal{X}_{(m)}, \mathcal{Y}_{(m)})^2 \right), \tag{5.3}$$

where, d presents a cordial distance between row-spaces on the *Grassmann manifold*.

Arguably, in [167], it is mentioned that the presented factor kernel must be positive definite (also discussed in Chapter 3). The distance d should be meaningful so that the product kernel k (as defined in (5.3)) can be equivalently restated as the RBF kernel . This closely resembles (4.5) but differs in that the Euclidean norm is replaced by a non-Euclidean distance function defined as:

$$d(\mathcal{X}, \mathcal{Y}) = \sqrt{\sum_{m \in <M>} d(\mathcal{X}_{(m)}, \mathcal{Y}_{(m)})^2}. \tag{5.4}$$

As any unitarily invariant metric on a *Grassmannian manifold* is intrinsically linked to the concept of *principal angles.* The principal angles between row spaces can be definde recursively and among many principal angle based measured distances, the projection Frobenius norm (chordal distance [46]) with a one-to-one relation between supbspace and its orthogonal projection. As in the case of MLSVD, the orthogonal projection of subspaces is given by $\mathbf{V}_{\mathcal{Y}}^{(m)}\mathbf{V}_{\mathcal{Y}}^{(m)+}$, hence the clarification of the (5.2).

5.2.2 The Weighted Subspace Exponential Kernel

One main disadvantage of the subspace kernel is that the matrices $\mathbf{V}_{\mathcal{X}}^{(m)}$ containing the row-space information of each unfolding can be very large: Especially for higher-orders M, they suffer from the *curse of dimensionality* as their size grows with order $\mathcal{O}(M-1)$. Additionally, much of the information in these components is redundant and the row-spaces of the unfoldings are highly correlated. As already discussed, directly replacing the projections onto the row-space by those onto the column-spaces is not an option: While the curse of dimensionality would be broken, all information contained in the core component $\mathcal{G}$ of the Tucker tensor would be lost. This is confirmed in our synthetic numerical experiments below.

The DuSK uses a similar strategy as the subspace kernel: Here, we compare all the feature vectors in the CP decomposition. DuSK therefore also performs well if most of the information is in the column spaces, i.e., in the feature vectors. The main contribution of this article is an improved tensor kernel for Tucker tensors that includes information of the core tensor $\mathcal{G}$ while breaking the curse of dimensionality, retaining the strengths of the subspace kernel and DuSK while allowing for a much more efficient computation.

We first observe that in the computation of the SVD of a matricized tensor $\mathcal{X}_{(m)}$, we can shift any power of the singular values into either the left or the right singular matrices:

$$\mathcal{X}_{(m)} = \mathbf{U}^{(m)}\mathbf{\Sigma}^{(m)}(\mathbf{V}^{(m)})^T = \mathbf{U}^{(m)}\left(\mathbf{\Sigma}^{(m)}\right)^p\left(\mathbf{\Sigma}^{(m)}\right)^{1-p}(\mathbf{V}^{(m)})^T$$

for any $p \in \mathbb{R}$ (assuming no zero singular values, or defining $0^0 = 1$). Using this, we can distribute singular values over the Tucker factors in the HOSVD and we use the resulting *rescaled* features

$$\bar{\mathbf{U}}^{(m)} := \mathbf{U}^{(m)}\left(\mathbf{\Sigma}^{(m)}\right)^p \frac{\|\mathbf{\Sigma}^{(m)}\|_F^{1/M}}{\|(\mathbf{\Sigma}^{(m)})^p\|_F} \tag{5.5}$$

for the computation of the kernel. This means that for all choices $p \in [0,1]$, the norm of the tensor is distributed equally over the M feature matrices

$$\|\mathcal{X}\| = \|\mathbf{\Sigma}^{(1)}\|_F = \cdots = \|\mathbf{\Sigma}^{(M)}\|_F = \prod_{m=1}^{M} \|\bar{\mathbf{U}}^{(m)}\|_F,$$

which provides accurate classification in practice while using only the row-space components. We can choose different values for p: $p = 0$ yields unweighted feature vectors of equal importance, while $p = 1$ would mean a full reweighting according to the importance of the features in the data (see Figure 5.5).

Since the projections onto the row-spaces are invariant under rescaling of the features, we compute the Euclidean distances instead and sum over the exponential kernels of all combinations, noting that if the distances are large, these terms will be negligible. The result is similar to DuSK, but it uses the feature vectors from the Tucker decomposition and the order of the sum over the ranks and the product over the tensor order is reversed:

$$K(\mathcal{X},\mathcal{Y}) = \prod_{m=1}^{M} \sum_{i,j=1}^{R_m} k((\mathbf{u}_{\mathcal{X}}^{(m)})_i, (\mathbf{u}_{\mathcal{Y}}^{(m)})_j) = \prod_{m=1}^{M} \sum_{i,j=1}^{R_m} \exp\left(-\frac{1}{2g^2}\left\|\bar{\mathbf{u}}_{\mathcal{X},i}^{(m)} - \bar{\mathbf{u}}_{\mathcal{Y},j}^{(m)}\right\|_F^2\right), \quad (5.6)$$

where, as in the subspace kernel, we distinguish the SVD of the two tensors, writing $\mathcal{X}_{(m)} = \mathbf{U}_{\mathcal{X}}^{(m)}\Sigma_{\mathcal{X}}^{(m)}\mathbf{V}_{\mathcal{X}}^{(m)^T}$ and $\mathcal{Y}_{(m)} = \mathbf{U}_{\mathcal{Y}}^{(m)}\Sigma_{\mathcal{Y}}^{(m)}\mathbf{V}_{\mathcal{Y}}^{(m)^T}$. We call this kernel the *weighted subspace exponential kernel* or WSEK for short. We also considered leaving the order of the sum and the product the same as in DuSK, but this would need further considerations if the Tucker ranks across the modes are not all the same. Furthermore, our experiments have not shown any significant difference in classification accuracy with respect to the order of $\prod_{m=1}^{M}$ and $\sum_{i,j=1}^{R_m}$.

Algorithm 5.2 summarizes the computation of the reweighted HOSVD with some parameter $p \in [0,1]$: First, we compute the SVD of the m-th matricization of $\mathcal{X}$. Then, the sign ambiguity of the feature vectors is addressed and the reweighting (5.5) is performed. Finally, the the core tensor $\mathcal{G}$ is updated using the tensor-matrix-product defined in [183].

5.2.3 Computational Complexity of the Different Kernels

For a large number N of data points, for large tensor order M or large mode dimensions I_m, computing the different tensor kernels can be very time-consuming. If the data input is already given in CP format, computing the DuSK is not too expensive. But most often, the data is given as a full tensor. In these cases, it is preferred to compute first a TT or Tucker decomposition of the tensor and then convert it into CP, in order to circumvent the aforementioned numerical issues with the computation of the CP decomposition.

Algorithm 5.2: Weighted HOSVD.

```
Require: Given tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$, Tucker ranks $R_1, \ldots, R_M$, weighting power $p$
    (default $p = 1/M$).
Ensure: Tucker factors $\bar{\mathbf{U}}^{(1)}, \ldots, \bar{\mathbf{U}}^{(M)}$ and core $\mathcal{G}$.
    Initialize $\mathcal{G} = \mathcal{X}$.
    for $m = 1$ to $M$ do
        Step 1: Computing uniqueness-enforced HOSVD
        Compute SVD $\left[\mathbf{U}^{(m)}, \mathbf{\Sigma}^{(m)}, (\mathbf{V}^{(m)})^\top\right] = \mathrm{svd}(\mathcal{X}_{(m)})$,
        where $\mathbf{\Sigma}^{(m)} = \mathrm{diag}(\sigma_1^{(m)}, \sigma_2^{(m)}, \ldots, \sigma_{I_m}^{(m)})$
        for $r_m = 1$ to $R_m$ do
            $i^*_{r_m} = \arg\max_{i=1,\ldots,I_m} |u^{(m)}_{i,r_m}|$
            $\hat{\mathbf{u}}^{(m)}_{r_m} := \mathbf{u}^{(m)}_{r_m} / \mathrm{sign}(u^{(m)}_{i^*_{r_m},r_m})$
        end for
        $\hat{\mathbf{U}}^{(m)} = [\hat{\mathbf{u}}^{(m)}_1, \hat{\mathbf{u}}^{(m)}_2, \ldots, \hat{\mathbf{u}}^{(m)}_{R_m}]$
        Step 2: Computing norm weighted factors
        $\bar{\mathbf{U}}^{(m)} = \hat{\mathbf{U}}^{(m)} \left(\mathbf{\Sigma}^{(m)}\right)^p \frac{\|\mathbf{\Sigma}^{(m)}\|_F^{1/M}}{\|(\mathbf{\Sigma}^{(m)})^p\|_F}$
        where $\mathbf{\Sigma}_p^{(m)} = \mathrm{diag}((\sigma_1^{(m)})^p, (\sigma_2^{(m)})^p, \ldots, (\sigma_{R_m}^{(m)})^p)$
        $\mathcal{G} \leftarrow \mathtt{ttm}(\mathcal{G}, (\bar{\mathbf{U}}^{(m)})^{-1}, m)$        [183]
    end for
```

Here, however only notice is on the complexity of the kernel computation with respect to the given ranks. The maximal dimensions or ranks are denoted by $I = \max_{m=1,\ldots,M} I_m$, $R_{Tucker} = \max_{m=1,\ldots,M} R_m$, and $R_{TT} = \max_{m=1,\ldots,M-1} r_m$. The ranks are also to be understood as the maximal respective rank of all data inputs.

Table 5.2 summarizes the computational complexity for a single entry of the kernel matrix. Notice that all kernels except the Gaussian kernel can only be computed if the tensor is in a low-rank format. Also, a tensor is interpreted in CP format to be a Tucker tensor with diagonal core tensor $\mathcal{G}$ and thus the subspace kernel and WSEK are computed using the factor matrices. Furthermore, in the cases of Tucker and TT tensors, the complexity using naive matrix multiplication ($\mathcal{O}(n^3)$) is reported.

Here observe that for large tensor order M, computation of the Gauss kernel is prohibitive if it is not done in a low-rank format. If the CP rank R is small, all kernels can be computed efficiently. However, if the CP decomposition has to be obtained by conversion from Tucker or TT, these ranks can be large and DuSK suffers from the curse of dimensionality. WSEK is even more efficient than the subspace kernel. The CPU times for the numerical experiments are reported in Sec. 5.4.

Table 5.2: Theoretical complexity of computing a single entry of the different kernel matrices from data given in different formats. Note that some kernel-format combinations are not defined.

	Full	CP	Tucker	TT
Gaussian	$\mathcal{O}(I^M)$	$\mathcal{O}(MIR^2)$	$\mathcal{O}(MR_{Tucker}^{(M+1)} + MIR_{Tucker}^2)$	$\mathcal{O}(MIR_{TT}^3)$
DuSK	–	$\mathcal{O}(MIR^2)$	$\mathcal{O}(MIR_{Tucker}^{2M})$	$\mathcal{O}(MIR_{TT}^{2(M-1)})$
Subspace	–	$\mathcal{O}(MI^{M-1}R^2)$	$\mathcal{O}(MI^{M-1}R_{Tucker}^2)$	–
WSEK	–	$\mathcal{O}(MIR^2)$	$\mathcal{O}(MIR_{Tucker}^2)$	–

5.3 A Numerical Study on Synthetic Data

As mentioned above, knowledge about the tensor structure can and should be exploited when choosing the tensor kernel K. In this section, focus is on exploring why the DuSK performs well in many cases by comparing it to the Gaussian kernel and the subspace kernel in a synthetic experimental setting. Furthermore, Here it is seen that the proposed WSEK retains the advantages of DuSK, while outperforming it in cases where DuSK is less suitable.

5.3.1 Interpreting CP and Tucker

The CP decomposition of a tensor can be seen as a special case of the Tucker decomposition with a diagonal core tensor. More precisely, if the matrix ranks of the factor matrices $\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(M)}$ are smaller than R, a Tucker decomposition of the tensor with Tucker ranks $R_m < R$ for $m = 1, \dots, M$ is found. The core is then no longer diagonal, but it will have many zero entries if $R < R_1 + \cdots + R_M$.

In any case, for $m = 1, \dots, M$, the factor matrix $\mathbf{A}^{(m)}$ in the CP format, or the leaf $\mathbf{U}^{(m)}$ in the Tucker format, spans the subspace of $\mathbb{R}^{I_m}$ that the data of the m-th mode lies in (the columns of the m-mode matricization of the full tensor also span this subspace). ALthough, the Tucker core may become dense if CP factors are decomposed. The information that is stored in the Tucker tensor (and by inclusion also in the CP tensor) is therefore twofold: What are the subspaces that input data lies in? This information is stored in the orthogonalized leafs of the Tucker tensor. And what combination of feature vectors is present (and to what degree) in the data? This information is stored in the core tensor $\mathcal{G}$.

This observation can be exploited when designing a tensor kernel for KSTM. The subspace kernel introduced in §5.2 uses the highly correlated column spaces of the unfoldings. Using the smaller row spaces instead would mean that we can only see the leafs of the Tucker tensor. DuSK includes the core data via the norm equilibration. The Gaussian kernel uses the whole tensor, but it is less sensible in spotting the subspaces, because the tensor is simply vectorized, and this information is hidden. The proposed WSEK includes information of the core tensor because the feature vectors are weighted with the p-th power of the corresponding singular values.

5.3.2 A Synthetic Experiment

The considerations are substantiated by creating two artificial experimental scenarios: In one (the *leaf*-scenario), all the information necessary for classification is stored in the leafs (*i.e.* the subspaces) and in the other (the *core*-scenario), all the information is in the core of the Tucker tensor. Then the performance of the aforementioned kernels on this data for different noise levels and tensor ranks has tested.

The detailed experimental setup is as follows: Let $M = 3$ and $I_1 = I_2 = I_3 = 100$. For different Tucker ranks $R_1 = R_2 = R_3 = r_{approx} = 1, \ldots, 10$, simulate the approximation of a tensor with Tucker ranks $R'_1 = R'_2 = R'_3 = r_{exact} = 3$ plus some noise. That is, the core $\mathcal{G}$ of the simulated tensor will have size $r_{approx} \times r_{approx} \times r_{approx}$ and the leafs will have sizes $I_m \times r_{approx}$ for $m = 1, 2, 3$. The core consists of random noise that is normally distributed with mean 0 and variance ϑ^2 (the noise level to be chosen later). To the small upper-left-and-foremost cube of size $\min(r_{approx}, 3) \times \min(r_{approx}, 3) \times \min(r_{approx}, 3)$, add the information tensor. In the *core*-scenario, this is the same tensor for all samples in the same class, generated by drawing the entries from a standard normal distribution. In the *leaf*-scenario, the information tensor is different for all samples (also drawn from the standard normal distribution).

The leafs of size $I_m \times r_{approx}$ also consist of random noise (normally distributed with mean 0 and variance ϑ^2) and to the first $\min(r_{approx}, 3)$ columns add vectors $\cos(\pi * \nu * \mathbf{v})$, where $\nu \in \mathbb{R}^{100}$ is a uniform discretization of $[-1, 1]$ and the frequency ν is picked uniformly at random (with mean 0 and variance 1). In the *leaf*-scenario, these frequencies are the same for all samples in the same class, and in the *core*-scenario, these frequencies are different for all samples. After the construction, the leafs are orthogonalized (using the QR-decomposition and discarding the R-matrix), so that the resulting Tucker tensor is already in the form of a HOSVD.

The reasoning is that these two scenarios yield Tucker tensors of rank r_{approx} that are approximations of noisy tensors of rank r_{exact}, and the cluster information is stored exclusively in either the core or the leafs. 100 samples in two classes with 50 samples each for each noise level $\vartheta^2 = 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1$ and Tucker ranks $r_{approx} = 1, 3, 5, 10$ are generated. Continually the SVM is performed 20 times with 5-fold cross validation in order to determine the hyperparameters $C \in \{2^{-8:1:8}\}$ (the soft-margin parameter) and $g \in \{2^{-4:1:12}\}$ (the variance parameter in the Gaussian kernel).

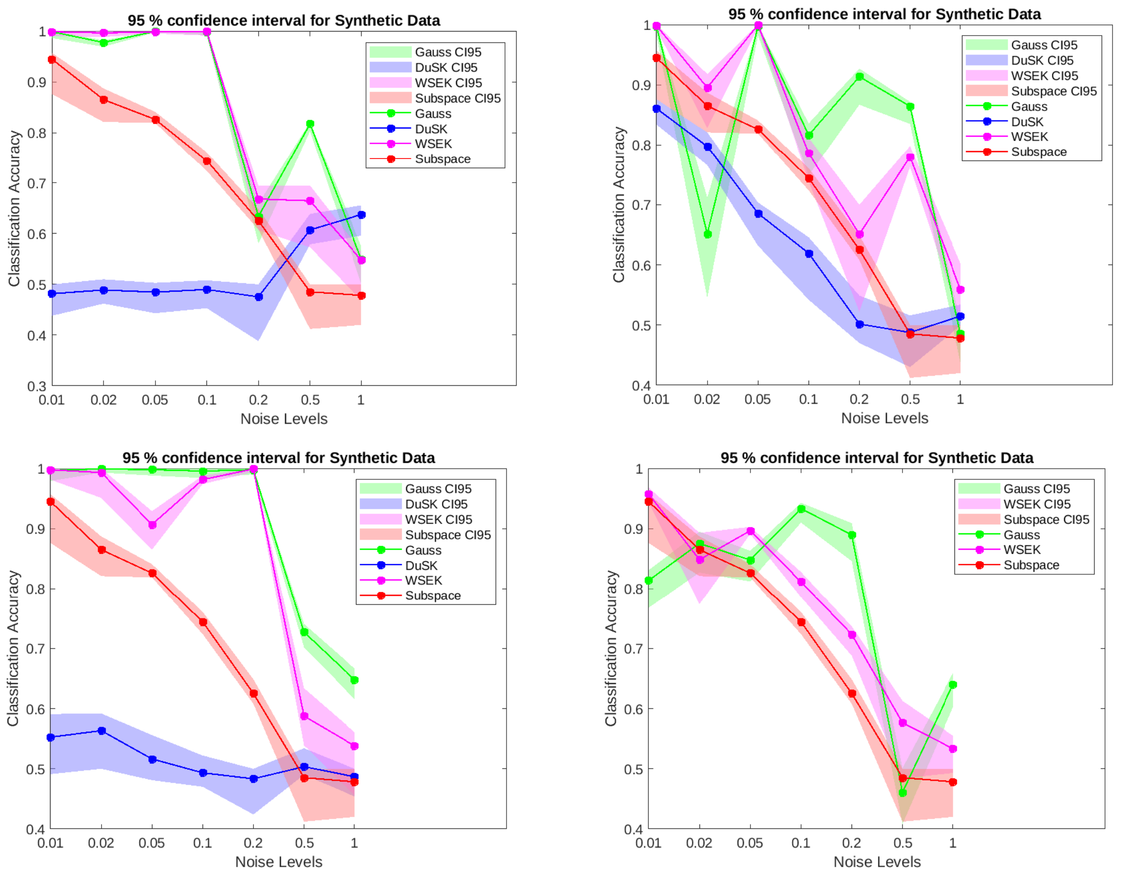


Figure 5.1: Classification Accuracy with 95% confidence interval for different noise levels in the *core*-scenario of the generated synthetic data with different rank truncation $r_{approx} = 1, 3, 5, 10$ (subspace kernel doesn't have any rank truncation).

5.3.3 Results and Interpretation

In Fig. 5.1 and Fig. 5.2, the results for the two experimental scenarios are marked-out. In both cases, the test error (*i.e.* the classification accuracy on the test set) for the ranks $r_{approx} = 1, 3, 5, 10$ are plotted. For each rank, the accuracies of the different kernels for all noise levels in one picture are drawn. The computation of DuSK was too expensive in the case $r_{approx} = 10$ as this requires the computation of norms of tensors with CP rank 1000 many times.

In the *core*-scenario, the Gaussian kernel outperforms both DuSK and the Subspace kernel. The WSEK performs similarly to the Gaussian kernel. In the *leaf*-scenario, the Subspace kernel gives 100% accuracy in all cases, and all but the Gaussian kernel performed very well on this data.

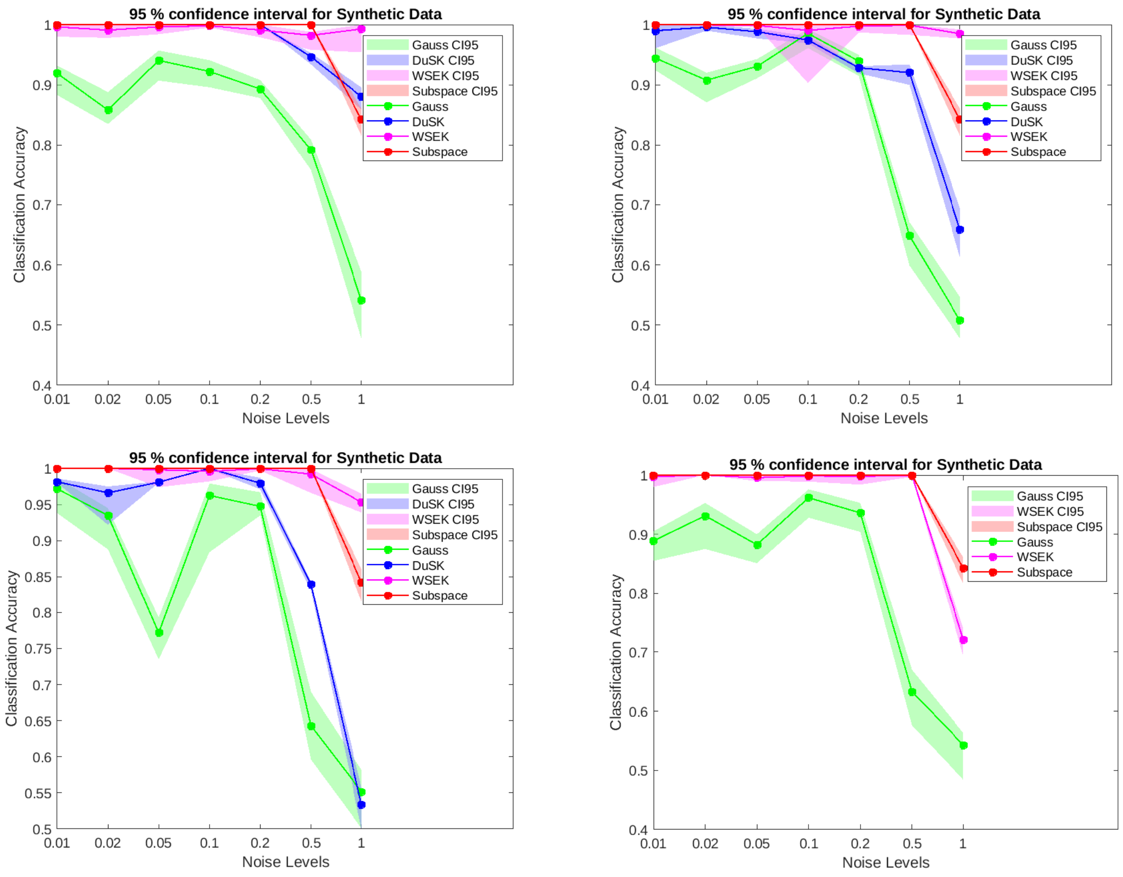


Figure 5.2: Classification Accuracy with 95% confidence interval for different noise levels in the *leaf*-scenario of the generated synthetic data with different rank truncation $r_{approx} = 1, 3, 5, 10$ (subspace kernel doesn't have any rank truncation).

These results are not surprising following the considerations in Sec. 5.3.1: The Subspace kernel sees only the information in the leafs, and it can therefore not perform well in the *core*-scenario. The DuSK kernel includes extra information so that it performs better in the *core*-scenario (especially when the rank is correctly guessed, $r_{approx} = r_{exact} = 3$) but still not as good as the Gaussian kernel. In the *leaf*-scenario, all the information is in the subspaces and therefore both DuSK and the Subspace kernel perform very well. Here, the Gaussian kernel performed much worse than all other kernels. The WSEK was designed to do well in both scenarios and this is shown also in these experiments. The next chapter exhibits that especially the *leaf*-scenario is realistic: In the ADNI dataset explored below, the subspace kernel performs better than the Gaussian kernel and it even outperforms DuSK. The conclusion is that including information on the m-mode subspaces is crucial for the design of a tensor kernel. This is also a possible explanation why DuSK has performed well in many settings. The mounted new kernel however outperforms DuSK in all offered experiments.

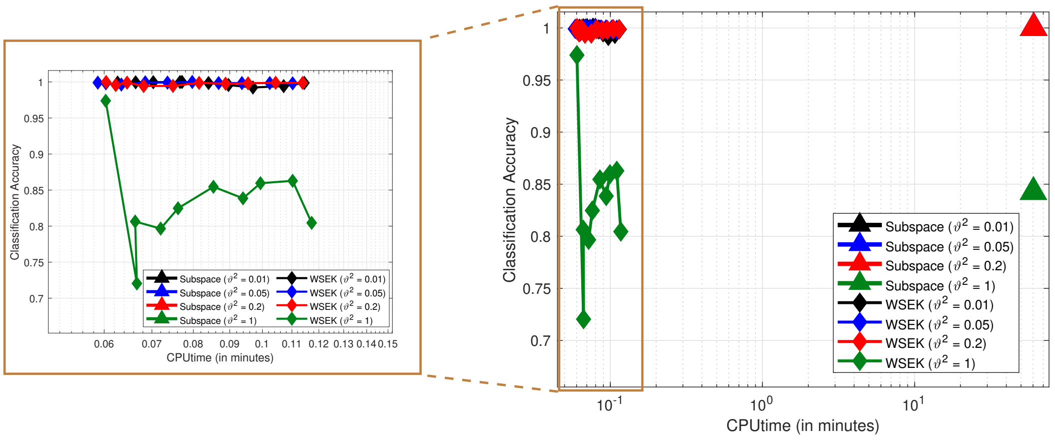


Figure 5.3: The plot shows the CPU time vs classification accuracy for synthetic data (*leaf*-case) with truncation rank from 1 to 10 between. The comparison is done between proposed WSEK (5.2.2) and subspace kernel ([167]). The subspace is referred as subspace in the plot and has no rank truncation. Therefore, there is only one point for each noise level.

Running Time Figures 5.3 and 5.4 present a comparison of processing time between our proposed WSEK (§5.2.2) and the subspace kernel [167]. The graphs clearly illustrate that the CPU time required for the subspace kernel is significantly higher in both the *leaf* and *core* scenarios. However, the classification performance remains almost similar for both kernels across varying levels of noise in both scenarios (*leaf* and *core*).

WSEK incorporates rank truncation, and even with lower ranks, the classification performance matches the state-of-the-art for generated synthetic data.

5.4 Classification of Real Datasets

In this section, the performance of the discussed tensor kernels is tested on two real world datasets: The ADNI dataset contains fMRI images of patients with and without Alzheimer's disease and the ADHD dataset contains fMRI images of ADHD patients and healthy subjects. Here using the KSTM with the different kernels to distinguish the two classes of subjects in each dataset.

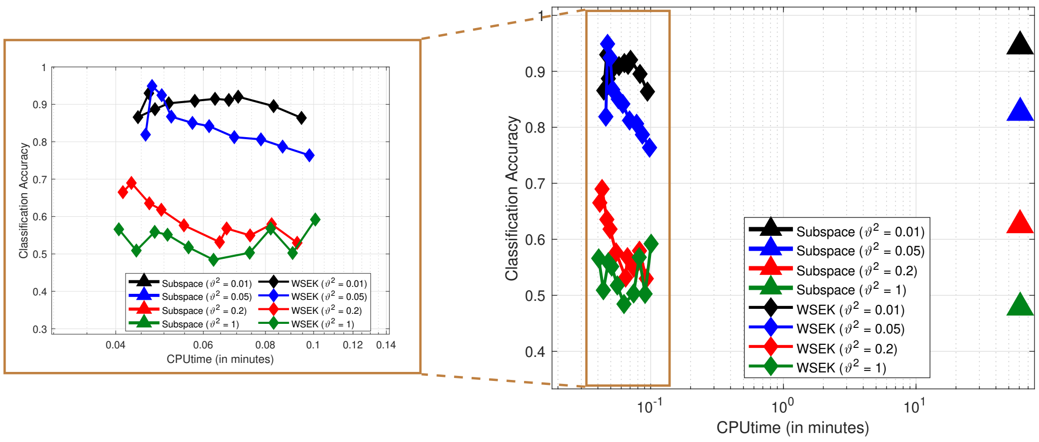


Figure 5.4: The plot shows the CPU time vs classification accuracy for synthetic data (*core*-case) with truncation rank from 1 to 10 between. The comparison is done between proposed WSEK (5.2.2) and subspace kernel ([167]). The subspace is referred as subspace in the plot and has no rank truncation. Therefore, there is only one point for each noise level.

All numerical experiments have been done in `MATLAB 2019b`. Low-rank tensor approximations are computed using `TT-Toolbox` citettTool and `tensor toolbox` citetenTool. All experiments have executed on a computer cluster which is equipped with `2 TB NVMe SSD` Hard disk, $2\times$`Intel Xeon Skylake Silver 4210R` CPUs with 10 cores per CPU, and `768 GB DDR4 ECC` of RAM. The hyperparameters in KSTM (3.24) are tuned similarly to Synthetic experiments (see Section 5.3.2), the only difference is that the results for real data experiments for each of the rank $R \in \{1, 2, \cdots, 10\}$ are reported, where $R_1 = R_2 = R_3 = R$. The SVM problem (3.24) is solved using the `svmtrain` function from LIBSVM citelibsvm library, and `svmpredict` computes the classification accuracy using (3.26) and (3.27).

5.4.1 Numerical results

In this section, the results for the two fMRI datasets are summarized:

- **p-parameter:** One of the factors influencing the computation of the WSEK (§5.2.2) is the p value used when distributing weights (singular values) across Tucker factors. Figure 5.5 illustrates the classification accuracy of ADNI and ADHD datasets for various p values in (5.5), which rescales the Tucker factors. In the case of ADNI data, a noticeable distinction is observed for the selected p

value, whereas ADHD exhibits no improvement, especially concerning the highest classification accuracy achieved. The best method for ADHD is Subspace Kernel.

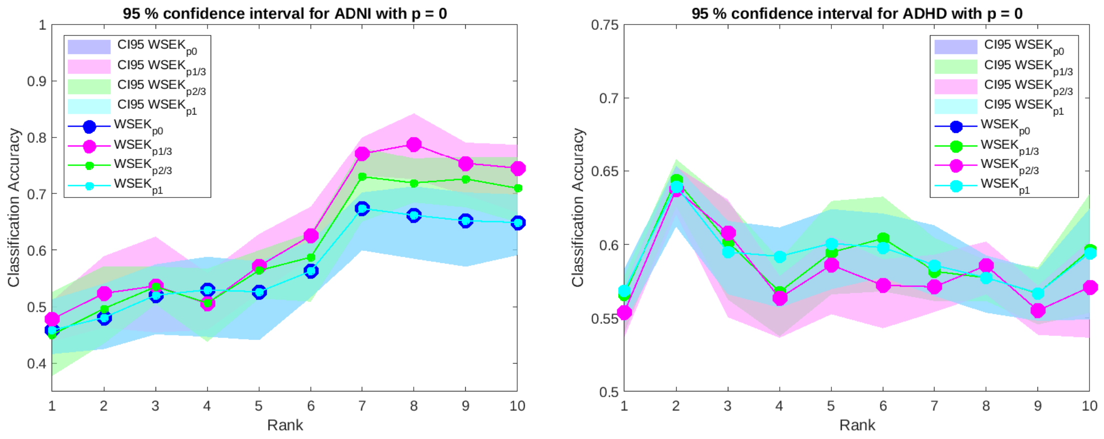


Figure 5.5: Classification Accuracy with 95% confidence interval for different p values in WSEK for ADNI (left) and ADHD (right) dataset with rank truncation $R \in [1, 10]$

- **Classification accuracy:** In Fig. 5.6, the average classification accuracy resulting from the cross validation is laid-out. In Table 5.3, the best classification accuracy between rank $[1, 10]$ is laid-out. On both datasets the proposed WSEK gives the best average classification accuracy (79% for ADNI and 64% for ADHD) compared to other state-of-the-art tensor kernels. Notice that the accuracy of the subspace kernel improves for higher ranks in the ADHD dataset and is then similar to that of WSEK, but this choice of rank is very high for a low-rank truncation method. On the other hand, the proposed kernel gives good classification accuracy already at rank 2. The Gaussian kernel was computed for different Tucker approximations of the full tensor. Using the full tensor in the computation of the kernel did not improve the accuracy. As in (Chapter 4), the DuSK kernel was computed using a CP approximation of the full tensor (CP-DuSK), because computing the Tucker decomposition and then converting to CP yielded high CP ranks and DuSK was too slow.

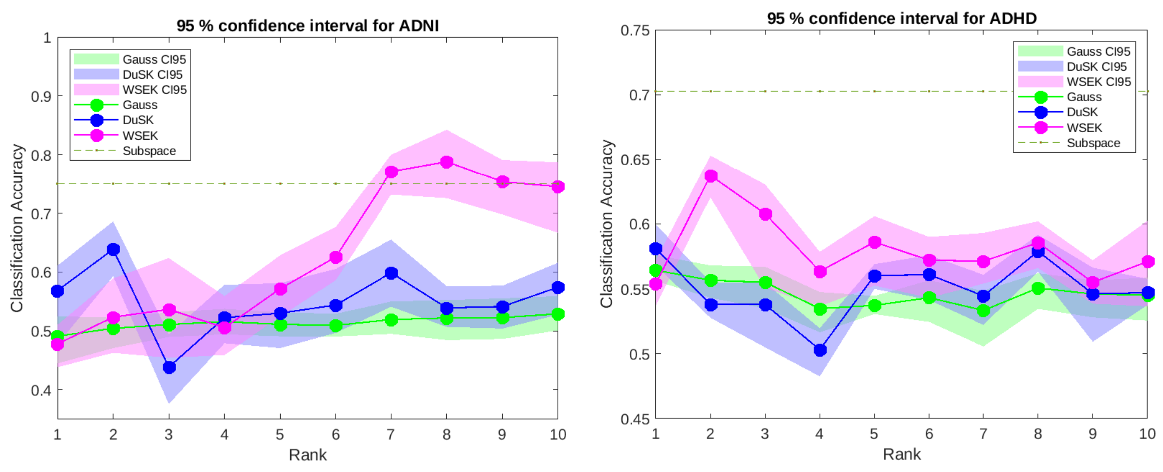


Figure 5.6: (left) Comparison of mean classification accuracy with variance for the different kernels with different rank truncation for ADNI dataset, (right) Comparison of mean classification accuracy with variance for the different kernels with different rank truncation for ADHD dataset.

Table 5.3: Maximum average classification accuracy in percentage ± standard deviation for different methods, data sets, and rank $R \in [1, 10]$. The values for TTCP-DuSK are taken from [98] for comparison.

Methods	ADNI	ADHD
Gauss	53	50
CP-DuSK	64 ± 0.05 (R = 5)	58 ± 0.02 (R = 6)
TTCP-DuSK	73 ± 0.03 **(R = 4)**	63 ± **0.01** (R = 5)
TuckerCP-DuSK	75 ± **0.02** (R = 7)	63 ± **0.01** (R = 8)
Tucker Subspace ([167])	75 ± 0.03 (R = full)	**70** ±**0.01** (R = full)
WSEK	**79** ± 0.03 (R = 8)	64 ±**0.01** **(R = 2)**

Note: Figure 5.7, as described, computes the classification accuracy with a 95% confidence interval for the ADNI and ADHD datasets. This approach involves performing the standard Tucker decomposition on the input data, converting it into CP format, and then applying the DuSK kernel over the factor matrices. The corresponding accuracy is also mentioned in Table 5.3 as TuckerCP-DuSK. However, this method suffers from significant slowness. As a result, the ADHD experiment reached its maximum capacity before reaching rank 10, which would have corresponded to rank 1000 in the case of TuckerCP. In practical terms, this method is not sustainable.

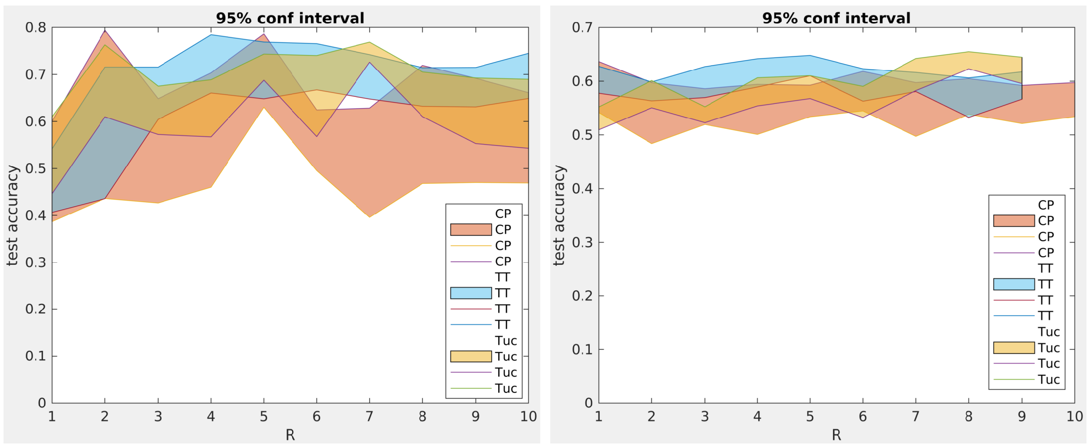


Figure 5.7: Comaprison of 95% confidence interval for DuSK with CP, TT, and Tucker decompositions: (left) ADNI dataset (right) ADHD dataset.

Remark 5.1:
The Figure 5.7 represents an early work performed during, where experiments setting is according to previous Chapter 4. The experiments here were exptremenly slow due to the computation of Kernel on each run separately and showed clear lack of upgrade in terms of efficiency. Therefore, further exploration with Tucker-CP DuSK wasn‘t focused. ◊

- **Running Time:** Table 5.4 and Table 5.5 show the running times for the computation of the different kernels on the ADNI and ADHD datasets respectively. For small tensors, computing the Gauss kernel is fast. Computation of DuSK on TTCP however quickly becomes prohibitive and takes a long time in these experiments. The WSEK is faster for all the experiments while subpace becomes slower for ADNI and even more for bigger dataset as ADHD.

- **Statistic comparison:** Table 5.3 and Figure 5.6 present the variance associated with the mean accuracy values. Notably, the WSEK STM demonstrates a favorable balance between classification accuracy and variance. Specifically, for the ADNI dataset, it achieves the highest classification accuracy. Conversely, in the case of ADHD, it achieves improved accuracy at substantially lower ranks, accompanied by reduced variance. A plausible explanation for the enhanced performance of the subspace method in the ADHD context could be attributed to the larger dataset size.

Table 5.4: Comparison of CPU time for $R \in [1, 10]$ for ADNI dataset. The values for TTCP-DuSK are taken from [98] for comparison. The time comparison is computed based on the originally presented work [71, 167].

Kernel	Format	Parameters	CPUtime for ADNI
Gaussian	Ktensor	C, g	30 seconds
DuSK	CP (§3.3.1)	$C, g, \ R$	17 minutes
DuSK	TTCP (§5.1.3)	$C, g, \ R_{TT}$	3.5 hours
Subspace	MLSVD (§5.2.1)	$C, g, \ R_{full}$	6 minutes
WSEK	SqrtmHOSVD (Alg. 5.2)	$C, g, \ R_{Tucker}$	50 seconds

Table 5.5: Comparison of CPU time for $R \in [1, 10]$ for ADHD dataset. The values for TTCP-DuSK are taken from [98] for comparison. The time comparison is computed based on the originally presented work [71, 167].

Kernel	Format	Parameters	CPUtime for ADHD
Gaussian	Ktensor	C, g	4 minutes
DuSK	CP (§3.3.1)	$C, g, \ R$	1.3 hours
DuSK	TTCP (§5.1.3)	$C, g, \ R_{TT}$	–
Subspace	MLSVD (§5.2.1)	$C, g, \ R_{full}$	59 minutes
WSEK	SqrtmHOSVD (Alg. 5.2)	$C, g, \ R_{Tucker}$	13 minutes

5.5 Chapter Summary

The previous chapter introduced an initial approach towards a robust classification model based on TT decomposition and computation of the kernel required in CP format. This work is an extension that not only focuses on constructing a kernel function or "factor kernel" but also pays attention to finding a solution for it. This chapter centres on another essential form of tensor decomposition: Tucker. Since the primary concern is to identify a kernel function or feature space that captures the dominant features of the tensor input while preserving the data structure, Section §5.2 presents an existing kernel based on Tucker decomposition. As mentioned in the previous chapter, DuSK has a special form that captures features uniquely. Combining Tucker decomposition-based subspace kernel (§5.2.1) with DuSK form (§4.2.5) leads to a new discovery. This new *Weighted Subspace Exponential Kernel* (§5.2.2) relies on a specific form of Tucker decomposition called *Weighted HOSVD*. The Weighted HOSVD includes the *uniqueness* property and a weighted distribution of singular values (Algorithm 5.2). Furthermore, Section §5.3 analyses the advantages and drawbacks of the newly discovered WSEK in various scenarios. The study concludes positively with numerical experiments on real-world data in Section §5.4.

CHAPTER 6

SIMULTANEOUSLY-UPDATED SUPPORT TENSOR MACHINE

6.1 Introduction 91
6.2 Regularized Tensor Machine Learning Model 93
6.2.1 Regularized Support Tensor Machine 93
6.3 Optimizing Regularized Models 99
6.3.1 Primal Dual SVM Relation 100
6.3.2 Partial Simultaneous Update using Gradient Descent 103
6.3.3 Full Simultaneous Update using Gradient Descent 104
6.3.4 Gradient Descent Enhancement 106
6.3.5 Full Simultaneous update TT-MMK 108
6.4 Numerical Assessment 109
6.4.1 Convergence Analysis 110
6.5 Chapter Summary 111

Contents

6.1 Introduction

When dealing with STL models, such as STMs, it is often useful to consider both classification performance and dimensionality reduction together. Low-rank approximation-based STM models typically operate on a well-posed low-dimensionality subspace, which is defined by significant low-rank feature vectors, also known as principal components. The standard version of KSTM (§3.4.3) has already been explained. The methodology presented here goes beyond the previously presented solutions of KSTM (§3.4.3) by introducing an additional constraint that is a regularized term. The rationale for this extension is elaborated further in [73]. The proposed research consider improving the accuracy (rather than computational complexity) of the schemes (Chapter 4, Chapter 5) and exploring joint optimization of the TT cores and SVM weights. Similar to the neural network compression in the TT format [126], such targeted iterative refinement of the TT decomposition may improve prediction accuracy. However, the significant contribution of this chapter lies in the theoretical development of low-rank methods in kernel-based machine learning models.

Main Novelty: The existing tensor decomposition methods developed so far are highly problem-specific, and there is a growing interest in finding a universal and optimal low-rank tensor factorization method. It is evident that the development of various STL models is motivated by the fact that tensors contain spatial structures that are lost when the data is vectorized. In Chapter 4, the TT-CP decomposition is introduced, which utilizes stable matrix-SVD based algorithms to obtain decomposed cores with properties of the CP decomposition, as well as a simple structure.

The main idea behind the work presented in this chapter is that finding the best low-rank tensor decomposition for the STM objective function collectively is better than finding the best fit separately.

However, the formulation of the specified abstraction leads to a regularized STM form of standard STMs. The optimization approach used for the STM, along with low-rank tensor factorization, was introduced in [73] through a separate update of CP factorization and STM update. This chapter extends the same idea to the full primal problem of STM and solves the simultaneously updated STM and TTCP factorization. However, finding a solution to the full primal STM is not straightforward due to the non-convex nature of the objective function. While using the TTCP format avoids the NP-hardness issue associated with finding the best low-rank approximation, solving the non-convex optimization problem could still lead to NP-hardness.

The most significant contribution in this chapter is a "primal-dual relation" of the STM model in TTCP factorization. This avoids the computational issue of finding the nonlinear mapping (Ψ) explicitly by using the previously mentioned kernel trick. This leads to a plausible solution of the full-primal optimization, which is Projected Gradient Descent (PGD) (Algorithm 6.2). The PGD is an iterative approach for constraint non-convex objective functions and requires gradients with respect to each parameter. This computation is also explained in this chapter. Theoretical analysis and some open questions regarding the convergence of PGD for the simultaneously-updated STM are further discussed topics of this contributed work.

6.2 Regularized Tensor Machine Learning Model

Regularized tensor models aims to reduce the complexity of STL models. The modification from standard STLs to regularized STLs are mostly done through constraints (restrictions) on the model weight parameters ($\mathcal{W}$). This is particularly advantageous for issues with numerous features but a few data samples. Regularized linear tensor models can be generally formulated as

$$\min_{\mathcal{W},b} f(\mathcal{X}, y) = \min_{\mathcal{W},b} \ell(\mathcal{X}, y \mid \mathcal{W}, b) + \gamma R(\mathcal{W}). \tag{6.1}$$

The expression $\ell(\mathcal{X}, y; |; \mathcal{W}, b)$ represents a loss or error function, while $R(\mathcal{W})$ is a regularization term. The parameter $\gamma > 0$ determines the balance between the contributions of the loss and regularization terms. In the case of STMs, a hinge loss in the form of $l(\hat{y}) = \max(0, 1 - y\hat{y})$ is commonly used, where $\hat{y}$ is the prediction and y is the true label. Depending on the choice of regularization term, such as Frobenius norm (standard Tikhonov regularization) or ℓ_1 norm (sparsity constraints), the objective function can be tailored to different applications.

Tensors offer more flexibility than vectors and matrices in terms of sparsity profiles. Instead of imposing global sparsity for the entire tensor, it is possible to impose sparsity on slices or fibers. The rank properties of tensors, similar to matrices, can also be utilized, and are richer and more complex due to the multidimensional structure of tensors. In addition to sparsity constraints, a low-rank structure of tensor data can be exploited as a regularizer, such as the canonical or Tucker decomposition of $\mathcal{W}$. The low-rank structure of tensor data has been successfully used in various applications, including missing data imputation [110] and subspace clustering [205]. Apart from the low-rank properties of data itself, low-rank regularization can also be applied to learning coefficients in classification tasks [203]. The low-rank constraint for the tensor $\mathcal{W}$ can also be formulated through the tensor norm, in the form [197], tensor norms is the *tensor nuclear norm* [110] or the *(overlapped) trace norm* [196], *latent trace norm* [180].

However, these methods have their own drawbacks. Computation becomes infeasible for very large-scale applications, or solving sub-optimization problems are not handy as it requires numerous variables in latent space. In general, dealing with tensor nuclear norm does not come as easily as matrix nuclear norm. Therefore, solutions for regularized STMs need to be explored from fresh perspectives.

6.2.1 Regularized Support Tensor Machine

One intriguing navigation to solution of regularized model is to change the choice of the regularized model. As perceived previously, nonlinear models have the ability to characterize complex nonlinear dependencies in data. Therefore, unravelling the solution is done by going back to the roots, that is by embedding the data into another higher-dimensional space and finding a linear separating hyperplane in there.

For a given input tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ the TTCP decomposition is (let say) $\mathcal{T}$, then the nonlinear "maximum marginal" classification problem with Ψ as a nonlinear function can be formulated in the following way:

$$J(\mathcal{T}, \Psi(\mathcal{T})) = \underset{\mathcal{W}, b}{\text{minimize}} \quad \frac{\langle \mathcal{W}, \mathcal{W} \rangle}{2} + C \sum_{i=1}^{N} L(y_i, \langle \mathcal{W}, \Psi(\mathcal{T}_i) \rangle + b), \tag{6.2}$$

where L is the loss function (hinge loss), $\mathcal{W}$ are weight parameters, C, b are trade-off parameter and bias term respectively. The decision function is then given as $f(\mathcal{X}_i) = \text{Sign}(\langle \mathcal{W}, \Psi(\mathcal{T}_i) \rangle + b)$, $i = 1, \ldots, N$. The approximation constraint is considered as $\Omega(\mathcal{T})$ on the tensor data, hence the (6.2) can be written as,

$$\underset{\Psi(\cdot)}{\text{argmin}} \quad J(\mathcal{T}, \Psi(\mathcal{T})) + \gamma \Omega(\mathcal{T}). \tag{6.3}$$

Extending the computational relation of the Lagrangian (2.15) for the tensor format and combining (6.2) and (6.3), the following will be the principal optimization problem to solve ($\boldsymbol{\xi}$ is a slack variable for non-separable case §2.2.1),

$$\begin{aligned} &\min_{\mathcal{W}, b, \boldsymbol{\xi}, \mathbf{T}^{(\tilde{m}, i)}} \frac{\langle \mathcal{W}, \mathcal{W} \rangle}{2} + C \sum_{i=1}^{N} \xi_i + \gamma \sum_{i=1}^{N} \Omega(\mathcal{T}_i) \\ &\text{subject to} \quad y_i(\langle \mathcal{W}, \Psi(\mathcal{T}_i) \rangle + b) \geqslant 1 - \xi_i, \quad \xi_i \geqslant 0, \quad i = 1, 2, \ldots, N \end{aligned} \tag{6.4}$$

The Lagrange function becomes as follows,

$$\begin{aligned} L_p = \frac{1}{2} ||\mathcal{W}||^2 &+ C \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \boldsymbol{\alpha}_i [y_i(\langle \mathcal{W}, \Psi(\mathcal{T}_i) \rangle + b) - (1 - \xi_i)] - \sum_{i=1}^{N} \mu_i \xi_i \\ &+ \gamma \sum_{i=1}^{N} \Omega(\mathcal{T}_i), \quad \wedge \quad \Omega(\mathcal{T}_i) = \left\| \mathcal{X}_i - [\![\mathbf{T}^{(1,i)}, \mathbf{T}^{(2,i)}, \cdots, \mathbf{T}^{(M,i)}]\!] \right\|_F^2, \end{aligned} \tag{6.5}$$

where $\Omega(\mathcal{T}_i)$ is minimized w.r.t. $\mathcal{W}, b, \xi_i, \mathbf{T}^{(\tilde{m},i)}$. As in the earlier Chapter 4, the value of $\langle \Psi(\mathcal{T}_i), \Psi'(\mathcal{T}_j) \rangle$ is computed in terms of the TT-CP factorization. Therefore, a nonlinear mapping from the space of tensors to a *tensor product RKHS* (§3.4.2, [169]) $\Psi \colon \mathbb{R}^{I_1} \times \cdots \times \mathbb{R}^{I_M} \mapsto \mathbb{F}$ consists of separate feature maps acting on different CP factors is taken as follows,

$$\Psi : \boldsymbol{\mathcal{X}} \mapsto \Psi : \sum_{r=1}^{R} \mathbf{t}_r^{(1)} \otimes \mathbf{t}_r^{(2)} \otimes \cdots \otimes \mathbf{t}_r^{(M)} \mapsto \sum_{r=1}^{R} \phi(\mathbf{t}_r^{(1)}) \otimes \phi(\mathbf{t}_r^{(2)}) \otimes \cdots \otimes \phi(\mathbf{t}_r^{(M)}). \tag{6.6}$$

Note: Each iteration for updating low-rank approximation while optimizing regularized STM, $\Omega(\boldsymbol{\mathcal{T}}_i) = \left\| \boldsymbol{\mathcal{X}}_i - [\![\mathbf{T}^{(1,i)}, \mathbf{T}^{(2,i)}, \cdots, \mathbf{T}^{(M,i)}]\!] \right\|_F^2$ need storing $\mathcal{O}(I^M)$. Therefore, while computing the CP form of $\boldsymbol{\mathcal{X}}_i$ has linear storage in terms of dimension.

In order to solve L_P, the KKT conditions are applied and these are as follows:

1. Primal Feasibility:
$$(1 - \xi_i - y_i(\boldsymbol{\mathcal{W}}^T \Psi(\boldsymbol{\mathcal{T}}_i) + b)) \leqslant 0$$
$$-\xi_i \leqslant 0 \implies \xi_i \geqslant 0,$$
2. Dual feasibility:
$$\boldsymbol{\alpha}_i \geqslant 0, \quad \mu_i \geqslant 0,$$
3. Complementary slackness:
$$\boldsymbol{\alpha}_i(1 - \xi_i - y_i(\langle \boldsymbol{\mathcal{W}}^T \Psi(\boldsymbol{\mathcal{T}}_i)\rangle + b)) = 0,$$
4. Derivatives:
$$\delta_{\boldsymbol{\mathcal{W}}} L_P = 0, \quad \delta_b L_P = 0, \quad \delta_{\xi_i} L_P = 0 \quad \delta_{\mathbf{T}^{(\tilde{m},i)}} L_P = 0.$$

Now combining all the KKT conditions, Some explicit formats for computation of variables are obtained. Those are given as follows,

$$\delta_{\boldsymbol{\mathcal{W}}} L_P = \boldsymbol{\mathcal{W}} - \sum_{i=1}^{N} \boldsymbol{\alpha}_i y_i \Psi(\boldsymbol{\mathcal{T}}_i) \tag{6.7}$$
$$\delta_b L_P = \sum_{i=1}^{N} \boldsymbol{\alpha}_i y_i$$
$$\delta_{\xi_i} L_P = C - \mu_i - \boldsymbol{\alpha}_i$$
$$\delta_{\mathbf{T}^{(\tilde{m},i)}} L_P = -\boldsymbol{\alpha}_i y_i \langle \boldsymbol{\mathcal{W}}, \Psi'(\boldsymbol{\mathcal{T}}_i)\rangle + \gamma \delta_{\mathbf{T}^{(\tilde{m},i)}} \left(\Omega(\boldsymbol{\mathcal{T}}_i)\right), \tag{6.8}$$

combining equations (6.7) and (6.8) together, the following equation comes out as a result,

$$\delta_{\mathbf{T}^{(\tilde{m},i)}} L_P \implies -\sum_{j=1}^{N} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y_i y_j \langle \Psi(\boldsymbol{\mathcal{T}}_i), \Psi'(\boldsymbol{\mathcal{T}}_j)\rangle + \gamma \delta_{\mathbf{T}^{(\tilde{m},i)}} \left(\Omega(\boldsymbol{\mathcal{T}}_i)\right) \tag{6.9}$$

Gradient Computation I: Equation (6.9) is combination of two terms with $\delta_{\mathbf{T}^{(\tilde{m},i)}}\left(\Omega(\mathcal{T}_i)\right)$ as a second term. The gradient w.r.t. each TT-CP factor $\mathbf{T}^{(\tilde{m},i)}$ can be straight forward in terms of a Hadamard product. This is given as follows,

$$\begin{aligned}\delta_{\mathbf{T}^{(\tilde{m},i)}}\left(\Omega(\mathcal{T}_i)\right) &= \delta_{\mathbf{T}^{(\tilde{m},i)}}\left(\left\|\mathcal{X}_i - [\![\mathbf{T}^{(1,i)}, \mathbf{T}^{(2,i)}, \cdots, \mathbf{T}^{(M,i)}]\!]\right\|^2\right)\\ &= 2\left(\left(\mathbf{T}^{(\tilde{m},i)}\left(\ast_{m=1,m\neq\tilde{m}}^{M}\mathbf{T}^{(m,i)^T}\mathbf{T}^{(m,i)}\otimes I\right)\right) - \left(x_{(i)}^{T}\left(\bigodot_{m=1,i\neq\tilde{m}}^{M}\mathbf{T}^{(m,i)}\otimes I\right)\right)\right)\\ &= 2\left(\left(\mathbf{T}^{(\tilde{m},i)}\left(\ast_{m=1,m\neq\tilde{m}}^{M}\mathbf{T}^{(m,i)^T}\mathbf{T}^{(m,i)}\otimes I\right)\right) - \left(x_{(i)}{}^{T}\left(\bigodot_{m=1,i\neq\tilde{m}}^{M}\mathbf{T}^{(m,i)}\otimes I\right)\right)\right).\end{aligned} \tag{6.10}$$

Hadamard product: The computation of the Hadamard (element-wise) product, $\mathcal{Z} = \mathcal{X} \circledast \mathcal{Y}$, of two tensors, $\mathbf{X}$ and $\mathbf{Y}$, of the same order and the same size can be performed very efficiently in the TT format by expressing the slices of the cores, $\mathcal{Z} \in \mathbb{R}^{R_{m-1}\times I_m\times R_m}$, as

$$\mathbf{Z}_{i_m}^{(m)} = \mathbf{X}_{i_m}^{(m)} \otimes \mathbf{Y}_{i_m}^{(m)}, \quad m = 1,\ldots,M, \;\; i_m \in \langle M\rangle. \tag{6.11}$$

This increases the TT ranks for the tensor $\mathcal{Z}$ to at most $R_m\tilde{R}_m$, $m \in \langle M\rangle$, but the associated computational complexity can be reduced from being exponential in M, $\mathcal{O}(I^M)$, to being linear in both I and M, $\mathcal{O}(IM(R\tilde{R})^2))$.

Gradient Computation II: The computation of first term from equation (6.9) is a bit tricky. Earlier in Section §2.3.2 the *kernel trick* is explained that can be used to write (6.2) into dual format and approximate $\langle\Psi(\mathcal{T}_i), \Psi(\mathcal{T}_j)\rangle$ by $\langle\Psi(\mathcal{T}_i), \Psi(\mathcal{T}_j)\rangle \approx K(\mathcal{T}_i, \mathcal{T}_j)$. Therefore, inheriting this idea into primal form leads to optimizing kernel formation (and eventually *factor kernels*) and learn non-linearity from data itself without explicitly knowing the nonlinear function (Ψ). Now, the main issue is with the computation of $\langle\Psi(\mathcal{T}_i), \Psi'(\mathcal{T}_j)\rangle$. This is computed with use of the following equality,

$$\langle\Psi(\mathcal{X}_i), \Psi(\mathcal{X}_j)\rangle = K(\mathcal{X}_i, \mathcal{X}_j) \implies \langle\Psi(\mathcal{X}_i), \Psi'(\mathcal{X}_j)\rangle = \frac{1}{2}\delta K(\mathcal{X}_i, \mathcal{X}_j).$$

Eventually the computation of $\langle\Psi(\mathcal{T}_i), \Psi^{'}(\mathcal{T}_j)\rangle$ depends on the computation of the gradient of the kernel function w.r.t. each of the "TT-CP" factors $(\mathbf{T}^{(\tilde{m},i)})$.

The work proposed in this paper is an extension of standard KSTM (§3.4.3). Hence, continuation of using *reproducing kernels* mentioned in Chapter 4 and Chapter 5 in sections §4.2.5, §5.2 are preferred. For instance, as the "TT-CP" decomposition is taken with DuSK (§4.2.5, [71]) is adapted here for the further computation of the gradient. This allows a way to exploit the fact that the data is given in the TT-CP format to aid the classification. However, the feature function $\Psi(\boldsymbol{\mathcal{X}})$ is to be defined implicitly through a kernel function.

$$\begin{aligned}\langle \Psi(\boldsymbol{\mathcal{T}}_i), \Psi(\boldsymbol{\mathcal{T}}_j)\rangle &= K(\boldsymbol{\mathcal{T}}_i, \boldsymbol{\mathcal{T}}_j)\\ &= K\left(\sum_{p=1}^{R}\prod_{m=1}^{M}\phi(\mathbf{T}_p^{(m,i)}), \sum_{q=1}^{R}\prod_{m=1}^{M}\phi(\mathbf{T}_q^{(m,j)})\right) \qquad (6.12)\\ &= \sum_{p,q=1}^{R}\prod_{m=1}^{M} k\left(\mathbf{T}^{(m,i)}(:,p), \mathbf{T}^{(m,j)}(:,q)\right). \qquad (6.13)\end{aligned}$$

Reformulating (6.12) in the following manner,

$$\begin{aligned}K(\boldsymbol{\mathcal{T}}_i, \boldsymbol{\mathcal{T}}_j) &= \sum_{p,q=1}^{R}\prod_{m=1}^{M}\exp\left(-\frac{\left\|(\mathbf{T}^{(m,i)}(:,p) - \mathbf{T}^{(m,j)}(:,q))\right\|^2}{2g^2}\right) \qquad (6.14)\\ &= \sum_{p,q=1}^{R}\exp\left(-\sum_{m=1}^{M}\frac{\left\|(\mathbf{T}^{(m,i)}(:,p) - \mathbf{T}^{(m,j)}(:,q))\right\|^2}{2g^2}\right) \qquad (6.15)\\ &= \sum_{p,q=1}^{R} u_{ij}^{pq} \qquad (6.16)\\ &= \mathbf{1}^T\mathbf{U}_{ij}\mathbf{1}, \qquad (6.17)\end{aligned}$$

where $\mathbf{U}_{ij} \in \mathbb{R}^{R\times R}$ and $\mathbf{1}$ is a vector of all ones. Also, we have

$$\mathbf{U}_{ij}(p,q) = \exp\left(-\sum_{m=1}^{M}\frac{\left\|(\mathbf{T}^{(m,i)}(:,p) - \mathbf{T}^{(m,j)}(:,q))\right\|^2}{2g^2}\right),$$

$$\text{or,}\quad \mathbf{U}_{ij}(p,q) = \exp\left(-\sum_{m=1}^{M}\frac{\left\|(\mathbf{T}_p^{(m,i)} - \mathbf{T}_q^{(m,j)})\right\|^2}{2g^2}\right). \qquad (6.18)$$

The derivative of the kernel function w.r.t. the TT-CP core can be computed in vector format as follows,

$$
\begin{aligned}
\frac{\partial K_{ij}}{\partial \mathbf{T}^{mk}_{lp}} &= \sum_{\tilde{p}\tilde{q}} \left(u^{\tilde{p}\tilde{q}}_{ij} \Big(\frac{\sum_{\tilde{l}} (\mathbf{T}^{(m,i)}_{\tilde{l}\tilde{p}} - \mathbf{T}^{(m,j)}_{\tilde{l}\tilde{q}})}{g^2} \Big) \delta_{l\tilde{l}} \delta_{p\tilde{p}} \delta_{i\tilde{k}} + u^{\tilde{p}\tilde{q}}_{ij} \Big(\frac{\sum_{\tilde{l}} (\mathbf{T}^{(m,j)}_{\tilde{l}\tilde{q}} - \mathbf{T}^{(m,i)}_{\tilde{l}\tilde{p}})}{g^2} \Big) \delta_{l\tilde{l}} \delta_{p\tilde{q}} \delta_{j\tilde{k}} \right) \\
&= \sum_{\tilde{q}} u^{p\tilde{q}}_{kj} \left(- \frac{(\mathbf{T}^{(m,k)}_{lp} - \mathbf{T}^{(m,j)}_{l\tilde{q}})}{g^2} \right) + \sum_{\tilde{p}} u^{\tilde{p}p}_{ik} \left(- \frac{(\mathbf{T}^{(m,k)}_{lp} - \mathbf{T}^{(m,i)}_{l\tilde{p}})}{g^2} \right) \\
&= \frac{-1}{g^2} \sum_{q} u^{pq}_{kj} (\mathbf{T}^{(m,k)}_{lp} - \mathbf{T}^{(m,j)}_{lq}) + u^{pq}_{ki} (\mathbf{T}^{(m,k)}_{lp} - \mathbf{T}^{(m,i)}_{lq}) \\
&= \frac{-1}{g^2} \sum_{q} (u^{pq}_{kj} + u^{pq}_{ki}) \mathbf{T}^{(m,k)}_{lp} - u^{pq}_{kj} \mathbf{T}^{(m,j)}_{lq} - u^{pq}_{ki} \mathbf{T}^{(m,i)}_{lq} \qquad (6.19)
\end{aligned}
$$

By going back to the computation of (6.9) and combining it with (6.10) and (6.19), the ensuing matrix formulation will be achieved,

$$
\begin{aligned}
\delta_{\mathbf{T}^{(\tilde{m},i)}} L_P = & - \underbrace{\sum_{j=1}^{N} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y_i y_j \left(\delta_{\mathbf{T}^{(\tilde{m},i)}} K\Big(\boldsymbol{\mathcal{T}}_i, \boldsymbol{\mathcal{T}}_j\Big) \right)}_{\mathcal{J}} \\
& + 2\gamma \left(\mathbf{T}^{(\tilde{m},i)} \left(\circledast_{m \neq \tilde{m}} \mathbf{T}^{(m,i)^T} \mathbf{T}^{(m,i)} \right) - \boldsymbol{\mathcal{X}}^{(i)}_{(\tilde{m})} \Big(\bigodot_{m \neq \tilde{m}} \mathbf{T}^{(m,i)} \Big) \right), \quad m \in \langle M \rangle. \qquad (6.20)
\end{aligned}
$$

Note: Equation (6.20) includes the opposite sign compare to (6.8) because the solution of Regularized STM lies in computing Gradient Ascent in dual form rather than Gradient Descent.

The compact and easy to understand form of (6.20) is written as follows,

$$\mathcal{J} = \sum_{j=1}^{N} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y_i y_j \delta_{\mathbf{T}^{(\tilde{m},i)}} K\Big(\boldsymbol{\mathcal{T}}_j, \boldsymbol{\mathcal{T}}_i\Big) = \mathbf{1}^T \tilde{\mathbf{C}} \mathbf{1},$$

$$\delta_{\mathbf{T}^{(\tilde{m},i)}} \mathcal{J} = \frac{1}{g^2} \Big[\mathbf{T}^{(\tilde{m})} \cdot diag(\tilde{\mathbf{C}} \cdot \mathbf{1}) - \mathbf{T}^{(\tilde{m})} \tilde{\mathbf{C}}\Big],$$

$$\tilde{\mathbf{C}} = \tilde{\mathbf{B}} \otimes \tilde{\mathbf{U}}\ ,\ \tilde{\mathbf{B}} = \mathbf{E}_R \otimes \mathbf{B},\ \mathbf{B}_{ij} = \boldsymbol{\alpha}_i y_i \boldsymbol{\alpha}_j y_j,\ \text{and } \tilde{\mathbf{C}} = \tilde{\mathbf{C}}^T,$$

$$\tilde{\mathbf{U}}_i = \exp\left(\frac{-1}{2g^2} \sum_{m=1}^{M} \sum_{s=1}^{I_m} \Big((\mathbf{T}^{(m,i)}(s,:))^T - \mathbf{T}^{(m,i)}(s,:)\Big)^2\right)$$

Computation of the gradient w.r.t. $\mathbf{T}^{(\tilde{m},1)}$ is as follows,

$$\delta_{\mathbf{T}^{(\tilde{m},i)}} K\Big(\boldsymbol{\mathcal{T}}_j, \boldsymbol{\mathcal{T}}_i\Big) = \mathbf{1} \delta_{\mathbf{T}^{(\tilde{m},i)}} \mathbf{U}_{ij} \mathbf{1}.$$

The gradient $\delta_{\mathbf{T}^{(\tilde{m},i)}} L_P$ using (6.19) in matrix form is written as follows,

$$\delta_{\mathbf{T}^{(\tilde{m},i)}} L_P = \frac{1}{g^2} \sum_{j=1}^{N} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y_i y_j \Big((\mathbf{T}^{(\tilde{m},i)} - \mathbf{T}^{(\tilde{m},j)}) \mathbf{U}_{ij}\Big)$$

Hence, the value of $\delta_{\boldsymbol{\mathcal{T}}_i}$ is given as follows,

$$\begin{aligned} \delta_{\mathbf{T}^{(\tilde{m},i)}} L_P = {} & \frac{1}{g^2} \sum_{j=1}^{N} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y_i y_j \Big((\mathbf{T}^{(\tilde{m},i)} - \mathbf{T}^{(\tilde{m},j)}) \mathbf{U}_{ij}\Big) \\ & + 2\gamma \left(\mathbf{T}^{(\tilde{m},i)} \Big(\circledast_{m \neq \tilde{m}} \mathbf{T}^{(m,i)^T} \mathbf{T}^{(m,i)}\Big) - \boldsymbol{\mathcal{X}}^{(i)}_{(\tilde{m})} \Big(\bigodot_{m \neq \tilde{m}} \mathbf{T}^{(m,i)}\Big)\right), \quad m \in \langle M \rangle. \end{aligned} \tag{6.21}$$

After having a compact form of the derivatives of the primal Lagrangian, it is necessary to find a way to use it in order to solve primal formulation (6.5). The next section discuss this in detail.

6.3 Optimizing Regularized Models

Optimizing a regularized Support Tensor Machine (STM) involves finding the optimal values for the model's parameters while considering a regularization term. The goal is to minimize the loss function, which measures the model's prediction error, while also controlling the complexity of the model using the regularization term. The regularization term helps prevent overfitting and promotes a more generalizable model.

To optimize a regularized STM, an iterative optimization approach is commonly used. This approach solves a series of optimization subproblems in each iteration until convergence is achieved. The specific optimization algorithm employed depends on the problem and the chosen regularization technique. Gradient-based methods are often a good choice for non-convex, non-linear optimization issues like SVMs. These methods update the model's parameters iteratively by computing the gradients of the loss function with respect to the parameters and adjusting the parameters in the direction of the steepest descent. The regularization term is incorporated into the gradient computation to control the complexity of the model.

When optimizing a regularized STM, there are several important considerations. First, the computational time required to find a solution of the desired quality and constraints, such as achieving a low error rate, can be high. Second, achieving theoretical efficiency, stability, and generalization can be challenging. Third, dealing with large datasets and high-dimensional data may pose storage issues. Lastly, finding a simple and effective approach can also be difficult.

6.3.1 Primal Dual SVM Relation

In the era of deep learning (DL), where unconstrained optimization problems are highly non-convex, it is crucial to explore the challenges associated with such problems. Efficient solutions to large-scale non-convex problems provide rich structures that can be exploited to address NP-hard problems [117, 123], which are often not achievable by solving relaxed convex problems. Apart from DL, algorithms that operate in high-dimensional spaces or work with nonlinear models, such as tensor-based models, are prone to be non-convex [78]. Some direct approaches for non-convex optimization, such as projected gradient descent (see chapter 2 [78]) and alternating minimization [114], have achieved remarkable success in various domains, typically outperforming relaxed approaches. These algorithms are simple, scalable, efficient, and fast in practice. However, the convergence of these heuristic approaches is still poorly understood.

The explicit computational cost of the nonlinear embedding function (Ψ) makes these non-convex optimizations even more expensive. Therefore, it becomes necessary to consider a dual formulation as a substitute for the primal objective function. This produces a convex quadratic programming (QP) objective function with bounded constraints, as discussed in §2.2.1. However, in the above-mentioned regularized STM, reaching such a well-defined problem is not straightforward. Hence, the following proposition provides a very good compromise between primal and dual for the full primal (6.8).

Additionally, the pre-computed gradient with respect to the "TTCP" factorization and the mentioned KKT conditions easily lead to the use of gradient-based iterative methods such as PGD.

The dual form from the Lagrangian for kernelised SVM has been explained earlier in Chapter chapter 2 in §2.3.3 ((2.22)) as well as for kernelised STM in Chapter chapter 3 in §3.4.3 ((3.24)). From there we can write vectorised form of Lagrangian (denoted as L_D) as follows,

$$L_D \colon \sum \boldsymbol{\alpha}_i - \frac{1}{2}\sum_{ij} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y_i y_j K(\mathbf{t}_i, \mathbf{t}_j). \tag{6.22}$$

Proposition 6.1 (Primal-Dual objective in the TTCP factors): The STM model has strong duality in CP factors. This means,

$$\delta_{\mathbf{T}^{(m,i)}} \left(L_P \Big|_{\mathcal{W}=\sum_{j=1}^{N} \boldsymbol{\alpha}_j y_j \Psi(\mathcal{T}_j)} \right) = \left(\delta_{\mathbf{T}^{(m,i)}} L_P\right)\Big|_{\mathcal{W}=\sum_{j=1}^{N} \boldsymbol{\alpha}_j y_j \Psi(\mathcal{T}_j)} = \delta_{\mathbf{T}^{(m,i)}} L_D,$$

where L_D is the dual form of the Lagrangian. ◊

Proof. **Substitute conditions and then differentiate Lagrangian:** This mean focusing on the first term that is given as follows,

$$\delta_{\mathbf{T}^{(m,i)}} \left(L_P \Big|_{\mathcal{W}=\sum_{j=1}^{N} \boldsymbol{\alpha}_j y_j \Psi(\mathcal{T}_j)} \right)$$

Dual cost: the vectorized form of dual Lagrangian function is taken as,

$$L_D \colon \sum \boldsymbol{\alpha}_i - \frac{1}{2}\sum_{ij} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y_i y_j K(\mathbf{t}_i, \mathbf{t}_j), \tag{6.23}$$

where kernel matrix is computed using RBF,

$$K(\mathbf{t}_i, \mathbf{t}_j) = \sum_{pq} \exp\left(- \sum_{m=1}^{M} \frac{\left\| \mathbf{t}_p^{mi} - \mathbf{t}_q^{mj} \right\|}{2g^2} \right),$$

The derivative of L_D depends on computing derivative of kernel matrix w.r.t. each TTCP factor vector $\mathbf{t}_{\tilde{p}}^{\tilde{m}i}$,

$$\delta_{\mathbf{t}_{\tilde{p}}^{\tilde{m}i}} K(\mathbf{t}_i, \mathbf{t}_j) = \sum_{pq} \mathbf{U}_{ij}(p,q) \left[\frac{-1}{2g^2} \delta_{\mathbf{t}_{\tilde{p}}^{\tilde{m}i}} \left\| \mathbf{t}_p^{\tilde{m}i} - \mathbf{t}_q^{\tilde{m}j} \right\|^2 \right],$$

where $\mathbf{U}_{ij}$ (6.18) is proposed earlier and included here to make it equation look more elegent and easy to understand,

$$\delta_{\mathbf{t}_{\tilde{p}}^{\tilde{m}\tilde{i}}} K(\mathbf{t}_i, \mathbf{t}_j) = \sum_{pq} \mathbf{U}_{ij}(p,q) \left[\frac{-1}{g^2} \left(\mathbf{t}_p^{\tilde{m}i} - \mathbf{t}_q^{\tilde{m}j} \right) \left(\delta_{\tilde{p}p}\delta_{\tilde{i}i} - \delta_{\tilde{p}q}\delta_{\tilde{i}i} \right) \right],$$

the right hand side can be reformulated as follows,

$$\frac{-1}{g^2} \left(\sum_q \mathbf{U}_{ij}(\tilde{p},q)\mathbf{t}_{\tilde{p}}^{\tilde{m}i}\delta_{\tilde{i}i} - \sum_q \mathbf{U}_{ij}(\tilde{p}q)\mathbf{t}_q^{\tilde{m}j}\delta_{i\tilde{i}} \right) + \frac{1}{g^2} \left(-\sum_p \mathbf{U}_{ij}(p,\tilde{p})\mathbf{t}_p^{\tilde{m}i}\delta_{\tilde{i}j} + \sum_p \mathbf{U}_{ij}(p,\tilde{p})\mathbf{t}_{\tilde{p}}^{\tilde{m}j}\delta_{\tilde{i}j} \right),$$

hence, (6.23) can be modified as following,

$$\begin{aligned} \delta_{t_{\tilde{p}}^{\tilde{m}\tilde{i}}} L_D = {} & \frac{1}{2g^2} \left[\sum_{qj} \boldsymbol{\alpha}_{\tilde{i}}\boldsymbol{\alpha}_j y_{\tilde{i}} y_j \mathbf{U}_{\tilde{i}j}(\tilde{p},q)\mathbf{t}_{\tilde{p}}^{\tilde{m}\tilde{i}} - \sum_{qj} \boldsymbol{\alpha}_{\tilde{i}}\boldsymbol{\alpha}_j y_{\tilde{i}} y_j \mathbf{U}_{\tilde{i}j}(\tilde{p},q)\mathbf{t}_q^{\tilde{m}j} \right] \\ & - \frac{1}{2g^2} \left[\sum_{pi} \boldsymbol{\alpha}_i\boldsymbol{\alpha}_{\tilde{i}} y_i y_{\tilde{i}} \mathbf{U}_{i\tilde{i}}(p,\tilde{p})\mathbf{t}_p^{\tilde{m}i} - \sum_{pi} \boldsymbol{\alpha}_i\boldsymbol{\alpha}_{\tilde{i}} y_i y_{\tilde{i}} \mathbf{U}_{i\tilde{i}}(p,\tilde{p})\mathbf{t}_{\tilde{p}}^{\tilde{m}\tilde{i}} \right], \end{aligned}$$

furthermore,

$$\delta_{\mathbf{t}_{\tilde{p}}^{\tilde{m}\tilde{i}}} K(\mathbf{t}_i, \mathbf{t}_j) = \frac{1}{g^2} \sum_{qj} \boldsymbol{\alpha}_{\tilde{m}}\boldsymbol{\alpha}_j y_{\tilde{i}} y_j \mathbf{U}_{\tilde{i}j}(\tilde{p},q) \left[\mathbf{t}_{\tilde{p}}^{\tilde{m}\tilde{i}} - \mathbf{t}_q^{\tilde{m}j} \right]. \tag{6.24}$$

Differentiate Lagrangian then substitute conditions: This concerns the second term that is given as,

$$\left(\delta_{\mathbf{T}^{(m,i)}} L_P \right) \Big|_{\mathcal{W} = \sum_{j=1}^{N} \boldsymbol{\alpha}_j y_j \Psi(\mathcal{T}_j)}$$

Taking equalities from (6.14) and (6.18), the following equations are computed,

$$\mathbf{U}_{ij}(p,q) = \exp \left(-\sum_{m=1}^{M} \frac{\left\| (\mathbf{t}^{(m,i)}(:,p) - \mathbf{t}^{(m,j)}(:,q)) \right\|^2}{2g^2} \right),$$

computing gradient of $\mathbf{U}$ and from this reaching to the computation of gradient of kernel matrix. The gradient of $\mathbf{U}$ in vector form is as follows,

$$\delta_{\mathbf{t}_{\tilde{p}}^{\tilde{m}\tilde{i}}} \mathbf{U}_{ij}(p,q) = \mathbf{U}_{ij}(p,q) \left[\frac{-1}{2g^2} \delta_{\mathbf{t}_{\tilde{p}}^{\tilde{m}i}} \left(\left\| \mathbf{t}_p^{\tilde{m}i} - \mathbf{t}_q^{\tilde{m}j} \right\|^2 \right) \right]$$

hence, gradient of $\mathbf{K}$ is as follows,

$$\delta_{\mathbf{t}_{\tilde{p}}^{\tilde{m}i}} K_{ij} = \frac{-1}{g^2}\left[\sum_q \mathbf{U}_{ij}(q,\tilde{p})\mathbf{t}_{\tilde{p}}^{\tilde{m}i} - \sum_q \mathbf{U}_{ij}(q,\tilde{p})\mathbf{t}_q^{\tilde{m}j}\right],$$

this implies,

$$\implies \frac{1}{g^2}\sum_q \mathbf{U}_{ij}(q,\tilde{p})\left[\mathbf{t}_q^{\tilde{m}j} - \mathbf{t}_{\tilde{p}}^{\tilde{m}i}\right],$$

the gradient of primal Lagrangian concludes as,

$$\delta_{\mathbf{t}_{\tilde{p}}^{\tilde{m}i}} L_P = \sum_{j=1}^{N} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y_i y_j \left(\frac{1}{g^2}\sum_{q=1}^{R} \mathbf{U}_{ij}(\tilde{p},q)\left[\mathbf{t}_{\tilde{p}}^{\tilde{m}i} - \mathbf{t}_q^{\tilde{m}j}\right]\right). \tag{6.25}$$

Henceforth from (6.24), and (6.25),

$$\delta_{\mathbf{t}_p^{mi}} L_P = \delta_{\mathbf{t}_p^{mi}} L_D. \qquad \square$$

Hence, the Lagrangian (primal) mentioned in (6.5) can be reformulated in the dual form using above proposition (6.1) in the following objective function,

$$F = \sum_{i=1}^{N} \boldsymbol{\alpha}_i - \frac{1}{2}\sum_{i,j=1}^{N} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y_i y_j K(\boldsymbol{\mathcal{T}}_i, \boldsymbol{\mathcal{T}}_j) - \gamma \sum_{i=1}^{N} \Omega(\boldsymbol{\mathcal{T}}_i) \tag{6.26}$$

When equation (6.26) is closely examined, it can be observed that the partial part is primarily associated with the solving of the TT-MMK method proposed in Chapter 4. In addition to TT-MMK, the low-rank approximation is simultaneously updated as an additional term. Consequently, this objective function is referred to as the Simultaneously-updated STM.

6.3.2 Partial Simultaneous Update using Gradient Descent

The partial simultaneous update occurs when the value of γ in (6.26) is set to 0. This means that the TT-CP factorization term $\Omega(\boldsymbol{\mathcal{X}})$ is not updated simultaneously. This remaining problem is exactly similar to optimizing the dual problem (3.24) using `LIBSVM`, as mentioned in §3.4.3. The Sequential Minimization Optimization (SMO) method, used in the LIBSVM library, optimizes the KSTM (3.24). In this proposed work, the focus of the developed method is on using the PGD method instead of the standard SMO for solving the SVM objective function. The overview, advantages, drawbacks, and intricacies of PGD are explained further in this section.

Note: One important observation to make is that initializing the full objective function (6.26) from the previous state-of-the-art (TT-CP based) method (TT-MMK, §4.2.6) almost surely achieves the same accuracy for any kind of tensorial dataset.

6.3.3 Full Simultaneous Update using Gradient Descent

For solving the objective function given in (6.26), each parameter is updated by fixing all except one optimization parameters. Typically, the initilization of the hyperparameters has a significant impact for reaching to optimal solution. As (6.26) is a composition of TT-MMK and a regualrization term. Therefore, the two hyperparameters $\boldsymbol{\alpha}, b$ can be initialized from the optimized TT-MMK solution. The gradients w.r.t. each parameter are computed as follows,

1. **Updating $\mathbf{T}^{(\tilde{m},i)}$:** As mentioned that the PGD method is used for solving optimization problem mentioned in (6.26). Hence, the derivative w.r.t. $\mathbf{T}^{(\tilde{m},i)}$ will be given as follows,

$$\frac{\partial F}{\partial \mathbf{T}^{(\tilde{m},i)}} = \frac{\partial \Omega}{\partial \mathbf{T}^{(\tilde{m},i)}} + \frac{\partial P}{\partial \mathbf{T}^{(\tilde{m},i)}} \tag{6.27}$$

   where from equation (6.26), we have

$$P(\boldsymbol{\mathfrak{T}}_i) = \boldsymbol{\alpha}_i - \frac{1}{2}\sum_{j=1}^{N} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y_i y_j K(\boldsymbol{\mathfrak{T}}_i, \boldsymbol{\mathfrak{T}}_j)$$

   The computation of the gradient w.r.t. the tensor factor $\boldsymbol{\mathfrak{T}}_i$ can be computed using $\delta L_{\mathbf{T}^{(\tilde{m},i)}}$ from (6.21).

2. **Updating $\boldsymbol{\alpha}_i$:**

$$\begin{aligned}
L(\boldsymbol{\alpha}_k) &= \boldsymbol{\alpha}_k - \frac{1}{2}\sum_{j=1}^{N} \boldsymbol{\alpha}_j y_j \boldsymbol{\alpha}_k y_k k(x_k, x_j) \\
&= \boldsymbol{\alpha}_k - \frac{1}{2}\boldsymbol{\alpha}_k y_k \boldsymbol{\alpha}_k y_k k(x_k, x_k) - \sum_{j=1, j\neq k}^{N} \boldsymbol{\alpha}_j y_j \boldsymbol{\alpha}_k y_k k(x_k, x_j) \\
\tfrac{\partial L}{\partial \boldsymbol{\alpha}_k} &= 1 - \boldsymbol{\alpha}_k y_k^2 k(x_k, x_k) - y_k \sum_{j=1, j\neq k}^{N} \boldsymbol{\alpha}_j y_j k(x_k, x_j) \\
\tfrac{\partial L}{\partial \boldsymbol{\alpha}_k} &= 1 - y_k \sum_{j=1}^{N} \boldsymbol{\alpha}_j y_j k(x_k, x_j).
\end{aligned} \tag{6.28}$$

Gradient of L w.r.t. $\boldsymbol{\alpha}$ is $\Delta(L) = \left[\frac{\partial L}{\partial \boldsymbol{\alpha}_1}, \frac{\partial L}{\partial \boldsymbol{\alpha}_2}, \ldots, \frac{\partial L}{\partial \boldsymbol{\alpha}_N}\right]$,

$$\frac{\partial F}{\partial \boldsymbol{\alpha}_i} = 1 - y_i \sum_{j=1}^{N} \boldsymbol{\alpha}_j y_j K(\mathfrak{T}_i, \mathfrak{T}_j). \tag{6.29}$$

The projection gradient method works well when the objective function is convex and the feasible domain S is a set of box constraints such as bound on the optimization variable.

Projected Gradient Descent for Bounded Constraints

The PGD is an iterative approach where the *feasible direction* $d \neq 0$ is updated along the direction vector.

Algorithm 6.1: Projected Gradient Descent

```
**Initialize** : $\mathbf{x}_0$
**Output:** Optimized $f(\mathbf{x}_i)$ s.t. $a_i < \mathbf{x}_i < b_i, \quad \forall i \in < M >$
**Update:** $\mathbf{x}_{k+1} = P(\mathbf{x}_k - \eta_k \nabla f(\mathbf{x}_k))$, where $\eta$ is the step size (learning rate)
**Projection :** $P[i] \begin{cases} \mathbf{x}_i & a_i < \mathbf{x}_i < b_i \\ b_i & \mathbf{x}_i \geqslant b_i \\ a_i & \mathbf{x}_i \leqslant a_i \end{cases}$
```

The algorithm 6.1 converges to a solution that minimizes the objective function while satisfying the bounded constraints. PGD is particularly useful in scenarios where the solution must lie within a predefined range, such as in optimization problems with physical or practical constraints.

The computation of the learning rate η and its specific selection lead to different kinds of PGD methods. One such method, "the Armijo rule along the projection arc," was introduced in [18], originating from [17]. Later, this approach of finding the best learning rate proved to be highly effective in a wide range of fields, from machine learning to basic advancements in algorithms such as NMF [109].

Lemma 6.2:
Let $\Omega \subseteq \mathbb{R}^n$ and suppose that $f \in C^2(\Omega)$. Then let

$$x^{k+1} = x^k + \eta^k d^k, \quad \text{where } \langle d^k, \nabla f(x^k) \rangle < 0.$$

Then for *sufficiently small* $\mathbf{s} > 0$ one has

$$f(x^{k+1}) < f(x^k).$$

The Armijo rule says [19],

$$f(x^k) + \langle \nabla f(x^k), d^k \rangle < f(x^k) + s\eta \langle \nabla f(x^k), d^k \rangle$$

Hence, the use of PGD has extend to the KSVM in [87], and with Armijo rule ([10]) in [19]. In current scenario, we extend the KSVM approach on the KSTM while using *TT-factors*, and Lemma 6.2. In the current case the bounded constraint is $\mathcal{S} = \{\boldsymbol{\alpha} \geqslant 0 \ \& \ \sum_{i=1}^{N} \boldsymbol{\alpha}_i y_i = 0\}$.

Algorithm 6.2: Projected Gradient Descent for KSTM.

```
**Input:** Obj($\mathfrak{X}$), $\left[\boldsymbol{\alpha}_0; \mathbf{T}_0^{(\tilde{m},i)}; b_0\right]$
**Output:** Optimal value for $\left[\boldsymbol{\alpha}; \mathbf{T}^{(\tilde{m},i)}; b\right]$
Set $v_k = \left[\boldsymbol{\alpha}_k; \mathbf{T}_k^{(\tilde{m},i)}; b_k\right] = \left[P_S(\boldsymbol{\alpha}_0); P_S(\mathbf{T}^{(\tilde{m},i)_0}); P_S(b_0)\right]$
**for** $k = 0, 1, \ldots k_{max}$ **do**
    Search direction: $d_k = P_S(v_k - \frac{\partial \text{Obj}(\mathfrak{X})}{\partial v_k}) - v_k$
    Step size $st_k = \max_{s \in \{0,1,2,\ldots\}} st^s$, such that
    $\text{Obj}(v_k + st^s * d_k) \leqslant \text{Obj}(v_k) + cst^s(\frac{\partial \text{Obj}(\mathfrak{X})}{\partial v_k})^T d_k, \quad c \in (0,1)$
    Providing $\left(\frac{\partial \text{Obj}(\mathfrak{X})}{\partial v_k}^T\right) d_k < 0$
    $v_{k+1} = v_k + st_k d_k$
    Convergence tolerance: $\|(d_k)\|_2 \leqslant \text{tol}$, or $t_k \leqslant \text{tol}$, or $d_k$ is non descent
**end for**
```

This used condition of projected gradient methods, ensures the sufficient decrease of the function value per iteration. This leads to minimal of the function without diverging or oscillating.

6.3.4 Gradient Descent Enhancement

The approach to find the optimal solution for the objective function in (6.26) would be to first compute the *TTCP Fcatorization* with the initialization of *Unconstrained TTCP* fcatorization from the Chapter 4. This computation is independent to the STM update. Problem can be broken into following two steps,

- Step 1: **TTCP Factorization:** This form is achieved from equation (3.6) and mentioned in Chapter 4,
  - Step 1: **TT Decomposition:** Finding the best low-rank approximation (Initialization) (Computing the TT decomposition using the uniqueness enforcing

TT-SVD Algorithm 4.1),

$$\min_{\mathfrak{C}^{(m)}} \left\| \mathfrak{X} - \langle\!\langle \tilde{\mathfrak{C}}^{(1)}, \tilde{\mathfrak{C}}^{(2)}, \cdots, \tilde{\mathfrak{C}}^{(M)} \rangle\!\rangle \right\|_F^2$$

– Step 2: **Exact TT-CP Expansion**

$$\sum_{r_0,\ldots,r_M} \mathfrak{C}^{(1)}_{r_0,i_1,r_1} \mathfrak{C}^{(2)}_{r_1,i_2,r_2} \cdots \mathfrak{C}^{(M)}_{r_{M-1},i_M,r_M} = \sum_{r=1}^{R} \hat{\mathbf{T}}^{(1)}_{i_1,r} \hat{\mathbf{T}}^{(2)}_{i_2,r} \cdots \hat{\mathbf{T}}^{(M)}_{i_M,r}$$

by merging the ranks $r_1, r_2, \ldots r_M$ into one index such that,

$$r = r_1 + (r_2 - 1)R_1 + \ldots + (r_M - 1) \prod_{\ell=1}^{M-1} R_\ell, \;\; r = 1, \ldots, R, \;\; R = R_1 \cdots R_M,$$

and introducing the CP factors

$$\hat{\mathbf{T}}^{(m)}_{i_m,r} = \mathfrak{C}^{(m)}_{r_{m-1},i_m,r_m}, \quad m = 1, \ldots, M.$$

– Step 3: **Norm Equilibration:**

$$n_r = \left\| \hat{\mathbf{T}}^{(1)}_r \right\| \cdots \left\| \hat{\mathbf{T}}^{(M)}_r \right\|,$$

and distribute this norm equally among the factors,

$$\mathbf{T}^{(m)}_r := \frac{\hat{\mathbf{T}}^{(m)}_r}{\left\| \hat{\mathbf{T}}^{(m)}_r \right\|} \cdot n_r^{1/M}, \qquad m = 1, 2, \cdots, M.$$

- Step 2: Solving **dual-STM optimization using PGD**

$$\max_{\boldsymbol{\alpha}, \mathfrak{T}^{(m)}_i} \; \sum_{i=1}^{N} \boldsymbol{\alpha}_i - \frac{1}{2} \sum_{i,j=1}^{N} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y_i y_j K(\mathfrak{T}_i, \mathfrak{T}_j) - \gamma \sum_{i=1}^{N} \Omega(\mathfrak{T}_i)$$
$$\text{subject to } \; 0 \leqslant \boldsymbol{\alpha}_i \leqslant C, \;\; \sum_{i=1}^{N} \boldsymbol{\alpha}_i y_i = 0, \quad i \in 1, 2, \ldots, N. \qquad (6.30)$$

These above-mentioned steps are re-formulated in an Algorithm 6.3.

Algorithm 6.3: Simultaneously-update for TT-MMK (SimUpTT-MMK).

```
Input: data $\{\mathcal{X}_n\}_{n=1}^{N}\mathbb{R}^{I_1\times I_2\times\ldots\times I_M}$, TTCP-rank $R$.
Output: Optimized Parameter $\mathbf{T}^{(m,n)}, \boldsymbol{\alpha}, b$
for $n = 1$ to $N$ do
    for m = 1 to N do
        Compute TTCP approximation of $\mathcal{X}_n \cong [\![\mathbf{T}^{(1,n)}, \mathbf{T}^{(2,n)}, \cdots, \mathbf{T}^{(M,n)}]\!]$ as mentioned in Algorithm 4.2
        Initialize $\boldsymbol{\alpha}$ and $b$ from TT-MMK Chapter 4
        Compute kernel matrix $K$ using (6.15)
        Updating $\mathbf{T}^{(m,n)}, \boldsymbol{\alpha}$ and $b$ by Algorithm 6.2
    end for
end for
```

6.3.5 Full Simultaneous update TT-MMK

The PGD (Algorithm 6.2) plausibly returns the optimal solution. This is immediately true if the objection function (6.26) is β-smooth as discussed now.

Theorem 6.3:
[22] Let F be $\beta-$smooth on $\mathbb{R}^m$ and bounded from below. Let $st_k = st = \frac{1}{\beta}$ for all $k \in d$. Then for every $k \in d$,

$$\min_{i\leqslant k}\|\nabla F(\theta_i)\|_2 \leqslant \left(\frac{2\beta}{k+1}(F(\theta_0) - F(\theta_{k+1}))\right)^{1/2} = \mathcal{O}(k^{-1/2}). \tag{6.31}$$

$\diamondsuit$

As gradient of the objective is Lipschitz if it is $\beta-$smooth. Therefore, dual form is bounded below. This proof can give theoretical guarantee to convergence of the method. Once the optimized parameters (Algorithm 6.3) are computed then the decision function can be used for predicting labels of tensor input. The decision function in dual form is given as follows,

$$f(\mathcal{T}) = \sum_{i=1}^{N}\boldsymbol{\alpha}_i y_i \Psi(\mathcal{T}_i)^T\Psi(\mathcal{T}) + b = \sum_{i=1}^{N}\boldsymbol{\alpha}_i y_i K(\mathcal{T}_i, \mathcal{T}) + b \quad \text{by (6.12)}$$

Using, (6.15)where
$$f(\mathcal{T}) = \sum_{i=1}^{N}\boldsymbol{\alpha}_i y_i\left(\sum_{p,q=1}^{R}\exp\left(-\sum_{\tilde{m}=1}^{M}\frac{\left\|(\mathbf{T}^{(\tilde{m},i)}(:,p) - \mathbf{T}^{(\tilde{m},i)}(:,q))\right\|^2}{2g^2}\right)\right) + b, \tag{6.32}$$

where, $\mathbf{T}^{(\tilde{m})}$ belongs to a new point $\boldsymbol{\mathfrak{X}}$. The function in equation (6.32) shows the *nonlinear decision boundary* for *Primal Support Tensor Train Machine*. The value of b is given as follows (only for those indices i that must be constrained, $0 < \boldsymbol{\alpha}_i < C$, and value of b for each of these constrained indices i would be same),

$$\begin{aligned} b &= y_i - \boldsymbol{\mathcal{W}}^T \Psi(\boldsymbol{\mathfrak{T}}_i) \\ &= y_i - \sum_{j=1}^{N} \boldsymbol{\alpha}_j y_j \langle \Psi(\boldsymbol{\mathfrak{T}}_j), \Psi(\boldsymbol{\mathfrak{T}}_i) \rangle \\ &= y_i - \sum_{j=1}^{N} \boldsymbol{\alpha}_j y_j K(\boldsymbol{\mathfrak{T}}_i, \boldsymbol{\mathfrak{T}}_j) \end{aligned} \tag{6.33}$$

or average over support indices N_s such that $b = \frac{1}{N_s} \sum_{i=1}^{N_s} (= y_i - \sum_{j=1}^{N} \boldsymbol{\alpha}_j y_j K(\boldsymbol{\mathfrak{T}}_i, \boldsymbol{\mathfrak{T}}_j))$.

6.4 Numerical Assessment

- **Synthetic Data Collection:** The dataset is collected as mentioned in Chapter 6 with the noise in frequency (*leaf*). For this case, 30 samples are taken with $50-50$ division of positive and negative labels. The data size is $[50, 50, 50]$ with the noise level 0.1.

- **Parameter Tuning:** The entire SimUpTT-MMK model depends on some parameters and hyperparameters. First, to simplify the selection of TT-CP ranks, we take all TT-CP ranks equal to the same value $R \in \{1, 2, \dots 10\}$. The other two parameters are $\boldsymbol{\alpha}$ and b. The $\boldsymbol{\alpha}$ and b are initialized from the dual solution of STM from our previous work Chapter 4. The trade-off parameter C, the scale parameter γ, and the Gaussian kernel parameter g are also optimization parameters. For tuning $R, \boldsymbol{\alpha}, b$ and γ to the best classification accuracy, the *k-fold cross validation* (§2.4.1) with $k = 5$. Along with this, all computations are supposed to be repeated 50 times and average the accuracy over these runs. The other two optimization parameters (g, C) lies within the range $\{2^{-8}, 2^{-7}, \dots, 2^{7}, 2^{8}\}$. While the hyperparameter γ is taken from $\{2^{-3}, 2^{-2}, \dots, 2^{2}, 2^{3}\}$. The another crucial hyperparameter take role in PGD method, the so-called 'stepsize'. The stepsize is initialized within $\{0.5, 1\}$ range and reduced it by using line-search method in PGD approach as mentioned in Algorithm 6.2. This ensures a confident and reproducible comparison of different techniques.

6.4.1 Convergence Analysis

Observations as optimization fails to give better results. Some observation for the experimental analysis has been made. Firstly, often the stepsize reaches the threshold before reaching the optimum of objective function. Second, the objective function might be overfitiing the data, reason being amount of data is small and data might be sparse. The explanation for this issue is that there is a bog gap between training and test error. This means model has high variance and finding best trade-off between bias and varinace is a tricky scenario. Third, the objective function is a combination of SVM part and low-rank approximation. Therefore, it is plausible that PGD is reaching to "singularity" rather reaching to an optimum. Fourth, the best of the worst classification is achieved mostly on the edge point of the grid of γ, let say if $\gamma \in \{2^{-3}, 2^{-2}, \ldots, 2^{3}\}$ then high chances that optimum value is achieved on either 2^{-3} or 2^{3}. Also, as much as the range is increased the shift of optimum moves according to the shift and again to the edges. This also give a supportable reasoning to the argument of reaching to *singularity.* Last but not least observation which is in favour of proposed method is that as much as the noise increase in data, the classification accuracy decreases.

Plausible Solutions as mentioned earlier the heuristic non-convex approaches have issue with convergence. Usually PGD stop making progress in optimizing objective function when "it actually converges to optimum" but *vanishing gradient* could be one issue to stop progress toward optimum. An inspiration of finding solution to this concern could come from one of the approach used in DL to deal with non-convex optimization. The route to reach to the convergence at apt time is to find a safeguard such as early stopping, more robust truncation conditions. In case of convex optimization it only happens when problem solution reaches to global optimum. However, for non-convex, non-smooth case, another issue could be that each time optimum get stuck at the local non-minimum minima. As PGD algorithm converge to global optimum in polynomial time with a linear rate of convergence for well-structured objective function and constraint sets. Therefore, the requirement of a well-defined optimization training algorithm is needed.

Analysis So far the theoretical explanation given in this chapter provides an advantage for a generic approach to Regularized STM models. As mentioned in [42], the Tucker format has an edge over CP-based decomposition for STM models. Tucker decomposition is a modelling technique that is particularly well-suited for high-dimensional data with a limited number of available samples. Additionally, it offers a more parsimonious and compact model. In the case of skewed data in terms of dimensions or significant differences in the sizes of different modes, by fully exploiting the multilinear ranks, Tucker decomposition allows for the freedom to choose a different rank for each mode, which can be useful in such situations. Moreover, Tucker decomposition explicitly models the interaction between factor matrices in different modes, enabling a finer grid search over a larger modelling space. This can help to identify complex patterns and relationships in the data that might be missed by other modelling techniques. Overall, Tucker decomposition is a powerful tool for analysing and modelling high-dimensional data, offering important advantages over other methods.

Note: Given certain advantage of Tucker model and proposed new kernel in Chapter 5 can give further advancement in classification accuracy if aligned in Regularized STM formulation. At present, only concern is to develop a well-defined optimization regime.

6.5 Chapter Summary

This chapter represents the final part of the contributed work in this thesis. The introduction section (§6.1) provides an overview of the main novelty behind the research idea. It discusses the importance of having a regularized model and highlights the challenges involved in finding the optimal solution.

The subsequent section (§6.2) delves deeper into these models and explores the traditional approaches for dealing with them. An alternative approach to address regularized models is presented in §6.2.1. This approach involves computing the complex gradient of the Lagrangian (primal) function with respect to each parameter of the objective function.

The following section (§6.3) utilizes the computed gradient and introduces an iterative method for non-convex optimization. Prior to discussing the iterative method, the main contribution of this work is presented, which is the “primal-dual SVM relation” in TTCP format (§6.3.1). Furthermore, the full simultaneously-updated STM model and the corresponding algorithms (Algorithm 6.2, 6.3) for finding solutions are introduced in §6.3.3 and §6.3.5 respectively.

The suggested theorems and lemmas provide additional theoretical guarantees. In section §6.4, a synthetic experiment is explained along with crucial observations resulting from running these experiments. This further emphasizes the need for a comprehensible model and solution approaches.

CHAPTER 7 SUMMARY AND OUTLOOK

7.1 Conclusions . . . 113
7.2 Future Research Directions . . . 115
7.2.1 Direction 1: Neural Support Tensor Machine (Multiple Kernel Learning) . . . 115
7.2.2 Direction 3: Gene Expression Recovery . . . 119
7.3 Outlook . . . 122

Contents

7.1 Conclusions

Chapter 1 makes a connection between two fields and explains the issues to solve a non-linear boundary binary classification model. It shows a substantial necessity to include tensor algebra into such a classification model. After providing a prerequisite detailed description of forming the model in Chapter 2 and Chapter 3, this thesis predominantly focuses on three different directions related to solving the KSTM model (§3.4.3), distributed along three chapters (4, 5, 6).

Chapter 4 centers on developing a new low-rank tensor approximation that is grounded on the TT decomposition. Along with it, the chapter proposes a new kernel model for SVM classification of tensor input data. The kernel extends the DuSK (§4.2.5) approach [72, 71] to the TT decomposition of the input tensor with enforced uniqueness and norm distribution. The TT decomposition is considered more reliable than the CP decomposition used in the original DuSK kernel. Using fMRI and Hyperspectral Image datasets, higher classification accuracy is demonstrated for the new TT-MMK method, even with unsophisticated choices of the TT ranks, for a wide range of classification problems. It is found that each component of the proposed scheme, including uniqueness-enforced TT, TT-CP expansion, and norm equilibration, is crucial for achieving this accuracy.

The main part of Chapter 5 is figuring out the reasoning for the success of DuSK (§4.2.5), especially when using the CP (TT-CP) format of the tensor input data. The after-effect of this analysis led to the development of a new *reproducing kernel*, the WSEK (§5.2.2). This relies on a weighted format of Tucker decomposition rather than the CP format. A detailed numerical study on the synthetic data experiments (§5.3) shows the persuasiveness and feebleness of the WSEK kernel, and real-world experiments (ADNI and ADHD datasets) show superior performance of WSEK both in terms of classification accuracy and running time. It is concluded that the classification information of the datasets is mostly hidden in the subspaces of the Tucker decomposition (hence the previously observed good performance of DuSK) but that classification can be improved by taking the singular values into account (as done in WSEK). Furthermore, computing the Tucker decomposition of tensor inputs is straightforward and efficient, resulting in an all-around very robust tensor kernel.

So far, research has focused either on defining a better kernel function in high-dimensional embedded space or defining a computationally efficient way to construct these reproducing kernels by developing better feature extraction techniques (low-rank approximation). The third happy hunting ground for the KSTM model would be to find a better optimization regime while combining both aspects of improving a model (kernel function and feature extraction). Chapter 6 addresses this issue and develops a solution for the highly non-linear KSTM model by proposing a new optimization regime. The idea of using simultaneous update of TTCP factors with SVM variable update goes one step ahead of the standard STM model. The proposition of the primal-dual relationship in the TTCP format of the regularized STM is a novel work of this chapter. The proposed algorithms, lemmas, and theorems provide the theoretical guarantee of convergence to a local minimum (in fact, global) of the non-convex optimization problem.

The prominent highlight of offered research work is attaining 15% increment in classification accuracy for highly complex fMRI data with small data samples notwithstanding.

Supplemental Contribution [84]:

In this work, authors introduced a method that combines a well-established nonlinear identification technique, specifically the bilinear approach, with the advantages of neural networks (NNs). Fitting bilinear systems accurately requires recovering the corresponding Markov parameters from input and output measurements, followed by a realization algorithm similar to the one proposed by Isidori. However, in this approach, NNs are utilized as a surrogate data simulator to generate input-output (i/o) data sequences. Authors then employ classical realization theory to construct an interpretable bilinear model, which can be further utilized to optimize engineering processes through robust simulations and control design. This integration of the bilinear method with NNs provides a novel framework for enhanced system identification and engineering applications. The use of NNs as a data simulator offers flexibility and efficiency in generating i/o data, while the bilinear model allows for interpretable and optimized simulations and control designs.

Note: While providing a detailed description of the work mentioned in [84] is beyond the scope of this thesis, it can be regarded as an application of machine learning (ML) models, specifically neural networks (NNs) with fewer hidden layers, such as support vector machines (SVMs).

7.2 Future Research Directions

The successful projects are included as contributed work in the thesis. In this section, several research ideas that were explored during the timeline of doctoral studies are presented, along with some remaining open questions. Some of these ideas have compelling models that directly align with the current research and the need for Generalized AI. The aforementioned model, as well as these futuristic models, have the potential to be useful in various applications such as 3D image analysis (e.g., MRI, EEG), anomaly detection, signal processing, medical equipment technology, and disease diagnosis (e.g., cancer, Alzheimer).

7.2.1 Direction 1: Neural Support Tensor Machine (Multiple Kernel Learning)

The one reason for the success of Weighted HOSVD and its related WSEK is the equal distribution of weights to all the feature vectors. This approach involves automatically optimizing a highly nonlinear objective function to assign weights to each neuron in the neural network. From these two techniques, an immediate intuition arises regarding the development of a kernel function based on low-rank approximation, such as TTCP (Algorithm 4.2) or Weighted HOSVD (Algorithm 5.2), which can learn automatically and provide optimum weighting to the most important feature vectors.

However, running an NN-based model can be computationally expensive for large-scale problems in high-dimensional space. In addition, other optimization issues, such as overfitting and vanishing gradients, may easily occur and lead to solutions worse than those presented in the state-of-the-art methods (see Chapter 4, Chapter 5 and Chapter 6).

One solution to address the mentioned issues is to utilize *Transfer Learning* (TL) [20], which is a popular technique in current DL research and has proven to be computationally efficient. TL focuses on applying the knowledge gained while solving one task to a related task. Motivated by this technique, the idea of developing the TT-MMK method serves as a solid foundation for further advancements. Hence, the next subsection explains the concept of TT-MMK or TTCP-based NNs.

TTCP-based Neural Network implementation

The implementation of the 3D-CNN model on ADHD data using the TTCP decomposition of the input data is shown in Figure 7.1. The dataset details are mentioned in Chapter 4, specifically in Section §4.3.1. The Figure 7.1 shows classification accuracy only for 20 epochs. The detailed description of the NN model is as follows:

Implementation Details for ADHD Dataset: A TensorFlow-based CNN implementation is developed for comparison with TT-MMK. The network design is inspired by VGG16 [170], a well-performing architecture for binary and multi-class image classification problems. In our case, a modified version of the VGG16 model is used to handle the three-dimensional input data (ADHD fMRI). The fully connected output layer with a sigmoid activation function provides the binary classification result. Since our input data has smaller dimensions compared to the original VGG16 input shape of $(224\times224\times3)$, adjustments are made, such as reducing the kernel size for convolution kernels and pooling steps. Additionally, 2D convolutions are replaced with 3D convolutions to accommodate the given data. The model has $6,808,157$ trainable parameters and follows an 80-20 split for training and validation datasets. During performance checks, the model is trained for 20 epochs, with a maximum observed validation accuracy of 62.5

Before defining the CNN for TT-CP decomposition, a simpler NN is implemented for the TT decomposed data. A multi-input model is developed for this purpose, where the three TT cores are fed into three separate branches of the network. Further, the flattened TT cores are used and fed into a sequential arrangement of fully connected layers. The outputs from each branch are concatenated by adding them to the end of the preceding one, creating a long tensor that is then fed into the succeeding dense layers. The architecture of the dense layers is also modified to include convolutional layers, considering the challenge posed by the small shape of the input tensor. Moreover, the concatenation scheme of the branch outputs is being reviewed to better represent the data. However, this model tends to overfit the training data due to the chosen fully connected architecture. It consists of $826,581$ trainable parameters.

Note: The efficient STTM in Chapter 4, contains three main steps in the algorithm. Among those, applying uniqueness of SVD in t3f library is not straight forward. Building a NN where taking TT-CP (Alg. 4.2)/Weighted HOSVD (Alg. 5.2) decompositions can be computed by evaluating three convolution layers separately (for 3D data) and they can be concatenated into one. This concatenation would resemble the kernel approximation in the STM method.

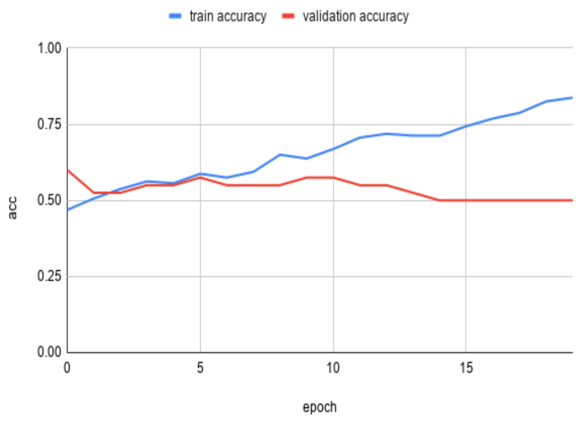


Figure 7.1: Classification accuracy of ADHD data for 20 epochs.

In Figure 7.1, the training-MSE is not achieved to be 100%, the reasons could be less amount of training data, missing uniqueness enforcement on TT factors, and not yet having a comprehensible model or hyperparameter tuning. These perspectives are yet to be looked into for future research work. The diamond in the rough version of Neural STM can be depicted as in Figure 7.2.

One of the advantages of working with DL is the flexibility it offers in terms of incorporating complexity, such as the number of layers or the structure of the proposed model. In the previous chapters, the Gaussian kernel was used, which is a common choice in the ML community. However, this fixed kernel approach may not capture all aspects of the data. In other words, certain rare features may be overlooked due to the lack of an appropriate kernel choice. To address this, a kernel layer can be added as an additional hidden layer in the Neural STM. This idea bears similarity to the concept of Multiple Kernel Learning (MKL) [143, 58], but without the need for manual intervention. The mathematical formulation in Neural STM for kernel computation is as follows,

$$K(\mathfrak{X}_i, \mathfrak{X}_j) = \sum_{p=1}^{R} \sum_{q=1}^{R} \zeta_{pq} \prod_{\tilde{m}=1}^{M} \underbrace{\exp\left(\frac{-\left\| (\mathbf{T}^{(\tilde{m},i)}(:,p) - \mathbf{T}^{(\tilde{m},j)}(:,q) \right\|^2}{2g^2} \right)}_{W^{(\tilde{m})}}, \tag{7.1}$$

where, $\mathfrak{X}$ are the tensor data points with $\mathfrak{T}$ as tensor factorization (TTCP or Tucker) and ζ are the weight parameter in the corresponding hidden layer. The MKL would also include different kernel function as an additional layer with additional weight parameter (η). The mathematical formulation is in (7.2). The depiction of this idea could look like as in Figure 7.3,

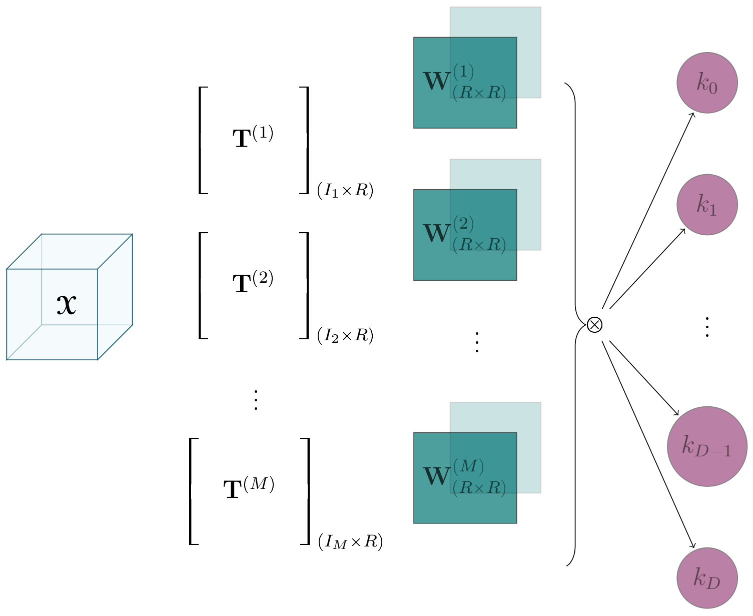


Figure 7.2: TT-MMK Kernel Approximation.

$$K_\eta(\mathcal{X}_i, \mathcal{X}_j) = \sum_{l=1}^{L} \eta_l \underbrace{\left(\sum_{p=1}^{R} \sum_{q=1}^{R} \zeta_{pq} \prod_{\tilde{m}=1}^{M} \underbrace{\exp\left(\frac{-\left\| \left(\mathbf{T}^{(\tilde{m},i)}(:,p) - \mathbf{T}^{(\tilde{m},j)}(:,q)\right\|^2}{2g^2} \right)}_{W^{(\tilde{m})}} \right)}_{\text{Weighted DuSK}}, \tag{7.2}$$

The concept of combining STM as a TL approach to develop a more flexible and generic model is an intriguing idea that holds promise for future projects. This approach has the potential to enhance the adaptability and versatility of models, allowing them to leverage knowledge from different domains and transfer it effectively to new tasks. Implementing and exploring this concept further could open up exciting avenues for research and contribute to the development of advanced and adaptable machine learning models.

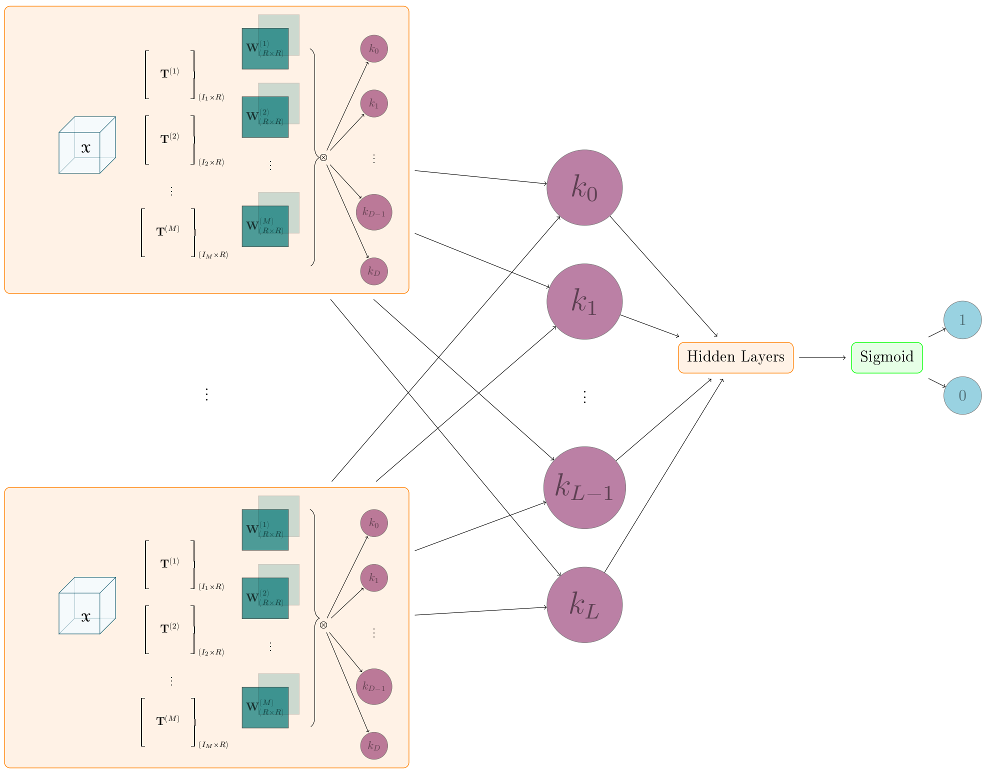


Figure 7.3: Neural Support Tensor Network.

Tensor Kernel Learning in Gaussian Mixture Models [94]: This leads to another future direction of research that is if all the kernels are taken as Gaussian then multiple kernel learning would eventually be *Gaussian mixture models* for low-rank tensor decomposition or multivariate Gaussian mixture models.

7.2.2 Direction 3: Gene Expression Recovery

An exciting direction for future research is the use of tensor-based models.

In the field of computational biology, it's not just about finding the best features; it's also about truly understanding what these features mean. Plus, interpreting the results statistically is crucial for making these methods work effectively in real-world scientific experiments.

The single-cell RNA sequencing data in Genomics, structured as a three-dimensional tensor denoted as $\mathfrak{X} \in \mathbb{R}^{D\times C\times G}$. Here, D represents the number of donors, C signifies the cell type, and G characterizes the gene expression profiles across diverse donors. This dataset prompts a multitude of intriguing questions spanning from unsupervised and semi-supervised to supervised learning. Hence, use of low-rank tensor factorization is an obvious further step to deal with complexity in data that is caused by multidimensionality. Already, tensor factorizations have found their application in genomics [83]. An advancement in this direction is evident in [174], where the HOSVD method is employed. This way by using the knowledge from Chapter 4 and Chapter 5 to address challenges to build an appropriate model for single-cell genomics data. These models could be more robust. However, beyond appropriately modeling data distribution and noise, the inherent randomness of tensor factorization algorithms poses a challenge, resulting in disparate outcomes across multiple runs and impeding interpretability and reproducibility [95]. Thus, the development of a tensor factorization method that not only captures biologically meaningful gene expression patterns but also yields factors with stronger correlation becomes imperative.

In many areas like healthcare and genomics, it's common to have data with lots of zeros, making it sparse. But the usual way of breaking down this data into smaller pieces, using something called tensor factorization, can be unreliable when the data is like this [35]. To fix this, people have come up with a new way called Bayesian Tensor Factorization (BTF). This new way has some benefits, like using what we already know, picking the best model, and telling us how sure we can be. But it's important to remember that this new way can be slow and needs a lot of careful planning, which might need the help of an expert.

Another problem is that the usual methods, which break down the data in a special way, might not work well for finding complex patterns in the data that are not straightforward. To deal with this, a possible solution is to combine this special method with another technique called kernelization. This would need careful designing of how the technique works, especially for finding these not-so-obvious patterns. In the future, we can try to use known techniques like DuSK and WSEK to fill in this gap.

One idea is to use WSEK to compare two pieces of data by considering all the parts together, while in the case of single-cell RNA data, we might focus on how related the different parts are. The new technique we come up with could be a mix of DuSK and WSEK, finding the right balance between them.

Let us take HOSVD (§3.3.2) or SqrtmHOSVD (Alg. 5.2) of tensors $\mathcal{X} \approx \mathcal{G} \times_1 \times \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$ and $\mathcal{Y} \approx \mathcal{H} \times_1 \times \mathbf{D} \times_2 \mathbf{E} \times_3 \mathbf{F}$. First normalize $\mathcal{G}$ and $\mathcal{H}$ such that the entries are between 0 and 1, so we will consider them as probabilities.

$$\sum_{i_1,j_1=1}^{R_1} \sum_{i_2,j_2=1}^{R_2} \sum_{i_3,j_3=1}^{R_3} |\mathcal{G}_{i,j,k} - \mathcal{H}_{i,j,k}| k(\mathbf{a}_{i_1}, \mathbf{d}_{j_1}) k(\mathbf{b}_{i_2}, \mathbf{e}_{j_2}) k(\mathbf{c}_{i_3}, \mathbf{f}_{j_3}). \tag{7.3}$$

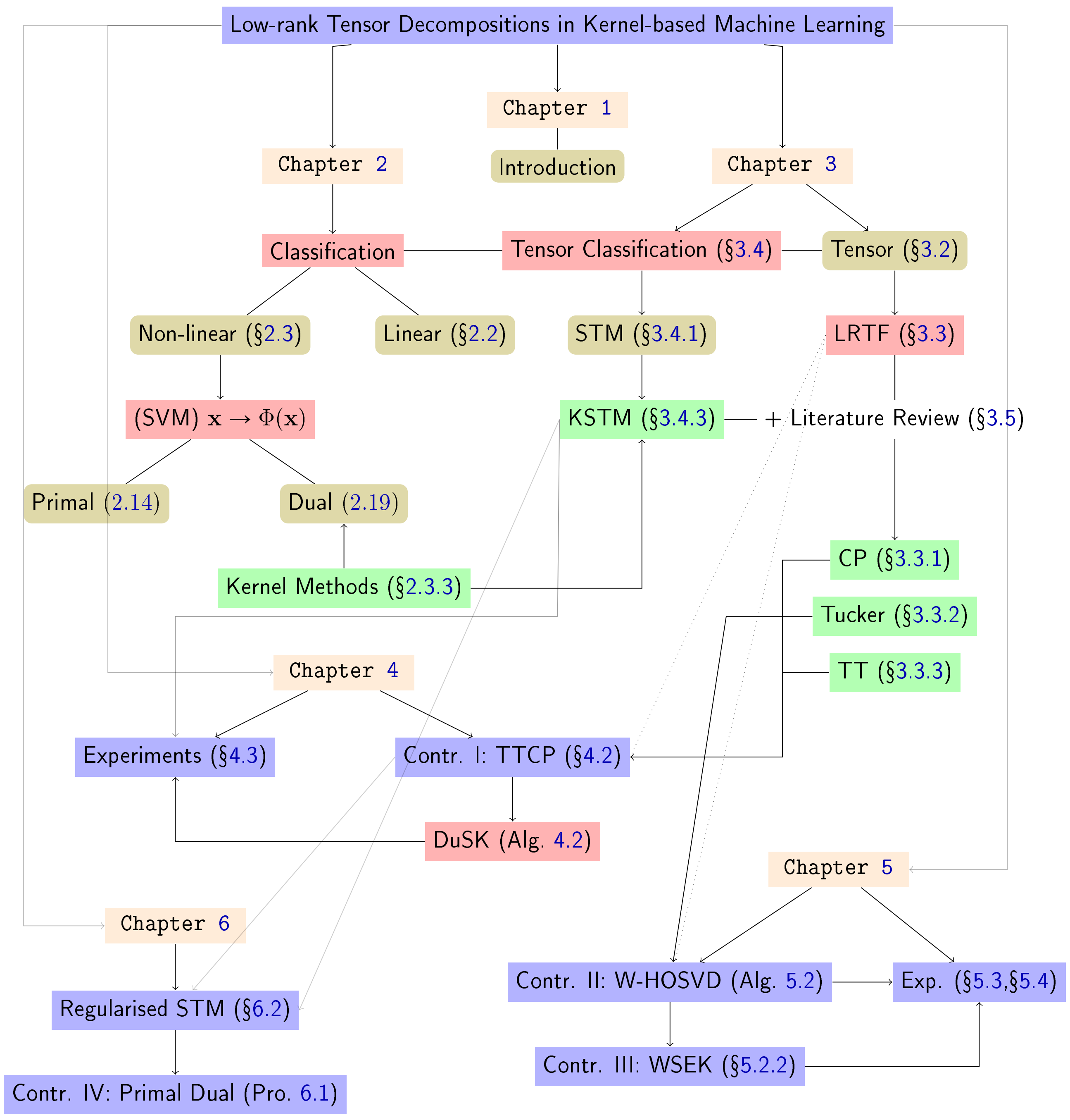
This would possibly provide the nonlinear relationship between factors and would also give a new direction to specialized low-rank approximation used in genomics data. Although, interpretability of these factors is not straight forward, therefore using a more advanced form of BTF with Kernel methods would lead to profound knowledge of single-cell RNA sequence data. This can help healthcare field to better understand the disease and in future could lead to finding better cure. Further, [101] presents detailed review of use of Kernel methods in genetic. The work presented in [192] go more into the direction of data analysis for gene expression using kernel methods.

Note: The proposed work in this thesis has implications for nonlinear boundary classification and opens up less-explored avenues. One potential direction is to apply the concepts to regression problems, specifically Support Vector/Tensor Regression or Kernel Ridge Regression, where the negative log likelihood loss can be used for classification ($< 0.5 \in$ class 1, $\geqslant 0.5 \in$ class 2) or regression. This approach has numerous applications, including image analysis, image segmentation, prediction or analysis of time-series data, sensor networks, and Natural Language Processing.

Another promising prospect is to explore models that incorporate different norms for the low-rank tensor approximation regularized term. This variation proves useful in tackling challenges related to Tensor Completion or missing data issues. The applications of such a model extend to recommender systems, video inpainting, brain imaging, social network analysis, environmental data analysis, genomics, and bioinformatics, among others.

Apart from the practical applications of the proposed work, this thesis also contributes to the extension of knowledge in the field of Numerical Multilinear Algebra. The developed algorithms serve as valuable tools for the wider community working in field of numeric.

7.3 Outlook


Low-rank Tensor Decompositions in Kernel-based Machine Learning
Chapter 1
Introduction
Chapter 2
Chapter 3
Classification
Tensor Classification (§3.4)
Tensor (§3.2)
Non-linear (§2.3)
Linear (§2.2)
STM (§3.4.1)
LRTF (§3.3)
(SVM) $\mathbf{x} \to \Phi(\mathbf{x})$
KSTM (§3.4.3)
+ Literature Review (§3.5)
Primal (2.14)
Dual (2.19)
Kernel Methods (§2.3.3)
CP (§3.3.1)
Tucker (§3.3.2)
TT (§3.3.3)
Chapter 4
Experiments (§4.3)
Contr. I: TTCP (§4.2)
DuSK (Alg. 4.2)
Chapter 5
Chapter 6
Contr. II: W-HOSVD (Alg. 5.2)
Exp. (§5.3,§5.4)
Regularised STM (§6.2)
Contr. III: WSEK (§5.2.2)
Contr. IV: Primal Dual (Pro. 6.1)

BIBLIOGRAPHY

[1] *Multi-way Analysis with Applications in the Chemical Sciences*, John Wiley & Sons, Ltd., 2004.

[2] *Multi-Way Array (Tensor) Factorizations and Decompositions*, John Wiley & Sons, Ltd, 2009, ch. 7, pp. 337–432.

[3] *Tensor Spaces and Numerical Tensor Calculus*, Springer Berlin, Heidelberg, 02 2012.

[4] E. Acar, T. G. Kolda, and D. M. Dunlavy, *All-at-once optimization for coupled matrix and tensor factorizations*, ArXiv, abs/1105.3422 (2011). 4, 6, 72

[5] E. Acar and B. Yener, *Unsupervised multiway data analysis: A literature survey.*, IEEE Trans. Knowl. Data Eng., 21 (2009), pp. 6–20. 4, 36

[6] I. Affleck, T. Kennedy, E. H. Lieb, and H. Tasaki, *Rigorous results on valence-bond ground states in antiferromagnets*, Phys. Rev. Lett., 59 (1987), pp. 799–802. 38

[7] S. A. Al-Baiyat, *Model reduction of bilinear systems described by input/output difference equation*, International Journal of Systems Science, 35 (2004), pp. 503 – 510. ix

[8] M. A. Alvarez, L. Rosasco, and N. D. Lawrence, *Kernels for vector-valued functions: a review*, 2012. 15, 16

[9] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky, *Tensor decompositions for learning latent variable models*, J. Mach. Learn. Res., 15 (2014), p. 2773–2832. 36

[10] L. Armijo, *Minimization of functions having lipschitz continuous first partial derivatives*, Pacific Journal of Mathematics, 16 (1976). 106

[11] S. Axler, *Measures*, Springer International Publishing, Cham, 2020, pp. 13–72. 15

[12] B. W. Bader, T. G. Kolda, et al., *Tensor toolbox for matlab version 3.4*, 2022.

[13] G. Ballard, A. R. Benson, A. Druinsky, B. Lipshitz, and O. Schwartz, *Improving the numerical stability of fast matrix multiplication*, SIAM Journal on Matrix Analysis and Applications, 37 (2016), pp. 1382–1418. 29

[14] G. Ballard, A. Druinsky, N. Knight, and O. Schwartz, *Hypergraph partitioning for parallel sparse matrix-matrix multiplication*, in Proceedings of the 27th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '15, New York, NY, USA, 2015, Association for Computing Machinery, p. 86–88. 29

[15] J. A. Bazerque, G. Mateos, and G. B. Giannakis, *Nonparametric low-rank tensor imputation*, IEEE Statistical Signal Processing Workshop (SSP), (2012), pp. 876–879.

[16] A. Berlinet and C. Thomas-Agnan, *Reproducing Kernel Hilbert Space in Probability and Statistics*, Springer, Boston, MA, 2004. 16

[17] D. P. Bertsekas, *On the goldstein-levitin-polyak gradient projection method*, IEEE Transactions on Automatic Control, 21 (1976), pp. 174–184. 105

[18] ——, *Nonlinear Programming*, Athena Scientific, Belmont, MA 02178-9998, 2nd ed., 1999. 105

[19] R. Bessi and H. Soumare, *Gradient projection algorithms for optimization problems on convex sets and application to svm*, (2021). ffhal-02877016v4. 106

[20] S. Bozinovski, *Reminder of the first paper on transfer learning in neural networks*, 44(3) (2020), p. 291. 115

[21] R. Bro, *Parafac. tutorial and applications*, Chemometrics and Intelligent Laboratory Systems, 38 (1997), pp. 149–171. 6

[22] S. Bubeck, *Convex optimization: Algorithms and complexity*, Foundations and Trends in Machine Learning, 8 (2015), pp. 231–357. 108

[23] C. J. C. Burges and D. Crisp, *Uniqueness of the svm solution*, in Advances in Neural Information Processing Systems, S. Solla, T. Leen, and K. Müller, eds., vol. 12, MIT Press, 1999. 19

[24] D. Cai, X. He, and J. Han, *Learning with tensor representation*, (2006).

[25] D. Cai, X. He, J.-R. Wen, J. Han, and W.-Y. Ma, *Support tensor machines for text categorization*, the University of Illinois at Urbana-Champaign Computer Science Department, (2006). 46

[26] C. F. Caiafa and A. Cichocki, *Generalizing the column–row matrix decomposition to multi-way arrays*, Linear Algebra and its Applications, 433 (2010), pp. 557–573. 4

[27] C. Carmeli, E. De Vito, A. Toigo, and V. Umanitá, *Vector valued reproducing kernel hilbert spaces and universality*, Analysis and Applications, 08 (2011). 15, 16

[28] J. D. Carroll and J. J. Chang, *Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition*, Psychometrika, 35 (1970), pp. 283–319. 5, 33

[29] C.-C. Chang and C.-J. Lin, *Libsvm: A library for support vector machines*, ACM Trans. Intell. Syst. Technol., 2 (2011). 47

[30] O. Chapelle, *Training a support vector machine in the primal*, Neural Computation, 19 (2007), pp. 1155–1178.

[31] C. Chen, K. Batselier, C.-Y. Ko, and N. Wong, *A support tensor train machine*, in 2019 International Joint Conference on Neural Networks (IJCNN), IEEE, 2019, pp. 1–8. 49, 53, 72

[32] C. Chen, K. Batselier, W. Yu, and N. Wong, *Kernelized support tensor train machines*, Pattern Recognition, 122 (2022), p. 108337. 51, 63, 66

[33] S. Chen, S. A. Billings, and P. M. Grant, *Non-linear system identification using neural networks*, International Journal of Control, 51 (1990), pp. 1191–1214. ix

[34] T.-L. Chen, D. Chang, S. Huang, H. Chen, C. Lin, and W. Wang, *Integrating multiple random sketches for singular value decomposition*, arXiv: Numerical Analysis, (2016). 37

[35] E. C. Chi and T. G. Kolda, *On tensors, sparsity, and nonnegative factorizations*, SIAM Journal on Matrix Analysis and Applications, 33 (2012), pp. 1272–1299. 120

[36] J. H. Choi and S. Vishwanathan, *Dfacto: Distributed factorization of tensors*, in Advances in Neural Information Processing Systems, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds., vol. 27, Curran Associates, Inc., 2014. 35

[37] A. Cichocki, *Tensor decompositions: new concepts in brain data analysis?*, Journal of the Society of Instrument and Control Engineers, 50 (2011), pp. 507–516. 6, 72

[38] A. Cichocki, S. Cruces, and S.-i. Amari, *Generalized alpha-beta divergences and their application to robust nonnegative matrix factorization*, Entropy, 13 (2011), pp. 134–170. 48

[39] A. Cichocki, S. Cruces, and S. ichi Amari, *Log-determinant divergences revisited: Alpha-beta and gamma log-det divergences*, Entropy, 17 (2015), pp. 2988–3034. 48

[40] 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*, FNT in Machine Learning, 9 (2016), pp. 249–429. 6, 32, 41, 72

[41] A. Cichocki, D. Mandic, A.-H. Phan, C. Caiafa, G. Zhou, Q. Zhao, and L. D. Lathauwer, *Tensor decompositions for signal processing applications from two-way to multiway component analysis*, Signal Processing Magazine, IEEE, 32 (2014). 6, 37

[42] 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, 9 (2017), pp. 431–673. 6, 32, 110

[43] N. Cohen, O. Sharir, and A. Shashua, *On the expressive power of deep learning: A tensor analysis*, in 29th Annual Conference on Learning Theory, V. Feldman, A. Rakhlin, and O. Shamir, eds., vol. 49 of Proceedings of Machine Learning Research, Columbia University, New York, New York, USA, 23–26 Jun 2016, PMLR, pp. 698–728. 4

[44] P. Comon, *Tensors : A brief introduction*, IEEE Signal Processing Magazine, 31 (2014), pp. 44–53. 37

[45] P. G. Constantine and D. F. Gleich, *Tall and skinny qr factorizations in mapreduce architectures*, in Proceedings of the Second International Workshop on MapReduce and Its Applications, MapReduce '11, New York, NY, USA, 2011, Association for Computing Machinery, p. 43–50. 37

[46] J. H. Conway, R. H. Hardin, and N. J. A. Sloane, *Packing lines, planes, etc.: Packings in grassmannian spaces*, Experimental Mathematics, 5 (1996), pp. 139–159. 77

[47] C. Cortes and V. Vapnik, *Support-vector networks*, Machine Learning, 20 (1995), pp. 273–297. 2, 20

[48] D. R. Cox, *The regression analysis of binary sequences*, Journal of the Royal Statistical Society. Series B (Methodological), 20 (1958), pp. 215–242. 7

[49] L. De Lathauwer, B. De Moor, and J. Vandewalle, *On the best rank-1 and rank-(r1 ,r2 ,. . .,rn) approximation of higher-order tensors*, SIAM Journal on Matrix Analysis and Applications, 21 (2000), pp. 1324–1342. 37

[50] V. de Silva and L.-H. Lim, *Tensor rank and the ill-posedness of the best low-rank approximation problem*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1084–1127. 34, 53

[51] E. De Vito, V. Umanità, and S. Villa, *An extension of mercer theorem to matrix-valued measurable kernels*, Applied and Computational Harmonic Analysis, 34 (2013), pp. 339–351. 16

[52] S. Dolgov and B. Khoromskij, *Two-level qtt-tucker format for optimized tensor calculus*, SIAM Journal on Matrix Analysis and Applications, 34 (2013), pp. 593–623. 4, 41, 75

[53] S. V. Dolgov and D. V. Savostyanov, *Alternating minimal energy methods for linear systems in higher dimensions*, SIAM Journal on Scientific Computing, 36 (2014), pp. A2248–A2271. 28

[54] H. Fanaee-T and J. Gama, *Tensor-based anomaly detection: An interdisciplinary survey*, Knowledge-Based Systems, 98 (2016), pp. 130–147. 37

[55] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, *qpoases: A parametric active-set algorithm for quadratic programming*, Mathematical Programming Computation, 6 (2014), pp. 327–363. 12

[56] F. Girosi, *Regularization theory, radial basis functions and networks*, in From Statistics to Neural Networks, V. Cherkassky, J. H. Friedman, and H. Wechsler, eds., Berlin, Heidelberg, 1994, Springer Berlin Heidelberg, pp. 166–187. 44

[57] G. H. Glover, *Overview of functional magnetic resonance imaging*, Neurosurgery Clinics of North America, 22 (2011), pp. 133 – 139. 3

[58] M. Gönen and E. Alpaydin, *Multiple kernel learning algorithms*, Journal of Machine Learning Research, 12 (2011), pp. 2211–2268. 117

[59] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, Adaptive computation and machine learning, MIT Press, 2016. 8

[60] L. Grasedyck, D. Kressner, and C. Tobler, *A literature survey of low-rank tensor approximation techniques*, GAMM-Mitteilungen, 36 (2013), pp. 53–78. 29, 37

[61] T. Guo, L. Han, L. He, and X. Yang, *A ga-based feature selection and parameter optimization for linear support higher-order tensor machine*, Neurocomputing, 144 (2014), pp. 408 – 416.

[62] W. Guo, I. Kotsia, and I. Patras, *Tensor learning for regression*, IEEE Transactions on Image Processing, 21 (2012), pp. 816–827. 49

[63] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, 2023. 12

[64] Y. h. Taguchi and T. Turki, *Mathematical formulation and application of kernel tensor decomposition based unsupervised feature extraction*, Knowledge-Based Systems, 217 (2021), p. 106834.

[65] N. Halko, P. G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Review, 53 (2011), pp. 217–288. 37

[66] N. Halko, P.-G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Review, 53 (2011), pp. 217–288.

[67] Z. Hao, L. He, B. Chen, and X. Yang, *A linear support higher-order tensor machine for classification*, IEEE Transactions on Image Processing, 22 (2013), pp. 2911–2920. 43, 49

[68] R. A. Harshman, *Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis*, UCLA Working Papers in Phonetics, 16 (1970), pp. 1–84. xix, 5, 33, 35

[69] J. Håstad, *Tensor rank is np-complete*, Automata, Languages and Programming, (1989), pp. 451–460.

[70] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer Series in Statistics, Springer New York Inc., New York, USA, 2001. 7, 8, 11, 13, 16, 22, 23, 24, 49

[71] L. He, X. Kong, P. S. Yu, X. Yang, A. B. Ragin, and Z. Hao, *Dusk: A dual structure-preserving kernel for supervised tensor learning with applications to neuroimages*, Proceedings of the 2014 SIAM International Conference on Data Mining (SDM), (2014), pp. 127–135. xvii, 49, 53, 59, 60, 62, 66, 89, 97, 113

[72] L. He, C.-T. Lu, H. Ding, S. Wang, L. Shen, P. S. Yu, and A. B. Ragin, *Multi-way multi-level kernel modeling for neuroimaging classification*, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2017), pp. 6846–6854. 53, 57, 59, 60, 62, 66, 113

[73] L. He, C.-T. Lu, G. Ma, S. Wang, L. Shen, P. S. Yu, and A. B. Ragin, *Kernelized support tensor machines*, Proceedings of the 34th International Conference on Machine Learning-Volume 70, (2017), pp. 1442–1451. 53, 72, 91, 92

[74] P. Hewson, *Applied multiway data analysis*, Journal of the Royal Statistical Society Series A: Statistics in Society, 172 (2009), pp. 941–942.

[75] F. L. Hitchcock, *The expression of a tensor or a polyadic as a sum of products*, Journal of Mathematics and Physics, 6 (1927), pp. 164–189. 4, 33

[76] ——, *Multiple invariants and generalized rank of a p-way matrix or tensor*, Journal of Mathematics and Physics, 7 (1928), pp. 39–79. 4, 33

[77] T. Huckle, K. Waldherr, and T. Schulte-Herbrüggen, *Computations in quantum tensor networks*, Linear Algebra and its Applications, 438 (2013), pp. 750–781. Tensors and Multilinear Algebra. 38

[78] P. Jain and P. Kar, *Non-convex optimization for machine learning*, Foundations and Trends® in Machine Learning, 10 (2017), pp. 142–336. 100

[79] H. P. Jakobsen, *Tensor products, reproducing kernels, and power series*, Journal of Functional Analysis, 31 (1979), pp. 293–305. 44

[80] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R*, Springer, 2013. 8, 9, 13, 22

[81] M. James, *Xvi. functions of positive and negative type, and their connection the theory of integral equations*, Series A, Containing Papers of a Mathematical or Physical Character, 209 (1909), p. 415–446. 16

[82] B. Jones, *Statistical methods for engineers and scientists*, Royal Statistical Society. Journal. Series A: General, 148 (2018), pp. 390–390. 7

[83] I. Jung, M. Kim, S. Rhee, S. Lim, and S. Kim, *Monti: A multi-omics non-negative tensor decomposition framework for gene-level integrative analysis*, Frontiers in Genetics, 12 (2021). 120

[84] D. S. Karachalios, I. V. Gosea, K. Kour, and A. C. Antoulas, *Bilinear realization from input-output data with neural networks*, 2022. iii, 114, 115

[85] L. Karlsson, D. Kressner, and A. Uschmajew, *Parallel algorithms for tensor completion in the cp format*, Parallel Computing, 57 (2016), pp. 222–234. 35

[86] V. A. Kazeev, B. N. Khoromskij, and E. E. Tyrtyshnikov, *Multilevel toeplitz matrices generated by tensor-structured vectors and convolution with logarithmic complexity*, SIAM Journal on Scientific Computing, 35 (2013), pp. A1511–A1536. 4

[87] M. A. Khamis, S. Im, B. Moseley, K. Pruhs, and A. Samadian, *A relational gradient descent algorithm for support vector machine training*, CoRR, abs/2005.05325 (2020). 106

[88] B. Khan, H. Fatima, A. Qureshi, S. Kumar, A. Hanan, J. Hussain, and S. Abdullah, *Drawbacks of artificial intelligence and their potential solutions in the healthcare sector*, Biomedical Materials and Devices, 8 (2023), pp. 1–8. Epub ahead of print. 4

[89] V. Khoromskaia and B. N. Khoromskij, *Tensor Numerical Methods in Quantum Chemistry*, De Gruyter, Berlin, Boston, 2018.

[90] B. N. Khoromskij, *O(d log n)-quantics approximation of n-d tensors in high-dimensional numerical modeling*, Constructive Approximation, 34 (2011), pp. 257–280. 41

[91] H. A. L. Kiers, *Towards a standardized notation and terminology in multiway analysis*, Journal of Chemometrics, 14 (2000), pp. 105–122. 33

[92] D. Kim, S. Sra, and I. S. Dhillon, *Fast Newton-type Methods for the Least Squares Nonnegative Matrix Approximation Problem*, pp. 343–354. 35

[93] H.-J. Kim, E. Ollila, V. Koivunen, and H. V. Poor, *Robust iteratively reweighted Lasso for sparse tensor factorizations*, in 2014 IEEE Workshop on Statistical Signal Processing (SSP), 2014, pp. 420–423. 35

[94] M. Kirstein, D. Sommer, and M. Eigel, *Tensor-train kernel learning for gaussian processes*, in Proceedings of the Eleventh Symposium on Conformal and Probabilistic Prediction with Applications, vol. 179 of Proceedings of Machine Learning Research, PMLR, 24–26 Aug 2022, pp. 253–272. 119

[95] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Review, 51 (2009), pp. 455–500. xix, 6, 28, 35, 36, 37, 72, 120

[96] I. Kotsia and I. Patras, *Support Tucker Machines*, CVPR 2011, (2011), pp. 633–640. 49, 66, 72

[97] K. Kour, S. Dolgov, P. Benner, M. Stoll, and M. Pfeffer, *A weighted subspace exponential kernel for support tensor machines*, 2023. iii, 6

[98] K. Kour, S. Dolgov, M. Stoll, and P. Benner, *Efficient structure-preserving support tensor train machine*, Journal of Machine Learning Research, 24 (2023), pp. 1–22. iii, xvii, 6, 72, 76, 87, 89

[99] D. Kressner, M. Steinlechner, and A. Uschmajew, *Low-rank tensor methods with subspace correction for symmetric eigenvalue problems*, SIAM Journal on Scientific Computing, 36 (2014), pp. A2346–A2368. 4

[100] J. B. Kruskal, *Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics*, Linear Algebra and its Applications, 18 (1977), pp. 95–138. 34, 35

[101] N. B. Larson, J. Chen, and D. J. Schaid, *A review of kernel methods for genetic association studies*, Genetic Epidemiology, 43 (2019), pp. 122–136. 121

[102] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, *A multilinear singular value decomposition*, SIAM Journal on Matrix Analysis and Applications, 21 (2000), pp. 1253–1278. 37

[103] V. Lebedev, Y. Ganin, M. Rakhuba, I. V. Oseledets, and V. S. Lempitsky, *Speeding-up convolutional neural networks using fine-tuned CP-decomposition*, CoRR, abs/1412.6553 (2014). 53

[104] Y. Lecun, *Une procédure d'apprentissage pour réseau à seuil asymétrique*, Proceedings of Cognitiva 85, Paris, (1985), pp. 599–604. 2

[105] N. Lee and A. Cichocki, *Fundamental tensor operations for large-scale data analysis using tensor network formats*, Multidimensional Systems and Signal Processing, (2016). 28, 36

[106] E. L. Lehmann and G. Casella, *Theory of Point Estimation*, Springer-Verlag, New York, NY, USA, second ed., 1998. 18

[107] J. Li, C. Battaglino, I. Perros, J. Sun, and R. Vuduc, *An input-adaptive and in-place approach to dense tensor-times-matrix multiply*, in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '15, New York, NY, USA, 2015, Association for Computing Machinery. 29

[108] A. P. Liavas and N. D. Sidiropoulos, *Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers*, IEEE Transactions on Signal Processing, 63 (2015), pp. 5450–5463. 35

[109] C.-J. Lin, *Projected gradient methods for nonnegative matrix factorization*, Neural Computation, 19 (2007), pp. 2756–2779. 105

[110] J. Liu, P. Musialski, P. Wonka, and J. Ye, *Tensor completion for estimating missing values in visual data*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 35 (2013), pp. 208–220. 93

[111] X. Liu, T. Guo, L. He, and X. Yang, *A low-rank approximation-based transductive support tensor machine for semisupervised classification*, IEEE Transactions on Image Processing, 24 (2015), pp. 1825–1838. 72

[112] X. Liu, T. Guo, L. He, and X. Yang, *A low-rank approximation-based transductive support tensor machine for semisupervised classification*, IEEE Transactions on Image Processing, 24 (2015), pp. 1825–1838.

[113] Y. Liu, F. Wu, Y. Zhuang, and J. Xiao, *Active post-refined multimodality video semantic concept detection with tensor representation*, Proceedings of the 16th ACM International Conference on Multimedia, (2008), p. 91–100.

[114] S. Lloyd, *Least squares quantization in pcm*, IEEE Transactions on Information Theory, 28 (1982), pp. 129–137. 100

[115] L. Luo, Y. Xie, Z. Zhang, and W.-J. Li, *Support matrix machines*, in Proceedings of the 32nd International Conference on Machine Learning, F. Bach and D. Blei, eds., vol. 37 of Proceedings of Machine Learning Research, Lille, France, 07–09 Jul 2015, PMLR, pp. 938–947. 21

[116] P. McCullagh and J. Nelder, *Generalized Linear Models, Second Edition*, Chapman and Hall/CRC Monographs on Statistics and Applied Probability Series, Chapman & Hall, 1989. 8

[117] R. Meka, P. Jain, C. Caramanis, and I. S. Dhillon, *Rank minimization via online learning*, in Proceedings of the 25th International Conference on Machine Learning, ICML, New York, NY, USA, 2008, Association for Computing Machinery, pp. 656–663. 100

[118] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*, MIT Press, Cambridge, MA, USA, 1969. 2

[119] J. Mocks, *Topographic components model for event-related potentials and some biophysical considerations*, IEEE Transactions on Biomedical Engineering, 35 (1988), pp. 482–484. 33

[120] MOSEK ApS, *MOSEK ApS: The MOSEK optimization toolbox for MATLAB manual. Version 8.0 (Revision 57)*, 2017. Accessed 6 Feb 2020. 12

[121] M. Mørup, *Applications of tensor (multiway array) factorizations and decompositions in data mining*, WIREs Data Mining and Knowledge Discovery, 1 (2011), pp. 24–40. 36

[122] J. G. Nagy and M. E. Kilmer, *Kronecker product approximation for three-dimensional imaging applications*, 2004. 6

[123] B. K. Natarajan, *Sparse approximate solutions to linear systems*, SIAM J. Comput., 24 (1995), pp. 227–234. 100

[124] A. Nestor et al., *The face of image reconstruction: Progress, pitfalls, prospects*, Trends in cognitive sciences, 24 (2020), pp. 747–759. 3

[125] D. Nion and L. D. Lathauwer, *Fast communication: An enhanced line search scheme for complex-valued tensor decompositions. application in ds-cdma*, Signal Processing, 88 (2008), pp. 749–755. 34

[126] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov, *Tensorizing neural networks*, Advances NIPS 28, (2015), pp. 442–450. 92

[127] A. Novikov, M. Trofimov, and I. V. Oseledets, *Exponential machines*, arXiv preprint arXiv:1605.03795, (2016). 53

[128] N. Orsini, R. Bellocco, and S. Greenland, *Generalized least squares for trend estimation of summarized dose-response data*, The Stata Journal, 6 (2006), pp. 40–57.

[129] R. Orús, *A practical introduction to tensor networks: Matrix product states and projected entangled pair states*, Annals of Physics, 349 (2014), pp. 117–158. 38

[130] I. Oseledets and S. Dolgov, *Tt-toolbox*, 2023.

[131] I. Oseledets and E. Tyrtyshnikov, *Tt-cross approximation for multidimensional arrays*, Linear Algebra and its Applications, 432 (2010), pp. 70–88. 38

[132] I. V. Oseledets, *Approximation of $2^d \times 2^d$ matrices using tensor decomposition*, SIAM Journal on Matrix Analysis and Applications, 31 (2010), pp. 2130–2145. 41

[133] I. V. Oseledets, *Tensor-train decomposition*, SIAM Journal on Scientific Computing, 33 (2011), pp. 2295–2317. xix, 38, 39, 40, 54, 55, 58

[134] 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, 31 (2009), pp. 3744–3759. 4

[135] D. B. Parker, *Learning-logic*, Tech. Rep. TR-47, Center for Comp. Research in Economics and Management Sci., MIT, 1985. 2

[136] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac, *Matrix product state representations*, Quantum Info. Comput., 7 (2007), p. 401–430. 38

[137] A. H. Phan and A. Cichocki, *Extended HALS algorithm for nonnegative Tucker decomposition and its applications for multiway analysis and classification*, Neurocomputing, 74 (2011), pp. 1956–1969. Adaptive Incremental Learning in Neural Networks Learning Algorithm and Mathematic Modelling Selected papers from the International Conference on Neural Information Processing 2009 (ICONIP 2009). 37

[138] A.-H. Phan, A. Cichocki, I. V. 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, (2019), pp. 1–15.

[139] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes, *Bilinear classifiers for visual recognition*, Advances in Neural Information Processing Systems 22, (2009), pp. 1482–1490. 49, 72

[140] J. Platt, *Sequential minimal optimization : A fast algorithm for training support vector machines*, Microsoft Research Technical Report, (1998). 10, 13

[141] R. Poldrack and M. Farah, *Progress and challenges in probing the human brain*, Nature, 526 (2015), pp. 371–379. 3

[142] P. M. R. Rifkin and A. Verri, *A note on support vector machine degeneracy*, Springer Berlin Heidelberg, 1999. 19

[143] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet, *Simplemkl*, Journal of Machine Learning Research, 9 (2008), pp. 2491–2521. 117

[144] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, Adaptive Computation and Machine Learning, MIT Press, Cambridge, MA, USA, jan 2006.

[145] P. Refaeilzadeh, L. Tang, and H. Liu, *Cross-Validation*, Springer US, Boston, MA, 2009. 22

[146] T. Rohwedder and A. Uschmajew, *On local convergence of alternating schemes for optimization of convex problems in the tensor train format*, SIAM Journal on Numerical Analysis, 51 (2013), pp. 1134–1162.

[147] L. Rosasco, E. D. Vito, A. Caponnetto, M. Piana, and A. Verri, *Are loss functions all the same?*, Neural Computation, 16 (2004), pp. 1063–1076. 24

[148] F. Rosenblatt, *The perceptron: A probabilistic model for information storage and organization in the brain.*, Psychological Review, 65 (1958), pp. 386–408. 2

[149] S. Rosset, J. Zhu, and T. J. Hastie, *Boosting as a regularized path to a maximum margin classifier*, J. Mach. Learn. Res., 5 (2004), pp. 941–973. 24

[150] W. Rudin, *Functional Analysis*, McGraw-Hill, 2nd ed., 1990. 15

[151] D. Rumelhart, G. Hinton, and R. Williams, *Learning intenal representations by error propagation.*, Parallel Distributed Processing, 1 (1987), pp. 318–362. 2

[152] D. E. Rumelhart, J. L. McClelland, and P. R. Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, The MIT Press, 07 1986.

[153] H. G. Rumelhart, D. and R. Williams, *Learning representations by back-propagating errors*, Nature, 323 (1986), p. 533–536. 2

[154] B. Schölkopf, *The kernel trick for distances*, in Advances in Neural Information Processing Systems, T. Leen, T. Dietterich, and V. Tresp, eds., vol. 13, MIT Press, 2000. 15

[155] B. Schölkopf, R. Herbrich, and A. J. Smola, *A generalized representer theorem*, Computational Learning Theory, (2001), pp. 416–426. 15, 47, 66

[156] B. Schölkopf and A. J. Smola, *Learning with kernels : support vector machines, regularization, optimization, and beyond*, Adaptive computation and machine learning, MIT Press, 2002. 17, 22, 44

[157] U. Schollwöck, *Matrix product state algorithms: Dmrg, tebd and relatives*, 2013. 38

[158] K. Seeliger, U. Güçlü, L. Ambrogioni, Y. Güçlütürk, and M. A. van Gerven, *Generative adversarial networks for reconstructing natural images from brain activity*, NeuroImage, 181 (2018), pp. 775–785. 3

[159] A. Shashua and A. Levin, *Linear image coding for regression and classification using the tensor-rank principle*, Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, 1 (2001), pp. I–I. 6

[160] G. Shen, K. Dwivedi, K. Majima, T. Horikawa, and Y. Kamitani, *End-to-end deep image reconstruction from human brain activity*, Frontiers in computational neuroscience, 13 (2019), p. 21. 3

[161] N. Sidiropoulos, *Generalizing Carathéodory's uniqueness of harmonic parameterization to N dimensions*, IEEE Trans. Inf. Theory, 47 (2001), pp. 1687–1690. 36

[162] N. Sidiropoulos, *Low-rank decomposition of multi-way arrays: a signal processing perspective*, in Processing Workshop Proceedings, 2004 Sensor Array and Multichannel Signal, 2004, pp. 52–58. 35

[163] N. Sidiropoulos, R. Bro, and G. Giannakis, *Parallel factor analysis in sensor array processing*, IEEE Transactions on Signal Processing, 48 (2000), pp. 2377–2388. 36

[164] N. D. Sidiropoulos and R. Bro, *On the uniqueness of multilinear decomposition of n-way arrays*, Journal of Chemometrics, 14 (2000). 35

[165] N. D. Sidiropoulos, L. D. Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, *Tensor decomposition for signal processing and machine learning*, IEEE Transactions on Signal Processing, 65 (2017), pp. 3551–3582. 6, 36, 37

[166] M. Signoretto, L. D. Lathauwer, and J. A. K. Suykens, *Learning tensors in reproducing kernel hilbert spaces with multilinear spectral penalties*, ArXiv, abs/1310.4977 (2013). 44

[167] M. Signoretto, E. Olivetti, L. D. Lathauwer, and J. A. K. Suykens, *A kernel-based framework to tensorial data analysis*, Neural Networks, 24 (2011), pp. 861 – 874. Artificial Neural Networks: Selected Papers from ICANN 2010. xvi, xvii, 48, 72, 76, 84, 85, 87, 89

[168] ——, *Classification of multichannel signals with cumulant-based kernels*, IEEE Transactions on Signal Processing, 60 (2012), pp. 2304–2314. 6, 48, 72

[169] M. Signoretto, Q. Tran Dinh, L. De Lathauwer, and J. A. K. Suykens, *Learning with tensors: a framework based on convex optimization and spectral regularization*, Machine Learning, 94 (2014), pp. 303–351. 60, 72, 94

[170] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition.*, in ICLR, Y. Bengio and Y. LeCun, eds., 2015. 116

[171] M. Sørensen, L. D. Lathauwer, and A. Stegeman, *Canonical polyadic decomposition with orthogonality constraints*, 2012. 35

[172] I. Steinwart, D. Hush, and C. Scovel, *An explicit description of the reproducing kernel hilbert spaces of gaussian rbf kernels*, IEEE Transactions on Information Theory, 52 (2006), pp. 4635–4643. 16, 18

[173] M. Sørensen and L. De Lathauwer, *Blind signal separation via tensor decomposition with Vandermonde factor Part I:: Canonical Polyadic decomposition*, IEEE Transactions on Signal Processing, 61 (2013), pp. 5507–5519. 6, 36

[174] Y.-H. Taguchi and T. Turki, *Tensor-decomposition-based unsupervised feature extraction applied to prostate cancer multiomics data*, Genes, 11 (2020), p. 1493. 120

[175] J. Tang, A. LeBel, S. Jain, et al., *Semantic reconstruction of continuous language from non-invasive brain recordings*, Nature Neuroscience, 26 (2023), pp. 858–866. 3

[176] D. Tao, X. Li, W. Hu, S. Maybank, and X. Wu, *Supervised tensor learning*, Proceedings of the Fifth IEEE International Conference on Data Mining, (2005), pp. 450–457. 43, 46

[177] ——, *Supervised tensor learning*, Knowledge and Information Systems, (2007), pp. 1–42. 43, 49

[178] R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society (Series B), 58 (1996), pp. 267–288.

[179] F. A. Tobar, S.-Y. Kung, and D. P. Mandic, *Multikernel least mean square algorithm*, IEEE Transactions on Neural Networks and Learning Systems, 25 (2014), pp. 265–277.

[180] R. Tomioka and T. Suzuki, *Convex tensor decomposition via structured schatten norm regularization*, in Advances in Neural Information Processing Systems, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, eds., vol. 26, Curran Associates, Inc., 2013. 93

[181] V. Tresp, C. Esteban, Y. Yang, S. Baier, and D. Krompass, *Learning with memory embeddings*, 2016. 36

[182] L. R. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311. 36

[183] N. Vannieuwenhoven, R. Vandebril, and K. Meerbergen, *A new truncation strategy for the higher-order singular value decomposition*, SIAM Journal on Scientific Computing, 34 (2012), pp. A1027–A1052. xix, 37, 38, 78, 79

[184] R. VanRullen and L. Reddy, *Reconstructing faces from fmri patterns using deep generative neural networks*, Communications biology, 2 (2019), p. 193. 3

[185] V. Vapnik, *The nature of statistical learning theory*, Springer-Verlag, New York, 1995. 10

[186] V. Vapnik, *Statistical Learning Theory*, Wiley-Interscience, 1998. 10, 11

[187] M. Vasilescu and D. Terzopoulos, *Multilinear image analysis for facial recognition*, in 2002 International Conference on Pattern Recognition, vol. 2, 2002, pp. 511–514 vol.2. 6

[188] Verstraete, Frank and Murg, V and Cirac, JI, *Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems*, ADVANCES IN PHYSICS, 57 (2008), pp. 143–224. 38

[189] G. Vidal, *Efficient classical simulation of slightly entangled quantum computations*, Phys. Rev. Lett., 91 (2003), p. 147902. 38

[190] H. Wang and N. Ahuja, *Facial expression decomposition*, Proceedings Ninth IEEE International Conference on Computer Vision, (2003), pp. 958–965 vol.2. 6

[191] L. Wang, K. Xie, T. Semong, and H. Zhou, *Missing data recovery based on tensor-cur decomposition*, IEEE Access, 6 (2018), pp. 532–544. 4

[192] X. Wang, E. P. Xing, and D. J. Schaid, *Kernel methods for large-scale genomic data analysis*, Briefings in Bioinformatics, 16 (2015), pp. 183–192. 121

[193] Y. Wang, H.-Y. Tung, A. Smola, and A. Anandkumar, *Fast and guaranteed tensor decomposition via sketching*, in Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, NIPS'15, Cambridge, MA, USA, 2015, MIT Press, p. 991–999. 36

[194] Z. Wang and S. Chen, *New least squares support vector machines based on matrix patterns*, Neural Processing Letters, 26 (2007), pp. 41–56.

[195] S. R. White, *Density-matrix algorithms for quantum renormalization groups*, Phys. Rev. B, 48 (1993), pp. 10345–10356. 38

[196] K. Wimalawarne, M. Sugiyama, and R. Tomioka, *Multitask learning meets tensor factorization: task imputation via convex optimization*, in Advances in Neural Information Processing Systems, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds., vol. 27, Curran Associates, Inc., 2014. 93

[197] K. Wimalawarne, R. Tomioka, and M. Sugiyama, *Theoretical and experimental analyses of tensor-based regression and classification*, Neural Computation, 28 (2016), pp. 686–715. 93

[198] L. Wolf, H. Jhuang, and T. Hazan, *Modeling appearances with low-rank SVM*, IEEE Conference on Computer Vision and Pattern Recognition, (2007). 49

[199] P. Wolfe, *The simplex method for quadratic programming*, Econometrica: Journal of the Econometric Society, (1959), pp. 382–398. 12

[200] M. Wu and J. Farquhar, *A subspace kernel for nonlinear feature extraction*, Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07), (2007), pp. 1125–1130. 53

[201] Z. Xu, F. Yan, and Y. Qi, *Infinite tucker decomposition : Nonparametric bayesian models for multiway data analysis*, ICML, Madison, WI, USA, 2012, Omnipress, pp. 1675–1682. 4

[202] S. Yan, D. Xu, Q. Yang, L. Zhang, X. Tang, and H.-J. Zhang, *Multilinear discriminant analysis for face recognition*, IEEE Transactions on Image Processing, 16 (2007), pp. 212–220.

[203] R. Yu and Y. Liu, *Learning from multiway data: Simple and efficient tensor regression*, in Proceedings of The 33rd International Conference on Machine Learning, M. F. Balcan and K. Q. Weinberger, eds., vol. 48 of Proceedings of Machine Learning Research, New York, New York, USA, 20–22 Jun 2016, PMLR, pp. 373–381. 93

[204] D. Zeng, S. Wang, Y. Shen, and C. Shi, *A ga-based feature selection and parameter optimization for support tucker machine*, Procedia Computer Science, 111 (2017), pp. 17 – 23. The 8th International Conference on Advances in Information Technology. 49

[205] C. Zhang, H. Fu, S. Liu, G. Liu, and X. Cao, *Low-rank tensor constrained multiview subspace clustering*, in Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), ICCV '15, USA, 2015, IEEE Computer Society, p. 1582–1590. 93

[206] L. Zhang, L. Zhang, D. Tao, and X. Huang, *A multifeature tensor for remote-sensing target recognition*, IEEE Geoscience and Remote Sensing Letters, 8 (2011), pp. 374–378.

[207] Y. Zhang, Z. Han, and Y. Tang, *Color image denoising based on low-rank tensor train*, in International Conference on Graphic and Image Processing, 2019. 4

[208] Z. Zhang and T. W. Chow, *Maximum margin multisurface support tensor machines with application to image classification and segmentation*, Expert Systems with Applications, 39 (2012), pp. 849 – 860.

[209] Q. Zhao, G. Zhou, T. Adali, L. Zhang, and A. Cichocki, *Kernelization of tensor-based models for multiway data analysis: Processing of multidimensional structured data*, IEEE Signal Processing Magazine, 30 (2013), pp. 137–148. 48, 72

[210] Q. Zhao, G. Zhou, T. Adali, L. Zhang, and A. Cichocki, *Kernelization of tensor-based models for multiway data analysis: Processing of multidimensional structured data*, IEEE Signal Processing Magazine, 30 (2013), pp. 137–148.

[211] Q. Zhao, G. Zhou, T. Adali, L. Zhang, and A. Cichocki, *Kernel-based tensor partial least squares for reconstruction of limb movements*, in 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 3577–3581.

[212] Q. Zhao, G. Zhou, S. Xie, L. Zhang, and A. Cichocki, *Tensor ring decomposition*, 2016. 28

[213] G. Zhou and A. Cichocki, *Canonical polyadic decomposition based on a single mode blind source separation*, IEEE Signal Processing Letters, 19 (2012), pp. 523–526. 35

[214] H. Zhou, L. Li, and H. Zhu, *Tensor regression with applications in neuroimaging data analysis*, Journal of the American Statistical Association, 108 (2013), pp. 540–552. 49

[215] Y. Zhou, X. Mei, W. Li, and J. Huang, *Classification of resting-state fmri datasets based on graph kernels*, 2nd International Conference on Image, Vision and Computing (ICIVC), (2017), pp. 665–669.

[216] Y. Zniyed, O. Karmouda, R. Boyer, J. Boulanger, A. L. de Almeida, and G. Favier, *Chapter 14 - structured tensor train decomposition for speeding up kernel-based learning*, (2022), pp. 537–563.