Tensor-Based Kernel Methods

Wesel, F.

DOI
10.4233/uuid:19c0d433-dba9-40e1-a45c-d56d1140bbfa

Publication date
2025

Document Version
Final published version

Citation (APA)
Wesel, F. (2025). *Tensor-Based Kernel Methods*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:19c0d433-dba9-40e1-a45c-d56d1140bbfa

TENSOR-BASED KERNEL MODELS

FREDERIEK WESEL

TENSOR-BASED KERNEL METHODS

TENSOR-BASED KERNEL METHODS

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Friday, the 14th of November, 2025, at 10:00 o'clock

by

Frederiek WESEL

Master of Science in Applied Mathematics,
Delft University of Technology, The Netherlands
born in Bordighera, Italy

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. dr. ir. J.W. van Wingerden,	Delft University of Technology, *promotor*
Dr. B. Hunyadi,	Delft University of Technology, *promotor*
Dr. ir. K. Batselier,	Delft University of Technology, *copromotor*
Independent members:	
Dr. M. Kok,	Delft University of Technology
Prof. dr. ir. G.J.T. Leus,	Delft University of Technology
Prof. dr. ir. J. Suykens,	Katholieke Universiteit Leuven
Dr. N. Wong	The University of Hong Kong
Prof. dr. ir. T. Keviczky,	Delft University of Technology, *reserve member*

Keywords: Machine Learning, Kernel Machines, Gaussian Processes, Tensors, Tensor Decompositions, Tensor Networks
Printed by: ProefschriftMaken.nl

ISBN 978-94-6518-163-9

An electronic copy of this dissertation is available at
https://doi.org/10.4233/uuid:19c0d433-dba9-40e1-a45c-d56d1140bbfa.

CONTENTS

Summary **vii**

Samenvatting **ix**

1. Introduction **1**
- 1.1. Notation and Tensors . . . 3
- 1.2. Tensor Networks . . . 4
- 1.3. Kernel Machines . . . 7
- 1.4. Contributions of this Thesis . . . 10

2. Fourier Features and Tensor Decompositions **17**
- 2.1. Introduction . . . 18
- 2.2. Related work . . . 19
- 2.3. Learning with Fourier features . . . 21
- 2.4. Numerical Experiments . . . 26
- 2.5. Conclusion . . . 31

3. Structured Inducing Points and Tensor Decompositions **37**
- 3.1. Introduction . . . 38
- 3.2. Grid-Structured Kernel Machines . . . 43
- 3.3. Experiments . . . 48
- 3.4. Conclusion . . . 52

4. Quantized Features and Tensor Decompositions **61**
- 4.1. Introduction . . . 62
- 4.2. Background . . . 64
- 4.3. Quantizing Polynomial and Fourier Features . . . 66
- 4.4. Quantized Tensor Network Kernel Machines . . . 70
- 4.5. Numerical Experiments . . . 73
- 4.6. Conclusion . . . 77
- A. Definitions . . . 83
- B. Proofs . . . 83
- C. Faster Multi-Convex Optimization Algorithms . . . 86

D. Numerical Experiments 87

5. Tensor Networks and Kernel Machines as Gaussian Processes **89**
5.1. Introduction 90
5.2. Background 91
5.3. TN-Constrained Kernel Machines as GPs 96
5.4. Numerical Experiments 101
5.5. Related Work 103
5.6. Conclusion 104
A. Introduction 109
B. TN-Constrained Kernel Machines as GPs 110
C. Numerical Experiments 117

6. A Kernelizable Primal-Dual Formulation of the MLSVD **119**
6.1. Introduction 120
6.2. Background 121
6.3. A Primal-Dual formulation for the MLSVD 122
6.4. Connection with Other Methods 130
6.5. Related work 134
6.6. Conclusions, Further Research and Applications 135
A. A Primal-dual formulation for the MLSVD 141

G. Conclusion **145**
G.1. Further Work 146

Curriculum Vitæ **151**

List of Publications **153**

SUMMARY

In an era where data-driven decision-making is becoming increasingly central to societal progress, the ability to efficiently process and analyze vast amounts of information is paramount. From healthcare and climate modeling to financial forecasting and autonomous systems, the demand for scalable and interpretable machine learning models has never been greater. However, as the complexity and dimensionality of data continue to grow, traditional machine learning approaches often struggle to keep pace. Kernel machines, a class of models renowned for their theoretical elegance and practical effectiveness, are no exception. While they offer powerful tools for learning complex patterns, their computational and memory requirements often scale prohibitively with the dimensionality of the data, limiting their applicability to real-world problems. This challenge is particularly acute in domains where high-dimensional data is the norm, such as medical imaging, genomics, and natural language processing.

At the heart of this issue lies the curse of dimensionality: as the number of features or dimensions in the data increases, the computational resources required to train and deploy kernel machines grow exponentially. This not only restricts their use in resource-constrained environments but also hinders their adoption in time-sensitive applications where rapid decision-making is critical. Moreover, the interpretability of these models, essential for gaining trust and actionable insights, often diminishes as their number of parameters grows. These limitations underscore a need for innovative approaches that can enhance the scalability and efficiency of kernel machines without compromising their predictive power.

This thesis addresses these challenges by exploring the intersection of *Tensor Networks* (TNs) and kernel machines, two fields that have traditionally evolved in parallel. Tensor networks, with their ability to efficiently represent high-dimensional data through low-rank structures, offer a promising avenue for overcoming the scalability limitations of kernel machines. By integrating the principles of TNs into kernel-based learning, this thesis aims to unlock new possibilities for modeling high-dimensional data while maintaining computational tractability and interpretability which characterizes kernel machines.

The contributions of this thesis are organized around three central questions, each addressing a critical aspect of the relationship between TNs and kernel ma-

chines. First, we investigate how TNs can be used to accelerate and enhance the scalability of kernel machines, while implicitly learning from a kernel function approximated to machine precision. Through a series of methodological advancements, we demonstrate that imposing low-rank TN constraints on the model weights can significantly reduce computational complexity while preserving the expressive power of kernel machines and enabling to learn from features that approximate the kernel function up to machine precision for both stationary and non-stationary kernels. Second, we explore the theoretical connections between TN-constrained kernel machines and Gaussian processes, shedding light on the conditions under which these models converge and generalize. Finally, we establish a novel optimization framework that characterizes a specific TN, the *Multilinear Singular Value Decomposition* (MLSVD), in terms of primal and dual problems, paving the way for new algorithmic developments and applications.

By exploring connections between TNs and kernel machines, this thesis not only advances the theoretical foundations of machine learning but also provides practical tools for addressing some of the most pressing challenges in data science. The methodologies developed here have the potential to democratize access to powerful machine learning models, enabling their use in a wider range of applications and ultimately contributing to societal progress in fields where data-driven insights are critical.

SAMENVATTING

In een tijdperk waarin datagestuurde besluitvorming steeds centraler staat in de maatschappelijke vooruitgang, is het vermogen om grote hoeveelheden informatie efficiënt te verwerken en analyseren van cruciaal belang. Van gezondheidszorg en klimaatmodellering tot financiële voorspellingen en autonome systemen, de vraag naar schaalbare en interpreteerbare machine learning-modellen is nog nooit zo groot geweest. Echter, naarmate de complexiteit en dimensionaliteit van data blijven toenemen, hebben traditionele machine learning-benaderingen vaak moeite om bij te blijven. Kernel machines, een klasse van modellen die bekend staan om hun theoretische elegantie en praktische effectiviteit, vormen hierop geen uitzondering. Hoewel ze krachtige tools bieden voor het leren van complexe patronen, schalen hun reken- en geheugeneisen vaak onhoudbaar met de dimensionaliteit van de data, wat hun toepasbaarheid op real-world problemen beperkt. Deze uitdaging is vooral acuut in domeinen waar hoogdimensionale data de norm is, zoals medische beeldvorming, genomica en natuurlijke taalverwerking.

De kern van dit probleem ligt in de zogenaamde vloek van dimensionaliteit: naarmate het aantal kenmerken of dimensies in de data toeneemt, groeien de benodigde rekenbronnen om kernel machines te trainen en in te zetten exponentieel. Dit beperkt niet alleen hun gebruik in omgevingen met beperkte middelen, maar belemmert ook hun adoptie in tijdgevoelige toepassingen waar snelle besluitvorming cruciaal is. Bovendien neemt de interpreteerbaarheid van deze modellen, essentieel voor het verkrijgen van vertrouwen en bruikbare inzichten, vaak af naarmate hun complexiteit toeneemt. Deze beperkingen onderstrepen een dringende maatschappelijke behoefte aan innovatieve benaderingen die de schaalbaarheid, efficiëntie en interpreteerbaarheid van kernel machines kunnen verbeteren zonder in te leveren op hun voorspellende kracht.

Deze scriptie gaat deze uitdagingen aan door het snijvlak van *Tensor Networks* (TNs) en kernel machines te verkennen, twee velden die traditioneel parallel zijn geëvolueerd. Tensornetwerken, met hun vermogen om hoogdimensionale data efficiënt weer te geven via laag-rang structuren, bieden een veelbelovende weg om de schaalbaarheidsbeperkingen van kernel machines te overwinnen. Door de principes van TNs te integreren in kernel-gebaseerd leren, streven we ernaar

nieuwe mogelijkheden te ontsluiten voor het modelleren van hoogdimensionale data, terwijl we rekenefficiëntie en interpreteerbaarheid behouden.

De bijdragen van deze scriptie zijn georganiseerd rond drie centrale vragen, elk gericht op een kritisch aspect van de relatie tussen TNs en kernel machines. Ten eerste onderzoeken we hoe TNs kunnen worden gebruikt om kernel machines te versnellen en hun schaalbaarheid te verbeteren, terwijl ze impliciet leren van een kernelfunctie die tot machineprecisie wordt benaderd. Door een reeks methodologische vooruitgangen tonen we aan dat het opleggen van laagrang TN-beperkingen aan de modelgewichten de rekencomplexiteit aanzienlijk kan verminderen, terwijl de expressieve kracht van kernel machines behouden blijft en het mogelijk wordt om te leren van kenmerken die de kernelfunctie benaderen tot op machineprecisie, zowel voor stationaire als niet-stationaire kernels. Ten tweede verkennen we de theoretische verbanden tussen TN-beperkte kernel machines en Gaussische processen, waardoor we inzicht krijgen in de voorwaarden waaronder deze modellen convergeren en generaliseren. Tot slot stellen we een nieuw optimalisatiekader op dat een specifiek TN, de *Multilinear Singular Value Decomposition* (MLSVD), karakteriseert in termen van primale en duale problemen, wat de weg vrijmaakt voor nieuwe algoritmische ontwikkelingen en toepassingen.

Door de verbanden tussen TNs en kernel machines te verkennen, draagt deze scriptie niet alleen bij aan de theoretische fundamenten van machine learning, maar biedt het ook praktische tools om enkele van de meest urgente uitdagingen in de datawetenschap aan te pakken. De hier ontwikkelde methodologieën hebben het potentieel om toegang tot krachtige machine learning-modellen te democratiseren, waardoor hun gebruik in een breder scala aan toepassingen mogelijk wordt en uiteindelijk bijdraagt aan maatschappelijke vooruitgang in domeinen waar datagestuurde inzichten cruciaal zijn.

1

INTRODUCTION

In an era where data-driven decision-making is becoming increasingly central to societal progress, the ability to efficiently process and analyze vast amounts of information is paramount [1]. From healthcare [2] and climate modeling [3] to financial forecasting [4], the demand for scalable and interpretable machine learning models has never been greater. These models are not only tools for scientific discovery but also enablers of transformative technologies that impact everyday life. For instance, in healthcare, they facilitate early disease detection and personalized treatment plans by analyzing complex medical data. In climate science, they help predict extreme weather events and inform mitigation strategies by processing vast amounts of environmental data. In finance, they enable real-time risk assessment and fraud detection by learning from high-dimensional transactional data. Similarly, in autonomous systems, they underpin perception and decision-making capabilities by interpreting sensory inputs in real time.

However, as the complexity and dimensionality of data continue to grow, traditional machine learning approaches often struggle to keep pace. Kernel machines, a class of models renowned for their theoretical elegance and practical effectiveness, are no exception. While they offer powerful tools for learning complex patterns, their computational and memory requirements often scale prohibitively with the dimensionality of the data, limiting their applicability to real-world problems. This challenge is particularly acute in domains where high-dimensional data is the norm, such as medical imaging, genomics, and natural language processing. For example, in medical imaging, the analysis of high-resolution scans requires models that can handle millions of voxels efficiently [5]. In genomics, the interpretation of gene expression data involves thousands of features, necessitating models that can scale without sacrificing accuracy [6]. In natural language processing, the representation of text data in high-dimensional embedding spaces demands models that can operate effectively within these spaces

[7].

At the heart of this issue lies the curse of dimensionality: as the number of features or dimensions in the data increases, the computational resources required to train and deploy kernel machines scale unfavourably [8, 9]. From a dual optimization perspective, said scaling is at least quadratic with the number of datapoints. This scaling arises because kernel methods typically rely on pairwise comparisons between data points, a fundamental operation that underlies their ability to capture complex relationships in the data. Specifically, kernel methods compute similarity measures between all pairs of data points, resulting in a kernel matrix whose size scales quadratically with the number of datapoints. This kernel matrix must be stored and manipulated, leading to significant computational and memory costs that quickly become intractable for large, possibly high-dimensional datasets. This quadratic scaling makes kernel methods impractical for modern datasets, which often consist of millions or billions of datapoints.

Moreover, the challenges of kernel methods extend beyond the kernel matrix itself. In order to break down the aforementioned quadratic complexity from quadratic to linear in the number of datapoints, a common strategy is to approximate the kernel function in terms of basis functions [10, 11]. A large number of basis functions are often required to approximate the kernel function effectively, particularly for high-dimensional input data. In many cases, the number of basis functions required to bound the approximation error grows exponentially with the dimensionality of the input data. [12, 13]. This exponential growth in the number of basis functions translates directly into an exponential increase in the number of model parameters, making it infeasible to handle data that is even moderately high-dimensional. As a result, while kernel methods are theoretically powerful and capable of learning highly complex functions, their practical application is limited by these scalability issues.

These limitations not only restrict the use of kernel machines in resource-constrained environments, such as edge computing devices or low-power systems, but also hinders their adoption in time-sensitive applications where rapid decision-making is critical. In fields like autonomous driving or real-time fraud detection, delays caused by computationally intensive models can have severe consequences, ranging from safety risks to financial losses. These limitations underscore a need for innovative approaches that can enhance the scalability and efficiency of kernel machines without compromising their predictive power and explainability.

This thesis addresses these challenges by exploring the intersection of *Tensor Networks* (TNs) and kernel machines, two fields that have traditionally evolved in parallel. TNs, with their ability to efficiently represent high-dimensional data through low-rank structures, offer a promising avenue for overcoming the scal-

ability limitations of kernel machines [14]. By integrating the principles of TNs into kernel-based learning, we aim to unlock new possibilities for modeling nonlinear high-dimensional and highly-sampled data while maintaining computational tractability. The methodologies developed here have the potential to democratize access to powerful machine learning models, enabling their use in a wider range of applications and ultimately contributing to societal progress in fields where data-driven insights are critical.

The remainder of this introduction are structured as follows. In Section 1.1 we briefly introduce the main concepts and notations regarding tensors adopted throughout the remainder of the thesis. In Section 1.2 we introduce the fundamentals regarding TNs, while in Section 1.3 we touch on the fundamental themes concerning kernel machines. Finally, in Section 1.4 we describe the layout of the rest of this thesis.

1.1 NOTATION AND TENSORS

A tensor of order D is an array with entries indexed with D indices. Types of tensors that are possibly familiar to the reader are tensors of order zero (scalars), tensors of order one (vectors), and tensors of order 2 (matrices). Henceforth the word tensors will generally refer to tensors of order higher than two. Throughout this thesis, tensors are represented using uppercase bold italics, e.g. $\mathcal{W}$, matrices with uppercase bold letters, e.g. $\boldsymbol{W}$, and vectors with lowercase bold letters, e.g. $\boldsymbol{w}$. Scalars are represented by lowercase letters, e.g. w, unless they denote the upper limit of an index, in which case they are uppercase. Element $m_1, m_2, \ldots, m_D$ of a tensor $\mathcal{W} \in \mathbb{C}^{M_1 \times M_2 \times \cdots \times M_D}$ of order D is denoted as $w_{m_1,m_2,\ldots,m_D}$. Vectorization, i.e. the process of converting a tensor into a vector, is achieved by flattening the tensor along a specified order of its indices. For a tensor $\mathcal{W} \in \mathbb{C}^{M_1 \times M_2 \times \cdots \times M_D}$, its vectorized form is denoted as $\mathrm{vec}(\mathcal{W}) \in \mathbb{C}^{M_1 M_2 \cdots M_D}$, such that

$$\mathrm{vec}(\mathcal{W})_m = w_{m_1,m_2,\ldots,m_D},$$

with $m = m_1 + \sum_{d=2}^{D}(m_d - 1)\prod_{k=1}^{d} M_k$. Likewise, its inverse, the tensorization operator $\mathrm{ten}(\cdot, M_1, M_2, \ldots, M_D) : \mathbb{C}^{M_1 M_2 \cdots M_D} \to \mathbb{C}^{M_1 \times M_2 \times \ldots M_D}$ is defined such that

$$\mathrm{ten}(\boldsymbol{w}, M_1, M_2, \ldots, M_D)_{m_1,m_2,\cdots,m_D} = w_m.$$

For simplicity, in the rest of the thesis it will be denoted in short form as $\mathrm{ten}(\cdot)$ unless otherwise necessary. The mode-d unfolding of a tensor $\mathcal{W} \in \mathbb{C}^{M_1 \times M_2 \times \cdots \times M_D}$

is denoted as $\boldsymbol{W}_{(d)}$, and is defined such that

$$w_{(d)\,m_d,m_1m_2\cdots m_D} = \text{ten}\,(\boldsymbol{W})_{m_1,m_2,\ldots,m_D}\,. \tag{1.1}$$

In the following, the left Kronecker product between two matrices $\boldsymbol{W}_1 \in \mathbb{C}^{M_1\times M_2}$ and $\boldsymbol{W}_2 \in \mathbb{C}^{N_1\times N_2}$ is denoted with $\otimes$ and defined such that

$$(\boldsymbol{W}_1 \otimes \boldsymbol{W}_2)_{m_1n_1,m_2n_2} = w_{1\,m_1,m_2}\; w_{2\,n_1,n_2}. \tag{1.2}$$

1.2 TENSOR NETWORKS

Tensor Networks (TNs) or *Tensor Decompositions* (TDs) extend the concept of rank from matrices to tensors, also known as higher-order arrays. Any matrix $\boldsymbol{W} \in \mathbb{R}^{M_1\times M_2}$ can be decomposed as

$$\boldsymbol{W} = \boldsymbol{W}_1\boldsymbol{W}_2^{\mathrm{T}}, \tag{1.3}$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{M_1\times R}$ and $\boldsymbol{W}_2 \in \mathbb{R}^{M_2\times R}$ and R is the *matrix rank*, upper bounded by

$$R \leq \min(M_1, M_2). \tag{1.4}$$

An important class of matrix decompositions are the so-called *rank-revealing* decompositions, which as their name suggests, decompose a matrix in terms of other matrices which allow to infer the rank by inspection. The most well-known one is the *Singular Value Decomposition* (SVD).

Definition 1.2.1 (SVD [15])**.** A matrix $\boldsymbol{W} \in \mathbb{C}^{M_1\times M_2}$ has a rank-R SVD if

$$w_{m_1,m_2} = \sum_{r=1}^{R} w_{1\,m_1,r}\; w_{2\,m_2,r}\; s_r,$$

where $\boldsymbol{W}_1 \in \mathbb{C}^{M_1\times R}$ and $\boldsymbol{W}_2 \in \mathbb{C}^{M_2\times R}$ such that $\boldsymbol{W}_1^{\mathrm{T}}\boldsymbol{W}_1 = \boldsymbol{W}_2^{\mathrm{T}}\boldsymbol{W}_2 = \boldsymbol{I}_R$ are matrices of *singular vectors* and $\boldsymbol{s} \in \mathbb{R}^R{}_+$ is the vector of *singular values*.

In tensors, there is not one unique concept of *tensor rank*. It is in fact possible to *decompose* a tensor in multiple *cores* using a different approaches, which in turn define their own notion of rank. Some of these decompositions are rank-revealing, others are not. We will here briefly review the most common TNs, which will be of use in the rest of this manuscript.

CANONICAL POLYADIC DECOMPOSITION

The *Canonical Polyadic Decomposition* (CPD), also known as *CANonical DECOMPosition* (CANDECOMP) or *PARAllel FACtor* (PARAFAC) analysis, was independently introduced as a method to factorize a tensor into a sum of component rank-1 tensors, gaining prominence as a a way to uncover unique hidden factors in multiway data, becoming foundational in tensor decomposition methods. The definition of the CPD is provided below.

Definition 1.2.2 (CPD [16])**.** A D-dimensional tensor $\mathscr{W} \in \mathbb{C}^{M_1 \times M_2 \times \cdots \times M_D}$ has a rank-R CPD if

$$w_{m_1,m_2,\ldots,m_D} = \sum_{r=1}^{R} \prod_{d=1}^{D} w_{d\,m_d,r}\, s_r.$$

The cores of this particular network are D factor matrices $\boldsymbol{W}_d \in \mathbb{C}^{M_d \times R}$, and the vector $\boldsymbol{s} \in \mathbb{R}^R$ typically contains the norms of the columns of the factor matrices. The storage complexity $P = R\sum_{d=1}^{D} M_d$ of a rank-R CPD is therefore $\mathcal{O}(DMR)$, where $M = \max(M_1, M_2, \ldots, M_D)$.

Compared to the SVD, the orthogonality constraint is absent. Notably, not all tensor admit a CPD [17, Section 3.3], and determining the CPD rank is a *Nondeterministic Polynomial-hard* (NP-HARD) problem [18]. The CPD is however unique under mild conditions (Krushkal conditions) [19]. The first applications of the CPD are historically rooted in psychometrics and chemometrics. More recent fields of application are signal processing (source separation, multichannel data analysis), spectroscopy, image compression, recommendation systems, compression of *Deep Neural Networks* (DNNs), and others. We redirect the interested reader in legacy applications to the comprehensive survey by Kolda and Bader [20]. More recent applications can be found in the survey by Panagakis *et al.* [21].

TENSOR TRAIN DECOMPOSITION

The *Tensor Train* (TT) decomposition can be traced back to concepts developed by White [22], particularly the *Matrix Product State* (MPS) in quantum physics, which were introduced as a way to represent quantum states efficiently in high-dimensional spaces. The idea of representing multi-dimensional data using low-rank tensor factorizations was formalized as the TT decomposition, building on previous tensor decomposition methods but optimizing for high-dimensional data by expressing tensors as a sequence of smaller, lower-dimensional tensors. This reinvention bridged a gap between quantum physics representations and

numerical methods for high-dimensional data. We provide the definition of the TT decomposition below.

Definition 1.2.3 (TT [23])**.** A D-dimensional tensor $\mathcal{W} \in \mathbb{C}^{M_1 \times M_2 \times \cdots \times M_D}$ admits a rank-$(R_1 := 1, R_2, \ldots, R_D, R_{D+1} := R_1)$ TT if

$$w_{m_1,m_2,\ldots,m_D} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_D=1}^{R_d} \prod_{d=1}^{D} w_{d\, r_d, m_d, r_{d+1}}.$$

The *cores* of a TT are the D 3-dimensional tensors $\mathcal{W}_d \in \mathbb{C}^{R_d \times M_d \times R_{d+1}}$. The case $R_1 > 1$ is also called a *Tensor Ring* (TR) [24]. The values $R_2, \ldots, R_D$ are called the TT ranks and are upper bounded by $R_d \leq \min\left(\prod_{p=1}^{d-1} M_p, \prod_{p=d+1}^{D} M_p\right)$. The storage complexity $P = \sum_{d=1}^{D} M_d R_d R_{d+1}$ of a TT is then $\mathcal{O}(DMR^2)$, where $M = \max(M_1, M_2, \ldots, M_D)$.

Unlike the CPD, the TT decomposition is not unique under trivial rescaling, meaning that the CPD might be a more suitable choice in certain circumstances, e.g. in unsupervised learning. Applications of the TT decomposition are data compression [25], deep learning [26, 27], signal processing [28], scientific computing [29], multivariate data analysis [30], and recommendation systems [31]. The interested reader is redirected to the monograph by Cichocki *et al.* [32] survey by Panagakis *et al.* [21].

TUCKER DECOMPOSITION

The Tucker decomposition is a higher-order generalization of the SVD for tensors, where a tensor is factored into a core tensor multiplied by matrices along each mode.

Definition 1.2.4 (Tucker Decomposition [33, 34])**.** A D-dimensional tensor $\mathcal{W} \in \mathbb{C}^{M_1 \times M_2 \times \cdots \times M_D}$ admits a rank-$(R_1, R_2, \ldots, R_D)$ Tucker decomposition if

$$w_{m_1,m_2,\ldots,m_D} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_D=1}^{R_d} \prod_{d=1}^{D} w_{d\, m_d, r_d}\; s_{r_1,r_2,\ldots,r_D}.$$

The cores of a Tucker decompositions are the D factor matrices $\boldsymbol{W}_d \in \mathbb{C}^{M_d \times R_d}$ and the *core tensor* $\mathcal{S} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_D}$. The values $(R_1, R_2, \ldots, R_D)$ are called the Tucker ranks and are upper bounded by $R_d \leq \min\left(M_d, \prod_{p=1}^{d-1} M_p \prod_{p=d+1}^{D} M_p\right)$. The storage complexity $P = \sum_{d=1}^{D} M_d R_d + \prod_{d=1} R_d$ of a Tucker decomposition is then $\mathcal{O}(R^D)$.

An important type of Tucker decomposition is the *Multilinear Singular Value Decomposition* (MLSVD). The MLSVD encompasses two ulterior constraints, namely the semi-orthogonality of the factor matrices, i.e. $\boldsymbol{W}_d^{\mathrm{T}}\boldsymbol{W}_d = \boldsymbol{I}_{R_d}$ and the all-orthogonality of the core, i.e. $\boldsymbol{S}_{(d)}{}^{\mathrm{T}}\boldsymbol{S}_{(d)}$ is a positive diagonal matrix with non-increasing entries. Due to the semi-orthogonality of the factor matrices, it is hence sometimes regarded as a generalization of *Principal Component Analysis* (PCA) to tensors. Unlike the CPD, determining the Tucker decomposition ranks of a tensor and carrying out its decompositions are operations that can be accomplished in polynomial time e.g. using MLSVD algorithm [35]. Example applications are chemometrics [36], signal processing [28] and deep learning [21]. As previously the case, the interested reader is redirected to the surveys by Kolda and Bader [20] and Panagakis *et al.* [21].

A TN is *underparametrized* if $P \ll \prod_{d=1}^{D} M_d$, i.e. it can represent a tensor with fewer parameters than the number of entries of the tensor. Other TNs are the hierarchical hierarchical Tucker [37, 38] decomposition, block-term decompositions [39, 40], *Projected Entangled Pair States* (PEPS) [41] and *Multi-scale Entanglement Renormalization Ansatz* (MERA) [42].

1.3 KERNEL MACHINES

Supervised learning is one of the main paradigms in *Machine Learning* (ML). In its most standard setting, supervised learning is characterized by the presence of a labeled dataset $\{(\boldsymbol{x}_n, y_n)_{n=1}^{N}\}$, which consists of N i.d.d. inputs-output pairs, where $\boldsymbol{x}_n \in \mathbb{R}^D$ are the inputs and $y_n \in \mathbb{R}$ are the outputs. The learning goal is to find a model $f(\cdot, \boldsymbol{w}) : \mathbb{R}^D \to \mathbb{R}$, typically parameterized in terms of of weights $\boldsymbol{w} \in \mathbb{R}^P$, which describes the relationship between inputs and outputs and enables to make prediction on unseen inputs.

Training a model consists then in finding a model $f(\cdot, \boldsymbol{w}^*)$ such that the input-output relationship is best described according to a measure of loss. One common approach to do so is by performing *Regularized Loss Minimization* (RLM), i.e. seeking a set of weights which minimizes the regularized loss

$$\boldsymbol{w}^* := \operatorname*{argmin}_{\boldsymbol{w}} \sum_{n=1}^{N} \ell(f(\boldsymbol{x}_n), y_n) + r(\boldsymbol{w}). \tag{1.5}$$

Here $\ell(\cdot,\cdot) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ is a measure of loss, sometimes called loss function which penalizes models that deviate from the data, and $r(\cdot) : \mathbb{R}^P \to \mathbb{R}$ is an explicit regularization term, which penalizes the model complexity. For context, examples of

model parametrization are *Feed-Forward Neural Network* (FNN), *Convolutional Neural Network* (CNN) or *Support Vector Machine* (SVM), while examples of regularization are lasso, ridge or elastic net. Other common optimization strategies are *Empirical Risk Minimization* (ERM), *Structural Risk Minimization* (SRM).

1.3.1 LINEAR MODELS

As their names suggests, linear models model the relationship between the inputs and outputs as linear in both the data and the model weights

$$f(\boldsymbol{x}, \boldsymbol{w}) = \langle \boldsymbol{x}, \boldsymbol{w} \rangle . \tag{1.6}$$

Linear models are foundational in both statistical learning and machine learning, due to their simplicity and interpretability. They assume that the relationship between the inputs and the outputs is linear, which can be particularly useful in situations where the underlying process governing the data is indeed linear, or when the goal is to provide an interpretable approximation of a more complex system. Notably, the computational cost of evaluating Eq. (1.6) and its gradient is of $\mathcal{O}(D)$, i.e. the scaling is linear in the dimensionality of the input feature and of model parameters. Examples of linear models which solve Eq. (1.5) with different loss functions are *Least-Squares Support Vector Machine* (LS-SVM) [43], *Ridge Regression* (RR) [44], *Lasso Regression* (LR) [45], and others such as *linear discriminant analysis* (LDA) and Poisson regression [9].

1.3.2 KERNEL MACHINES

Kernel machines provide a straightforward strategy for linear models to handle nonlinear data by mapping the inputs into a typically higher-dimensional space using a nonlinear feature map $\boldsymbol{\varphi}(\cdot) : \mathbb{R}^D \to \mathbb{R}^M$

$$f(\boldsymbol{x}, \boldsymbol{w}) = \left\langle \boldsymbol{\varphi}(\boldsymbol{x}), \boldsymbol{w} \right\rangle , \tag{1.7}$$

where $\boldsymbol{w} \in \mathbb{R}^M$ are the model weights. Arguably the most common choices of features are polynomials or Fourier features, as they guarantee that kernel machines can approximate any nonlinear function if the number of basis function is high enough [46]. The computational cost associated with computing Eq. (1.7) and its gradient is of $\mathcal{O}(M)$, i.e. linear with the number of features. Importantly, placing a normal prior on the weights $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda})$ yields the *weight-space* or parametric formulation of *Gaussian Processes* (GPs) [11]. A GP is a normal distribution over

a class of functions, parameterized in terms of a mean and covariance (kernel) function.

1.3.3 TENSOR-NETWORK CONSTRAINED KERNEL MACHINES

TNs-constrained kernel machines, as the name suggests, add an additional constraint to Eq. (1.7) which forces the model weights to be a vectorized TN of *low rank*, and considers features which have Kronecker product structure. The resulting optimization problem is then

$$\boldsymbol{w}^* := \underset{\boldsymbol{w}}{\operatorname{argmin}} \sum_{n=1}^{N} \ell(\langle \boldsymbol{\varphi}_1(\boldsymbol{x}) \otimes \boldsymbol{\varphi}_2(\boldsymbol{x}) \otimes \cdots \otimes \boldsymbol{\varphi}_Q(\boldsymbol{x}), \boldsymbol{w} \rangle, y_n) + r(\boldsymbol{w}). \tag{1.8}$$

$$\text{subject to: } \text{TN-rank}(\text{ten}(\boldsymbol{w})) = \boldsymbol{r}, \tag{1.9}$$

where $\boldsymbol{r}$ here indicates the vector containing the additional hyperparameters which are the ranks of the TN, and each $\boldsymbol{\varphi}_q(\cdot) : \mathbb{R}^D \to \mathbb{R}^{M_d}$ maps the inputs or a subset of the inputs to a higher space. Said models were first considered by Wahls *et al.* [47] and Stoudenmire and Schwab [48] (TT and Fourier features), Chen *et al.* [49] and Novikov, Oseledets, and Trofimov [50] (TT and polynomials). Said models have the advantage of being able to learn from a large $\prod_{q=1}^{Q} M_q$ number of features, with a restricted number of model parameters, which is fully determined by the choice of TN and of TN rank. Additionally, said models have been shown to posses some degree of regularization due to the imposed low-rank structure, often achieving better generalization than kernel machines without the additional TN constraint [47, 49, 50].

1.3.4 PRIMAL-DUAL OPTIMIZATION AND KERNELIZATION

Training a kernel machine in a supervised learning setting amounts typically to solve Eq. (1.5), which is also called the *primal optimization*. Due to the i.i.d. assumption, training with a stochastic first-order gradient-based method has a computational cost equal to the number of model parameters, i.e. $\mathcal{O}(P)$, which is independent of the number of inputs and outputs in the training set. If the loss function and regularization term are convex, a global optimum can be found, possibly closed form, e.g. if the loss function is the sum of squares and with lasso or ridge regularization. Alternatively, one can solve the associated *dual optimiza-*

tion problem, which requires to compute the inner products

$$k(\boldsymbol{x}, \boldsymbol{x}_n) := \left\langle \boldsymbol{\varphi}(\boldsymbol{x}), \boldsymbol{\varphi}(\boldsymbol{x}_n) \right\rangle, \tag{1.10}$$

for all inputs in the training set. This requires a computational cost of at least $\mathcal{O}(N)$, i.e. which is linear and fixed in terms of the size of the training set, rendering the dual optimization problem only advantageous if $N \ll M$. Importantly, it is not necessary to compute the inner products in Eq. (1.10) explicitly as long as we can compute the *kernel function* $k(\cdot,\cdot)$ directly. This is the key idea behind the *kernel trick*, which allows us to implicitly map the data into a high-dimensional space without ever explicitly calculating the feature mapping $\boldsymbol{\varphi}(\cdot)$. As a result, we can perform optimization in the dual space without having to deal with the computational burden of working with the potentially infinite-dimensional feature space.

1.4 CONTRIBUTIONS OF THIS THESIS

In this thesis, we broadly explore the relation between TNs and kernel machines, i.e. the models in Eq. (1.7). In particular we investigate:

How can tensor networks be used to accelerate and enhance the scalability of kernel machines?

We explore this idea in Chapters 2 to 4, where we train kernel machines with the additional constraint that the tensorized model weights are a *low-rank* TN (Eq. (1.8)).

More specifically, in Chapter 2 we consider models defined by weighted products of Fourier features, which are commonly used in literature to approximate stationary kernel functions. These approximations can be very accurate, but require an exponential number of basis functions in the dimensionality of the data. By casting the problem as a TN-constraint kernel machine, i.e. by introducing the additional constraint that the tensorized model weights are a low-rank TN, we enable learning kernel machines with linear complexity in the dimensionality of the data from stationary kernel functions which are implicitly approximated up to machine precision.

In Chapter 3, we explore the same idea in the context of *Nyström* or *inducing point*-based approximations of the kernel function, which can be regarded as choosing the kernel function itself as a basis function. In particular, we notice that when considering product kernels, placing the inducing

points on a Cartesian grid yields features which have Kronecker product structure. By imposing a low-rank TN constraint on the model weights, we can develop kernel machines that achieve linear complexity with respect to the dimensionality of the data. These models effectively learn from a product kernel function that is approximated to machine precision.

In Chapter 4, we observe that some Fourier and polynomial basis functions contain further latent product structure, i.e. can be *quantized* (not to be confused with the practice of training models with lower machine precision), or in other words, tensorized along a number of fictitious modes. As a consequence, the model parameters in the associated kernel machine can be quantized as well. Thanks to low-rank TN constraint imposed on the weight tensor characterized by an artificially increased number of modes, the resulting quantized model is able to describe more patterns for the same number of parameters. We show that, for the same number of model parameters, the resulting quantized TN-constrained kernel machines have a higher bound on the *Vapnik–Chervonenkis* (VC)-dimension (the largest set of points that can be classified in all possible ways) as opposed to their non-quantized counterparts, at no additional computational cost, while learning from identical features. In simpler terms, this means that quantized models can learn from the exact same functions of non-quantized models and build a model which generalizes better with fewer model parameters and thus faster.

In which ways do the resulting tensor network-constrained kernel machines relate to Gaussian processes?

In Chapter 5, we establish a formal connection between TN-constrained kernel machines and GPs in a novel way, distinct from previous approaches that treated the TN as a constraint on the GP. We observe that the TN-constrained models converge to a GP when the limit of the ranks tends to infinity and when an appropriate prior is specified on their cores. A CPD-constrained model converges faster to the GP than a TT-constrained model. The proposed priors, unlike the ones that are implicitly considered in Chapters 2 to 4, are less strict, and allow the model to better generalize or overfit more depending on the training regime the model is in, providing an alternative regularization strategy to the interested practitioner.

Is it possible to characterize tensor networks in terms of a primal and dual optimization problem?

In Chapter 6, we establish a primal optimization problem whose associated

dual optimization problem is a Tucker decomposition (Definition 1.2.4) with the additional constraints of semi-orthogonality on the factor matrices, and all-orthogonality of the core tensors, in other words, the MLSVD. In the spirit of kernel machines, said primal optimization problem describes the MLSVD parametrically in terms of model weights, paving the way for more computationally and memory efficient implementations and for the kernelization of the decomposition. We also discuss several lines of application.

Other Contributions Additional contributions that are not integral to this thesis can be found listed in the List of Publications.

REFERENCES

[1] E. Brynjolfsson, L. M. Hitt, and H. H. Kim. *Strength in Numbers: How Does Data-Driven Decisionmaking Affect Firm Performance?* SSRN Scholarly Paper. Rochester, NY, Apr. 2011. Social Science Research Network: 1819486.

[2] S. Secinaro, D. Calandra, A. Secinaro, V. Muthurangu, and P. Biancone. "The Role of Artificial Intelligence in Healthcare: A Structured Literature Review". In: *BMC Medical Informatics and Decision Making* 21.1 (Apr. 2021), p. 125.

[3] C. Huntingford, E. S. Jeffers, M. B. Bonsall, H. M. Christensen, T. Lees, and H. Yang. "Machine Learning and Artificial Intelligence to Aid Climate Change Research and Preparedness". In: *Environmental Research Letters* 14.12 (Nov. 2019), p. 124007.

[4] S. Bahoo, M. Cucculelli, X. Goga, and J. Mondolo. "Artificial Intelligence in Finance: A Comprehensive Review through Bibliometric and Content Analysis". In: *SN Business & Economics* 4.2 (Jan. 2024), p. 23.

[5] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. "3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*. Ed. by S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. Unal, and W. Wells. Cham: Springer International Publishing, 2016, pp. 424–432.

[6] J. Gui and H. Li. "Penalized Cox Regression Analysis in the High-Dimensional and Low-Sample Size Settings, with Applications to Microarray Gene Expression Data". In: *Bioinformatics* 21.13 (July 2005), pp. 3001–3008.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. ukasz Kaiser, and I. Polosukhin. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[8] Y. Bengio, O. Delalleau, and N. Roux. "The Curse of Highly Variable Functions for Local Kernel Machines". In: *Advances in Neural Information Processing Systems*. Vol. 18. MIT Press, 2005.

[9] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, July 2006.

[10] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Nov. 2002.

[11] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press, 2006.

[12] J. Hensman, N. Durrande, and A. Solin. “Variational Fourier Features for Gaussian Processes”. In: *The Journal of Machine Learning Research* 18.1 (Jan. 2017), pp. 5537–5588.

[13] A. Solin and S. Särkkä. “Hilbert Space Methods for Reduced-Rank Gaussian Process Regression”. In: *Statistics and Computing* 30.2 (Mar. 2020), pp. 419–446.

[14] A. Cichocki. “Era of Big Data Processing: A New Approach via Tensor Networks and Tensor Decompositions”. In: *arXiv:1403.2048 [cs]* (Aug. 2014). arXiv: 1403.2048 [cs].

[15] G. H. Golub and C. F. Van Loan. *Matrix Computations (3rd Ed.)* USA: Johns Hopkins University Press, Oct. 1996.

[16] F. L. Hitchcock. “The Expression of a Tensor or a Polyadic as a Sum of Products”. In: *Journal of Mathematics and Physics* 6.1-4 (1927), pp. 164–189.

[17] T. G. Kolda. “Orthogonal Tensor Decompositions”. In: *SIAM Journal on Matrix Analysis and Applications* 23.1 (Jan. 2001), pp. 243–255.

[18] J. Håstad. “Tensor Rank Is NP-Complete”. In: *Journal of Algorithms* 11.4 (Dec. 1990), pp. 644–654.

[19] N. D. Sidiropoulos and R. Bro. “On the Uniqueness of Multilinear Decomposition of N-Way Arrays”. In: *Journal of Chemometrics* 14.3 (2000), pp. 229–239.

[20] T. G. Kolda and B. W. Bader. “Tensor Decompositions and Applications”. In: *SIAM Review* 51.3 (Aug. 2009), pp. 455–500.

[21] Y. Panagakis, J. Kossaifi, G. G. Chrysos, J. Oldfield, M. A. Nicolaou, A. Anandkumar, and S. Zafeiriou. “Tensor Methods in Computer Vision and Deep Learning”. In: *Proceedings of the IEEE* 109.5 (May 2021), pp. 863–890.

[22] S. R. White. “Density Matrix Formulation for Quantum Renormalization Groups”. In: *Physical Review Letters* 69.19 (Nov. 1992), pp. 2863–2866.

[23] I. V. Oseledets. “Tensor-Train Decomposition”. In: *SIAM Journal on Scientific Computing* 33.5 (Jan. 2011), pp. 2295–2317.

[24] Q. Zhao, G. Zhou, S. Xie, L. Zhang, and A. Cichocki. “Tensor Ring Decomposition”. In: *arXiv:1606.05535 [cs]* (June 2016). arXiv: 1606.05535 [cs].

[25] K. Batselier, W. Yu, L. Daniel, and N. Wong. “Computing Low-Rank Approximations of Large-Scale Matrices with the Tensor Network Randomized SVD”. In: *SIAM Journal on Matrix Analysis and Applications* 39.3 (Jan. 2018), pp. 1221–1244.

[26] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov. “Tensorizing Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc., 2015.

[27] A. Tjandra, S. Sakti, and S. Nakamura. “Compressing Recurrent Neural Network with Tensor Train”. In: *2017 International Joint Conference on Neural Networks (IJCNN)*. May 2017, pp. 4451–4458.

[28] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos. “Tensor Decomposition for Signal Processing and Machine Learning”. In: *IEEE Transactions on Signal Processing* 65.13 (July 2017), pp. 3551–3582.

[29] B. N. Khoromskij. “Tensors-Structured Numerical Methods in Scientific Computing: Survey on Recent Advances”. In: *Chemometrics and Intelligent Laboratory Systems* 110.1 (Jan. 2012), pp. 1–19.

[30] I. V. Oseledets and E. E. Tyrtyshnikov. “Algebraic Wavelet Transform via Quantics Tensor Train Decomposition”. In: *SIAM Journal on Scientific Computing* 33.3 (Jan. 2011), pp. 1315–1328.

[31] C. Yin, B. Acun, C.-J. Wu, and X. Liu. “TT-Rec: Tensor Train Compression for Deep Learning Recommendation Models”. In: *Proceedings of Machine Learning and Systems* 3 (Mar. 2021), pp. 448–462.

[32] A. Cichocki, A.-H. Phan, Q. Zhao, N. Lee, I. V. Oseledets, M. Sugiyama, and D. Mandic. “Tensor Networks for Dimensionality Reduction and Large-Scale Optimizations. Part 2 Applications and Future Perspectives”. In: *Foundations and Trends® in Machine Learning* 9.6 (2017), pp. 249–429. arXiv: 1708.09165.

[33] L. R. Tucker. “Implications of Factor Analysis of Three-Way Matrices for Measurement of Change”. In: *Problems in measuring change* 15.122-137 (1963), p. 3.

[34] L. R. Tucker. “Some Mathematical Notes on Three-Mode Factor Analysis”. In: *Psychometrika* 31.3 (Sept. 1966), pp. 279–311.

[35] L. De Lathauwer, B. De Moor, and J. Vandewalle. “A Multilinear Singular Value Decomposition”. In: *SIAM Journal on Matrix Analysis and Applications* 21.4 (Jan. 2000), pp. 1253–1278.

[36] R. Bro and A. K. Smilde. “Principal Component Analysis”. In: *Analytical Methods* 6.9 (2014), pp. 2812–2831.

[37] W. Hackbusch and S. Kühn. “A New Scheme for the Tensor Representation”. In: *Journal of Fourier Analysis and Applications* 15.5 (Oct. 2009), pp. 706–722.

[38] L. Grasedyck. “Hierarchical Singular Value Decomposition of Tensors”. In: *SIAM Journal on Matrix Analysis and Applications* 31.4 (Jan. 2010), pp. 2029–2054.

[39] L. De Lathauwer. “Decompositions of a Higher-Order Tensor in Block Terms—Part I: Lemmas for Partitioned Matrices”. In: *SIAM Journal on Matrix Analysis and Applications* 30.3 (Jan. 2008), pp. 1022–1032.

[40] L. De Lathauwer. “Decompositions of a Higher-Order Tensor in Block Terms—Part II: Definitions and Uniqueness”. In: *SIAM Journal on Matrix Analysis and Applications* 30.3 (Jan. 2008), pp. 1033–1066.

[41] F. Verstraete and J. I. Cirac. *Renormalization Algorithms for Quantum-Many Body Systems in Two and Higher Dimensions*. July 2004. arXiv: cond-mat/0407066.

[42] G. Evenbly and G. Vidal. “Algorithms for Entanglement Renormalization”. In: *Physical Review B* 79.14 (Apr. 2009), p. 144108.

[43] J. Suykens and J. Vandewalle. “Least Squares Support Vector Machine Classifiers”. In: *Neural Processing Letters* 9.3 (June 1999), pp. 293–300.

[44] C. Saunders, A. Gammerman, and V. Vovk. “Ridge Regression Learning Algorithm in Dual Variables”. In: *Proceedings of the Fifteenth International Conference on Machine Learning*. ICML ’98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., July 1998, pp. 515–521.

[45] V. Roth. “The Generalized LASSO”. In: *IEEE Transactions on Neural Networks* 15.1 (Jan. 2004), pp. 16–28.

[46] B. Hammer and K. Gersmann. “A Note on the Universal Approximation Capability of Support Vector Machines”. In: *Neural Processing Letters* 17.1 (Feb. 2003), pp. 43–53.

[47] S. Wahls, V. Koivunen, H. V. Poor, and M. Verhaegen. “Learning Multidimensional Fourier Series with Tensor Trains”. In: *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. Dec. 2014, pp. 394–398.

[48] E. M. Stoudenmire and D. J. Schwab. “Supervised Learning with Tensor Networks”. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. Curran Associates Inc., Dec. 2016, pp. 4806–4814.

[49] Z. Chen, K. Batselier, J. A. K. Suykens, and N. Wong. “Parallelized Tensor Train Learning of Polynomial Classifiers”. In: *IEEE Transactions on Neural Networks and Learning Systems* 29.10 (Oct. 2018), pp. 4621–4632.

[50] A. Novikov, I. Oseledets, and M. Trofimov. “Exponential Machines”. In: *Bulletin of the Polish Academy of Sciences: Technical Sciences; 2018; 66; No 6 (Special Section on Deep Learning: Theory and Practice); 789-797* (2018).

2

LARGE-SCALE LEARNING WITH FOURIER FEATURES AND TENSOR DECOMPOSITIONS

Random Fourier features provide a way to tackle large-scale machine learning problems with kernel methods. Their slow Monte Carlo convergence rate has motivated the research of deterministic Fourier features whose approximation error can decrease exponentially in the number of basis functions. However, due to their tensor product extension to multiple dimensions, these methods suffer heavily from the curse of dimensionality, limiting their applicability to one, two or three-dimensional scenarios. In our approach we overcome said curse of dimensionality by exploiting the tensor product structure of deterministic Fourier features, which enables us to represent the model parameters as a low-rank tensor decomposition. We derive a monotonically converging block coordinate descent algorithm with linear complexity in both the sample size and the dimensionality of the inputs for a regularized squared loss function, allowing to learn a parsimonious model in decomposed form using deterministic Fourier features. We demonstrate by means of numerical experiments how our low-rank tensor approach obtains the same performance of the corresponding nonparametric model, consistently outperforming random Fourier features.

This chapter has been published as:
F. Wesel and K. Batselier. "Large-Scale Learning with Fourier Features and Tensor Decompositions". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 17543–17554

2.1 INTRODUCTION

Kernel methods such as *Support Vector Machines* (SVMs) and *Gaussian Processes* (GPs) are commonly used to tackle problems such as classification, regression and dimensionality reduction. Since they can be universal function approximators [2], kernel methods have received renewed attention in the last few years and have shown equivalent or superior performance to *Neural Networks* (NNs) [3, 4, 5]. The main idea behind kernel methods is to lift the data into a higher-dimensional (or even infinite-dimensional) Reproducing Kernel Hilbert Space by means of a *feature map* $\boldsymbol{\varphi}(\cdot) : \mathbb{R}^D \to \mathbb{R}^M$. Considering then the pairwise similarities between the mapped data allows to tackle problems which are highly nonlinear in the original sample space. This can be done equivalently by considering a *kernel function* $k(\cdot,\cdot) : \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}$ such that $\langle \boldsymbol{\varphi}(\boldsymbol{x}), \boldsymbol{\varphi}(\boldsymbol{x'}) \rangle = k(\boldsymbol{x}, \boldsymbol{x'})$ and performing thus said mapping implicitly. Although effective at learning nonlinear patterns in the data, kernel methods are known to scale poorly as the number of data points N increases. For example, when considering *Kernel Ridge Regression* (KRR), *Gaussian Process Regression* (GPR) [6] or *Least-Squares Support Vector Machine* (LS-SVM) [7, 8] training usually consist in inverting the *Gram matrix* $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, which encodes the pairwise relation between all data. As a consequence, the associated storage complexity is $\mathcal{O}(N^2)$ and the computational complexity is $\mathcal{O}(N^3)$, rendering these methods unfeasible for large data. In order to lower the computational cost, *data-dependent* methods approximate the kernel function by means of M data-dependent basis functions. Due to their reduced-rank formulation, the computational complexity is reduced to $\mathcal{O}(NM^2)$ for $N \gg M$. However, e.g. for the Nyström method [9], the convergence rate is only of $\mathcal{O}\left(\frac{1}{\sqrt{M}}\right)$ [10] limiting its effectiveness. As the name suggests, *data-independent* methods approximate the kernel function by M data-independent basis functions. A good example is the celebrated *Random Fourier Features* (RFF) approach by Rahimi and Recht [11], where the authors propose for stationary kernels a low-dimensional *random* mapping $\boldsymbol{z}(\cdot) : \mathbb{R}^D \to \mathbb{R}^M$ such that

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \langle \boldsymbol{\varphi}(\boldsymbol{x}), \boldsymbol{\varphi}(\boldsymbol{x'}) \rangle \approx \langle \boldsymbol{z}(\boldsymbol{x}), \boldsymbol{z}(\boldsymbol{x'}) \rangle . \tag{2.1}$$

As in the case of data-dependent methods, the reduced-rank formulation allows for a computational complexity of $\mathcal{O}(NM^2)$ for $N \gg M$. Probabilistic error bounds on the approximation are provided which result in a convergence rate of $\mathcal{O}\left(\frac{1}{\sqrt{M}}\right)$, which is again the Monte Carlo rate.

Improvements in this sense were achieved by considering *deterministic* fea-

tures resulting from dense quadrature methods [12, 13, 14] and kernel eigenfunctions [15, 16]. These methods are able to achieve *exponentially* decreasing upper bounds on uniform convergence guarantees when certain conditions are met. However, for a D-dimensional input space these methods take the tensor product of D vectors, resulting in an exponential increase in the number of basis functions and thus of model weights, effectively limiting the applicability of deterministic features to low-dimensional data. In this work we consider deterministic features. In order to take advantage of the tensor product structure which arises when mapping the inputs to $\mathscr{H}$, we represent the weights as a low-rank tensor. This allows us to *learn* the inter-modal relations in the tensor product of (low-dimensional) Hilbert spaces, avoiding the exponential computational and storage complexities in D. In this way we are able to obtain a *linear* computational complexity in both the number of samples *and* in the input dimension during training, without having to resort to the use of sparse grids or additive modeling of kernels.

The main contribution of this work is in lifting the curse of dimensionality affecting deterministic Fourier features by modeling the weights as a low-rank tensor. This enables the efficient solution of large-scale and high-dimensional kernel learning problems. We derive an iterative algorithm under the exemplifying case of regularized squared loss and test it on regression and classification problems.

2.2 RELATED WORK

Fourier Features (FFs) are a collection of data-independent methods that leverage Bochner's theorem [17] from harmonic analysis to approximate stationary kernels by numerical integration of their spectral density $p(\cdot)$:

$$k(\boldsymbol{x},\boldsymbol{x}') \overset{\text{Stationarity}}{:=} k(\boldsymbol{x}-\boldsymbol{x}') \overset{\text{Bochner}}{=} \int p(\boldsymbol{\omega}) \exp\left(\left\langle i\boldsymbol{\omega}, (\boldsymbol{x}-\boldsymbol{x}')\right\rangle\right) d\boldsymbol{\omega} \overset{\text{FF}}{\approx} \left\langle z(\boldsymbol{x}), z(\boldsymbol{x}')\right\rangle. \tag{2.2}$$

Rahimi and Recht [11] proposed to approximate the integral by Monte Carlo integration i.e. by drawing M random frequencies $\boldsymbol{\omega} \sim p(\cdot)$. In their work they show how the method converges uniformly at the Monte Carlo rate. See [18] for an overview of RFF. In order to achieve faster convergence and a lower sample complexity, a multitude of approaches that rely on deterministic numerical quadrature of the Fourier integral were developed. These methods generally consider

product kernels whose spectral density factors in the frequency domain, which enables in turn to factor the Fourier integral. The resulting deterministic Fourier features are then the tensor product of D one-dimensional deterministic features.

For example, in [12] the authors give an analysis of the sample complexity of features resulting from dense *Gaussian Quadrature* (GQ). In [13] the authors present a similar quadrature approach for kernels whose spectral density factors over the dimensions of the inputs. They provide an explicit construction for their *Quadrature Fourier Features* relying on a dense Cartesian grid, and note that their method, as well as GQ, can attain exponentially decreasing uniform convergence bounds in the total number of basis functions per dimensions M [13, Theorem 1]. To avoid the curse of dimensionality, they make use of additive modeling of kernels. *Variational Fourier Features* (VFF) [15] are derived probabilistically in a one-dimensional setting for Matérn kernels by projecting a GP onto a set of Fourier basis. An extension to multiple dimensions when considering product kernels is then achieved by taking the tensor product of the one-dimensional features, incurring however again in exponentially rising computational costs in D. In what they call *Hilbert–space Gaussian Process* (HGP) [16] the authors diagonalize stationary kernels in terms of the eigenvalues and eigenfunctions of the Laplace operator with Dirichlet boundary conditions. Due to the multiplicative structure of the eigenfunctions of the Laplace operator, the complexity of the basis function increases exponentially in D when one considers product kernels. Like with other deterministic approximations, bounds on the uniform convergence error which decrease exponentially in M can be achieved [16, Theorem 8].

Tensor decompositions have been used extensively in machine learning in order to benefit from structure in data [19], in particular to obtain an efficient representation of the model parameters. In the context of GPR, the tensor product structure which arises when the inputs are located on a Cartesian grid have been exploited to speedup inference [20]. In their variational inference framework, [21] proposed to parameterize the posterior mean of a GP by means of a tensor decomposition in order to exploit the tensor product structure which arises when interpolating the kernel function. In the context of NNs, [22] proposed to represent the weights in deep NNs as a tensor decomposition to speed up training. A similar approach was carried out in case of recurrent NNs [23] [24].

Tensor decompositions have also been used to learn from simple features. In [25] multivariate functions are learned from a Fourier basis, which corresponds to assuming a uniform spectral density in Eq. (2.2). Motivated by spin vectors in quantum mechanics, [26] consider trigonometric feature maps which they argue induce uninformative kernels. On a similar note, [27] and [28] leverage the

tensor product structure of polynomial mappings to induce a polynomial kernel and consider model parameters in decomposed form. While these existing approaches are able to overcome the curse of dimensionality affecting tensor product feature maps by modeling the parameter tensor as a tensor network, they consider simple and empirical feature maps which induce uninformative kernels. In the following section we show how deterministic Fourier features can be used for supervised learning in both large and high-dimensional scenarios by assuming that the model weights are a low-rank tensor, thereby linking tensor decompositions with stationary product kernels.

2.3 LEARNING WITH FOURIER FEATURES

2.3.1 THE MODEL

In this article we assume models of the form

$$f(\boldsymbol{x}, \boldsymbol{w}) = \left\langle \boldsymbol{\varphi}(\boldsymbol{x}), \boldsymbol{w} \right\rangle. \tag{2.3}$$

Here $\boldsymbol{x} \in \mathbb{R}^D$ is the input vector, $\boldsymbol{\varphi}(\cdot) : \mathbb{R}^D \to \mathbb{R}^M$ is a feature map, $\boldsymbol{w} \in \mathbb{R}^M$ are the model weights, $y \in \mathbb{R}$ is the corresponding available observation. Learning such a kernelized model consists in finding as set of weights $\boldsymbol{w}$ such that

$$\sum_{n=1}^{N} \ell\left(f(\boldsymbol{x}_n, \boldsymbol{w}), y_n\right) + r(\boldsymbol{w}), \tag{2.4}$$

is minimized. Here, $\ell(\cdot,\cdot) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ is a symmetric and positive loss function and $r(\boldsymbol{w})$ is a regularization term. A variety of *primal* machine learning formulations arise when considering different combinations of loss functions and regularization terms. For example, considering a ridge regularization term and squared loss leads to KRR [8], while considering hinge loss leads to SVM [29], and so on. Furthermore, applying the *kernel trick* when considering $\boldsymbol{\varphi}(\cdot)$ recovers the nonparametric formulation which, depending on the choice of kernel, enables to perform inference using infinitely many basis functions with a computational complexity of $\mathcal{O}(N^3)$. Considering instead a low-dimensional finite mapping $\boldsymbol{z}(\cdot) : \mathbb{R}^D \to \mathbb{R}^M$ such as RFF or the Nyström method leads a primal approach and to computational savings with a computational complexity of $\mathcal{O}\left(NM^2\right)$ for $N \gg M$. However, as already discussed, these low-dimensional random mappings converge at the slow Monte Carlo rate, motivating our approach.

In order to approximate the kernel function with faster and possibly *exponential* convergence, we consider deterministic, finite-dimensional features which are the tensor product of D vectors. For a given D-dimensional input point $\boldsymbol{x}$ we define the deterministic feature mapping $\boldsymbol{z}(\cdot) : \mathbb{R}^D \to \mathbb{R}^{M_1 M_2 \cdots M_D}$ as

$$\boldsymbol{z}(\boldsymbol{x}) = \boldsymbol{z}_1(x_1) \otimes \boldsymbol{z}_2(x_2) \cdots \otimes \boldsymbol{z}_D(x_D), \tag{2.5}$$

where $\boldsymbol{z}_d$ is a deterministic mapping applied to the d-th dimension. The dimension of the feature space is $\prod_{d=1}^{D} M_d$. It is easy to verify by applying the kernel trick that the features of Eq. (2.5) yield product kernels. The mapping $\boldsymbol{z}(\cdot)$ encompasses for instance the mappings derived by quadrature [12, 13, 14] and by projection [15, 16]. Note that these mappings cover many popular kernels such as the Gaussian and Matérn kernels. To give a concrete example, in the framework of A. Solin and S. Särkkä [16, Equation 60], for input data centered in a hyperbox $[-U_1, U_1] \times \cdots \times [-U_D, U_D]$, the Gaussian kernel is approximated by means of D tensor products of weighted sinusoidal basis functions with frequencies lying on a harmonic scale such that:

$$\boldsymbol{z}_d(x_d)_{i_d} = \frac{1}{\sqrt{U_d}}\, p\left(\frac{\pi i_d}{2U_d}\right) \sin\left(\frac{\pi i_d (x_d + U_d)}{2U_d}\right),\ i_d = 1, 2, \ldots, M_d. \tag{2.6}$$

Here $p(\cdot)$ is the spectral density of the Gaussian kernel with one-dimensional inputs, which is known in closed-form [6, page 83]. This deterministic mapping then approximates the Gaussian kernel function [16, Equation 59] such that

$$k\left(\boldsymbol{x}, \boldsymbol{x}'\right) \approx \left\langle \boldsymbol{z}(\boldsymbol{x}), \boldsymbol{z}(\boldsymbol{x}') \right\rangle = \left\langle \mathrm{ten}\left(\boldsymbol{z}(\boldsymbol{x})\right), \mathrm{ten}\left(\boldsymbol{z}(\boldsymbol{x}')\right) \right\rangle_{\mathrm{F}},$$

and converges uniformly with exponentially decreasing bounds [16, Theorem 8]. We therefore use $\boldsymbol{z}(\cdot)$ instead of $\boldsymbol{\varphi}(\cdot)$ in Eq. (2.3) to obtain

$$f(\boldsymbol{x}, \boldsymbol{w}) = \left\langle \boldsymbol{z}(\boldsymbol{x}), \boldsymbol{w} \right\rangle = \left\langle \mathrm{ten}\left(\boldsymbol{z}(\boldsymbol{x})\right), \mathrm{ten}\left(\boldsymbol{w}\right) \right\rangle_{\mathrm{F}}. \tag{2.7}$$

Learning the exponential number of model parameters $\mathrm{ten}(\boldsymbol{w})$ in Eq. (2.7) under a hinge loss leads to SVM, while considering a squared loss leads to the primal formulation of KRR, which we will consider as exemplifying case from here on:

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \sum_{n=1}^{N} \left(\left\langle \mathrm{ten}\left(\boldsymbol{z}(\boldsymbol{x}_n)\right), \mathrm{ten}\left(\boldsymbol{w}\right) \right\rangle_{\mathrm{F}} - y_n \right)^2 + \lambda ||\mathrm{ten}\left(\boldsymbol{w}\right)||_{\mathrm{F}}^2. \tag{2.8}$$

Since the number of elements M in $\mathrm{ten}(\boldsymbol{w})$ and $\mathrm{ten}(\boldsymbol{z})$ grows exponentially in D,

this primal approach is advantageous compared to the nonparametric *dual* approach only if $\prod_{d=1}^{D} M_d \ll N$, limiting it to low-dimensional inputs. In order to lift this curse of dimensionality, we propose to represent and to learn $\text{ten}(\boldsymbol{w})$ directly as a low-rank tensor decomposition. A low-rank tensor decomposition allows us to exploit redundancies in $\text{ten}(\boldsymbol{w})$ in order to obtain a parsimonious model with a storage complexity that scales linearly in both M and D. The low-rank structure will, as explicitly shown in the experiments, act as a form of regularization by limiting the total number of degrees of freedom of the model.

2.3.2 LOW-RANK TENSOR DECOMPOSITIONS

Tensor Networks (TNs) can be seen as generalizations of the *Singular Value Decomposition* (SVD) of a matrix to tensors [30]. Three common tensor decompositions are the Tucker decomposition [31, 32], the *Tensor Train* (TT) decomposition [33] and the *Canonical Polyadic Decomposition* (CPD) [34, 35], where each of them encompasses different properties of the SVD. In this subsection we briefly discuss these three decompositions and list both their advantages and disadvantages for modeling $\text{ten}(\boldsymbol{w})$ in Eq. (2.8). For a detailed exposition on tensor decompositions we refer the reader to [36] and the references therein. An important property of tensor decompositions is that they can always be written linearly in their components, which implies that applying a block coordinate descent algorithm to solve Eq. (2.8) results in a series of linear least-squares problems.

A rank-R CPD of $\mathscr{W}$ consists of D factor matrices $\boldsymbol{W}_d \in \mathbb{R}^{M_d \times R}$ and $\mathscr{S} \in \mathbb{R}^{R \times R \times \cdots \times R}$ is a superdiagonal core tensor, whose entries typically scale the columns of the factor matrices up to unit norm.

$$\text{vec}(\mathscr{W}) = (\boldsymbol{W}_1 \otimes \boldsymbol{W}_2 \otimes \cdots \otimes \boldsymbol{W}_D)\,\text{vec}(\mathscr{S})\,.$$

Note that the entries of the $\text{vec}(\mathscr{S})$ vector were absorbed in one of the factor matrices when writing the CPD as a sum of R terms. The CPD has been shown, in contrast to matrix factorizations, to be unique under mild conditions [37]. The storage complexity comes mainly from the D factor matrices and is therefore $\mathcal{O}(RMD)$. The Tucker decomposition generalizes the CPD in two ways. First, the Khatri-Rao product is replaced with a Kronecker product. The $\text{vec}(\mathscr{S})$ vector has to grow in length accordingly from R to R^D. Second, the R-dimension of each of the factor matrices is allowed to vary, resulting in a multi-linear rank

$(R_1, R_2, \ldots, R_D)$:

$$\boldsymbol{w} = (\boldsymbol{W}_1 \otimes \boldsymbol{W}_2 \otimes \cdots \otimes \boldsymbol{W}_D)\, \boldsymbol{s}.$$

The Tucker decomposition is inherently non-unique and its storage complexity $\mathcal{O}(R^D)$ is dominated by the $\boldsymbol{s}$ vector, which limits its use to low-dimensional input data. For this reason we do not consider the Tucker decomposition any further in this article.

The TT decomposition consists of D third-order tensors $\mathcal{W}_d \in \mathbb{R}^{R_d \times M \times R_{d+1}}$ such that

$$w_{i_1 i_2 \cdots i_D} = \sum_{r_1=1}^{R_1} \cdots \sum_{r_{D+1}=1}^{R_{D+1}} w_{1 r_1 i_1 r_2} \cdots w_{D r_D i_D r_{D+1}}. \tag{2.9}$$

The auxiliary dimensions $R_1, R_2, \ldots, R_{D+1}$ are called the TT-ranks. In order to ensure that the right-hand side of Eq. (2.9) is a scalar, the boundary condition $R_1 = R_{D+1} = 1$ is enforced. The TT decomposition is, just like the Tucker decomposition, non-unique and its storage complexity $R^2 MD$ is due to the D tensor components $\mathcal{W}^{(d)}$. Considering their storage complexity, both the CPD and TT decomposition are viable candidates to replace $\mathcal{W}$ in Eq. (2.8). The CPD-rank R and TT-ranks $R_2, \ldots, R_D$ are additional hyperparameters, which favors the CPD in practice. For the TT, one could choose $R_2 = R_3 = \cdots = R_D$ to reduce the number of additional hyperparameters but this constraint turns out in practice to lead to suboptimal results. For these reasons we limit the discussion of the learning algorithm to the CPD case.

2.3.3 TENSOR LEARNING WITH FOURIER FEATURES

We now wish to minimize the standard regularized squared loss function as in Eq. (2.8) with the additional constraint that the weight tensor has a rank-R CPD structure:

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \sum_{n=1}^{N} \left(\langle \operatorname{ten}(\boldsymbol{z}(\boldsymbol{x})), \operatorname{ten}(\boldsymbol{w}) \rangle_{\mathrm{F}} - y_n \right)^2 + \lambda ||\operatorname{ten}(\boldsymbol{w})||_{\mathrm{F}}^2, \tag{2.10}$$

$$\text{subject to:} \quad \text{CPD-rank}(\operatorname{ten}(\boldsymbol{w})) = R. \tag{2.11}$$

Note that if R equals the true CPD-rank of the underlying weight tensor then the exact solution of Eq. (2.8) would be obtained. In practice a low-rank solution for $\operatorname{ten}(\boldsymbol{w})$ achieves a sufficiently complex decision boundary that is practically indistinguishable from the full-rank solution, as is demonstrated in the experi-

ments. Imposing the rank-R reduces the number of unknowns from M^D to RMD and allows the application of a block coordinate descent algorithm (also known as alternating linear scheme), which is shown to converge monotonically [38, 39]. Each of the factor matrices $\boldsymbol{W}_d$ is optimized in an iterative fashion while keeping the others fixed. Such a factor matrix update is obtained by solving a linear least-squares problem with MR unknowns. In what follows we derive the linear problem for the d-th factor matrix $\boldsymbol{W}_d$. The data-fitting term $\langle \operatorname{ten}(\boldsymbol{z}(\boldsymbol{x})), \operatorname{ten}(\boldsymbol{w})\rangle_{\mathrm{F}}$ can be rewritten linearly in terms of the unknown factor matrix as

$$\begin{aligned}
\langle \operatorname{ten}(\boldsymbol{z}(\boldsymbol{x})), \operatorname{ten}(\boldsymbol{w})\rangle_{\mathrm{F}} &= \langle \operatorname{ten}(\boldsymbol{z}_1 \otimes \boldsymbol{z}_2 \cdots \otimes \boldsymbol{z}_D), \operatorname{ten}\left(\sum_{r=1}^{R} \boldsymbol{w}_{1r} \otimes \boldsymbol{w}_{2r} \otimes \cdots \otimes \boldsymbol{w}_{Dr}\right)\rangle_{\mathrm{F}} \\
&= \sum_{m_d=1}^{M_d} \sum_{r=1}^{R} \left(z_{d\,m_d} \sum_{i_1=1}^{M_1} w_{1\,m_1,r}\, z_{1\,m_1} \cdots \sum_{m_D=1}^{M} w_{D\,m_D,r}\, z_{D\,m_D} \right) w_{d\,m_d r} \\
&= \left(\boldsymbol{z}_d \otimes \left(\boldsymbol{z}_1{}^{\mathrm{T}} \boldsymbol{W}_1 \odot \cdots \odot \boldsymbol{z}_D{}^{\mathrm{T}} \boldsymbol{W}_D\right)\right)^{\mathrm{T}} \operatorname{vec}(\boldsymbol{W}_d) \\
&= \langle \boldsymbol{g}_d(\boldsymbol{x}), \operatorname{vec}(\boldsymbol{W}_d)\rangle_{\mathrm{F}}. \qquad (2.12)
\end{aligned}$$

where $\boldsymbol{g}_d(\boldsymbol{x}) \coloneqq \boldsymbol{z}_d \otimes \left(\boldsymbol{z}_1{}^{\mathrm{T}} \boldsymbol{W}_1 \odot \cdots \odot \boldsymbol{z}_D{}^{\mathrm{T}} \boldsymbol{W}_D\right)$. Similarly, for the regularization term:

$$\begin{aligned}
||\operatorname{ten}(\boldsymbol{w})||_{\mathrm{F}}^2 &= \langle \operatorname{ten}\left(\sum_{r=1}^{R} \boldsymbol{w}_{1r} \otimes \boldsymbol{w}_{2r} \otimes \cdots \otimes \boldsymbol{w}_{Dr}\right), \operatorname{ten}\left(\sum_{p=1}^{R} \boldsymbol{w}_{1p} \otimes \boldsymbol{w}_{2p} \otimes \cdots \otimes \boldsymbol{w}_{Dp}\right)\rangle_{\mathrm{F}} \\
&= \sum_{m_d=1}^{M_d} \sum_{r=1}^{R} \sum_{p=1}^{R} w_{d\,m_d,r} \left(\sum_{m_1=1}^{M_1} w_{1\,m_1,r}\, w_{1\,m_1,p} \cdots \sum_{m_D=1}^{M_D} w_{D\,m_D,r}\, w_{D\,m_D,p} \right) w_{d\,m_d,p} \\
&= \sum_{m_d=1}^{M_d} \sum_{r=1}^{R} \sum_{p=1}^{R} w_{d\,m_d,r} \left(\boldsymbol{W}_1^{\mathrm{T}} \boldsymbol{W}_1 \odot \cdots \odot \boldsymbol{W}_D^{\mathrm{T}} \boldsymbol{W}_D\right)_{r,p} w_{d\,m_d,p} \\
&= \operatorname{vec}(\boldsymbol{W}_d)^{\mathrm{T}} \left(\boldsymbol{I}_{M_d} \otimes \boldsymbol{H}_d\right) \operatorname{vec}(\boldsymbol{W}_d). \qquad (2.13)
\end{aligned}$$

Here $\boldsymbol{H}_d \coloneqq \left(\boldsymbol{W}_1{}^{\mathrm{T}} \boldsymbol{W}_1 \odot \cdots \odot \boldsymbol{W}_d{}^{\mathrm{T}} \boldsymbol{W}_D\right)$.
Substitution of the expressions Eq. (2.12) and Eq. (2.13) into Eq. (2.10) leads to a linear least-squares problem for $\operatorname{vec}(\boldsymbol{W}_d)$:

$$\underset{\operatorname{vec}(\boldsymbol{W}_d)}{\operatorname{argmin}} \sum_{n=1}^{N} \langle \boldsymbol{g}_d(\boldsymbol{x}_n), \operatorname{vec}(\boldsymbol{W}_d)\rangle_{\mathrm{F}} + \lambda \operatorname{vec}(\boldsymbol{W}_d) \left(\boldsymbol{I}_{M_d} \otimes \boldsymbol{H}_d\right)^{\mathrm{T}} \operatorname{vec}(\boldsymbol{W}_d). \qquad (2.14)$$

Its unique solution can be computed exactly in an efficient manner by solving the normal equations, requiring $NM^2R^2 + M^3R^3$ operations. Since we are solv-

ing a non-convex optimization problem that consists of a series of convex and exactly solvable sub-problems, our algorithm is monotonically decreasing, and, although it is not guaranteed to converge to the global optimum, it is the standard choice when dealing with tensor decompositions [30, 38]. The total computational complexity of the algorithm when $N \gg MR$ is then $\mathcal{O}(NDM^2R^2)$, rendering it suitable for problems which are large in both N and D, provided that R and M are small. The necessary memory is equal to $RMD + 2R^2M^2 + 2RM$ (storing the weight tensor $\mathcal{W}$ in decomposed form, the rank-RM Gram matrix and regularization matrix, the transformed responses and the solution of the linear system), leading to a storage complexity of $\mathcal{O}\left(NR^2M^2\right)$ for $RM \gg D$.

When learning, the selection of λ and of the kernel-related hyperparameters can be carried out by standard methods such as cross-validation. The choice of M and the additional hyperparameter R which we introduce are directly linked with the available computational budget. One should in fact choose M so that the model has access to a sufficiently complex set of basis functions to learn from. In practice we notice that for our choices of kernel function hyperparameters, at most $M = 40$ basis functions per dimension suffice. R can then be fixed accordingly in order to match the computational budget at hand. As we will show in the next section, learning is possible with small values of M and R.

2.4 NUMERICAL EXPERIMENTS

We implemented our *Tensor-Kernel Ridge Regression* (T-KRR) algorithm in Mathworks MATLAB 2021a (Update 1) [40] and tested it on several regression and classification problems. Our implementation can be freely downloaded from https://github.com/fwesel/T-KRR and allows reproduction of all experiments in this section. In our implementation we avoid constructing $\boldsymbol{g}_d(\cdot)$ and $\boldsymbol{H}_d$ from scratch at every iteration by updating their components iteratively. With the exception of the first experiment 2.4.1, we further speedup our implementation by considering only the diagonal of the regularization term $\boldsymbol{H}_d$. All experiments were run on a Dell Inc. Latitude 7410 laptop with 16 GB of RAM and an Intel Core i7-10610U CPU running at 1.80 GHz. In all our experiments the Gaussian kernel $k\left(\boldsymbol{x}, \boldsymbol{x'}\right) = \exp\left(-\|\boldsymbol{x}-\boldsymbol{x'}\|_2^2/2l^2\right)$ was approximated by considering the HGP [16] mapping of Eq. (2.6). In all experiments inputs were scaled to lie in a D-dimensional unit hypercube. When dealing with regression, the responses were standardized around the mean, while when considering binary classification inference was done by looking at the sign of the model response (LS-SVM [7]). One sweep of our T-KRR algorithm is defined as updating factor matrices in the order $1 \rightarrow D$

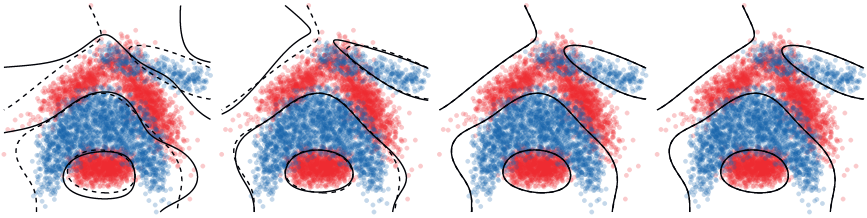
and then back from $D \to 1$. All initial factor matrices were initialized with standard normal numbers and normalized by dividing all entries with their Frobenius norm. For all experiments the number of sweeps of T-KRR algorithm are set to 10. In the following three experiments it is shown how our proposed T-KRR algorithm is stable and recovers the full KRR estimate in case of low number of frequencies M and low rank R, consistently outperforming RFF. Finally, our model exhibits very competitive performance on large-scale problems when compared with other kernel methods.

2.4.1 BANANA CLASSIFICATION

The banana dataset is a two-dimensional classification problem [41] consisting of $N = 5300$ datapoints. Since the data is two-dimensional, we consider $M = 12$ frequencies per dimension, which enables us to compare T-KRR with HGP. Furthermore since for tensors of order two (i.e. matrices) the CPD-rank is the matrix rank, we can deduce that our approach should recover the underlying HGP method with $R = M = 12$. In this example we fix the kernel hyperparameters of our method and of the HGP to $l = 1/2$ and $\lambda = 10^{-5}$ for visualization purposes. In Figure 2.1 we plot the decision boundary of T-KRR (full line) and of the associated HGP (dashed line). We can see that already when $R = 6$ the learned decision boundary is indistinguishable to the one of HGP, meaning that in this example it is possible to obtain a low-rank representation of the model weights. From a computational point of view, we solve a series of linear systems with MR unknowns instead of M^D. However due to the low-dimensionality of the dataset, the computational savings per iteration are very modest, i.e. for $R = 6$ and $M = 12$, we solve per iteration a linear system with 72 unknowns as opposed to the one-time solve of a linear system with 144 unknowns as in case of HGP. As we show in Subsection 2.4.2, the linear computational complexity in D of our algorithm results in ever-increasing performance benefits as the dimensionality of the problem becomes larger, where it becomes impossible to consider a full weight tensor.

2.4.2 MODEL PERFORMANCE WITH BASELINE

We consider five *University of California, Irvine* (UCI) [42] datasets in order to compare the performance of our model with RFF and the GPR/KRR baseline. For each dataset, we consider 90% of the data for training and the remaining 10% for testing. In particular, in all regression datasets, we first obtain an estimate of the lengthscale of the Gaussian kernel l and regularization term λ by log-marginal



(a) $R = 2$ (b) $R = 4$ (c) $R = 6$ (d) $R = M = 12$

Figure 2.1.: Classification boundary of the two-dimensional banana dataset for increasing CPD-ranks. In the last plot the chosen CPD-rank matches the true matrix rank. The dashed line is the HGP, while the full line is T-KRR.

likelihood optimization over the whole training set using the GPLM toolbox [43]. We subsequently train GPR/KRR, RFF and our method with the estimated hyperparameter. In case of classification, we choose the lengthscale l to be the sample mean of the sample standard deviation of our data (which is the default choice in the Matlab Machine Learning Toolbox and scikit-learn [44]) and $\lambda = 10^{-5}$. We repeat this procedure ten times over different random splits and report the sample mean and sample standard deviation of the predictive *Mean Squared Error* (MSE) and the misclassification rate for regression and classification respectively. In order to test the approximation capabilities of our approach, we set M and R such that $MR \ll N$ in order to benefit from computational gains. Furthermore, we note that due to the curse of dimensionality it is not possible to compare our approach with RFF while considering the same total number of basis functions $M = M^D$. Hence we select $M_{\text{RFF}} = MR$ such that the computational complexity of both methods is similar. Table 2.1 shows the MSE for different UCI datasets. We see that our approach comes close to the performance of full KRR and outperforms RFF on all datasets when considering the same number of model parameters. This is because our approach considers implicitly M^D frequencies to learn a low-rank parsimonious model which is fully described by MRD parameters, as opposed to RFF where the number of frequencies is the same as the number of model weights. Similarly to what is reported in [45, Figure 2] we observe lower performance of RFF on the Adult dataset than reported in [11]. Figure 2.2 plots the monotonically decreasing loss function as well as the corresponding misclassification rate while training with T-KRR on the Spambase dataset. After four sweeps the misclassification rate has converged. Since T-KRR is able to obtain similar performance as the KRR baseline on a range of small datasets, we proceed to tackle a

large-scale regression problem.

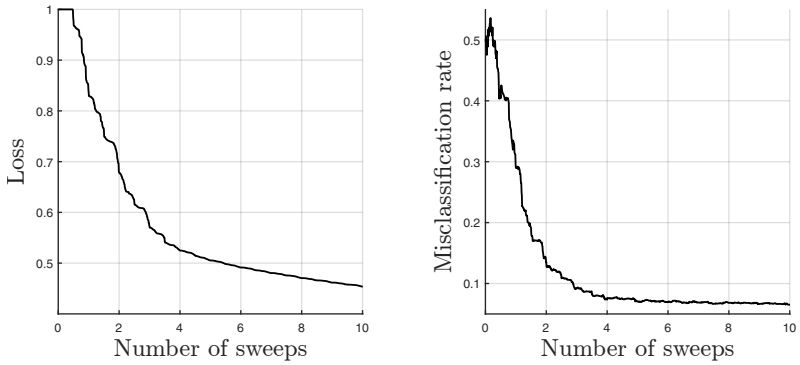


Figure 2.2.: Normalized loss and misclassification rate while training on the Spambase dataset.

Dataset	$N \downarrow$	D	M	R	RFF	T-KRR	KRR
Yacht	308	6	10	25	0.0021 ± 0.0018	0.0009 ± 0.0006	$\mathbf{0.0007} \pm 0.0004$
Energy	768	9	20	10	0.0275 ± 0.0075	0.0200 ± 0.0066	0.0200 ± 0.0087
Airfoil	1503	5	20	10	0.2180 ± 0.0336	0.1679 ± 0.0258	$\mathbf{0.1587} \pm 0.0232$
Spambase	4601	57	40	10	0.3620 ± 0.0188	0.0935 ± 0.0095	$\mathbf{0.0909} \pm 0.0115$
Adult	45222	96	40	10	0.3976 ± 0.0056	$\mathbf{0.1596} \pm 0.0046$	N/A

Table 2.1.: Predictive MSE (regression) and misclassification rate (classification) with one standard deviation for RFF, T-KRR and KRR on different UCI datasets.

2.4.3 LARGE-SCALE EXPERIMENT

In order to highlight the ability of our method to deal with large data we consider the Airline dataset. The Airline dataset [15, 46] is a large-scale regression problem originally considered in [46] which is often used to compare state-of-the-art GPR approximations due to its large size and its non-stationary features. The goal is in fact to predict the delay of a flight given eight features, which are the age of the airplane, route distance, airtime, departure time, arrival time, day of the week, day of the month, and month. We follow the same exact preprocessing steps as in the experiments in [15], [16] and [47], which consider subsets of data

of size $N = 10000, 100000, 1000000, 5929413$, each chosen uniformly at random. Training the model is then accomplished with $2/3N$ datapoints, with the remaining portion reserved for testing. The entire procedure is then repeated 10 times with random initialization in order to obtain a sample mean and sample standard deviation estimate of the MSE. Following exactly the approach in HGP [16], we consider $M = 40$ basis functions per dimension, with the crucial difference that T-KRR approximates a standard *product* Gaussian kernel, as opposed to an *additive* model. As was the case in the previous section for classification, we select the lengthscale of the kernel as the mean of the standard deviations of the eight features and choose $\lambda_N = 100/N$ for the different splits, where the dependency on N is suggested by standard learning theory. We then train four distinct models for $R = 5, 10, 15, 20$. Table 2.2 compares T-KRR with the best performing models found in the literature, which all (except for GPR) rely on low-rank approximation of the kernel function. The T-KRR method is able to recover the baseline GPR performance already with $R = 5$ and remarkably outperform all other approaches although we choose the hyperparameters naively and consider equal lengthscales l for all dimensions. Interestingly, a higher choice of R does result in better performance in all cases except for $N = 10000$, where model performance very much depends on the random split (note that we consider different splits for each experiment). The *Stochastic Variational Inference Gaussian Process* (SVIGP) [46] achieves comparable results to T-KRR, indicating that a performance gain is possible from product kernels. The difficulty with SVIGP is however that the kernel function is interpolated *locally* at M nodes in a data-dependent fashion, requiring an increasing amount of interpolation nodes to cover the whole domain to allow for good generalization. In contrast, T-KRR considers an exponentially large amount of basis functions in the frequency domain, and learns an efficient representation of the model weights. In light of the performance of the additive HGP and additive VFF models, we expect similar performance when considering other feature maps which induce stationary kernels. When considering the whole dataset, training our model with $R = 5$ takes 1565 ± 1 seconds on a laptop, while for $R = 20$ it takes 7141 ± 245 seconds. Reported training times of SVIGP indicate 18360 ± 360 seconds [15] on a cluster. *Variational Inducing Spherical Harmonics* (VISH) scales very favorably in terms of time, but since it is a data-independent method, we expect the predictive performance to be inferior for the same number of parameters. Since the computational complexity of our algorithm is dominated by matrix-matrix multiplications, we expect significant speedups when relying on GPU computations.

N	10000	100000	1000000	5929413
T-KRR ($R = 5$)	0.91 ± 0.10	0.82 ± 0.03	0.80 ± 0.02	0.800 ± 0.008
T-KRR ($R = 10$)	0.89 ± 0.05	0.80 ± 0.05	0.79 ± 0.02	0.785 ± 0.009
T-KRR ($R = 15$)	0.90 ± 0.07	0.80 ± 0.04	0.78 ± 0.02	0.773 ± 0.007
T-KRR ($R = 20$)	0.97 ± 0.15	$\mathbf{0.78} \pm 0.04$	$\mathbf{0.77} \pm 0.01$	$\mathbf{0.763} \pm 0.007$
Additive-HGP [16]	0.97 ± 0.14	0.80 ± 0.06	0.83 ± 0.02	0.827 ± 0.005
Additive-VFF [15]	0.89 ± 0.15	0.82 ± 0.05	0.83 ± 0.01	0.827 ± 0.004
SVIGP [46]	0.89 ± 0.16	0.79 ± 0.05	0.79 ± 0.01	0.791 ± 0.005
VISH [47]	0.90 ± 0.16	0.81 ± 0.05	0.83 ± 0.03	0.834 ± 0.055
GPR [6]	0.89 ± 0.16	N/A	N/A	N/A
Additive-GPR [48]	0.89 ± 0.16	N/A	N/A	N/A

Table 2.2.: Predictive MSE with one standard deviation for T-KRR.

2.5 CONCLUSION

In this work a framework to perform large-scale supervised learning with tensor decompositions which leverages the tensor product structure of deterministic Fourier features was introduced. Concretely, a monotonically decreasing learning algorithm with linear complexity in both sample size and dimensionality was derived. This algorithm leverages the efficient format of the CPD in combination with exponentially fast converging FFs which allow to implicitly approximate stationary product kernels up to machine precision. Numerical experiments show how the performance of the baseline KRR is recovered with a very limited number of parameters. The proposed method can handle problems which are both large in the number of samples as well as in their dimensionality, effectively enabling large-scale supervised learning with stationary product kernels. The biggest limitation of the current approach is that it is does not allow for uncertainty quantification, which motivates further work in that direction.

REFERENCES

[1] F. Wesel and K. Batselier. “Large-Scale Learning with Fourier Features and Tensor Decompositions”. In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 17543–17554.

[2] B. Hammer and K. Gersmann. “A Note on the Universal Approximation Capability of Support Vector Machines”. In: *Neural Processing Letters* 17.1 (Feb. 2003), pp. 43–53.

[3] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. “Deep Neural Networks as Gaussian Processes”. In: *International Conference on Learning Representations*. Feb. 2018.

[4] R. Novak, L. Xiao, Y. Bahri, J. Lee, G. Yang, J. Hron, D. A. Abolafia, J. Pennington, and J. Sohl-dickstein. “Bayesian Deep Convolutional Networks with Many Channels Are Gaussian Processes”. In: *International Conference on Learning Representations*. Sept. 2018.

[5] A. Garriga-Alonso, C. E. Rasmussen, and L. Aitchison. “Deep Convolutional Networks as Shallow Gaussian Processes”. In: *International Conference on Learning Representations*. Sept. 2018.

[6] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press, 2006.

[7] J. Suykens and J. Vandewalle. “Least Squares Support Vector Machine Classifiers”. In: *Neural Processing Letters* 9.3 (June 1999), pp. 293–300.

[8] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Nov. 2002.

[9] C. Williams and M. Seeger. “Using the Nyström Method to Speed Up Kernel Machines”. In: *Advances in Neural Information Processing Systems 13*. MIT Press, 2001, pp. 682–688.

[10] P. Drineas and M. W. Mahoney. “On the Nystrom Method for Approximating a Gram Matrix for Improved Kernel-Based Learning”. In: *Journal of Machine Learning Research* 6.72 (2005), pp. 2153–2175.

[11] A. Rahimi and B. Recht. “Random Features for Large-Scale Kernel Machines”. In: *Proceedings of the 20th International Conference on Neural Information Processing Systems*. Curran Associates Inc., Dec. 2007, pp. 1177–1184.

[12] T. Dao, C. D. Sa, and C. Ré. “Gaussian Quadrature for Kernel Features”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Red Hook, NY, USA: Curran Associates Inc., Dec. 2017, pp. 6109–6119.

[13] M. Mutný and A. Krause. “Efficient High Dimensional Bayesian Optimization with Additivity and Quadrature Fourier Features”. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS’18. Red Hook, NY, USA: Curran Associates Inc., Dec. 2018, pp. 9019–9030.

[14] P. F. Shustin and H. Avron. “Gauss-Legendre Features for Gaussian Process Regression”. In: *arXiv:2101.01137 [cs, math]* (Jan. 2021). arXiv: 2101.01137 [cs, math].

[15] J. Hensman, N. Durrande, and A. Solin. “Variational Fourier Features for Gaussian Processes”. In: *The Journal of Machine Learning Research* 18.1 (Jan. 2017), pp. 5537–5588.

[16] A. Solin and S. Särkkä. “Hilbert Space Methods for Reduced-Rank Gaussian Process Regression”. In: *Statistics and Computing* 30.2 (Mar. 2020), pp. 419–446.

[17] W. Rudin. *Fourier Analysis on Groups*. Interscience Tracts in Pure and Applied Mathematics 12. New York: Interscience Publishers, 1962. Chap. 285 pages 24 cm.

[18] F. Liu, X. Huang, Y. Chen, and J. A. K. Suykens. “Random Features for Kernel Approximation: A Survey on Algorithms, Theory, and Beyond”. In: *arXiv:2004.11154 [cs, stat]* (July 2020). arXiv: 2004.11154 [cs, stat].

[19] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos. “Tensor Decomposition for Signal Processing and Machine Learning”. In: *IEEE Transactions on Signal Processing* 65.13 (July 2017), pp. 3551–3582.

[20] E. Gilboa, Y. Saatçi, and J. P. Cunningham. “Scaling Multidimensional Inference for Structured Gaussian Processes”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.2 (Feb. 2015), pp. 424–436.

[21] P. Izmailov, A. Novikov, and D. Kropotov. “Scalable Gaussian Processes with Billions of Inducing Inputs via Tensor Train Decomposition”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR, Mar. 2018, pp. 726–735.

[22] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov. “Tensorizing Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc., 2015.

[23] Y. Yang, D. Krompass, and V. Tresp. “Tensor-Train Recurrent Neural Networks for Video Classification”. In: *International Conference on Machine Learning*. PMLR, July 2017, pp. 3891–3900.

[24] A. Tjandra, S. Sakti, and S. Nakamura. “Compressing Recurrent Neural Network with Tensor Train”. In: *2017 International Joint Conference on Neural Networks (IJCNN)*. May 2017, pp. 4451–4458.

[25] S. Wahls, V. Koivunen, H. V. Poor, and M. Verhaegen. “Learning Multidimensional Fourier Series with Tensor Trains”. In: *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. Dec. 2014, pp. 394–398.

[26] E. M. Stoudenmire and D. J. Schwab. “Supervised Learning with Tensor Networks”. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. Curran Associates Inc., Dec. 2016, pp. 4806–4814.

[27] A. Novikov, I. Oseledets, and M. Trofimov. “Exponential Machines”. In: *Bulletin of the Polish Academy of Sciences: Technical Sciences; 2018; 66; No 6 (Special Section on Deep Learning: Theory and Practice); 789-797* (2018).

[28] Z. Chen, K. Batselier, J. A. K. Suykens, and N. Wong. “Parallelized Tensor Train Learning of Polynomial Classifiers”. In: *IEEE Transactions on Neural Networks and Learning Systems* 29.10 (Oct. 2018), pp. 4621–4632.

[29] C. Cortes and V. Vapnik. “Support-Vector Networks”. In: *Machine Learning* 20.3 (Sept. 1995), pp. 273–297.

[30] T. G. Kolda and B. W. Bader. “Tensor Decompositions and Applications”. In: *SIAM Review* 51.3 (Aug. 2009), pp. 455–500.

[31] L. De Lathauwer, B. De Moor, and J. Vandewalle. “A Multilinear Singular Value Decomposition”. In: *SIAM Journal on Matrix Analysis and Applications* 21.4 (Jan. 2000), pp. 1253–1278.

[32] L. R. Tucker. “Some Mathematical Notes on Three-Mode Factor Analysis”. In: *Psychometrika* 31.3 (Sept. 1966), pp. 279–311.

[33] I. V. Oseledets. “Tensor-Train Decomposition”. In: *SIAM Journal on Scientific Computing* 33.5 (Jan. 2011), pp. 2295–2317.

[34] F. L. Hitchcock. “The Expression of a Tensor or a Polyadic as a Sum of Products”. In: *Journal of Mathematics and Physics* 6.1-4 (1927), pp. 164–189.

[35] J. B. Kruskal. “Three-Way Arrays: Rank and Uniqueness of Trilinear Decompositions, with Application to Arithmetic Complexity and Statistics”. In: *Linear Algebra and its Applications* 18.2 (Jan. 1977), pp. 95–138.

[36] 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”. In: *Foundations and Trends® in Machine Learning* 9.4-5 (2016), pp. 249–429.

[37] N. D. Sidiropoulos and R. Bro. “On the Uniqueness of Multilinear Decomposition of N-Way Arrays”. In: *Journal of Chemometrics* 14.3 (2000), pp. 229–239.

[38] P. Comon, X. Luciani, and A. L. F. de Almeida. “Tensor Decompositions, Alternating Least Squares and Other Tales”. In: *Journal of Chemometrics* 23.7-8 (July 2009), pp. 393–405.

[39] S. Holtz, T. Rohwedder, and R. Schneider. “The Alternating Linear Scheme for Tensor Optimization in the Tensor Train Format”. In: *SIAM Journal on Scientific Computing* 34.2 (Jan. 2012), A683–A713.

[40] Matlab. *9.10.0.1649659 (R2021a) Update 1*. Natick, Massachussetts: The MathWorks Inc., 2021.

[41] J. Hensman, A. Matthews, and Z. Ghahramani. “Scalable Variational Gaussian Process Classification”. In: *Artificial Intelligence and Statistics*. PMLR, Feb. 2015, pp. 351–360.

[42] D. Dua and C. Graff. *UCI Machine Learning Repository*. 2017.

[43] C. E. Rasmussen and H. Nickisch. “Gaussian Processes for Machine Learning (GPML) Toolbox”. In: *Journal of Machine Learning Research* 11.100 (2010), pp. 3011–3015.

[44] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay. “Scikit-Learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12.85 (2011), pp. 2825–2830.

[45] T. Yang, Y.-f. Li, M. Mahdavi, R. Jin, and Z.-H. Zhou. “Nyström Method vs Random Fourier Features: A Theoretical and Empirical Comparison”. In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., 2012.

[46] J. Hensman, N. Fusi, and N. D. Lawrence. “Gaussian Processes for Big Data”. In: *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*. UAI’13. AUAI Press, Aug. 2013, pp. 282–290.

[47] V. Dutordoir, N. Durrande, and J. Hensman. “Sparse Gaussian Processes with Spherical Harmonic Features”. In: *International Conference on Machine Learning*. PMLR, Nov. 2020, pp. 2793–2802.

[48] D. K. Duvenaud, H. Nickisch, and C. Rasmussen. “Additive Gaussian Processes”. In: *Advances in Neural Information Processing Systems* 24 (2011), pp. 226–234.

3

TENSOR-BASED KERNEL MACHINES WITH STRUCTURED INDUCING POINTS FOR LARGE AND HIGH-DIMENSIONAL DATA

Kernel machines are one of the most studied family of methods in machine learning. In the exact setting, training requires to instantiate the kernel matrix, thereby prohibiting their application to large-sampled data. One popular kernel approximation strategy which allows to tackle large-sampled data consists in interpolating product kernels on a set of grid-structured inducing points. However, since the number of model parameters increases exponentially with the dimensionality of the data, these methods are limited to small-dimensional datasets. In this work we lift this limitation entirely by placing inducing points on a grid and constraining the primal weights to be a low-rank Canonical Polyadic Decomposition *(*CPD*). We derive a block coordinate descent algorithm that efficiently exploits grid-structured inducing points. The computational complexity of the algorithm scales linearly both in the number of samples and in the dimensionality of the data for any product kernel. We demonstrate the performance of our algorithm on large-scale and high-dimensional data, achieving state-of-the art results on a laptop computer. Our results show that grid-structured approaches can work in higher-dimensional problems.*

This chapter has been published as:
F. Wesel and K. Batselier. "Tensor-Based Kernel Machines with Structured Inducing Points for Large and High-Dimensional Data". In: *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2023, pp. 8308–8320

3.1 INTRODUCTION

Kernel machines, such as *Support Vector Machines* (SVMs) [2] and *Gaussian Processes* (GPs) [3] are a family of machine learning methods that handle inference of nonlinear functions by lifting the data into a high and possibly infinite-dimensional feature space and performing linear inference therein. Because of their elegant formulation, their connection with reproducing kernel Hilbert spaces and the guarantees which stem from their convex optimization setting, they have become one of the most widely studied machine learning paradigms. Furthermore, kernel machines are known for their connections with neural networks [4, 5, 6] and for the fact that they can be universal function approximators for a suitable choice of kernel [7].

The main limitation of kernel machines is that training involves instantiating the kernel matrix which encodes the similarities between all mapped data. This results in a cost of at least $\mathcal{O}(N^2)$, limiting their applicability to small datasets. To obviate this problem, a number of low-rank approaches based on random features [8, 9] and inducing points [10, 11, 12, 13, 14, 15, 16] have been developed. Broadly speaking, these methods seek a rank-$M \ll N$ approximation of the kernel function, which enables faster inference at cost of $\mathcal{O}(NM^2)$. However, this scaling forces the modeler to trade-off accuracy of the kernel approximation with the ability to process large-scale data.

A popular family of approaches that is based on the Nyström approximation of kernel functions is *Structured Kernel Interpolation* (SKI) [17, 18, 19, 20, 21]. SKI-methods do not sacrifice accuracy for fast inference since they interpolate the kernel function globally on a regularly spaced grid in order to exploit the ensuing structure for computational gains. However, since the number of interpolation points are placed on a regular grid, and therefore the number of model parameters increases exponentially with the dimensionality of the data, these approaches are limited to small-dimensional datasets.

Recently, the CPD [22], a tool from multi-linear algebra, has been applied in the context of kernel machines to bypass the exponential growth of model parameters affecting Fourier features-based approximations of stationary kernels [23] by constraining the model weights to be a CPD of low rank. This low-rank constraint allows to learn a model with a *linear* number of model parameters in the dimensionality D, but requires knowledge of the spectral representation (Fourier transform) of the kernel of choice which in general can be unknown or non-analytical, requiring then further approximations. In this paper we develop a CPD-based approach to learn from *any* product kernel which allows grid-structured inducing points methods to scale to both large-sampled and high-

dimensional data. We exploit the tensor-product structure of grid-structured inducing point by constraining the model weights to be a CPD of low-rank. Under this constraint, we derive a block coordinate descent algorithm that allows for the efficient training of kernel machines. Our algorithm has a computational complexity of $\mathcal{O}(NDM^{\frac{2}{D}}R^2)$ and storage complexity of $\mathcal{O}(NR)$, where R is the rank of the tensor-decomposition which controls the time versus accuracy trade-off. We show through experiments that competitive results in terms of performance can be obtained on a laptop computer for data that is both large in sample size as well as in dimensionality.

In the context of supervised learning, the goal is to estimate a function $f(\cdot) : \mathbb{R}^D \to \mathbb{R}$ given only a finite set of i.i.d. input-output pairs $\left(\boldsymbol{x}_n, y_n\right)_{n=1}^N$ s.t. $\boldsymbol{x}_n \in \mathbb{R}^D$, $y \in \mathbb{R}$, $\forall n \in \{1, \ldots, N\}$ generated by some probability measure. After defining a measure of loss $\ell(\cdot,\cdot) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$, this can be accomplished by minimizing the (regularized) empirical risk

$$\sum_{n=1}^{N} \ell\left(f\left(\boldsymbol{x}_n, \boldsymbol{w}\right), y_n\right) + r(\boldsymbol{w}). \tag{3.1}$$

3.1.1 KERNEL MACHINES

Kernel machines model f as linear in the mapped data, i.e.

$$f\left(\boldsymbol{x}, \boldsymbol{w}\right) = \left\langle \boldsymbol{\varphi}\left(\boldsymbol{x}\right), \boldsymbol{w} \right\rangle. \tag{3.2}$$

Here $\boldsymbol{\varphi}(\cdot) : \mathbb{R}^D \to \mathbb{R}^M$ is the *feature map* which lifts the data in a high (and possibly infinite-dimensional) reproducing kernel Hilbert space where linear inference is possible, and $\boldsymbol{w} \in \mathbb{R}^M$ are the model weights. In practice, this explicit mapping can be avoided by considering a kernel function $k(\cdot,\cdot) : \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}$ such that $k\left(\boldsymbol{x}, \boldsymbol{x'}\right) = \left\langle \boldsymbol{\varphi}\left(\boldsymbol{x}\right), \boldsymbol{\varphi}\left(\boldsymbol{x'}\right) \right\rangle$. By the representer theorem [24] we have in fact that:

$$f\left(\boldsymbol{x}, \boldsymbol{\alpha}\right) = \sum_{n=1}^{N} \alpha_n k\left(\boldsymbol{x}, \boldsymbol{x}_n\right), \tag{3.3}$$

which implies that we only need to estimate multipliers $\boldsymbol{\alpha} \in \mathbb{R}^N$. Depending on the choice of loss function different kernel machines arise, for instance hinge loss leads to support vector machines, squared loss to *Kernel Ridge Regression* (KRR), which is the same estimator as the GP regression posterior mean. In case of mean squared error, Eq. (3.1) can be minimized exactly from the corresponding dual

optimization problem:

$$(\boldsymbol{K_{XX}} + \lambda \boldsymbol{I})\, \boldsymbol{\alpha} = \boldsymbol{y}. \tag{3.4}$$

In practice, since the kernel evaluations between all points need to be computed, the computational cost of training in the dual is at least $\mathcal{O}(N^2)$, limiting its usefulness to small-sampled data.

Structured Data One way to enable exact inference on large-scale data is to exploit existing structure in the data. A particular fortunate case arises when the data lies on a Cartesian grid $\boldsymbol{x}_1 \times \boldsymbol{x}_2 \times \cdots \times \boldsymbol{x}_D$ where each $\boldsymbol{x}_d \in \mathbb{R}^{N_d}$ such that $N = \prod_{d=1}^{D} N_d$, and when considering product kernels of the form:

$$k\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \prod_{d=1}^{D} k_d\left(x_d, x'_d\right), \tag{3.5}$$

where $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^D$. In this case the kernel matrix $\boldsymbol{K_{XX}}$ is the Kronecker product of small matrices $\boldsymbol{K_{x_d x_d}} \in \mathbb{R}^{N_d \times N_d}$ [25, Equation 5.7]:

$$\boldsymbol{K_{XX}} = \boldsymbol{K_{x_1 x_1}} \otimes \boldsymbol{K_{x_2 x_2}} \otimes \cdots \otimes \boldsymbol{K_{x_D x_D}}. \tag{3.6}$$

Storing the full kernel matrix $\boldsymbol{K_{XX}} \in \mathbb{R}^{N \times N}$ can then be avoided by storing smaller kernel matrices $\boldsymbol{K_{x_1 x_1}}, \ldots, \boldsymbol{K_{x_D x_D}}$ without ever computing the tensor-products. Exact training can then be accomplished with $\mathcal{O}(DN^{1+\frac{1}{D}})$ operations by exploiting the properties of the Kronecker product [25, 26].

Unstructured Data One way to enable faster inference to unstructured data is to consider the Nyström method [10, 27], whose key idea is to approximate the spectrum of the full kernel matrix $\boldsymbol{K_{XX}}$ by means of a restricted number of kernel evaluations at a subset of $M \ll N$ inducing points denoted by $\boldsymbol{M}$, traditionally sampled at random [10] from the data $\boldsymbol{X}$, defining hence the Nyström approximation [10, Equations 8-9]:

$$\boldsymbol{K_{XX}} \approx \boldsymbol{K_{XM}} \boldsymbol{L}^{-\mathrm{T}} \boldsymbol{L}^{-1} \boldsymbol{K_{MX}} =: \boldsymbol{K}_{\text{Nyström}}, \tag{3.7}$$

where $\boldsymbol{L}$ is such that $\boldsymbol{K_{MM}} = \boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$. Embedding Eq. (3.7) in Eq. (3.1) under the assumption of squared loss gives rise to a linear least-squares problem which can be solved from the normal equations:

$$\left(\boldsymbol{L}^{-1} \boldsymbol{K_{MX}} \boldsymbol{K_{XM}} \boldsymbol{L}^{-\mathrm{T}} + \lambda \boldsymbol{I}\right) \boldsymbol{w} = \boldsymbol{L}^{-1} \boldsymbol{K_{MX}} \boldsymbol{y}, \tag{3.8}$$

This formulation enables training at the computational cost of $\mathcal{O}(NM^2 + M^3)$ and storage cost of $\mathcal{O}(M^2)$. As argued previously however, these complexities force to chose between the accuracy of the approximation and the ability to process large-scale data, as $M \ll N$ for any computational gains. When one considers stationary product kernels, one naive approach would be to locate a large number of inducing points $M \gg N$ on a Cartesian grid and to exploit the ensuing Toeplitz (one-dimensional inputs) and Kronecker (higher-dimensional inputs) structures for computational gains, as in Eq. (3.6). Since this alone only alleviates the complexity associated with $\boldsymbol{K_{MM}}$, plenty of research has focused on approximating $\boldsymbol{K_{XM}}$ which accounts for the dominant $\mathcal{O}(NM^2)$ term. One of these methods is SKI [17]. In SKI, the cross-covariance matrix $\boldsymbol{K_{XM}}$ is approximated by local interpolation, i.e. $\boldsymbol{K_{XM}} \approx \boldsymbol{P}\boldsymbol{K_{MM}}$, where $\boldsymbol{P}$ is a sparse interpolation matrix with 2^D non-zero elements per row (in case of linear interpolation), giving rise to the SKI kernel

$$\boldsymbol{K_{XX}} \approx \boldsymbol{P}\boldsymbol{K_{MM}}\boldsymbol{P}^{\mathrm{T}} =: \boldsymbol{K}_{\mathrm{SKI}}. \tag{3.9}$$

When considering a stationary product kernel $\boldsymbol{K_{MM}}$ has a Toeplitz structure (one-dimension) or a Kronecker product structure of Toeplitz matrices (higher-dimensions). SKI takes advantage of these structures by approximately solving $(\boldsymbol{K}_{\mathrm{SKI}} + \lambda \boldsymbol{I})^{-1}\,\boldsymbol{y}$ using Krylov subspace methods which rely on matrix-vector products. Since $\boldsymbol{P}$ is sparse and $\boldsymbol{K_{MM}}$ is structured, each iteration of SKI costs only $\mathcal{O}(N + M\log M)$ operations and $\mathcal{O}(NM)$ memory. However, since M scales exponentially in D, SKI is limited to sets of data of small dimensionality $D < 5$ [17]. In order to mitigate this exponential dependency in D, Gardner *et al.* [19] approximate the kernel matrix of a stationary product kernel as the Hadamard product of rank-R SKI kernel matrices in order to perform fast matrix-vector products in a divide-and-conquer fashion. Although this approach overcomes the curse of dimensionality, it requires the storage of R copies (typically 30) of the dataset limiting its applicability to data of moderate dimensionality.

Recent extensions and improvement of the SKI framework are the handling of online data [20], its reformulation as a Bayesian linear regression problem [21] and the use of a permutohedral lattice instead of a Cartesian grid [28]. This latter approach reduces the number of neighboring points from 2^D to $D+1$, alleviating the curse of dimensionality by allowing training at $\mathcal{O}(D^2(N+M))$. However, this latter approach is most effective only for $D \leq 20$ [28] due to the quadratic scaling in D of the computational complexity and the decreasing accuracy of the kernel approximation as D increases.

3.1.2 TENSOR DECOMPOSITIONS

The most common tensor decompositions are the CPD [22, 29], the Tucker decomposition [30] and the *Tensor Train* (TT) decomposition [31]. A rank-R CPD decomposes a tensor $\text{ten}(\boldsymbol{w}) \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_D}$ such that

$$\boldsymbol{w} = (\boldsymbol{W}_1 \otimes \boldsymbol{W}_2 \otimes \cdots \otimes \boldsymbol{W}_D)\,\text{vec}(\mathscr{S})\,. \tag{3.10}$$

Here $\boldsymbol{W}_d \in \mathbb{R}^{M_d \times R}$ are the factor matrices and $\mathscr{S} \in \mathbb{R}^{R \times R \times \cdots \times R}$ is a superdiagonal core tensor, whose entries typically scale the columns of the factor matrices up to unit norm. The rank R is defined as the smallest R such that Eq. (3.10) holds exactly [22, 29].

Storing $\text{ten}(\boldsymbol{w})$ in decomposed form requires then only to store D factor matrices, requiring $R\sum_{d=1}^{D} M_d$ memory units as opposed to $M = \prod_{d=1}^{D} M_d$. Because of this compression, tensor decompositions have been used to reduce the number of model parameters in deep learning models by tensorizing and decomposing weights [32, 33, 34, 35], or compressing filters which have tensorial structure by definition, e.g. convolutions [36, 37, 38].

Tensor decompositions have also been used to reduce the exponential number of parameters which arise when learning from tensor-product feature maps by constraining the weight tensor to be a low-rank tensor decomposition. So far these models have considered trigonometric [39], polynomial [40, 41] and Fourier feature maps [23], where the latter are used to induce stationary product kernels. Furthermore, in the context of GPs, tensor decompositions have been used to reduce the complexity of multi-output GPs [42] and in the context of stochastic variational GPs to compress the mean of the variational posterior distribution [43]. However the former method is not designed to handle data which is larger in the number of samples, as it scales with $\mathcal{O}\left(DN^2\right)$, while the latter does not work for $D > 10$, as it becomes unpractical to store the interpolation grid, forcing to train on $D \leq 10$-dimensional embeddings of the data.

In the following section, building on the works of [17] and [23], we derive in the context of classical kernel machines a block-coordinate descent algorithm whose computational complexity at training scales with $\mathcal{O}(NDM^{\frac{2}{D}}R^2)$ requiring $\mathcal{O}(NR)$ memory, allowing to tackle large-scale and large dimensional problems as demonstrated experimentally in Section 3.3.

3.2 GRID-STRUCTURED KERNEL MACHINES

In our approach we consider product kernels (Eq. (3.5)) and Nyström inducing points (Eq. (3.7)), which inspired by Wilson and Nickisch [17] we place on a Cartesian grid $\boldsymbol{m}_1 \times \boldsymbol{m}_2 \times \cdots \times \boldsymbol{m}_D$, where each $\boldsymbol{m}_d \in \mathbb{R}^{M_d}$ and $M = \prod_{d=1}^{D} M_d$. This approximation recovers the underlying kernel as $M \to \infty$ [44, Theorem 1]. We have already seen in Eq. (3.6) how $\boldsymbol{K_{MM}}$ can be stored and manipulated efficiently by considering its Kronecker-product structure by indexing it as a tensor. This allows for the efficient computation of its Cholesky factor $\boldsymbol{L}$ [25, Theorem 5.2] from

$$\begin{aligned} &\boldsymbol{K}_{\boldsymbol{m}_1\boldsymbol{m}_1} \otimes \boldsymbol{K}_{\boldsymbol{m}_2\boldsymbol{m}2} \otimes \cdots \otimes \boldsymbol{K}_{\boldsymbol{m}_D\boldsymbol{m}_D} \\ =&\boldsymbol{L}_1\boldsymbol{L}_1{}^{\mathrm{T}} \otimes \boldsymbol{L}_2\boldsymbol{L}_2{}^{\mathrm{T}} \otimes \cdots \otimes \boldsymbol{L}_D\boldsymbol{L}_D{}^{\mathrm{T}} \\ =&\underbrace{(\boldsymbol{L}_1 \otimes \cdots \otimes \boldsymbol{L}_D)}_{\boldsymbol{L}} \underbrace{(\boldsymbol{L}_1 \otimes \cdots \otimes \boldsymbol{L}_D)^{\mathrm{T}}}_{\boldsymbol{L}^{\mathrm{T}}}, \end{aligned} \tag{3.11}$$

as it inherits the Kronecker-product structure. Similarly, each row $\boldsymbol{k}_{\boldsymbol{xM}}$ of $\boldsymbol{K_{XM}}$ has a tensor-product structure:

$$\boldsymbol{k}_{\boldsymbol{xM}} = \boldsymbol{k}_{x_1\boldsymbol{m}_1} \otimes \boldsymbol{k}_{x_2\boldsymbol{m}_2} \otimes \cdots \otimes \boldsymbol{k}_{x_D\boldsymbol{m}_D},$$

derivations can be found in the supplementary material. By the mixed-product property of the Kronecker-product [25, Equation 5.10] we have that:

$$\boldsymbol{k}_{\boldsymbol{xM}}\boldsymbol{L}^{-\mathrm{T}} = \boldsymbol{k}_{x_1\boldsymbol{m}_1}\boldsymbol{L}_1{}^{-\mathrm{T}} \otimes \cdots \otimes \boldsymbol{k}_{x_D\boldsymbol{m}_D}\boldsymbol{L}_D{}^{-\mathrm{T}}.$$

The computational benefits associated with this tensor-product structure can however not be exploited without further assumptions, as model evaluations $\langle \boldsymbol{k}_{\boldsymbol{xM}}\boldsymbol{L}^{-\mathrm{T}}, \boldsymbol{w} \rangle$ will require in fact still $\mathcal{O}(M)$ computations and the 'unpacking' of the tensor-product structure. In the next paragraph, we will show how one can leverage fully the tensorial structure of $\boldsymbol{k}_{\boldsymbol{xM}}\boldsymbol{L}^{-\mathrm{T}}$ by assuming that the model weights are a rank-R CPD tensor. This will allow to consider both large and high-dimensional datasets with grid-structured inducing points, by effectively reducing the number of model parameters from M to $RDM^{\frac{1}{D}}$.

We now wish to minimize the empirical risk Eq. (3.1) under a convex loss, with the additional constraint that the weight tensor ten$(\boldsymbol{w})$ has a rank-R CPD struc-

ture:

$$\min_{\boldsymbol{w}} \sum_{n=1}^{N} \ell\left(\langle \text{ten}\left(\boldsymbol{k}_{\boldsymbol{x}_n \boldsymbol{M}} \boldsymbol{L}^{-\mathrm{T}}\right), \text{ten}\left(\boldsymbol{w}\right)\rangle_{\mathrm{F}}, y_n\right) + \lambda ||\text{ten}\left(\boldsymbol{w}\right)||_{\mathrm{F}}^2, \tag{3.12}$$

$$\text{subject to: CPD-rank}\left(\text{ten}\left(\boldsymbol{w}\right)\right) = R, \tag{3.13}$$

where if R is chosen to be the true CPD rank of $\text{ten}\left(\boldsymbol{w}\right)$, the solution of Eq. (3.1) associated with Eq. (3.7) is recovered. In this case, $\text{ten}\left(\boldsymbol{w}\right)$ is furthermore also unique under mild conditions [45]. As we will see, this constraint enables to fully exploit the rank-1 CPD structure of $\boldsymbol{k}_{\boldsymbol{x}\boldsymbol{M}} \boldsymbol{L}^{-\mathrm{T}}$ by allowing to optimize one CPD factor matrix $\boldsymbol{W}_d$ of $\text{ten}\left(\boldsymbol{w}\right)$ at a time, enabling to tackle large-sampled and large-dimensional datasets with modest hardware. This is accomplished by exploiting the *multilinearity* of tensor decomposition which allows to express the empirical risk as a *linear* function of the d-th factor matrix $\boldsymbol{W}_d$. Minimizing the risk successively for each factor matrix yields a well-known block coordinate descent [22, 46, 47] algorithm for which each subproblem is convex and exhibits local linear convergence [48, Theorem 3.3]. Similar properties hold when constraining $\text{ten}\left(\boldsymbol{w}\right)$ to be a low-rank tensor train decomposition or Tucker decomposition. However, the number of elements in the Tucker decomposition scales exponentially in D, while the tensor train decomposition models explicitly the correlations between features, yielding for the same rank, different models depending on the ordering of the features. In contrast to other decompositions, our CPD-based approach enables to reduce the costs of storage by clever in-place updates. Furthermore, recent theoretical advances in the domain of tensor decomposition show that the *Vapnik–Chervonenkis* (VC) dimension and pseudodimension of models of the form of Eq. (3.2), where $\text{ten}\left(\boldsymbol{w}\right)$ is a rank-R tensor decomposition, are *independent* of the choice of decomposition of $\text{ten}\left(\boldsymbol{w}\right)$ and instead upper bounded by the number of parameters [49, Theorem 7], further motivating the choice of modeling $\text{ten}\left(\boldsymbol{w}\right)$ as a CPD. Following [23], we begin by showing how the risk can be expressed only as a function of $\boldsymbol{W}_d$. The model term can in fact be rewritten ex-

actly as

$$\begin{aligned}\langle \boldsymbol{k}_{\boldsymbol{xM}} \boldsymbol{L}^{-\mathrm{T}}, \boldsymbol{w}\rangle_{\mathrm{F}} &= \langle \mathrm{ten}\left(\boldsymbol{k}_{x_1 \boldsymbol{m}_1} \boldsymbol{L}_1{}^{-\mathrm{T}} \otimes \boldsymbol{k}_{x_2 \boldsymbol{m}_2} \boldsymbol{L}_2{}^{-\mathrm{T}} \otimes \cdots \otimes \boldsymbol{k}_{x_D \boldsymbol{m}_D} \boldsymbol{L}_D{}^{-\mathrm{T}}\right), \\ &\qquad \mathrm{ten}\left(\sum_{r=1}^{R} \boldsymbol{w}_{1r} \otimes \boldsymbol{w}_{2r} \otimes \cdots \otimes \boldsymbol{w}_{Dr}\right)\rangle \\ &= \sum_{m_d=1}^{M_d} \sum_{r=1}^{R} (\boldsymbol{k}_{x_d \boldsymbol{m}_d} \boldsymbol{L}_{D:m_d}^{-\mathrm{T}} \sum_{m_1=1}^{M_1} w_{1 m_1 r} \boldsymbol{k}_{x_1 \boldsymbol{m}_1} \boldsymbol{L}_{1:m_1}^{-\mathrm{T}} \\ &\quad \cdots \sum_{m_D=1}^{M_D} w_{D m_D r} \boldsymbol{k}_{x_D \boldsymbol{m}_D} \boldsymbol{L}_{D:m_D}^{-\mathrm{T}}) w_{d m_d r} \\ &= \mathrm{vec}(\boldsymbol{k}_{x_d \boldsymbol{m}_d} \boldsymbol{L}_d{}^{-\mathrm{T}} \otimes (\boldsymbol{k}_{x_1 \boldsymbol{m}_1} \boldsymbol{L}_1{}^{-\mathrm{T}} \boldsymbol{W}_1 \odot \cdots \odot \boldsymbol{k}_{x_D \boldsymbol{m}_D} \boldsymbol{L}_D{}^{-\mathrm{T}} \boldsymbol{W}_D))^{\mathrm{T}} \mathrm{vec}(\boldsymbol{W}_d) \\ &= \langle \boldsymbol{g}_d(\boldsymbol{x}), \mathrm{vec}\,(\boldsymbol{W}_d)\rangle_{\mathrm{F}}. \end{aligned} \tag{3.14}$$

Similarly, for the regularization term

$$\begin{aligned}||\mathrm{ten}\,(\boldsymbol{w})||_{\mathrm{F}}^2 &= \langle \mathrm{ten}\left(\sum_{r=1}^{R} \boldsymbol{w}_{1r} \otimes \boldsymbol{w}_{2r} \otimes \cdots \otimes \boldsymbol{w}_{Dr}\right), \mathrm{ten}\left(\sum_{p=1}^{R} \boldsymbol{w}_{1p} \otimes \boldsymbol{w}_{2p} \otimes \cdots \otimes \boldsymbol{w}_{Dp}\right)\rangle_{\mathrm{F}} \\ &= \sum_{m_d=1}^{M_d} \sum_{r=1}^{R} \sum_{p=1}^{R} w_{d m_d, r} (\sum_{m_1=1}^{M_1} w_{1 m_1, r}\, w_{1 m_1, p} \cdots \sum_{m_D=1}^{M_D} w_{D m_D, r}\, w_{D m_D, p})\, w_{d m_d, p} \\ &= \sum_{m_d=1}^{M_d} \sum_{r=1}^{R} \sum_{p=1}^{R} w_{d m_d, r} (\boldsymbol{W}_1^{\mathrm{T}} \boldsymbol{W}_1 \odot \cdots \odot \boldsymbol{W}_D^{\mathrm{T}} \boldsymbol{W}_D)_{r,p}\, w_{d m_d, p} \\ &= \mathrm{vec}\,(\boldsymbol{W}_d)^{\mathrm{T}} (\boldsymbol{I}_{M_d} \otimes \boldsymbol{H}_d)\, \mathrm{vec}\,(\boldsymbol{W}_d). \end{aligned} \tag{3.15}$$

Here $\boldsymbol{H}_d \coloneqq \left(\boldsymbol{W}_1{}^{\mathrm{T}} \boldsymbol{W}_1 \odot \cdots \odot \boldsymbol{W}_d{}^{\mathrm{T}} \boldsymbol{W}_D\right)$. Substitution of Eq. (3.14) and Eq. (3.15) into Eq. (3.12) leads to a convex optimization problem for $\mathrm{vec}\,(\boldsymbol{W}_d)$, consisting in practice to training a kernel machine with $RM^{\frac{1}{D}}$ model parameters:

$$\begin{aligned}\min_{\mathrm{vec}(\boldsymbol{W}_d)} \; & \ell\left(\left\langle \boldsymbol{g}_d\,(\boldsymbol{x}_n), \mathrm{vec}\,(\boldsymbol{W}_d)\right\rangle, y_n\right) \\ & + \lambda \left\langle \mathrm{vec}\left(\boldsymbol{W}_d{}^{\mathrm{T}} \boldsymbol{W}_d\right), \mathrm{vec}\,(\boldsymbol{H}_d)\right\rangle, \end{aligned} \tag{3.16}$$

which in case of squared loss can be solved exactly by means of the normal equations

$$\left(\boldsymbol{G}_d{}^{\mathrm{T}} \boldsymbol{G}_d + \lambda \boldsymbol{I} \otimes \boldsymbol{H}_d\right) \mathrm{vec}\,(\boldsymbol{W}_d) = \boldsymbol{G}_d{}^{\mathrm{T}} \boldsymbol{y}, \tag{3.17}$$

where $[\boldsymbol{G}_d]_{i,:} = \boldsymbol{g}_d\,(\boldsymbol{x}_i)$. The computational cost of solving Eq. (3.17) exactly is of

$\mathcal{O}(NM^{\frac{2}{D}}R^2 + M^{\frac{3}{D}}R^3)$, where the first term accounts for constructing the relevant squared matrices and the second term accounts for solving the linear system. The storage requirements are $\mathcal{O}(M^{\frac{2}{D}}R^2)$ if one builds up $\boldsymbol{G}_d{}^{\mathrm{T}}\boldsymbol{G}_d$ as a series of N rank-1 updates. Alternating the minimization by iterating across all factor matrices, i.e. for $d = 1, 2, \ldots, D$ yields the a block coordinate algorithm which is well studied in the tensor community [22, 48]. Section 3.3 shows how the algorithm converges to suitable minima in all our experiments and is numerically stable. The computational complexity of our proposed Algorithm *CPD-Structured Inducing Points* (CPD-SIP) is then $\mathcal{O}(NDM^{\frac{2}{D}}R^2 + DM^{\frac{3}{D}}R^3)$.

Implementation Details Note that we could have considered in Eq. (3.12) the linearly equivalent feature map $\boldsymbol{k}_{\boldsymbol{xM}}$ which e.g. under squared loss and without rank constraints gives rise to the following regularized linear least-squares problem $(\boldsymbol{K}_{\boldsymbol{XM}}{}^{\mathrm{T}}\boldsymbol{K}_{\boldsymbol{XM}} + \boldsymbol{K}_{\boldsymbol{MM}})\bar{\boldsymbol{w}} = \boldsymbol{K}_{\boldsymbol{XM}}{}^{\mathrm{T}}\boldsymbol{y}$, where $\bar{\boldsymbol{w}} = \boldsymbol{L}^{-\mathrm{T}}\boldsymbol{w}$. However this formulation is prone to numerical instability, as the singular values of $\boldsymbol{K}_{\boldsymbol{XM}}$ are not scaled by $\boldsymbol{L}$, as discussed in [3, Chapter 3.4.3]. As a consequence when embedded in CPD-SIP, $\boldsymbol{G}_d$ has a higher condition number and spirals out of control after a few iterations.

The naive storage cost of $\mathcal{O}(ND(M^{\frac{2}{D}}R^2))$ can be reduced by a factor D by updating $\boldsymbol{G}_d$ and $\boldsymbol{H}_d$ in-place and by locating the inducing points on a dimension-independent grid, see Algorithm CPD-SIP, where we denote these in-place updated matrices as $\boldsymbol{G}$ and $\boldsymbol{H}$ respectively. Since these operations are $\mathcal{O}(NR)$, they do not affect the computational complexity. The storage complexity can be further reduced by carrying out the updates for $\boldsymbol{G}$ (e.g. in line 9 of CPD-SIP) in batches of one or more rows of $\boldsymbol{k}_{\boldsymbol{xm}}\boldsymbol{L}^{-\mathrm{T}}$, which can be computed on-the-fly, bringing it down to $\mathcal{O}(NR)$. Of course, if memory is not an issue, speedup up to a constant factor can be easily obtained by caching $\boldsymbol{k}_{\boldsymbol{Xm}}\boldsymbol{L}^{-\mathrm{T}}$, however this is not a requirement by any means. A summary of the computational and storage complexities of various SKI-related methods is given in Table 3.1, where we can observe that the computational complexity of Algorithm scales linearly in N and D, while having a storage complexity which is independent on D. All methods except the Simplex-GP, whose complexity scales with $\mathcal{O}(D^2)$, place the inducing points on a Cartesian grid. As a result, either an exponential number of computations or R copies of the whole dataset are required, prohibiting to tackle large-dimensional (in this case $D > 20$ [28]) data.

Selecting the hyperparameters In CPD-SIP, the CPD-rank R is introduced as additional hyperparameter, which similarly to other SKI-based approaches [17, 19] we advocate to select based on the computational budget at hand: M should

Algorithm CPD-SIP.

Require: Inputs $\boldsymbol{X} \in \mathbb{R}^{N \times D}$, outputs $\boldsymbol{y} \in \mathbb{R}^{N}$, kernel function $k(\cdot,\cdot) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, loss $\ell(\cdot,\cdot) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_{+}$, number of basis $\hat{M} \in \mathbb{N}_{+} : \hat{M} := M^{\frac{1}{D}}$, CPD-rank $R \in \mathbb{N}_{+}$, max iterations $S \in \mathbb{N}_{+}$

Ensure: Factor matrices $\boldsymbol{W}_d \in \mathbb{R}^{\hat{M} \times R}$, $d = 1, 2, \ldots, D$

$s \leftarrow 0$
$\boldsymbol{G} \leftarrow \text{ones}(\hat{M}, R)$
$\boldsymbol{H} \leftarrow \text{ones}(N, R)$
Compute $\boldsymbol{K_{mm}}$ using Eq. (3.6)
$\boldsymbol{L} \leftarrow \text{chol}(\boldsymbol{K_{mm}})$ from Eq. (3.11)
for $d = D, D-1, \ldots, 1$ **do**
    $\boldsymbol{W}_d \leftarrow \text{randn}(\hat{M}, R)$
    $\boldsymbol{W}_d \leftarrow \boldsymbol{W}_d / ||\boldsymbol{W}_d||$
    $\boldsymbol{G} \leftarrow \boldsymbol{G} \odot (\boldsymbol{K_{Xm}} \boldsymbol{L}^{-\mathrm{T}} \boldsymbol{W}_d)$
    $\boldsymbol{H} \leftarrow \boldsymbol{H} \odot (\boldsymbol{W}_d{}^{\mathrm{T}} \boldsymbol{W}_d)$
end for
repeat
    $s \leftarrow s + 1$
    for $d = 1, 2, \ldots, D$ **do**
        $\boldsymbol{H} \leftarrow \boldsymbol{H} \oslash (\boldsymbol{W}_d{}^{\mathrm{T}} \boldsymbol{W}_d)$
        $\boldsymbol{G} \leftarrow \boldsymbol{G} \oslash (\boldsymbol{K_{Xm}} \boldsymbol{L}^{-\mathrm{T}} \boldsymbol{W}_d)$
        Solve Eq. (3.16) for $\text{vec}(\boldsymbol{W}_d)$
        $\boldsymbol{W}^{(d)} \leftarrow \text{ten}(\text{vec}(\boldsymbol{W}_d))$
        $\boldsymbol{H} \leftarrow \boldsymbol{H} \odot (\boldsymbol{W}_d{}^{\mathrm{T}} \boldsymbol{W}_d)$
        $\boldsymbol{G} \leftarrow \boldsymbol{G} \odot (\boldsymbol{K_{Xm}} \boldsymbol{L}^{-\mathrm{T}} \boldsymbol{W}_d)$
    end for
until Convergence or $s = S$

be chosen first as to provide an accurate representation of the kernel, possibly to machine precision. This can be ensured for instance by means of cross-validation on a portion of unseen data. R can then be chosen in order to fill in the remainder of the available computational and memory budget.

Method	Complexities	
	Computational	Storage
KRR [27]	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$
SKI [17, 21]	$\mathcal{O}(N + M \log M)$	$\mathcal{O}(NM)$
SKIP [19]	$\mathcal{O}(NDR + RM^{\frac{1}{D}} \log M + NR^3 \log D + NR^2)$	$\mathcal{O}(DNR)$
Simplex-GP [28]	$\mathcal{O}(ND + MD^2))$	$\mathcal{O}(MD)$
CPD-SIP	$\mathcal{O}(D(NM^{\frac{2}{D}}R^2 + M^{\frac{3}{D}}R^3))$	$\mathcal{O}(NR)$

Table 3.1.: Computational and storage complexities of various SKI-based approaches when exploiting stationary structure. For *Scalable Kernel Interpolation for Products* (SKIP) R is typically chosen to be between 20 and 100 [28].

3.3 EXPERIMENTS

We implemented CPD-SIPin MathWorks MATLAB. The implementation and instructions to reproduce the results are available at https://github.com/fwesel/CPD-SIP. We scale the inputs in order to lie in the unit hypercube $[0,1]^D$. In case of regression problems, we standardize the responses to have zero mean and unit variance. In case of binary classification, we consider only the sign ± 1 of the responses [50]. We initialize the factor matrices $\boldsymbol{W}_d$ with standard normal numbers and normalize them to have unit norm. In all our experiments we set the number of iterations to $S = 20$ (consistent with Wesel and Batselier [23], as we define iterations as half a sweep). All the experiments were run on the Intel Core i7-10610U 1.8 GHz (CPU) of a Dell Inc. Latitude 7410 laptop with 16 GB of RAM. In what follows we present a series of three numerical experiments. Therein we demonstrate how our algorithm is stable and recovers the underlying KRR baseline with small values of R. We show how it compares with other grid-structured approaches managing to extend their applicability to data large in dimensionality ($D = 384$) or sample-size ($N = 5\,000\,000$).

3.3.1 NON-STATIONARY KERNEL

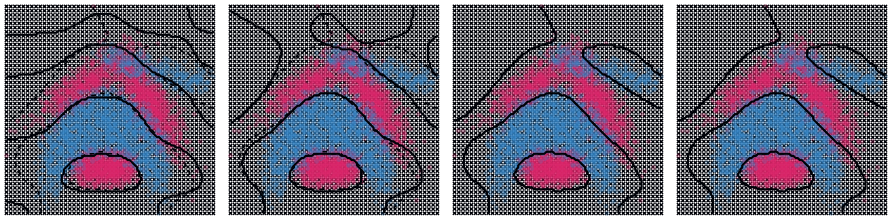
The Banana dataset [51] is a two-dimensional binary classification dataset which is often used in the context of low-rank kernel machines to visually demonstrate their characteristics. The dataset comprises $N = 5300$ data points roughly split in two classes. We consider the non-stationary separable polynomial kernel $k(\boldsymbol{x}, \boldsymbol{x'}) = \prod_{d=1}^{D}(1 + x_d x'_d)^5$, $\lambda = {}^{1\times10^{-6}}/_{N}$ and consider $M = 2500$ inducing points, 50 per di-

mension, located on an equidistant Cartesian grid. We then proceed to train a KRR classifier of Eq. (3.4) and CPD-SIP with the same hyperparameter λ. Since the problem is two-dimensional, ten$(\boldsymbol{w})$ is a matrix and has rank $R = M^{\frac{1}{D}} = 50$. In Fig. 3.1 we can observe that for low values of R the classification boundary is similar to the one of the KRR baseline where there is more data. This is because CPD-SIP seeks to minimize the empirical risk, and when provided with very few parameters it will seek to improve the classification boundary where the data is denser. We notice that already for $R = 6$ the classification boundary is indistinguishable from the one of KRR. As we will see next, the assumption of a small rank is valid also when dealing with higher-dimensional and large-sampled data.

3.3.2 COMPARISON WITH SKI

In order to compare our method with SKI, we consider seven *University of California, Irvine* (UCI) datasets [52], five of which are considered also by [28]. We compare our approach against SKIP [19] and Simplex-GP [28]. We consider the Gaussian kernel and locate $M = 10^D$ inducing points on a equidistant Cartesian grid and model $\boldsymbol{w}$ as a rank-20 CPD. In order to train a model in approximately the same function space, we select our hyperparameters l and λ by means of maximizing the log-likelihood of an exact GP model [53] constructed on a small random uniform subset of 2000 points. We then validate our model by means of 3-fold cross validation and report in Table 3.2 the standardized *Root Mean Squared Error* (RMSE) with one standard deviation. While training we keep track of the quality of our inducing-point approximation $\boldsymbol{k_{xM}}\boldsymbol{L}^{-\mathrm{T}}$ by sampling uniformly at random a subset $\boldsymbol{E}$ of 1000 points and computing the relative error $||\boldsymbol{K_{EE}} - \boldsymbol{K_{EM}}\boldsymbol{L}^{-\mathrm{T}}\boldsymbol{L}^{-1}\boldsymbol{K_{ME}}||/||\boldsymbol{K_{EE}}||$, which we report in Table 3.2. Here we can observe that the quality of the approximation approaches machine precision on many datasets, allowing the modeler to chose R according to the remaining computational budget.

In Table 3.2 we can observe that notwithstanding the sub-optimal choice of hyperparameters, our model is competitive in term of performance with the other inducing-points based approaches. Notably, although the seven considered datasets range vastly in the number of samples, they do not do so in the dimensionality, as all methods pay a heavy computational or storage-related price when scaling to higher-dimensional data. This is not the case of CPD-SIP which contrary to SKIP, does not *require* the contemporary storage of D $N \times R$ matrices, which allows us to tackle datasets of large dimensionality such as Slice with $D = 384$. Training our model on the laptop CPU requires then 11 274(189) s for the Song dataset



$R = 2$ $R = 3$ $R = 6$ $R = M^{\frac{1}{D}} = 50$

Figure 3.1.: Classification boundary of the two-dimensional Banana dataset for increasing CPD-ranks R for the non-stationary product kernel $k(\boldsymbol{x}, \boldsymbol{x'}) = \prod_{d=1}^{D}(1 + x_d x'_d)^5$. The dashed line is the KRR decision boundary while the full line corresponds to CPD-SIP. The black crosses are the locations of the inducing points. In the last plot the chosen CPD-rank matches the true (matrix) rank of ten ($\boldsymbol{w}$).

and 4724(198) s for the HouseElectric, compared to a per-epoch 1075(176) s of Simplex-GP [28, Table 4] on a Titan RTX (GPU) with 24 GB of RAM. Training on the Slice dataset took 226(2) s.

3.3.3 LARGE-SCALE CLASSIFICATION

In order to demonstrate the favorable complexity of CPD-SIP when dealing with a larger number of samples, we consider the *SUperSYmmetry* (SUSY) dataset [52, 54], an binary classification 18-dimensional dataset consisting of 5 000 000 samples, whose first 8 features consist of particle detector measurements, while the following 10 are high-level features engineered from the first 8. We consider $M = 20^D$ inducing points, and $R = 5, 10, 15, 20$. As is standard on this dataset, training is performed on the first 4 500 000 points and test on the remainder. We train both on only low-level and low-level plus high-level features. We use the Guassian kernel with l as the mean of the standard deviations of the features and $\lambda = {}^{2\times10^{-5}}/_{N}$ and report in Table 3.3 the *Area Under the Curve* (AUC), misclassification error and training time of our and other methods in literature. In Table 3.3 we can see that already for $R = 5$ our CPD-SIP scores similarly to *Variational Inducing Spherical Harmonics* (VISH) [55], whose reliance on numerically unstable spherical harmonics prohibits it however to be deployed on data with $D > 9$. For higher values of R, the performance rivals with *Deep Neural Networks* (DNNs). Others results on the dataset are from [56] where the authors obtain a misclassification rate of 20.1 % in 2400 s on a cluster with IBM POWER8 12-core CPUs and

Dataset			RMSE			Rel. Approx. Error
	$D\downarrow$	N	SKIP	Simplex GP	CPD-SIP	CPD-SIP
Precipitation	3	628474	1.032 ± 0.001	$\mathbf{0.939 \pm 0.001}$	0.974 ± 0.000	$(0.81 \pm 1.03) \times 10^{-2}$
Protein	9	45730	0.817 ± 0.012	$\mathbf{0.571 \pm 0.003}$	0.705 ± 0.004	$(1.94 \pm 1.74) \times 10^{-2}$
HouseElectric	11	2049280	NA	$\mathbf{0.079 \pm 0.002}$	0.084 ± 0.002	$(1.63 \pm 8.24) \times 10^{-14}$
Elevators	17	16599	0.447 ± 0.037	0.510 ± 0.018	$\mathbf{0.382 \pm 0.005}$	$(6.37 \pm 4.67) \times 10^{-15}$
KeggDirected	20	48827	0.487 ± 0.005	0.095 ± 0.002	$\mathbf{0.089 \pm 0.001}$	$(3.17 \pm 0.81) \times 10^{-13}$
Song	90	515345	NA	NA	$\mathbf{0.800 \pm 0.003}$	$(2.40 \pm 1.40) \times 10^{-5}$
Slice	386	53500	NA	NA	$\mathbf{0.094 \pm 0.002}$	$(2.79 \pm 1.65) \times 10^{-12}$

Table 3.2.: Predictive Standardized RMSE with one standard deviation on five UCI datasets [28, Table 2]

512 GB RAM.

3.4 CONCLUSION

In this work we build on the idea of placing inducing points on a Cartesian grid in order to exploit the computationally favorable arising tensorial structure. This allows us to obtain a good approximation of the kernel function on the whole domain, without sacrificing accuracy or the ability to tackle large-dimensional problems. In contrast to SKI and inducing points-related literature, we are in fact able to overcome the curse of dimensionality which affects both computational and storage-related complexities of these structured approaches by modeling the weights as a rank-R CPD. We show by means of numerical experiments how our approach is viable even on modest hardware. Note that all operations in CPD-SIP can be expressed as a series of matrix-vector products, enabling for efficient (multi-)GPU implementations. Furthermore, since our approach allows to learn from any product kernel, it allows to consider the SKI kernel of Eq. (3.9), which can be a product kernel depending on the the choice of interpolation strategy [17]. This could then allow for cheap caching of the features and further speedup by exploiting to the sparse structure in combination with stationary product kernels. One limitation of our approach is that its computational complexity scales with $\mathcal{O}(ND)$, prohibiting its application to data with a large number of samples *and* dimensionality, e.g. in case of categorical features. Another limitation, which we did not encounter in the experiments, is that the low-rank hypothesis is certainly not always justified, especially when dealing with highly complicated functions. We think that a possible remedy might be to seek for a different kernel space where the low-rank assumption would hold. Furthermore, although our approach is inherently non-probabilistic, our work allows to approximately carry out one of the two GP tasks, namely data fitting. Interesting further directions would be to investigate the regularizing effects of the low-rank constraint, to incorporate this exact approach in a probabilistic framework allowing for uncertainty quantification and possibly Bayesian model selection.

	AUC		1-Accuracy (%)	Time (s)
Technique	Low-level	Complete	Complete	Complete
BDT	0.850 ± 0.003	0.863 ± 0.003	NA	NA
Neural Network (NN)	0.867 ± 0.002	0.875 ± 0.001	NA	NA
Dropout NN	0.856 ± 0.001	0.873 ± 0.001	NA	NA
DNN	0.872 ± 0.001	0.876 ± 0.001	NA	NA
Dropout DNN	$\mathbf{0.876 \pm 0.001}$	$\mathbf{0.879 \pm 0.001}$	NA	NA
VISH	0.859 ± 0.001	NA	NA	NA
CPD-SIP ($R = 5$)	0.862 ± 0.002	0.872 ± 0.002	20.04 ± 0.01	1641 ± 21
CPD-SIP ($R = 10$)	0.867 ± 0.000	0.874 ± 0.000	19.82 ± 0.01	4650 ± 15
CPD-SIP ($R = 15$)	0.872 ± 0.000	0.875 ± 0.000	19.74 ± 0.00	6773 ± 52
CPD-SIP ($R = 20$)	0.872 ± 0.000	0.876 ± 0.000	19.68 ± 0.01	9446 ± 23

Table 3.3.: Predictive AUC, misclassification rate and training time with one standard deviation on the SUSY dataset. Results for *Bayesian Decision Tree* (BDT) and DNNs are from [54, Table 2], while the result for VISH is from [55, Table 3].

REFERENCES

[1] F. Wesel and K. Batselier. "Tensor-Based Kernel Machines with Structured Inducing Points for Large and High-Dimensional Data". In: *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2023, pp. 8308–8320.

[2] C. Cortes and V. Vapnik. "Support-Vector Networks". In: *Machine Learning* 20.3 (Sept. 1995), pp. 273–297.

[3] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press, 2006.

[4] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. "Deep Neural Networks as Gaussian Processes". In: *International Conference on Learning Representations*. Feb. 2018.

[5] R. Novak, L. Xiao, Y. Bahri, J. Lee, G. Yang, J. Hron, D. A. Abolafia, J. Pennington, and J. Sohl-dickstein. "Bayesian Deep Convolutional Networks with Many Channels Are Gaussian Processes". In: *International Conference on Learning Representations*. Sept. 2018.

[6] A. Garriga-Alonso, C. E. Rasmussen, and L. Aitchison. "Deep Convolutional Networks as Shallow Gaussian Processes". In: *International Conference on Learning Representations*. Sept. 2018.

[7] B. Hammer and K. Gersmann. "A Note on the Universal Approximation Capability of Support Vector Machines". In: *Neural Processing Letters* 17.1 (Feb. 2003), pp. 43–53.

[8] A. Rahimi and B. Recht. "Random Features for Large-Scale Kernel Machines". In: *Proceedings of the 20th International Conference on Neural Information Processing Systems*. Curran Associates Inc., Dec. 2007, pp. 1177–1184.

[9] Z. Yang, A. Wilson, A. Smola, and L. Song. "A La Carte – Learning Fast Kernels". In: *Artificial Intelligence and Statistics*. PMLR, Feb. 2015, pp. 1098–1106.

[10] C. Williams and M. Seeger. "Using the Nyström Method to Speed Up Kernel Machines". In: *Advances in Neural Information Processing Systems 13*. MIT Press, 2001, pp. 682–688.

[11] A. Smola and P. Bartlett. "Sparse Greedy Gaussian Process Regression". In: *Advances in Neural Information Processing Systems*. Vol. 13. MIT Press, 2001.

[12] L. Csató and M. Opper. “Sparse On-Line Gaussian Processes”. In: *Neural Computation* 14.3 (Mar. 2002), pp. 641–668.

[13] J. Quiñonero-Candela and C. E. Rasmussen. “A Unifying View of Sparse Approximate Gaussian Process Regression”. In: *Journal of Machine Learning Research* 6.65 (2005), pp. 1939–1959.

[14] E. Snelson and Z. Ghahramani. “Sparse Gaussian Processes Using Pseudo-inputs”. In: *Advances in Neural Information Processing Systems*. Vol. 18. MIT Press, 2006.

[15] M. Titsias. “Variational Learning of Inducing Variables in Sparse Gaussian Processes”. In: *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2009, pp. 567–574.

[16] G. Meanti, L. Carratino, L. Rosasco, and A. Rudi. “Kernel Methods through the Roof: Handling Billions of Points Efficiently”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 14410–14422.

[17] A. Wilson and H. Nickisch. “Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP)”. In: *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, June 2015, pp. 1775–1784.

[18] T. Nickson, T. Gunter, C. Lloyd, M. A. Osborne, and S. Roberts. “Blitzkriging: Kronecker-Structured Stochastic Gaussian Processes”. In: *arXiv:1510.07965 [stat]* (Oct. 2015). arXiv: 1510.07965 [stat].

[19] J. Gardner, G. Pleiss, R. Wu, K. Weinberger, and A. Wilson. “Product Kernel Interpolation for Scalable Gaussian Processes”. In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. PMLR, Mar. 2018, pp. 1407–1416.

[20] S. Stanton, W. Maddox, I. Delbridge, and A. G. Wilson. “Kernel Interpolation for Scalable Online Gaussian Processes”. In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. PMLR, Mar. 2021, pp. 3133–3141.

[21] M. Yadav, D. Sheldon, and C. Musco. “Faster Kernel Interpolation for Gaussian Processes”. In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. PMLR, Mar. 2021, pp. 2971–2979.

[22] T. G. Kolda and B. W. Bader. “Tensor Decompositions and Applications”. In: *SIAM Review* 51.3 (Aug. 2009), pp. 455–500.

[23] F. Wesel and K. Batselier. “Large-Scale Learning with Fourier Features and Tensor Decompositions”. In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 17543–17554.

[24] B. Schölkopf, R. Herbrich, and A. J. Smola. “A Generalized Representer Theorem”. In: *Computational Learning Theory*. Ed. by D. Helmbold and B. Williamson. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2001, pp. 416–426.

[25] Y. Saatchi. “Scalable Inference for Structured Gaussian Process Models”. PhD thesis. Cambridge: University of Cambridge, Nov. 2011.

[26] E. Gilboa, Y. Saatçi, and J. P. Cunningham. “Scaling Multidimensional Inference for Structured Gaussian Processes”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.2 (Feb. 2015), pp. 424–436.

[27] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Nov. 2002.

[28] S. Kapoor, M. Finzi, K. A. Wang, and A. G. G. Wilson. “SKIing on Simplices: Kernel Interpolation on the Permutohedral Lattice for Scalable Gaussian Processes”. In: *Proceedings of the 38th International Conference on Machine Learning*. PMLR, July 2021, pp. 5279–5289.

[29] F. L. Hitchcock. “The Expression of a Tensor or a Polyadic as a Sum of Products”. In: *Journal of Mathematics and Physics* 6.1-4 (1927), pp. 164–189.

[30] L. R. Tucker. “Some Mathematical Notes on Three-Mode Factor Analysis”. In: *Psychometrika* 31.3 (Sept. 1966), pp. 279–311.

[31] I. V. Oseledets. “Tensor-Train Decomposition”. In: *SIAM Journal on Scientific Computing* 33.5 (Jan. 2011), pp. 2295–2317.

[32] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov. “Tensorizing Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc., 2015.

[33] A. Tjandra, S. Sakti, and S. Nakamura. “Compressing Recurrent Neural Network with Tensor Train”. In: *2017 International Joint Conference on Neural Networks (IJCNN)*. May 2017, pp. 4451–4458.

[34] Y. Yang, D. Krompass, and V. Tresp. “Tensor-Train Recurrent Neural Networks for Video Classification”. In: *International Conference on Machine Learning*. PMLR, July 2017, pp. 3891–3900.

[35] M. Khodak, N. A. Tenenholtz, L. Mackey, and N. Fusi. “Initialization and Regularization of Factorized Neural Layers”. In: *International Conference on Learning Representations*. Sept. 2020.

[36] G. Favier and T. Bouilloc. “Parametric Complexity Reduction of Volterra Models Using Tensor Decompositions”. In: *2009 17th European Signal Processing Conference*. Aug. 2009, pp. 2288–2292.

[37] V. Lebedev, Y. Ganin, M. Rakhuba, I. V. Oseledets, and V. S. Lempitsky. “Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition”. In: *International Conference on Learning Representations*. Jan. 2015.

[38] K. Batselier, Z. Chen, and N. Wong. “Tensor Network Alternating Linear Scheme for MIMO Volterra System Identification”. In: *Automatica* 84 (Oct. 2017), pp. 26–35.

[39] E. M. Stoudenmire and D. J. Schwab. “Supervised Learning with Tensor Networks”. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. Curran Associates Inc., Dec. 2016, pp. 4806–4814.

[40] A. Novikov, I. Oseledets, and M. Trofimov. “Exponential Machines”. In: *Bulletin of the Polish Academy of Sciences: Technical Sciences; 2018; 66; No 6 (Special Section on Deep Learning: Theory and Practice); 789-797* (2018).

[41] Z. Chen, K. Batselier, J. A. K. Suykens, and N. Wong. “Parallelized Tensor Train Learning of Polynomial Classifiers”. In: *IEEE Transactions on Neural Networks and Learning Systems* 29.10 (Oct. 2018), pp. 4621–4632.

[42] S. Zhe, W. Xing, and R. M. Kirby. “Scalable High-Order Gaussian Process Regression”. In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2019, pp. 2611–2620.

[43] P. Izmailov, A. Novikov, and D. Kropotov. “Scalable Gaussian Processes with Billions of Inducing Inputs via Tensor Train Decomposition”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR, Mar. 2018, pp. 726–735.

[44] T. Evans and P. Nair. “Scalable Gaussian Processes with Grid-Structured Eigenfunctions (GP-GRIEF)”. In: *International Conference on Machine Learning*. PMLR, July 2018, pp. 1417–1426.

[45] N. D. Sidiropoulos and R. Bro. “On the Uniqueness of Multilinear Decomposition of N-Way Arrays”. In: *Journal of Chemometrics* 14.3 (2000), pp. 229–239.

[46] J. D. Carroll and J.-J. Chang. “Analysis of Individual Differences in Multidimensional Scaling via an N-Way Generalization of “Eckart-Young” Decomposition”. In: *Psychometrika* 35.3 (Sept. 1970), pp. 283–319.

[47] R. A. Harshman. “Foundations of the PARAFAC Procedure : Models and Conditions for An”. In: *UCLA Working Papers in Phonetics* 16 (1970), pp. 1–84.

[48] A. Uschmajew. “Local Convergence of the Alternating Least Squares Algorithm for Canonical Tensor Approximation”. In: *SIAM Journal on Matrix Analysis and Applications* 33.2 (Jan. 2012), pp. 639–652.

[49] B. Khavari and G. Rabusseau. “Lower and Upper Bounds on the Pseudo-Dimension of Tensor Network Models”. In: *Advances in Neural Information Processing Systems*. May 2021.

[50] J. Suykens and J. Vandewalle. “Least Squares Support Vector Machine Classifiers”. In: *Neural Processing Letters* 9.3 (June 1999), pp. 293–300.

[51] J. Hensman, A. Matthews, and Z. Ghahramani. “Scalable Variational Gaussian Process Classification”. In: *Artificial Intelligence and Statistics*. PMLR, Feb. 2015, pp. 351–360.

[52] D. Dua and C. Graff. *UCI Machine Learning Repository*. 2017.

[53] C. E. Rasmussen and H. Nickisch. “Gaussian Processes for Machine Learning (GPML) Toolbox”. In: *Journal of Machine Learning Research* 11.100 (2010), pp. 3011–3015.

[54] P. Baldi, P. Sadowski, and D. Whiteson. “Searching for Exotic Particles in High-Energy Physics with Deep Learning”. In: *Nature Communications* 5.1 (July 2014), p. 4308.

[55] V. Dutordoir, N. Durrande, and J. Hensman. “Sparse Gaussian Processes with Spherical Harmonic Features”. In: *International Conference on Machine Learning*. PMLR, Nov. 2020, pp. 2793–2802.

[56] J. Chen, H. Avron, and V. Sindhwani. “Hierarchically Compositional Kernels for Scalable Nonparametric Learning”. In: *Journal of Machine Learning Research* 18.66 (2017), pp. 1–42.

4

QUANTIZED FOURIER AND POLYNOMIAL FEATURES FOR MORE EXPRESSIVE TENSOR NETWORK MODELS

In the context of kernel machines, polynomial and Fourier features are commonly used to provide a nonlinear extension to linear models by mapping the data to a higher-dimensional space. Unless one considers the dual formulation of the learning problem, which renders exact large-scale learning unfeasible, the exponential increase of model parameters in the dimensionality of the data caused by their tensor-product structure prohibits to tackle high-dimensional problems. One of the possible approaches to circumvent this exponential scaling is to exploit the tensor structure present in the features by constraining the model weights to be an underparametrized tensor network. In this paper we quantize, i.e. further tensorize, polynomial and Fourier features. Based on this feature quantization we propose to quantize the associated model weights, yielding quantized models. We show that, for the same number of model parameters, the resulting quantized models have a higher bound on the VC-dimension as opposed to their non-quantized counterparts, at no additional computational cost while learning from identical features. We verify experimentally how this additional tensorization regularizes the learning problem by prioritizing the most salient features in the data and how it provides models with increased generalization capabilities. We finally benchmark our approach on large regression task, achieving state-of-the-art results on a laptop computer.

This chapter has been published as:
F. Wesel and K. Batselier. "Quantized Fourier and Polynomial Features for More Expressive Tensor Network Models". In: *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2024, pp. 1261–1269

4.1 INTRODUCTION

In the context of supervised learning, the goal is to estimate a function $f(\cdot) : \mathbb{R}^D \to \mathbb{R}$ given N input-output pairs $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$, where $\boldsymbol{x} \in \mathbb{R}^D$ and $y \in \mathbb{R}$. Kernel machines accomplish this by lifting the input data into a high-dimensional feature space by means of a *feature map* $\boldsymbol{z}(\cdot) : \mathbb{R}^D \to \mathbb{R}^M$ and seeking a linear relationship therein:

$$f(\boldsymbol{x}, \boldsymbol{w}) = \langle \boldsymbol{z}(\boldsymbol{x}), \boldsymbol{w} \rangle . \tag{4.1}$$

Training such a model involves the minimization of the regularized empirical risk given a convex measure of loss $\ell(\cdot,\cdot) : \mathbb{R}^M \times \mathbb{R} \to \mathbb{R}_+$

$$R_{\text{empirical}}(\boldsymbol{w}) = \frac{1}{N} \sum_{n=1}^{N} \ell\left(\langle \boldsymbol{z}(\boldsymbol{x}_n), \boldsymbol{w} \rangle, y_n\right) + \lambda \|\boldsymbol{w}\|^2 . \tag{4.2}$$

Different choices of loss yield the *primal* formulation of different kernel machines. For example, squared loss results in *Kernel Ridge Regression* (KRR) [2], hinge loss in *Support Vector Machines* (SVMs) [3], and logistic loss yields logistic regression. Different choices of the feature map $\boldsymbol{z}$ allow for modeling different nonlinear behaviors in the data. In this article we consider tensor-product features

$$\boldsymbol{z}(\boldsymbol{x}) = \bigotimes_{d=1}^{D} \boldsymbol{v}_d(x_d), \tag{4.3}$$

where $\boldsymbol{v}_d(\cdot) : \mathbb{C} \to \mathbb{C}^{M_d}$ is a feature map acting on each element of the d-th component x_d of $\boldsymbol{x} \in \mathbb{C}^D$. Here $\otimes$ denotes the *left* Kronecker product [4]. This tensor-product structure arises when considering product kernels [5, 6, 7], Fourier features [8], when considering B-splines [9] and polynomials [5].

Due to the tensor-product structure in Eq. (4.3), $\boldsymbol{z}(\cdot)$ maps an input sample $\boldsymbol{x} \in \mathbb{C}^D$ into an exponentially large feature vector $\boldsymbol{z}(\boldsymbol{x}) \in \mathbb{C}^{M_1 M_2 \cdots M_D}$. As a result, the model is also described by an exponential number of weights $\boldsymbol{w}$. This exponential scaling in the number of features limits the use of tensor-product features to low-dimensional data or to mappings of very low degree.

Both these computational limitations can be sidestepped entirely by considering the *dual* formulation of the learning problem in Eq. (4.2), requiring to compute the pairwise similarity of all data respectively by means of a kernel function $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{z}(\boldsymbol{x}), \boldsymbol{z}(\boldsymbol{x}') \rangle$. However, the dual formulation requires to instantiate the kernel matrix at a cost of $\mathcal{O}(N^2)$ and to estimate N Lagrange multipliers by solving a (convex) quadratic problem at a cost of at least $\mathcal{O}(N^2)$, prohibiting to tackle

large-scale data (large N). To lift these limitations, a multitude of research has focused on finding low-rank approximations of kernels by considering *random* methods such as polynomial sketching [10, 11, 12] and random features [13, 14, 15], which approximate the feature space with probabilistic approximation guarantees.

One way to take advantage of the existing tensor-product structure in Eq. (4.3) is by imposing a tensor network [16, 17] constraint on the weights $\boldsymbol{w}$. For example, using a polyadic rank-R constraint reduces the storage complexity of the weights from $\mathcal{O}(M^D)$ down to $\mathcal{O}(DMR)$ and enables the development of efficient learning algorithms with a computational complexity of $\mathcal{O}(DMR)$ per gradient descent iteration. This idea has been explored for polynomial [18, 19, 20, 21, 22] pure-power-1 polynomials [23], pure-power polynomials of higher degree [24], B-splines [9], and Fourier features [8, 25, 26, 27, 28, 29].

In this article, we improve on this entire line of research by deriving an exact *quantized* representation [30] of pure-power polynomials and Fourier features, exploiting their inherent Vandermonde structure. It is worth noting that *in this paper quantized means further tensorized*, and should not be confused with the practice of working with lower precision floating point numbers. By virtue of the derived quantized features, we are able to quantize the model weights. We show that compared to their non-quantized counterparts, quantized models can be trained with no additional computational cost, while learning from the same exact features. Most importantly, for the same number of model parameters the ensuing quantized models are characterized by higher upper bounds on the *Vapnik–Chervonenkis* (VC)-dimension, which indicates a potential higher expressiveness. While these bounds are in practice not necessarily met, we verify experimentally that:

1. Quantized models are indeed characterized by higher expressiveness. This is demonstrated in Section 4.5.1, where we show that in the underparameterized regime quantized models achiever lower test errors than the non-quantized models with identical features and identical total number of model parameters.

2. This additional structure regularizes the problem by prioritizing the learning of the peaks in the frequency spectrum of the signal (in the case of Fourier features) (Section 4.5.2). In other words, the quantized structure is learning the most salient features in the data first with its limited amount of available model parameters.

3. Quantized tensor network models can provide state-of-the-art performance

on large-scale real-life problems. This is demonstrated in Section 4.5.3, where we compare the proposed quantized model to both its non-quantized counterpart and other state-of-the-art methods, demonstrating superior generalization performance on a laptop computer.

4.2 BACKGROUND

We denote scalars in both capital and non-capital italics w, W, vectors in non-capital bold $\boldsymbol{w}$, matrices in capital bold $\boldsymbol{W}$ and tensors, also known as higher-order arrays, in capital italic bold font $\mathcal{W}$. Sets are denoted with calligraphic capital letters, e.g. $\mathcal{S}$. The m-th entry of a vector $\boldsymbol{w} \in \mathbb{C}^M$ is indicated as w_m and the $m_1 m_2 \dots m_D$-th entry of a D-dimensional tensor $\mathcal{W} \in \mathbb{C}^{M_1 \times M_2 \times \cdots \times M_D}$ as $w_{m_1 m_2 \dots m_D}$. We denote the complex-conjugate with superscript * and $\otimes$ denotes the *left* Kronecker product [4]. We employ zero-based indexing for all tensors. The Frobenius inner product between two D-dimensional tensors $\mathcal{V}, \mathcal{W} \in \mathbb{C}^{M_1 \times M_2 \times \cdots \times M_D}$ is defined as

$$\langle \mathcal{V}, \mathcal{W} \rangle_{\mathrm{F}} \coloneqq \sum_{m_1=0}^{M_1-1} \sum_{m_2=0}^{M_2-1} \cdots \sum_{m_D=0}^{M_D-1} v^*_{m_1 m_2 \dots m_D} w_{m_1 m_2 \dots m_D}. \tag{4.4}$$

We define the vectorization operator as $\mathrm{vec}(\cdot) : \mathbb{C}^{M_1 \times M_2 \times \cdots \times M_D} \to \mathbb{C}^{M_1 M_2 \cdots M_D}$ such that

$$\mathrm{vec}(\mathcal{W})_m = w_{m_1 m_2 \dots m_D},$$

with $m = m_1 + \sum_{d=2}^{D} m_d \prod_{k=1}^{d-1} M_k$. Likewise, its inverse, the tensorization operator $\mathrm{ten}(\cdot, M_1, M_2, \dots, M_D) : \mathbb{C}^{M_1 M_2 \cdots M_D} \to \mathbb{C}^{M_1 \times M_2 \times \dots M_D}$ is defined such that

$$\mathrm{ten}(\boldsymbol{w}, M_1, M_2, \dots, M_D)_{m_1 m_2 \cdots m_D} = w_m.$$

4.2.1 TENSOR NETWORKS

Tensor Networks (TNs) [4, 16, 31, 32] express a D-dimensional tensor $\mathrm{ten}(\boldsymbol{w}, M_1, M_2, \dots, M_D) =: \mathcal{W}$ as a multi-linear function of C *core* tensors, see Definition A.1 for a rigorous definition. Two commonly used TNs are the *Canonical Polyadic Decomposition* (CPD) and *Tensor Train* (TT).

Definition 4.2.1 (CPD [16, 33])**.** A D-dimensional tensor $\mathcal{W} \in \mathbb{C}^{M_1 \times M_2 \times \cdots \times M_D}$ has

a rank-R CPD if

$$w_{m_1 m_2 \ldots m_D} = \sum_{r=0}^{R-1} \prod_{d=1}^{D} w_{d\, m_d r}.$$

The cores of this particular network are $C = D$ matrices $\boldsymbol{W}_d \in \mathbb{C}^{M_d \times R}$. The storage complexity $P = R\sum_{d=1}^{D} M_d$ of a rank-R CPD is therefore $\mathcal{O}(DMR)$, where $M = \max(M_1, M_2, \ldots, M_D)$.

Definition 4.2.2 (TT [34])**.** A D-dimensional tensor $\mathcal{W} \in \mathbb{C}^{M_1 \times M_2 \times \cdots \times M_D}$ admits a rank-$(R_1 \coloneqq 1, R_2, \ldots, R_D, R_{D+1} \coloneqq R_1)$ TT if

$$w_{m_1 m_2 \ldots m_D} = \sum_{r_1=0}^{R_1-1} \sum_{r_2=0}^{R_2-1} \cdots \sum_{r_D=0}^{R_d-1} \prod_{d=1}^{D} w_{d\, r_d m_d r_{d+1}}.$$

The cores of a TT are the $C = D$ 3-dimensional tensors $\mathcal{W}_d \in \mathbb{C}^{R_d \times M_d \times R_{d+1}}$. The case $R_1 > 1$ is also called a *Tensor Ring* (TR) [35]. Throughout the rest of this article we will simply refer to the TT rank as $R = \max(R_2, \cdots, R_D)$. The storage complexity $P = \sum_{d=1}^{D} M_d R_d R_{d+1}$ of a TT is then $\mathcal{O}(DMR^2)$. A TN is *underparametrized* if $P \ll \prod_{d=1}^{D} M_d$, i.e. it can represent a tensor with fewer parameters than the number of entries of the tensor.

Other TNs are the Tucker decomposition [36, 37], hierarchical hierarchical Tucker [38, 39] decomposition, block-term decompositions [40, 41], *Projected Entangled Pair States* (PEPS) [42] and *Multi-scale Entanglement Renormalization Ansatz* (MERA) [43].

4.2.2 TENSORIZED KERNEL MACHINES

The tensor-product structure of features in Eq. (4.3) can be exploited by imposing a tensor network structure onto the tensorized model weights

$$\mathrm{ten}\,(\boldsymbol{w}, M_1, M_2, \ldots, M_D)\,.$$

Although generally speaking the tensorized model weights are not full rank, modeling them as an underparametrized tensor network allows to compute fast model responses when the feature map $\boldsymbol{z}(\cdot)$ is of the form of Eq. (4.3).

Theorem 4.2.3 (Tensorized kernel machine (TKM))**.** *Suppose* $\mathrm{ten}\,(\boldsymbol{w}, M_1, M_2, \ldots, M_D)$

is a tensor in CPD, TT *or* TR *form. Then model responses and associated gradients*

$$f(\boldsymbol{x}, \boldsymbol{w}) = \langle \bigotimes_{d=1}^{D} \boldsymbol{v}_d(x_d), \boldsymbol{w} \rangle_{\mathrm{F}},$$

can be computed in $\mathcal{O}(P)$ *instead of* $\mathcal{O}(\prod_{d=1}^{D} M_d)$*, where* $P = DMR$ *in case of* CPD*, and* $P = DMR^2$ *in case of* TT *or* TR*.*

Proof. See Section B.1. □

Results for more general TNs can be found in Section B.1. This idea has been explored for a plethora of different combinations of tensor-product features and tensor networks [8, 23, 24, 25, 28, 29, 44]. A graphical depiction of a TKM can be found in Fig. 4.1a: a full line denotes a summation along the corresponding index, while a dotted line denotes a Kronecker product. Training a kernel machine under such constraint yields the following nonconvex optimization problem:

$$\min_{\boldsymbol{w}} \frac{1}{N} \sum_{n=1}^{N} \ell(\langle \bigotimes_{d=1}^{D} \boldsymbol{v}_d(x_d), \boldsymbol{w} \rangle_{\mathrm{F}}, y_n) + \lambda ||\boldsymbol{w}||_{\mathrm{F}}^2, \tag{4.5}$$
$$\text{s.t. } \mathrm{ten}(\boldsymbol{w}, M_1, M_2, \ldots, M_D) \text{ is a tensor network.}$$

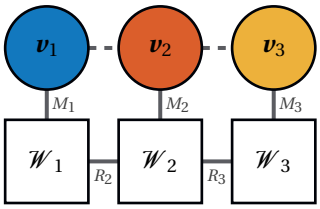
Common choices of tensor network-specific optimizers are the *Alternating Least-Squares* (ALS) [16, 45, 46, 47], the *Density Matrix Renormalization Group* (DMRG) [48] and Riemannian optimization [23, 49]. Generic first or second order gradient-based optimization method can also be employed.

4.3 QUANTIZING POLYNOMIAL AND FOURIER FEATURES

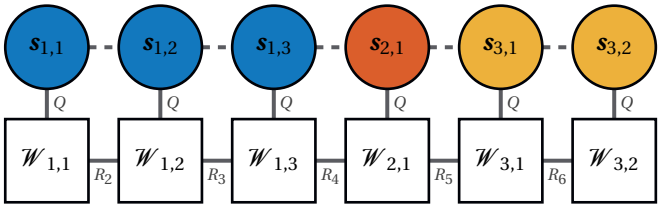
Before presenting the main contribution of this article, we first provide the definition of a pure-power polynomial feature map.

Definition 4.3.1 (Pure-power polynomial feature map [24])**.** For an input sample $\boldsymbol{x} \in \mathbb{C}^D$, the pure-power polynomial features $\boldsymbol{z}(\cdot) : \mathbb{C}^D \to \mathbb{C}^{M_1 M_2 \cdots M_D}$ of degree $(M_1 - 1, M_2 - 1, \ldots, M_D - 1)$ are defined as

$$\boldsymbol{z}(\boldsymbol{x}) = \bigotimes_{d=1}^{D} \boldsymbol{v}_d(x_d),$$



(a) TKM with TT-constrained weights.



(b) Corresponding QTKM with Q-quantized TT-constrained weights.

Figure 4.1.: TKM (Fig. 4.1a), and QTKM (Fig. 4.1b) with TT-constrained model weights. In these diagrams, each circle represent a vector which constitutes the pure-power feature map of Definition 4.3.1, and each square represents a TT core (Definition 4.2.2). The color coding relates the d-th feature with its quantized representation. A full connecting line denotes a summation along the corresponding index, while a dotted line denotes a Kronecker product, see Cichocki *et al.* [4] for a more in-depth explanation. Figure 4.1b depicts the case where $K_1 = Q^3$, $K_2 = Q$ and $K_3 = Q^2$. Notice how quantization allows to model correlations within each particular mode of the model weights, in this case explicitly by means of the TT ranks $(1, R_2, \ldots, R_6, 1)$.

with $\boldsymbol{v}_d(\cdot) : \mathbb{C} \to \mathbb{C}^{M_d}$ the Vandermonde vector

$$\boldsymbol{v}_d(x_d) = \left[1, x_d, x_d^2, \ldots, x_d^{M_d-1}\right].$$

The m_d-th element of the feature map vector $\boldsymbol{v}_d(x_d)$ is

$$v_d(x_d)_{m_d} = (x_d)^{m_d}, \quad m_d = 0, 1, \ldots, M_d - 1.$$

The definition of the feature map is given for degree $(M_1-1, M_2-1, \ldots, M_D-1)$ such that the feature map vector $z(\boldsymbol{x})$ has a length $M_1 M_2 \cdots M_D$. The Kronecker product in Definition 4.3.1 ensures that all possible combinations of products of monomial basis functions are computed, up to a total degree of $\sum_{d=1}^{D}(M_d-1)$. Compared to the more common affine polynomials, which are basis functions of the polynomial kernel $k(\boldsymbol{x}, \boldsymbol{x'}) = (b + \langle \boldsymbol{x}, \boldsymbol{x'} \rangle)^M$, pure-power polynomial features contain more higher-order terms. Similarly, their use is justified by the Stone-Weierstrass theorem [50], which guarantees that any continuous function on a locally compact domain can be approximated arbitrarily well by polynomials of increasing degree. Fourier features can be similarly defined by replacing the monomials with complex exponentials.

Definition 4.3.2. (Fourier Features) For an input sample $\boldsymbol{x} \in \mathbb{C}^D$, the Fourier feature map $\boldsymbol{\varphi}(\cdot) : \mathbb{C}^D \to \mathbb{C}^{M_1 M_2 \cdots M_D}$ with M_d basis frequencies $-M_d/2, \ldots, M_d/2-1$ per dimension is defined as

$$\boldsymbol{\varphi}(\boldsymbol{x}) = \bigotimes_{d=1}^{D} \left(c_d \, \boldsymbol{v}_d \left(e^{-\frac{2\pi j x_d}{L}} \right) \right),$$

where j is the imaginary unit, $c_d = e^{2\pi j x_d \frac{2+M_d}{2L}} \in \mathbb{C}$, $L \in \mathbb{R}$ is the periodicity of the function class and $\boldsymbol{v}^{(d)}(\cdot)$ are the Vandermonde vectors of Definition 4.3.1.

Fourier features are ubiquitous in the field of kernel machines as they are eigenfunctions of D-dimensional stationary product kernels with respect to the Lebesgue measure, see [51, Chapter 4.3] or [6, 7]. As such they are often used for the uniform approximation of such kernels in the limit of $L \to \infty$ and $M_1, M_2, \ldots, M_D \to \infty$ [8, Proposition 1].

We now present the first contribution of this article, which is an exact *quantized*, i.e. further tensorized, representation of pure-power polynomials and Fourier features. These quantized features allows for the quantization of the model weights, which enables to impose additional tensor network structure between features, yielding more expressive models for the same number of model parameters.

4.3.1 QUANTIZED FEATURES

In order to quantize pure-power polynomial features we assume for ease of notation that M_d can be written as some power $M_d = Q^{K_d}$, where both $Q, K_d \in \mathbb{N}$. The more general case involves considering the (prime) factorization of M_d and follows the same derivation steps albeit with more intricate notation.

Definition 4.3.3 (Quantized Vandermonde vector)**.** For $Q, k \in \mathbb{N}$, we define the quantized Vandermonde vector $\boldsymbol{s}_{d,k}(\cdot) : \mathbb{C} \to \mathbb{C}^Q$ as

$$\boldsymbol{s}_{d,k}(x_d) \coloneqq \left[1, x_d^{Q^{k-1}}, \ldots, x_d^{(Q-1)Q^{k-1}}\right].$$

The q-th element of $\boldsymbol{s}_{d,k}(x_d)$ is therefore

$$s_{d,k}(x_d)_q = (x_d)^{qQ^{k-1}}, \quad q = 0, 1, \ldots, Q-1.$$

Theorem 4.3.4 (Quantized pure-power-$(M_d - 1)$ polynomial feature map)**.** *Each Vandermonde vector $\boldsymbol{v}^{(d)}(x_d)$ can be expressed as a Kronecker product of K_d factors*

$$\boldsymbol{v}_d(x_d) = \bigotimes_{k=1}^{K_d} \boldsymbol{s}_{d,k}(x_d),$$

where $M_d = Q^{K_d}$.

Proof. From Definition 4.3.1 we have that

$$v_d(x_d)_{m_d} = (x_d)^{m_d}.$$

Assume that $M_d = Q^{K_d}$. We proceed by tensorizing $\boldsymbol{v}_d(x_d)$ along K_d dimensions, each having size Q. Then

$$\begin{aligned} v_d(x_d)_{m_d} &= \operatorname{ten}(v_d, Q, Q, \ldots, Q)_{q_1 q_2 \ldots q_{K_d}} \\ &= (x_d)^{\sum_{k=1}^{K_d} q_k Q^{k-1}} \\ &= \prod_{k=1}^{K_d} (x_d)^{q_k Q^{k-1}} \\ &= \prod_{k=1}^{K_d} s_{d,k}(x_d)_{q_k}. \end{aligned}$$

The last equality follows directly from Definition 4.3.3. Hence by the definition of Kronecker product, we have that

$$\boldsymbol{v}_d(x_d) = \bigotimes_{k=1}^{K_d} \boldsymbol{s}_{d,k}(x_d).$$

□

Note once more that in principle it is possible to tensorize with respect to K_d indices such that $M_d = Q_1 Q_2 \cdots Q_{K_d}$, but we restrain from doing so not to needlessly complicate notation. Theorem 4.3.4 allows then to quantize pure-power and Fourier features.

Corollary 4.3.5 (Quantized pure-power polynomials)**.** *For an input sample $\boldsymbol{x} \in \mathbb{C}^D$, the pure-power polynomial feature map can be expressed as*

$$\boldsymbol{z}(\boldsymbol{x}) = \bigotimes_{d=1}^{D} \bigotimes_{k=1}^{K_d} \boldsymbol{s}_{d,k}(x_d).$$

Corollary 4.3.6 (Quantized Fourier feature map)**.** *For an input sample $\boldsymbol{x} \in \mathbb{C}^D$, the Fourier feature map can be expressed as*

$$\boldsymbol{\varphi}(\boldsymbol{x}) = \bigotimes_{d=1}^{D} \bigotimes_{k=1}^{K_d} c_d^{\frac{1}{K_d}} \boldsymbol{s}_{d,k}\left(e^{-\frac{2\pi j x_d}{L}}\right),$$

where $c_d = e^{2\pi j x_d \frac{2+M_d}{2L}}$.

Note that when quantized, both pure-power and Fourier features admit an efficient storage complexity of $\mathcal{O}(DK) = \mathcal{O}(D \log M)$ instead of $\mathcal{O}(DM)$, where $K = \max(K_1, \ldots, K_D)$.

Example 4.3.7. Consider $D = 2$, $M_1 = 8 = 2^3$ $M_2 = 4 = 2^2$, then the Vandermonde vector of monomials up to total degree 10 is constructed from

$$\boldsymbol{z}(\boldsymbol{x}) = [1, x_1] \otimes \left[1, x_1^2\right] \otimes \left[1, x_1^4\right] \otimes [1, x_2] \otimes \left[1, x_2^2\right].$$

We now present the second contribution of this article, which is the quantization of the model weights associated with quantized polynomial and Fourier features. As we will see, these quantized models are more expressive given the number of model parameters and same exact features.

4.4 QUANTIZED TENSOR NETWORK KERNEL MACHINES

When not considering quantization, model weights allow for tensorial indexing along the D dimensions of the inputs, i.e. $\text{ten}(\boldsymbol{w}, M_1, M_2, \ldots, M_D)$. Corollary 4.3.5 and Corollary 4.3.6 allow to exploit the Kronecker product structure

of pure-power polynomial and Fourier features by further tensorizing the model weights of the tensor network-constrained kernel machines of Eq. (4.5)

$$\text{ten}(\boldsymbol{w}, \underbrace{Q, Q, \ldots, Q}_{\sum_{d=1}^{D} K_d \text{ times}}).$$

These further factorized model weights can then be constrained to be a tensor network, and learned by minimizing the empirical risk in the framework of Eq. (4.5). Training a kernel machine under this constraint results in the following nonlinear optimization problem:

$$\min_{\boldsymbol{w}} \frac{1}{N} \sum_{n=1}^{N} \ell(\langle \bigotimes_{d=1}^{D} \bigotimes_{k=1}^{K_d} \boldsymbol{s}_{d,k}(x_d), \boldsymbol{w} \rangle_{\text{F}}, y_n) + \lambda ||\boldsymbol{w}||_{\text{F}}^2, \tag{4.6}$$
$$\text{s.t. ten}(\boldsymbol{w}, Q, Q, \ldots, Q) \text{ is a tensor network.}$$

4

4.4.1 COMPUTATIONAL COMPLEXITY

In case of CPD, TT or TR-constrained and quantized model weights, model responses and associated gradients can be computed at the same cost as with non-quantized models:

Theorem 4.4.1 (Quantized tensorized kernel machine (QTKM))**.** *Consider pure-power and Fourier feature maps factorized as in Corollary 4.3.5 and Corollary 4.3.6 and suppose* $\text{ten}(\boldsymbol{w}, Q, Q, \ldots, Q)$ *is a tensor in* CPD, TT *or* TR *form. Then by Theorem 4.2.3, model responses and associated gradients*

$$f_{quantized}(\boldsymbol{x}, \boldsymbol{w}) = \langle \bigotimes_{d=1}^{D} \bigotimes_{k=1}^{K_d} \boldsymbol{s}_{d,k}(x_d), \boldsymbol{w} \rangle_{\text{F}},$$

can be computed in $\mathcal{O}(P)$ *instead of* $\mathcal{O}(\prod_{d=1}^{D} M_d)$*, where* $P = KDQR$ *in case of* CPD*, and* $P = KDQR^2$ *in case of* TT *or* TR*.*

Proof. See Section B.2. □

Results for more general TNs can be found in Section B.2. A graphical depiction of a QTKM can be found in Fig. 4.1b. Furthermore, when considering tensor network-specific optimization algorithms, the time complexity per iteration of training when optimizing Eq. (4.6) is lower compared to Eq. (4.5), as these meth-

ods typically optimize over a subset (typically one core) of model parameters, see Section C.

4.4.2 INCREASED MODEL EXPRESSIVENESS

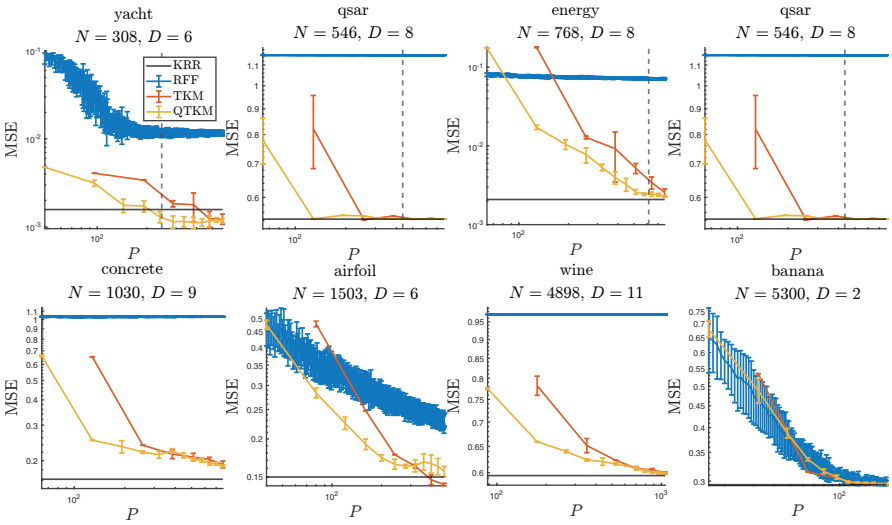


Figure 4.2.: Plots of the test mean squared error as a function of the number of model parameters P, for different real-life datasets. In blue, random Fourier features [14], in red tensorized kernel machines with Fourier features [8, 25, 27, 29], in yellow quantized kernel machines with Fourier features, with quantization $Q = 2$. The gray horizontal full line is the full unconstrained optimization problem, which corresponds to KRR. The grey vertical dotted line is set at $P = N$. It can be seen that for $P < N$ case, quantization allows to achieve better generalization performance with respect to the non-quantized case.

Constraining a tensor to be a tensor network allows to distill the most salient characteristics of the data in terms of an limited number of effective parameters without destroying its multi-modal nature. This is also known as the *blessing of dimensionality* [31] and is the general underlying concept behind tensor network-based methods. In the more specific context of supervised kernel machines, these well-known empirical considerations are also captured in the rigorous framework of VC-theory [52]. Khavari and Rabusseau [44, theorem 2] have

recently shown that the VC-dimension and pseudo-dimension of tensor network-constrained models of the form of Eq. (4.6) satisfies the following upper bound *irrespectively of the choice of tensor network*:

$$\text{VC}(f) \leq 2P\log(12|V|),$$

where $|V|$ is the number of vertices in the TN (see Definition A.1). Since quantization of the model weights increases the number of vertices in their tensor network representation, quantized models are characterized by higher upper bounds on the VC-dimension and pseudo-dimension *for the same number of model parameters*. For example, in the non-quantized case, parametrizing the TN as a CPD, TT or TR yields

$$\text{VC}(f) \leq 2P\log(12D),$$

while for the quantized case

$$\text{VC}(f_{\text{quantized}}) \leq 2P\log(12D\log M).$$

Hence, in case of CPD, TT and TR this additional possible model expressiveness comes at *no additional computational costs per iteration* when training with gradient descent (Theorems 4.2.3 and 4.4.1). Setting $Q = 2$ provides then in this sense an optimal choice for this additional hyperparameter, as it maximizes the upper bound. In the more general case where M_d is not a power of 2, this choice corresponds with the prime factorization of M_d. It should be noted that a higher VC-dimension does not imply better performance on unseen data. However as we will see in Sections 4.5.1 and 4.5.2 quantized models tend to outperform their counterparts in the underparameterized regime where TKMs are typically employed, as the gained expressiveness is put fully to good use and does not result in overfitting.

4.5 NUMERICAL EXPERIMENTS

In all experiments we consider a squared loss $\ell(f(\boldsymbol{x}), y) = |f(\boldsymbol{x}) - y|^2$, scale our inputs to lie in the unit box, and consider Fourier features (Definition 4.3.2) as they notably suffer less from ill-conditioning than polynomials. In all experiments we model the weight tensor as a CPD of rank R. We do not consider other TNs in the numerical experiments for three reasons: first, it has been shown that TTs are more suited to model time-varying functions such as dynamical systems and time series, as opposed to CPD [53]. Second, CPD adds only one hyperparameter

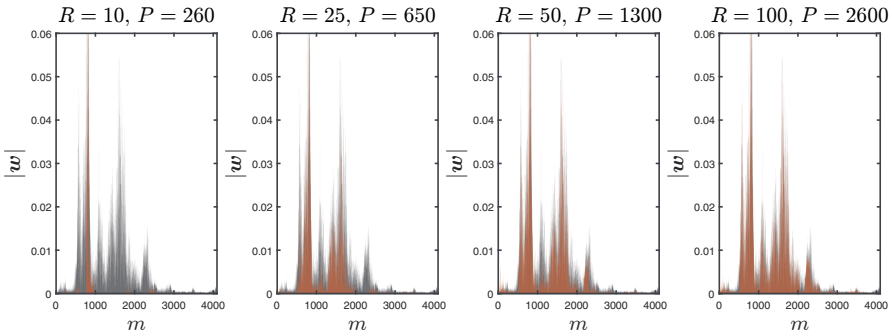


Figure 4.3.: Sound dataset. In red, plot of the magnitude of the quantized Fourier coefficients for different values of R and total number of model parameters P. The magnitude of the full unconstrained Fourier coefficients is shown in black. It can be observed that increasing the CPD rank R recovers the peaks of frequencies with the highest magnitude.

to our model as opposed to D hyperparameters for the TT or TR. Choosing these hyperparameters (TT ranks) is not trivial and can yield models with very different performance for the same total number of model parameters. Third, CPD-based models are invariant to reordering of the features as opposed to TT. We believe that this invariance is very much desired in the context of kernel machines. We solve the ensuing optimization problem using ALS [46]. The source code and data to reproduce all experiments is available at https://github.com/fwesel/QFF.

4.5.1 IMPROVED GENERALIZATION CAPABILITIES

In this experiment we verify the expected quantization to positively affect the generalization capabilities of quantized models. We compare QTKM (our approach) with TKM [8, 25, 27, 29], *Random Fourier Features* (RFF) [14], and with the full, unconstrained model (KRR) which is our baseline, as we are dealing in all cases with squared loss). For our comparison we select eight small *University of California, Irvine* (UCI) datasets [54]. This choice allows us to train KRR by solving its dual optimization problem and thus to implicitly consider $\prod_{d=1}^{D} M_d$ features. For each dataset, we select uniformly at random 80 % of the data for training, and keep the rest for test. We set $Q = 2$ and select the remaining hyperparameters (λ and L) by 3-fold cross validating KRR. We set the number of basis functions $M_d = 16$ uniformly for all d for all models, so that they learn from the

same representation (except for RFF, which is intrinsically random). We then vary the rank R of the non-quantized tensorized model from $R = 1, 2, \ldots, 6$ and train all other models such that their number of model parameters P is at most equal to the ones of the non-quantized model. This means that for TKM $P = R\sum_{d=1}^{D} M_d$, for QTKM $P = 2R\sum_{d=1}^{D} \log_2 M_d$ and for RFF P equals the number of random frequencies. To make sure that TKM and QTKM converge, we run ALS for a very large number of iterations (5000). We repeat the procedure 10 times, and plot the mean and standard deviation of the test *Mean Squared Error* (MSE) in Fig. 4.2.

In Fig. 4.2 one can observe that on all datasets, for the same number of model parameters P and identical features, the generalization performance of QTKM is equivalent or better in term of test MSE. An intuitive explanation for these results is that for equal P, quantization allows to explicitly model correlations within each of the D modes of the feature map, yielding models with increased learning capacity. We notice that while on most datasets the tensor-based approaches recover the performance of KRR, in one case, namely on the yacht dataset, the performance is better than baseline, pointing out at the regularizing effect of the quantized CPD model. Furthermore, on all datasets examined in Fig. 4.2 it can be observed that QTKM switches from underfitting to overfitting regime (first local optimum of the learning curve) before TKM, indicating that indeed its capacity is saturated with fewer model parameters. At that sweet spot, TKM is still underfitting and underperforming with respect to QTKM. For a further increase in model parameters both models exhibit double descent, as can be observed on the qsar, qsar_fish and airfoil datasets. Note that QTKM outperforms TKM in a similar fashion on the training set (Fig. 4 in the appendix), corroborating the presented analysis. In Fig. 4.2 it can also be seen that except on the examined 2-dimensional dataset, both tensor network are consistently outperforming RFF. As we will see in Section 4.5.2, these tensor network-based methods are able to find in a data-dependent way a parsimonious model representation given an exponentially large feature space. This is in contrast to random methods such as RFF, which perform feature selection prior to training and are in this sense oblivious to training data.

4.5.2 REGULARIZING EFFECT OF QUANTIZATION

We would like to gain insight in the regularizing effect caused by modeling the quantized weights as an underparametrized tensor network. For this reason we investigate how the Fourier coefficients are approximated as a function of the CPD rank in a one-dimensional dataset. In order to remove other sources of reg-

ularization, we set $\lambda = 0$. The sound dataset [55] is a one-dimensional time series regression task which comprises 60 000 sampled points of a sound wave. The training set consists of $N = 59\,309$ points, of which the remainder is kept for test. Based on the Nyquist–Shannon sampling theorem, we consider $M = 2^{13} = 8192$ Fourier features, which we quantize with $Q = 2$. We model the signal as a having unit period, hence set $L = 1$. The Fourier coefficients are modeled as a CPD tensor, with rank $R = 10, 25, 50, 100$ in order to yield underparametrized models ($P \ll M$). We plot the magnitude of the Fourier coefficients, which we obtain by minimizing Eq. (4.6) under squared loss.

We compare the magnitude of the quantized weights with the magnitude of the unconstrained model response, obtained by solving Eq. (4.2), in Fig. 4.3. From Fig. 4.3 we can see that for low values of R the quantized kernel machine does not recover the coefficients associated with the lowest frequencies, as a data-independent approach would. Instead, we observe that the coefficients which are recovered for lower ranks, e.g. in case of $R = 10$, are the peaks with the highest magnitude. This is explained by the fact that the additional modes introduced by $Q = 2$-quantization force the underparametrized tensor network to model the nonlinear relation between different basis which under squared-loss maximize the energy of the signal. As the rank increases, the increased model flexibility allows to model more independent nonlinearities. We can see that already for $R = 100$ the two spectra become almost indistinguishable. We report the relative approximation error of the weights and the standardized mean absolute error on the test set in Section D.2.

4.5.3 LARGE-SCALE REGRESSION

In order to showcase and compare out approach with existing literature in the realm of kernel machines, we consider the airline dataset [56], an 8-dimensional dataset which consists of $N = 5\,929\,413$ recordings of commercial airplane flight delays that occurred in 2008 in the USA. As is standard on this dataset [57], we consider a uniform random draw of $2/3N$ for training and keep the remainder for the evaluation of the MSE on the test set and repeat the procedure ten times. In order to capture the complicated nonlinear relation between input and output, we resort to consider $M_d = 64$ Fourier features per dimension, which we quantize with $Q = 2$. For this experiment, we set $L = 10$, $\lambda = 1 \times 10^{-10}$ and run the ALS optimizer for 25 epochs. We train three different QTKMs with $R = 20, 30, 40$.

We present the results in Table 4.1, where we can see that QTKM (our approach) is best at predicting airline delay in term of MSE. Other grid-based approaches,

such as *Variational Fourier Features* (VFF) [6] or *Hilbert–space Gaussian Process* (HGP) [7], are forced to resort to additive kernel modeling and thus disregard higher-order interactions between Fourier features pertaining to different dimension. Other *inducing points*-based methods such as *Stochastic Variational Inference Gaussian Process* (SVIGP) [56] or *Variational Inducing Spherical Harmonics* (VISH) [58] struggle find meaningful features in their exponentially large space. In contrast, QTKM is able to construct R data-driven explanatory variables based on an exponentially large set of Fourier features. When compared with its non-quantized counterpart TKM, we can see that our quantized approach outperforms it with approximately half of its model parameters. Training QTKM on the Intel Core i7-10610U CPU of a Dell Inc. Latitude 7410 laptop with 16 GB of RAM took 6613(40) s for $R = 20$ and took 13 039(114) s for $R = 40$.

Method	M	$P\downarrow$	MSE
VFF [6]	40	320	0.827 ± 0.004
HGP [7]	40	320	0.827 ± 0.005
VISH [58]	660	660	0.834 ± 0.055
SVIGP [56]	1000	1000	0.791 ± 0.005
Falkon [59]	10 000	10 000	0.758 ± 0.005
TKM ($R = 4$)	64	2048	0.789 ± 0.005
TKM ($R = 6$	64	3072	0.773 ± 0.006
TKM ($R = 8$)	64	4096	0.765 ± 0.007
QTKM ($R = 20$)	64	1920	0.764 ± 0.005
QTKM ($R = 30$)	64	2880	0.754 ± 0.005
QTKM ($R = 40$)	64	3840	$\mathbf{0.748 \pm 0.005}$

Table 4.1.: MSE for different kernel machines on the airline dataset with one standard deviation. We report the number of basis functions M per dimensions (in case of random approaches we simply report the total number of basis) and model parameters P. Notice that QTKM is able to parsimoniously predict airline delay with a restricted number of model parameters, achieving state-of-the art performance on this dataset.

4.6 CONCLUSION

We proposed to quantize Fourier and pure-power polynomial features, which allowed us to quantize the model weights in the context of tensor network-constrained

kernel machines. We verified experimentally the theoretically expected increase in model flexibility which allows us to construct more expressive models with the same number of model parameters which learn from the same exact features at the same computational cost per iteration.

Our approach can be readily incorporated in other tensor network-based learning methods which make use of pure-power polynomials or Fourier features.

REFERENCES

[1] F. Wesel and K. Batselier. "Quantized Fourier and Polynomial Features for More Expressive Tensor Network Models". In: *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2024, pp. 1261–1269.

[2] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Nov. 2002.

[3] C. Cortes and V. Vapnik. "Support-Vector Networks". In: *Machine Learning* 20.3 (Sept. 1995), pp. 273–297.

[4] 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". In: *Foundations and Trends® in Machine Learning* 9.4-5 (2016), pp. 249–429.

[5] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[6] J. Hensman, N. Durrande, and A. Solin. "Variational Fourier Features for Gaussian Processes". In: *The Journal of Machine Learning Research* 18.1 (Jan. 2017), pp. 5537–5588.

[7] A. Solin and S. Särkkä. "Hilbert Space Methods for Reduced-Rank Gaussian Process Regression". In: *Statistics and Computing* 30.2 (Mar. 2020), pp. 419–446.

[8] S. Wahls, V. Koivunen, H. V. Poor, and M. Verhaegen. "Learning Multidimensional Fourier Series with Tensor Trains". In: *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. Dec. 2014, pp. 394–398.

[9] R. Karagoz and K. Batselier. "Nonlinear System Identification with Regularized Tensor Network B-splines". In: *Automatica* 122 (Dec. 2020), p. 109300.

[10] N. Pham and R. Pagh. "Fast and Scalable Polynomial Kernels via Explicit Feature Maps". In: *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '13. Association for Computing Machinery, Aug. 2013, pp. 239–247.

[11] D. P. Woodruff. "Sketching as a Tool for Numerical Linear Algebra". In: *Foundations and Trends® in Theoretical Computer Science* 10.1–2 (Oct. 2014), pp. 1–157.

[12] M. Meister, T. Sarlos, and D. Woodruff. "Tight Dimensionality Reduction for Sketching Low Degree Polynomial Kernels". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019.

[13] C. Williams and M. Seeger. “Using the Nyström Method to Speed Up Kernel Machines”. In: *Advances in Neural Information Processing Systems 13*. MIT Press, 2001, pp. 682–688.

[14] A. Rahimi and B. Recht. “Random Features for Large-Scale Kernel Machines”. In: *Proceedings of the 20th International Conference on Neural Information Processing Systems*. Curran Associates Inc., Dec. 2007, pp. 1177–1184.

[15] Q. Le, T. Sarlos, and A. Smola. “Fastfood - Computing Hilbert Space Expansions in Loglinear Time”. In: *Proceedings of the 30th International Conference on Machine Learning*. PMLR, May 2013, pp. 244–252.

[16] T. G. Kolda and B. W. Bader. “Tensor Decompositions and Applications”. In: *SIAM Review* 51.3 (Aug. 2009), pp. 455–500.

[17] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos. “Tensor Decomposition for Signal Processing and Machine Learning”. In: *IEEE Transactions on Signal Processing* 65.13 (July 2017), pp. 3551–3582.

[18] G. Favier and T. Bouilloc. “Parametric Complexity Reduction of Volterra Models Using Tensor Decompositions”. In: *2009 17th European Signal Processing Conference*. Aug. 2009, pp. 2288–2292.

[19] S. Rendle. “Factorization Machines”. In: *2010 IEEE International Conference on Data Mining*. Dec. 2010, pp. 995–1000.

[20] M. Blondel, A. Fujino, N. Ueda, and M. Ishihata. “Higher-Order Factorization Machines”. In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., 2016.

[21] M. Blondel, V. Niculae, T. Otsuka, and N. Ueda. “Multi-Output Polynomial Networks and Factorization Machines”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[22] K. Batselier, Z. Chen, and N. Wong. “Tensor Network Alternating Linear Scheme for MIMO Volterra System Identification”. In: *Automatica* 84 (Oct. 2017), pp. 26–35.

[23] A. Novikov, I. Oseledets, and M. Trofimov. “Exponential Machines”. In: *Bulletin of the Polish Academy of Sciences: Technical Sciences; 2018; 66; No 6 (Special Section on Deep Learning: Theory and Practice); 789-797* (2018).

[24] Z. Chen, K. Batselier, J. A. K. Suykens, and N. Wong. “Parallelized Tensor Train Learning of Polynomial Classifiers”. In: *IEEE Transactions on Neural Networks and Learning Systems* 29.10 (Oct. 2018), pp. 4621–4632.

[25] E. M. Stoudenmire and D. J. Schwab. “Supervised Learning with Tensor Networks”. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. Curran Associates Inc., Dec. 2016, pp. 4806–4814.

[26] S. Efthymiou, J. Hidary, and S. Leichenauer. "TensorNetwork for Machine Learning". In: *arXiv:1906.06329 [cond-mat, physics:physics, stat]* (June 2019). arXiv: 1906.06329 [cond-mat, physics:physics, stat].

[27] N. Kargas and N. D. Sidiropoulos. "Supervised Learning and Canonical Decomposition of Multivariate Functions". In: *IEEE Transactions on Signal Processing* (2021), pp. 1–1.

[28] S. Cheng, L. Wang, and P. Zhang. "Supervised Learning with Projected Entangled Pair States". In: *Physical Review B* 103.12 (Mar. 2021), p. 125117.

[29] F. Wesel and K. Batselier. "Large-Scale Learning with Fourier Features and Tensor Decompositions". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 17543–17554.

[30] B. N. Khoromskij. "O(Dlog N)-Quantics Approximation of N-d Tensors in High-Dimensional Numerical Modeling". In: *Constructive Approximation* 34.2 (Oct. 2011), pp. 257–280.

[31] A. Cichocki. "Era of Big Data Processing: A New Approach via Tensor Networks and Tensor Decompositions". In: *arXiv:1403.2048 [cs]* (Aug. 2014). arXiv: 1403.2048 [cs].

[32] A. Cichocki, A.-H. Phan, Q. Zhao, N. Lee, I. V. Oseledets, M. Sugiyama, and D. Mandic. "Tensor Networks for Dimensionality Reduction and Large-Scale Optimizations. Part 2 Applications and Future Perspectives". In: *Foundations and Trends® in Machine Learning* 9.6 (2017), pp. 249–429. arXiv: 1708.09165.

[33] F. L. Hitchcock. "The Expression of a Tensor or a Polyadic as a Sum of Products". In: *Journal of Mathematics and Physics* 6.1-4 (1927), pp. 164–189.

[34] I. V. Oseledets. "Tensor-Train Decomposition". In: *SIAM Journal on Scientific Computing* 33.5 (Jan. 2011), pp. 2295–2317.

[35] Q. Zhao, G. Zhou, S. Xie, L. Zhang, and A. Cichocki. "Tensor Ring Decomposition". In: *arXiv:1606.05535 [cs]* (June 2016). arXiv: 1606.05535 [cs].

[36] L. R. Tucker. "Implications of Factor Analysis of Three-Way Matrices for Measurement of Change". In: *Problems in measuring change* 15.122-137 (1963), p. 3.

[37] L. R. Tucker. "Some Mathematical Notes on Three-Mode Factor Analysis". In: *Psychometrika* 31.3 (Sept. 1966), pp. 279–311.

[38] W. Hackbusch and S. Kühn. "A New Scheme for the Tensor Representation". In: *Journal of Fourier Analysis and Applications* 15.5 (Oct. 2009), pp. 706–722.

[39] L. Grasedyck. "Hierarchical Singular Value Decomposition of Tensors". In: *SIAM Journal on Matrix Analysis and Applications* 31.4 (Jan. 2010), pp. 2029–2054.

[40] L. De Lathauwer. "Decompositions of a Higher-Order Tensor in Block Terms—Part I: Lemmas for Partitioned Matrices". In: *SIAM Journal on Matrix Analysis and Applications* 30.3 (Jan. 2008), pp. 1022–1032.

[41] L. De Lathauwer. “Decompositions of a Higher-Order Tensor in Block Terms—Part II: Definitions and Uniqueness”. In: *SIAM Journal on Matrix Analysis and Applications* 30.3 (Jan. 2008), pp. 1033–1066.

[42] F. Verstraete and J. I. Cirac. *Renormalization Algorithms for Quantum-Many Body Systems in Two and Higher Dimensions*. July 2004. arXiv: cond-mat/0407066.

[43] G. Evenbly and G. Vidal. “Algorithms for Entanglement Renormalization”. In: *Physical Review B* 79.14 (Apr. 2009), p. 144108.

[44] B. Khavari and G. Rabusseau. “Lower and Upper Bounds on the Pseudo-Dimension of Tensor Network Models”. In: *Advances in Neural Information Processing Systems*. May 2021.

[45] P. Comon, X. Luciani, and A. L. F. de Almeida. “Tensor Decompositions, Alternating Least Squares and Other Tales”. In: *Journal of Chemometrics* 23.7-8 (July 2009), pp. 393–405.

[46] A. Uschmajew. “Local Convergence of the Alternating Least Squares Algorithm for Canonical Tensor Approximation”. In: *SIAM Journal on Matrix Analysis and Applications* 33.2 (Jan. 2012), pp. 639–652.

[47] S. Holtz, T. Rohwedder, and R. Schneider. “The Alternating Linear Scheme for Tensor Optimization in the Tensor Train Format”. In: *SIAM Journal on Scientific Computing* 34.2 (Jan. 2012), A683–A713.

[48] S. R. White. “Density Matrix Formulation for Quantum Renormalization Groups”. In: *Physical Review Letters* 69.19 (Nov. 1992), pp. 2863–2866.

[49] A. Novikov, M. Rakhuba, and I. Oseledets. “Automatic Differentiation for Riemannian Optimization on Low-Rank Matrix and Tensor-Train Manifolds”. In: *arXiv:2103.14974 [cs, math]* (Oct. 2021). arXiv: 2103.14974 [cs, math].

[50] L. De Branges. “The Stone-Weierstrass Theorem”. In: *Proceedings of the American Mathematical Society* 10.5 (1959), pp. 822–824. JSTOR: 2033481.

[51] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press, 2006.

[52] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer New York, 1998.

[53] V. Khrulkov, A. Novikov, and I. Oseledets. “Expressive Power of Recurrent Neural Networks”. In: *International Conference on Learning Representations*. Apr. 2018.

[54] D. Dua and C. Graff. *UCI Machine Learning Repository*. 2017.

[55] A. Wilson and H. Nickisch. “Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP)”. In: *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, June 2015, pp. 1775–1784.

[56] J. Hensman, N. Fusi, and N. D. Lawrence. “Gaussian Processes for Big Data”. In: *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*. UAI’13. AUAI Press, Aug. 2013, pp. 282–290.

[57] Y.-L. K. Samo and S. J. Roberts. "String and Membrane Gaussian Processes". In: *Journal of Machine Learning Research* 17.131 (2016), pp. 1–87.

[58] V. Dutordoir, N. Durrande, and J. Hensman. "Sparse Gaussian Processes with Spherical Harmonic Features". In: *International Conference on Machine Learning*. PMLR, Nov. 2020, pp. 2793–2802.

[59] G. Meanti, L. Carratino, L. Rosasco, and A. Rudi. "Kernel Methods through the Roof: Handling Billions of Points Efficiently". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 14410–14422.

A DEFINITIONS

Definition A.1 (TN [44])**.** Given a graph $G = (V, E, \dim)$ where V is a set of vertices, E is a set of edges and $\dim : E \to \mathbb{N}$ assigns a dimension to each edge, a tensor network assigns a core tensor $\mathscr{C}_v$ to each vertex of the graph, such that $\mathscr{C}_v \in \otimes_{e \in E_v} \mathbb{C}^{\dim(e)}$. Here $E_v = \{e \in E | v \in e\}$ is the set of edges connected to vertex v. The resulting tensor is a tensor in $\otimes_{e \in E \cap V} \mathbb{C}^{\dim(e)}$. The number of parameters of the tensor network is then $P = \sum_{v \in V} \prod_{e \in E_v} \dim(e)$.

B PROOFS

B.1 TENSOR NETWORK KERNEL MACHINE

Theorem B.1. *Suppose* $\mathrm{ten}(\boldsymbol{w}, \underbrace{M, M, \ldots, M}_{D \text{ times}})$ *is a tensor network. Then the dependency on* M *of the computational complexity for the model responses*

$$f(\boldsymbol{x}) = \langle \bigotimes_{d=1}^{D} \boldsymbol{v}_d(x_d), \boldsymbol{w} \rangle_{\mathrm{F}},$$

is $\mathcal{O}(M^t)$*, where* t *is the maximum number of singleton edges per core.*

Proof. Let t be the maximum number of singleton edges per core. Since taking the Frobenius inner product (Eq. (4.4)) involves summing over all singleton edges $\underbrace{M, M, \ldots, M}_{D \text{ times}}$, the required number of floating point operations will be $\mathcal{O}(M^t)$. □

Corollary B.2. *Suppose* $\text{ten}(\boldsymbol{w}, \underbrace{M, M, \ldots, M}_{D \text{ times}})$ *is a tensor network with* $t = 1$ *maximum number of singleton edges per core. Then the dependency on* M *of the computational complexity for the model responses*

$$f(\boldsymbol{x}) = \langle \bigotimes_{d=1}^{D} \boldsymbol{v}_d(x_d), \boldsymbol{w} \rangle_{\text{F}},$$

is of $\mathcal{O}(M)$.

Note that most used tensor networks such as CPD, Tucker, TT/TR, MERA, PEPS have $t = 1$. An example of a tensor network where t can be $t \geq 2$ or higher is hierarchical Tucker. In what follows we derive the computational complexity of the model responses of CPD and TT networks.

Theorem 4.2.3 (Tensorized kernel machine (TKM))**.** *Suppose* $\text{ten}(\boldsymbol{w}, M_1, M_2, \ldots, M_D)$ *is a tensor in* CPD, TT *or* TR *form. Then model responses and associated gradients*

$$f(\boldsymbol{x}, \boldsymbol{w}) = \langle \bigotimes_{d=1}^{D} \boldsymbol{v}_d(x_d), \boldsymbol{w} \rangle_{\text{F}},$$

can be computed in $\mathcal{O}(P)$ *instead of* $\mathcal{O}(\prod_{d=1}^{D} M_d)$*, where* $P = DMR$ *in case of* CPD*, and* $P = DMR^2$ *in case of* TT *or* TR.

Proof. Let $\text{ten}(\boldsymbol{w}, M_1, M_2, \ldots, M_D)$ be a tensor in CPD form. Then

$$\begin{aligned}
f(\boldsymbol{x}) &= \langle \bigotimes_{d=1}^{D} \boldsymbol{v}_d(x_d), \boldsymbol{w} \rangle_{\text{F}} \\
&= \sum_{m_1=0}^{M_1-1} \cdots \sum_{m_D=0}^{M_D-1} \prod_{d=1}^{D} v_{d m_d} \sum_{r=0}^{R-1} \prod_{d=1}^{D} w_{d m_d r} \\
&= \sum_{r=0}^{R-1} \sum_{m_1=0}^{M_1-1} \cdots \sum_{m_D=0}^{M_D-1} \prod_{d=1}^{D} v_{d m_d} w_{d m_d r} \\
&= \sum_{r=0}^{R-1} \prod_{d=1}^{D} \sum_{m_d=0}^{M_d-1} v_{d m_d} w_{d m_d r}.
\end{aligned}$$

Gradients can be computed efficiently by caching $\prod_{d=1}^{D} \sum_{m_d=0}^{M_d-1} v_{d m_d} w_{d r_{d-1} m_d r_d}$, $r = 1, \ldots, R$. Hence the computational complexity of the model responses and associ-

ated gradients is of $\mathcal{O}(DMR)$. Now let $\text{ten}(\boldsymbol{w}, M_1, M_2, \ldots, M_D)$ be a tensor in TT/TR form. Then

$$
\begin{aligned}
f(\boldsymbol{x}) &= \langle \bigotimes_{d=1}^{D} \boldsymbol{v}_d(x_d), \boldsymbol{w} \rangle_{\mathrm{F}} \\
&= \sum_{m_1=0}^{M_1-1} \cdots \sum_{m_D=0}^{M_D-1} \prod_{d=1}^{D} v_{d\,m_d} \sum_{r_1=0}^{R_1-1} \cdots \sum_{r_D=0}^{R_D-1} \prod_{d=1}^{D} w_{d\,r_{d-1}m_d r_d} \\
&= \sum_{r_1=0}^{R_1-1} \cdots \sum_{r_D=0}^{R_D-1} \sum_{m_1=0}^{M_1-1} \cdots \sum_{m_d=0}^{M_D-1} \prod_{d=1}^{D} v_{d\,m_d} w_{d\,r_{d-1}m_d r_d} \\
&= \sum_{r_1=0}^{R_1-1} \cdots \sum_{r_D=0}^{R_D-1} \prod_{d=1}^{D} \sum_{m_d=0}^{M_d-1} v_{d\,m_d} w_{d\,r_{d-1}m_d r_d},
\end{aligned}
$$

which is a sequence of matrix-matrix multiplications. Gradients can be computed efficiently by caching $\sum_{m_d=0}^{M_d-1} v_{d\,m_d} w_{d\,r_{d-1}m_d r_d}$, $r_d = 1,\ldots,R_d$, $d = 1,\ldots,D$. Hence the computational complexity of the model responses and associated gradients is of $\mathcal{O}(DMR^2)$, where $M = \max(M_1, M_2, \ldots, M_D)$ i.e. $\mathcal{O}(P)$ for both CPD and TT/TR. □

B.2 QUANTIZED TENSOR NETWORK KERNEL MACHINE

Theorem B.3. *Suppose* $\text{ten}(\boldsymbol{w}, \underbrace{Q, Q, \ldots, Q}_{DK \text{ times}})$ *is a tensor network. Then the dependency on* M *on the computational complexity of model responses*

$$
f(\boldsymbol{x}) = \langle \bigotimes_{d=1}^{D} \bigotimes_{k=1}^{K} \boldsymbol{s}_{d,k}(x_d), \boldsymbol{w} \rangle_{\mathrm{F}},
$$

is of $\mathcal{O}(M^{\frac{t}{\log Q}} \log M)$*, where* t *is the maximum number of singleton edges per core.*

Proof. Let Q be chosen such that $K = \log_Q M$. Let t be the maximum number of singleton edges per core. Taking the Frobenius inner product (Eq. (4.4)) involves summing over all singleton edges $\underbrace{Q, Q, \ldots, Q}_{D\log_Q M \text{ times}}$. Since $Q = \frac{1}{\log_Q M}$, the required number of floating point operations will be $\mathcal{O}(Q^t \log_Q M) = \mathcal{O}(M^{\frac{s}{\log Q}} \log M)$. □

Corollary B.4. *Suppose* $\text{ten}(\boldsymbol{w}, \underbrace{Q, Q, \ldots, Q}_{DK \text{ times}})$ *is a tensor network with* $t = 1$ *maximum number of singleton edges per core. Then the dependency on* M *on the computational complexity of model responses*

$$f(\boldsymbol{x}) = \langle \bigotimes_{d=1}^{D} \bigotimes_{k=1}^{K_d} \boldsymbol{s}_{d,k}(x_d), \boldsymbol{w} \rangle_{\text{F}},$$

is of $\mathcal{O}(\log M)$.

Theorem 4.4.1 (Quantized tensorized kernel machine (QTKM))**.** *Consider pure-power and Fourier feature maps factorized as in Corollary 4.3.5 and Corollary 4.3.6 and suppose* $\text{ten}(\boldsymbol{w}, Q, Q, \ldots, Q)$ *is a tensor in* CPD, TT *or* TR *form. Then by Theorem 4.2.3, model responses and associated gradients*

$$f_{quantized}(\boldsymbol{x}, \boldsymbol{w}) = \langle \bigotimes_{d=1}^{D} \bigotimes_{k=1}^{K_d} \boldsymbol{s}_{d,k}(x_d), \boldsymbol{w} \rangle_{\text{F}},$$

can be computed in $\mathcal{O}(P)$ *instead of* $\mathcal{O}(\prod_{d=1}^{D} M_d)$*, where* $P = KDQR$ *in case of* CPD*, and* $P = KDQR^2$ *in case of* TT *or* TR.

Proof. The proof follows from the proof of Theorem 4.2.3. Since instead of summing R times over $M_1, M_2, \ldots, M_D$ we are summing R times over $\underbrace{Q, Q, \ldots, Q}_{DK \text{ times}}$, a model response can be evaluated in $QKDR$ floating point operations for CPD and $QKDR^2$ floating point operations for TT. Since Q is a constant which does not dependent on M and $K = \log_Q M$, we have that the computational complexities are respectively $\mathcal{O}(\log MDR)$ and $\mathcal{O}(\log MDR^2)$ for CPD and TT/TR, where $K = \log M = \max(\log M_1, \log M_2, \ldots, \log M_D)$, i.e. $\mathcal{O}(P)$ for both CPD and TT/TR. □

C Faster Multi-Convex Optimization Algorithms

Quantized features allow to speedup Eq. (4.6) for a large class of multi-convex solvers such as ALS [16, 45, 46, 47], the DMRG [48] and Riemannian optimization [23, 49]. These solvers exploit the multi-linearity of tensor networks in order to express the empirical risk as a function of only one core of the weight tensor in tensor network form per iteration, also known as sub-problem. After solving

the ensuing optimization sub-problem, this procedure is repeated for for the remaining cores, defining one epoch. The whole procedure is then repeated until convergence. When a convex quadratic loss function is used, computational benefits associated with quantization arise as it enables to solve a series of quadratic problems exactly. This is common practice in literature, see for instance Wahls *et al.* [8], Novikov, Oseledets, and Trofimov [23], Chen *et al.* [24], and Wesel and Batselier [29].

In the exemplifying case of CPD, TT and tensor ring, for a fixed number of model parameters P, quantization allows to solve each sub-problem at a reduced computational cost of $\mathcal{O}(P^2/D^2)$ compared to a cost of $\mathcal{O}(P^2/D^2(\log M)^2)$. This yields a sub-problem complexity which is *independent* of M. A similar reduction follows for other one-layered networks. Quantifying the computational gains for other structures of tensor networks is less straightforward.

D NUMERICAL EXPERIMENTS

D.1 IMPROVED GENERALIZATION CAPABILITIES

We report in Fig. 4 the training error on the examined datasets in Section 4.5.1. As one can observe, QTKM outperforms TKM in terms of training error (Fig. 4) and test error (Fig. 4.2).

D.2 REGULARIZING EFFECT OF QUANTIZATION

In Table 2 we repeat the number of model parameters $P = 2\log_2 MR$, the compression ratio of the quantized model weights M/P, as well as the relative approximation error of the weights $||\boldsymbol{w}-\boldsymbol{w}_{\text{CPD}}||/||\boldsymbol{w}||$ and the standardized *Mean Absolute Error* (MAE) of the reconstruction error on the test set as a function of the CPD rank.

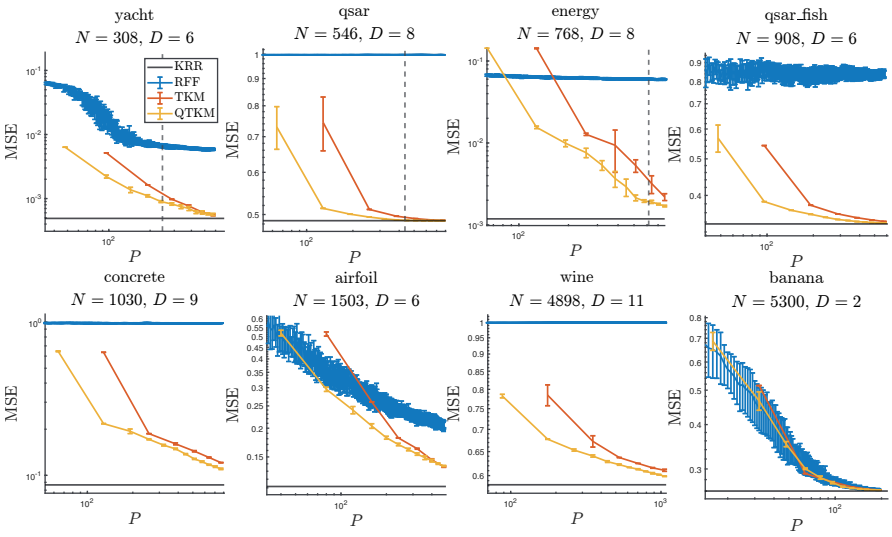


Figure 4.: Plots of the train MSE as a function of the number of model parameters P, for different real-life datasets. In blue, random Fourier features [14], in red tensorized kernel machines with Fourier features [8, 25, 27, 29], in yellow quantized kernel machines with Fourier features, with quantization $Q = 2$. The gray horizontal full line is the full unconstrained optimization problem, which corresponds to KRR. The grey vertical dotted line is set at $P = N$. It can be seen that for $P < N$ case, quantization allows to achieve better performance with respect to the non-quantized case on the training set (this figure) *and* on the test set (Fig. 4.2).

R	P	M/P	$\|\|\boldsymbol{w}-\boldsymbol{w}_{\text{CPD}}\|\|/\|\|\boldsymbol{w}\|\|$	MAE
10	260	31.5	0.841	0.579
25	650	12.6	0.712	0.571
50	1300	6.3	0.528	0.451
100	2600	3.1	0.310	0.182

Table 2.: Model parameters, compression ratio and relative approximation error of the weights, and standardized mean absolute error on the test data as a function of the CPD rank.

5

TENSOR NETWORK-CONSTRAINED KERNEL MACHINES AS GAUSSIAN PROCESSES

In this paper we establish a new connection between Tensor Network *(*TN*)-constrained kernel machines and* Gaussian Processes *(*GPs*). We prove that the outputs of* Canonical Polyadic Decomposition *(*CPD*) and* Tensor Train *(*TT*)-constrained kernel machines converge in the limit of large ranks to the same product kernel* GP *which we fully characterize, when specifying appropriate i.i.d. priors across their components. We show that* TT*-constrained models convergence faster to the* GP *compared to their* CPD *counterparts for the same number of model parameters. The convergence to the* GP *occurs as the ranks tend to infinity, as opposed to the standard approach which introduces* TNs *as an additional constraint on the posterior. This implies that the newly established priors allow the models to learn features more freely as they necessitate infinitely more parameters to converge to a* GP*, which is characterized by a fixed learning representation and thus no feature learning. As a consequence, the newly derived priors yield more flexible models which can better fit the data, albeit at increased risk of overfitting. We demonstrate these considerations by means of two numerical experiments.*

This chapter has been published as:
F. Wesel and K. Batselier. "Tensor Network-Constrained Kernel Machines as Gaussian Processes". In: *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics.* PMLR, Apr. 2025, pp. 2161–2169

5.1 INTRODUCTION

Tensor Networks [TNs, 2, 3, 4], a tool from multilinear algebra, extend the concept of rank from matrices to tensors allowing to represent an exponentially large object with a linear number of parameters. As such, TNs have been used to reduce the storage and computational complexities by compressing the model parameters of a range of models such as *Deep Neural Networks* (DNNs) [5], *Convolutional Neural Networks* (CNNs) [6, 7], *Recurrent Neural Networks* (RNNs) [8], *Graph Neural Networks* (GNNs) [9] and transformers [10].

Similarly, TNs have also found application in the context of kernel machines for supervised learning [11, 12, 13] as an additional constraint on the model posterior. Such models learn a low-rank nonlinear data-dependent representation from an exponentially large number of fixed features by means of a restricted number of parameters, and are as such characterized by an implicit source of regularization. Furthermore, storage and the evaluation of the model and its gradient require a linear complexity in the number of parameters, rendering these methods promising candidates for applications requiring both good generalization and scalability. Because of their intrinsic nonlinearity which prohibits closed-form Bayesian inference, the training of these models is typically accomplished in the *Maximum Likelihood* (ML) and *Maximum A Posteriori* (MAP) framework where the low-rank TN assumption is introduced as an additional nonlinear constraint in the optimization problem. In this setting the ensuing estimator recovers the solution that would be obtained without the TN constraint when the low-rank assumption is satisfied *exactly*, i.e. for *finite rank*.

In contrast, *Gaussian Processes* [GPs, 14] are an established framework for modeling functions which naturally allows the practitioner to incorporate prior knowledge. When considering i.i.d. observations and Gaussian likelihoods, GPs allow for the determination of the posterior in *closed-form*, which considerably facilitates tasks such as inference, sampling and the construction of sparse approximations among many others.

In this paper we establish a direct connection between TN-constrained kernel machines and GPs, thus solving an open question considered by Wesel and Batselier [13, 15]. We prove that the outputs of *Canonical Polyadic Decomposition* (CPD) and *Tensor Train* (TT)-constrained kernel machines converge in the limit of large ranks to the same product kernel GP which we fully characterize, when specifying appropriate i.i.d. priors across their components. This result allows us to derive that when placing such priors on their parameters, TT-constrained models achieve faster convergence to the GP compared to their CPD counterparts for the same number of model parameters. The convergence to the GP occurs

as the ranks tend to infinity, as opposed to the standard approach which introduces TNs as an additional constraint on the model posterior. This implies that the newly established priors allow the models to more autonomously learn features as they necessitate infinitely more parameters to converge to a GP, which is characterized by a *fixed* learning representation and thus no feature learning. Consequently, the newly derived priors yield more flexible models which can better fit the data and have a higher chance of overfitting. We showcase the convergence properties of both newly derived priors and their effect on MAP estimation by means of numerical experiments.

The rest of this paper is organized as follows. In Section 5.2 we provide a brief introduction to GPs and their approximations, TNs and TN-constrained kernel machines. In Section 5.3 we present our main result, i.e. the equivalence in the limit between TN-constrained kernel machines and product kernel GPs. In Section 5.4 we showcase the different convergence rates to the GP of CPD and TT and their effect on MAP estimation. We then provide a review of related work (Section 5.5) and a conclusion (Section 5.6). We discuss the notation used throughout the paper in Section A.1.

5.2 BACKGROUND

GPs are a collection of random variables such that any finite subset has a joint Gaussian distribution [14]. They provide a flexible formalism for modeling functions which inherently allows for the incorporation of prior knowledge and the production of uncertainty estimates in the form of a predictive distribution. More specifically, a GP is fully specified by a mean function $\mu(\cdot) : \mathbb{R}^D \to \mathbb{R}$, typically chosen as zero, and a covariance or kernel function $k(\cdot,\cdot) : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$:

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}), k(\boldsymbol{x},\cdot)).$$

Given a labeled dataset $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ consisting of N inputs $\boldsymbol{x}_n \in \mathbb{R}^D$ and i.i.d. noisy observations $y_n \in \mathbb{R}$, GPs can be used for modeling the underlying function f in classification or regression tasks by specifying a likelihood function. For example the likelihood

$$p(y_n \mid f(\boldsymbol{x}_n)) = \mathcal{N}(f(\boldsymbol{x}_n), \sigma^2), \tag{5.1}$$

yields a GP posterior which can be obtained in closed-form by conditioning the prior GP on the noisy observations. Calculating the mean and covariance of such

a posterior crucially requires instantiating and formally inverting the kernel matrix $\boldsymbol{K}$ such that $k_{n,m} := k(\boldsymbol{x}_n, \boldsymbol{x}_m)$. These operations respectively incur a computational cost of $\mathcal{O}(N^2)$ and $\mathcal{O}(N^3)$ and therefore prohibit the processing of large-sampled datasets.

5.2.1 BASIS FUNCTION APPROXIMATION

Aside from variational inference [16, 17] and iterative methods [18], a common approach in literature to circumvent the $\mathcal{O}(N^3)$ computational bottleneck is to project the GP onto a finite number of *Basis Functions* (BFs) [e.g., 14, 19]. This is achieved by approximating the kernel as

$$k(\boldsymbol{x}, \boldsymbol{x}') \approx \boldsymbol{\varphi}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{\Lambda}\, \boldsymbol{\varphi}(\boldsymbol{x}'), \tag{5.2}$$

where here $\boldsymbol{\varphi}(\boldsymbol{x}) : \mathbb{R}^D \to \mathbb{R}^M$ are (nonlinear) basis functions and $\boldsymbol{\Lambda} \in \mathbb{R}^{M \times M}$ are the BF weights. This finite-dimensional kernel approximation ensures a *degenerate* kernel [14], as it characterized by a finite number of non-zero eigenvalues. Its associated GP can be characterized equivalently as

$$f(\boldsymbol{x}) = \left\langle \boldsymbol{\varphi}(\boldsymbol{x}), \boldsymbol{w} \right\rangle, \quad \boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Lambda}), \tag{5.3}$$

wherein $\boldsymbol{w} \in \mathbb{R}^M$ are the model weights and $\boldsymbol{\Lambda}$ is the associated prior covariance. Once more considering a Gaussian likelihood (Eq. (5.1)) yields a closed-form posterior GP whose mean and covariance require only the posterior covariance matrix $(\sum_{n=1}^{N} \boldsymbol{\varphi}(\boldsymbol{x}_n)\boldsymbol{\varphi}(\boldsymbol{x}_n)^{\mathrm{T}} + \boldsymbol{\Lambda}^{-1})^{-1}$. This yields a computational complexity of $\mathcal{O}(NM^2 + M^3)$, which allows to tackle large-sampled data when $N \gg M$.

5.2.2 PRODUCT KERNELS

In the remainder of this paper we consider GPs with product kernels.

Definition 5.2.1 (Product kernel [14])**.** A kernel $k(\boldsymbol{x}, \boldsymbol{x}')$ is a product kernel if

$$k(\boldsymbol{x}, \boldsymbol{x}') = \prod_{q=1}^{Q} k^{(q)}(\boldsymbol{x}, \boldsymbol{x}'), \tag{5.4}$$

where each $k^{(q)}(\cdot, \cdot) : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is a valid kernel.

While many commonly used kernels are product kernels e.g. the Gaussian kernel and the polynomial kernel, product kernels provide a straightforward strategy

to extend one-dimensional kernels to the higher-dimensional case [14, 20]. The basis functions and prior covariance of product kernels can then be determined based on the basis function expansion of their constituents as follows.

Lemma 5.2.2 (Basis functions and prior covariances of product kernels)**.** *Consider the product kernel of Definition 5.2.1. Denote the basis functions and prior covariance of each factor* $k^{(q)}(\boldsymbol{x}, \boldsymbol{x}')$ *as* $\boldsymbol{\varphi}^{(q)}(\boldsymbol{x}) \in \mathbb{R}^{M_q}$ *and* $\boldsymbol{\Lambda}^{(q)} \in \mathbb{R}^{M_q \times M_q}$ *respectively, then the basis functions and prior covariance of* $k(\boldsymbol{x}, \boldsymbol{x}')$ *are*

$$\boldsymbol{\varphi}(\boldsymbol{x}) = \otimes_{q=1}^{Q} \boldsymbol{\varphi}^{(q)}(\boldsymbol{x}), \tag{5.5}$$

and

$$\boldsymbol{\Lambda} = \otimes_{q=1}^{Q} \boldsymbol{\Lambda}^{(q)}, \tag{5.6}$$

The inherent challenge in this approach stems from the exponential increase of the number of basis functions M and thus of model parameters as a function of the dimensionality of the input data, thereby restricting their utility to low-dimensional datasets.

Such structure arises for instance when dealing with Mercer expansions of product kernels, in the structured kernel interpolation framework [18, 21] variational Fourier features framework [20] and Hilbert-GP framework [22]. Alternative important approximation strategies which avoid this exponential scaling are random features [23, 24], inducing features [17, 19, 25, 26, 27, 28] and additive GPs [29, 30] which circumvent the outlined computational issue. All those approaches can be interpreted as projecting the GP on a set of BFs.

The performance of these methods however tends to deteriorate in higher dimensions, as they need to cover an exponentially large domain with a linear number of random samples or inducing points. These issues are some of the computational aspects of the *curse of dimensionality*, which renders it difficult to operate in high-dimensional feature spaces [31].

5.2.3 TENSOR NETWORKS

A recent alternative approach to remedy said curse of dimensionality affecting the exponentially increasing weights of the linear model in Eq. (5.3) consists in constraining the tensorized models weights $\text{ten}(\boldsymbol{w})$ to be a low-rank tensor network. TNs express a Q-dimensional tensor $\mathcal{W}$ as a multi-linear function of a number of *core* tensors. Two commonly used TNs are the CPD and TT, defined as follows.

Definition 5.2.3 (CPD [32])**.** A Q-dimensional tensor $\mathcal{W} \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_Q}$ has a rank-R CPD if

$$w_{m_1,m_2,\ldots,m_Q} = \sum_{r=1}^{R} \prod_{q=1}^{Q} w^{(q)}{}_{m_q,r}. \tag{5.7}$$

The cores of a CPD are the matrices $\boldsymbol{W}^{(q)} \in \mathbb{R}^{M_q \times R}$. Since a CPD tensor can be expressed solely in terms of its cores, its storage requires $P_{\texttt{CPD}} = R\sum_{q=1}^{Q} M_q$ parameters as opposed to $\prod_{q=1}^{Q} M_q$.

Definition 5.2.4 (TT [33])**.** A Q-dimensional tensor $\mathcal{W} \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_Q}$ admits a rank-$(R_0 \coloneqq 1, R_1, \ldots, R_Q \coloneqq 1)$ tensor train if

$$w_{m_1,m_2,\ldots,m_Q} = \sum_{r_0=1}^{R_0} \sum_{r_1=1}^{R_1} \cdots \sum_{r_Q=1}^{R_Q} \prod_{q=1}^{Q} w^{(q)}{}_{r_{q-1},m_q,r_q}. \tag{5.8}$$

The cores of a tensor train are Q 3-dimensional tensors $\mathcal{W}^{(q)} \in \mathbb{R}^{R_{q-1} \times M_q \times R_q}$ which yield $P_{\texttt{TT}} = \sum_{q=1}^{Q} R_{Q-1} M_Q R_Q$ parameters.

In the following we denote by $\texttt{TN}(\mathcal{W})$ a tensor which admits a general TN format, by $\texttt{CPD}(\mathcal{W})$ a tensor which is a rank-R CPD form and by $\texttt{TT}(\mathcal{W})$ a tensor in rank-$(R_0 \coloneqq 1, R_1, \ldots, R_Q \coloneqq 1)$ TT form. Lastly, we denote by $\texttt{R}_1(\mathcal{W})$ a tensor which is rank-1 CPD form or rank-$(1, 1, \ldots, 1)$ TT, as both are equivalent. Importantly, we refer to a tensor in general TN format $\texttt{TN}(\mathcal{W}) \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_Q}$ as *underparametrized* if its rank hyperparameters, e.g. R in case of CPD, are chosen such that its storage cost is less than $\prod_{q=1}^{Q} M_q$. This is crucial in order to obtain storage and computational benefits.

5.2.4 TENSOR NETWORK-CONSTRAINED KERNEL MACHINES

TNs have been used to reduce the number of model parameters in kernel machines (Eq. (5.3)) by tensorizing the BFs $\boldsymbol{\varphi}(\cdot)$ and model weights $\boldsymbol{w}$ and by constraining both to be underparameterized TNs. This approach lays its foundations on the fact that the Frobenius inner product of a tensorized vector is isometric with respect to the Euclidean inner product, i.e.

$$f(\boldsymbol{x}) = \left\langle \boldsymbol{\varphi}(\boldsymbol{x}), \boldsymbol{w} \right\rangle = \left\langle \text{ten}\left(\boldsymbol{\varphi}(\boldsymbol{x})\right), \text{ten}\left(\boldsymbol{w}\right) \right\rangle_{\text{F}}. \tag{5.9}$$

This isometry allows then to constrain the BFs and the model weights to be an underparameterized TN. Since product kernels yield an expansion in terms of

Kronecker-product BFs (Eq. (5.5)), they are a rank-1 TN by definition after tensorization. Embedding these relations yields an approximate model

$$f(\boldsymbol{x}) \approx f_{\mathtt{TN}}(\boldsymbol{x}) := \langle \mathtt{R}_1(\mathrm{ten}\left(\boldsymbol{\varphi}(\boldsymbol{x})\right)), \mathtt{TN}(\mathrm{ten}(\boldsymbol{w}))\rangle_{\mathrm{F}}, \tag{5.10}$$

characterized by lower storage and computational complexities. This approach has been proposed mostly for weights modeled as CPD [13, 15, 34] or TT [11, 12, 35, 36, 37] as they arguably introduce fewer rank hyperparameters (only one in case of CPD) and thus are in practice easier to work with compared to other TNs such as the *Multi-scale Entanglement Renormalization Ansatz* (MERA) [38].

We define such models as we will need them in detail in the next section, where we present our main contribution.

Definition 5.2.5 (CPD-constrained kernel machine)**.** The CPD-constrained kernel machine is defined as

$$f_{\mathtt{CPD}}(\boldsymbol{x}) := \langle \mathtt{R}_1(\mathrm{ten}\left(\boldsymbol{\varphi}(\boldsymbol{x})\right)), \mathtt{CPD}(\mathrm{ten}(\boldsymbol{w}))\rangle_{\mathrm{F}} \tag{5.11}$$

$$= \sum_{r=1}^{R} h_r(\boldsymbol{x}), \tag{5.12}$$

where the intermediate variables $h_r \in \mathbb{R}$ are defined as

$$h_r(\boldsymbol{x}) := \prod_{q=1}^{Q} \boldsymbol{\varphi}^{(q)}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{w}^{(q)}{}_{:,r}. \tag{5.13}$$

Similarly, we provide a definition for the TT-constrained kernel machine.

Definition 5.2.6 (TT-constrained kernel machine)**.** The TT-constrained kernel machine is defined as

$$f_{\mathtt{TT}}(\boldsymbol{x}) := \langle \mathtt{R}_1(\mathrm{ten}\left(\boldsymbol{\varphi}(\boldsymbol{x})\right)), \mathtt{TT}(\mathrm{ten}(\boldsymbol{w}))\rangle_{\mathrm{F}} \tag{5.14}$$

$$= \sum_{r_Q=1}^{R_Q} \sum_{r_{Q-1}=1}^{R_{Q-1}} \cdots \sum_{r_0=1}^{R_0} \prod_{q=1}^{Q} z^{(q)}_{r_{q-1},r_q}(\boldsymbol{x}), \tag{5.15}$$

where the intermediate variables $\boldsymbol{Z}^{(q)} \in \mathbb{R}^{R_{q-1} \times R_q}$ are defined element-wise as

$$z^{(q)}_{r_{q-1},r_q}(\boldsymbol{x}) := \sum_{m_q=1}^{M_q} \varphi^{(q)}_{m_q}(\boldsymbol{x})\, w^{(q)}_{r_{q-1},m_q,r_q}. \tag{5.16}$$

Evaluating CPD and TT-constrained kernel machines (Eq. (5.11), Eq. (5.14)) and

their gradients can be accomplished with $\mathcal{O}(P_{\text{CPD}})$ and $\mathcal{O}(P_{\text{TT}})$ computations, respectively. This allows the practitioner to tune the rank hyperparameter in order to achieve a model that fits in the computational budget at hand and that learns from the specified BFs.

From an optimization point-of-view, models in the form of Eq. (5.10) are considered in the ML [11, 36] and in the MAP setting [12, 13, 15, 34, 35, 37] and in the context of GP variational inference [39] where TTs are used to parameterize the variational distribution. In all these scenarios, TNs appear as *an additional constraint to the optimization problem*, and do hence not *define* a probabilistic model but merely approximate the chosen estimator (ML, MAP, etc.).

In the following section we present the main contribution of our work: we show how when placing i.i.d. priors on the cores of these CPD and TT-constrained model, they converge to a GP which we fully characterize. As we will see, beside connecting the TN-constrained kernel machines with GPs, this probabilistic characterization defines a different and less stringent type of regularization for such models.

5

5.3 TN-CONSTRAINED KERNEL MACHINES AS GPS

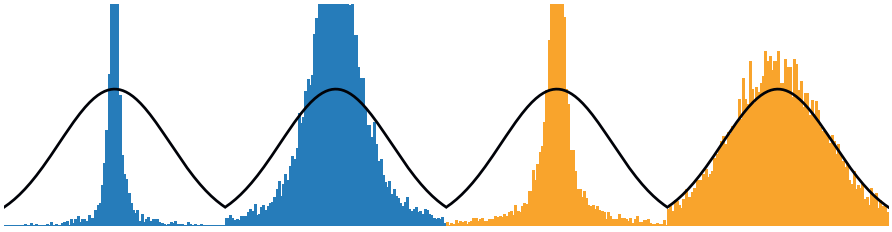


Figure 5.1.: Histograms of the empirical *Probability Density Function* (PDF) of CPD (blue) and TT (orange) models specified in Theorems 5.3.1 and 5.3.2 evaluated at a random point as a function of model parameters P for $D = 16$. The black line is the PDF of the GP. Notice how TT converges faster to the GP for the same number of model parameters P.

We commence to outline the correspondence between TN-constrained kernel machine and GPs, which makes use of the *Central Limit Theorem* (CLT). We begin by elucidating the simplest case, i.e. the CPD.

Theorem 5.3.1 (CPD-constrained kernel machine as GP)**.** *Consider the* CPD*-constrained kernel machine*

$$f_{\texttt{CPD}}(\boldsymbol{x}) := \langle \mathtt{R}_1(\mathrm{ten}\left(\boldsymbol{\varphi}(\boldsymbol{x})\right)), \texttt{CPD}(\mathrm{ten}\,(\boldsymbol{w}))\rangle_{\mathrm{F}}.$$

If each of the R columns $\boldsymbol{w}^{(q)}{}_{:,r} \in \mathbb{R}^{M_q}$ *of each* CPD *core is an i.i.d. random variable such that*

$$\mathbb{E}\left[\boldsymbol{w}^{(q)}_{:,r}\right] = \mathbf{0},$$
$$\mathbb{E}\left[\boldsymbol{w}^{(q)}_{:,r}\boldsymbol{w}^{(q)\mathrm{T}}_{:,r}\right] = R^{-\frac{1}{Q}}\boldsymbol{\Lambda}^{(q)},$$

then $f_{\texttt{CPD}}(\boldsymbol{x})$ *converges in distribution as* $R \to \infty$ *to the* GP

$$f_{\texttt{CPD}}(\boldsymbol{x}) \sim \mathcal{GP}\left(0, \prod_{q=1}^{Q} \boldsymbol{\varphi}^{(q)}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\Lambda}^{(q)}\boldsymbol{\varphi}^{(q)}(\cdot)\right).$$

Proof. See Section B.1. □

A similar result can be constructed for the TT case.

Theorem 5.3.2 (TT-constrained kernel machine as GP)**.** *Consider the* TT*-constrained kernel machine*

$$f_{\texttt{TT}}(\boldsymbol{x}) := \langle \mathtt{R}_1(\mathrm{ten}\left(\boldsymbol{\varphi}(\boldsymbol{x})\right)), \texttt{TT}(\mathrm{ten}\,(\boldsymbol{w}))\rangle_{\mathrm{F}}$$

If each of the $R_{q-1}R_q$ *fibers* $\boldsymbol{W}^{(q)}{}_{r_{q-1},:,r_q} \in \mathbb{R}^{M_q}$ *of each* TT *core is an i.i.d. random variable such that*

$$\mathbb{E}\left[\boldsymbol{W}^{(q)}_{r_{q-1},:,r_q}\right] = \mathbf{0},$$
$$\mathbb{E}\left[\boldsymbol{W}^{(q)}_{r_{q-1},:,r_q}\boldsymbol{W}^{(q)\mathrm{T}}_{r_{q-1},:,r_q}\right] = \frac{1}{\sqrt{R_{q-1}R_q}}\boldsymbol{\Lambda}^{(q)},$$

then $f_{\texttt{TT}}(\boldsymbol{x})$ *converges in distribution as sequentially* $R_1 \to \infty$, $R_2 \to \infty, \ldots, R_{Q-1} \to \infty$ *to the Gaussian process*

$$f_{\texttt{TT}}(\boldsymbol{x}) \sim \mathcal{GP}\left(0, \prod_{q=1}^{Q} \boldsymbol{\varphi}^{(q)}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\Lambda}^{(q)}\boldsymbol{\varphi}^{(q)}(\cdot)\right).$$

Proof. See Section B.1. □

Theorem 5.3.2 guarantees the convergence in distribution of $f_{\texttt{TT}}(\boldsymbol{x})$ to the GP of Eq. (5.3) by taking successive limits of each TT rank. Importantly, the same

convergence results also holds true if the TT ranks grow simultaneously, see Section B.3.

Both Theorems 5.3.1 and 5.3.2 are remarkable, as they imply that a GP which can be defined in terms of a finite number of $\prod_{q=1}^{Q} M_q$ weights $\boldsymbol{w}$ can be also obtained with an *infinite* number of model parameters P using the CPD-constrained model of Definition 5.2.5 or the TT-constrained model of Definition 5.2.6. Furthermore, Theorems 5.3.1 and 5.3.2 suggest that when the priors of Theorems 5.3.1 and 5.3.2 are placed on the model weights, CPD and TT-based models exhibit GP behavior in the overparameterized regime as their ranks tend towards infinity. GP behavior is characterized by a fixed learning representation $\boldsymbol{\varphi}(\cdot)$, which in case of the kernel in Theorems 5.3.1 and 5.3.2 is fully defined by the BFs and is hence data-independent. On the contrary, as we will see, in the finite rank regime both CPD and TT models are able to craft nonlinear features from the provided BFs, learning nonlinear latent patterns in the mapped data.

5.3.1 CONVERGENCE RATE TO THE GP

While both Theorem 5.3.1 and Theorem 5.3.2 guarantee convergence in distribution to the GP of Eq. (5.3), they do so at rates that differ in terms of the number of model parameters. Let us assume, for simplicity, that the number of basis functions is the same along each dimension, i.e., M, and that the $Q-1$ TT ranks equal R. It follows then that the number of CPD model parameters $P_{\text{CPD}} = MQR_{\text{CPD}}$ and the number of TT model parameters $P_{\text{TT}} = M(Q-2)R_{\text{TT}}^2 + 2MR_{\text{TT}} = \mathcal{O}(MQR_{\text{TT}}^2)$. Given the convergence rate of the CLT for the expression in Eq. (5.11) to the GP in Eq. (5.3), denoted as $\mathcal{O}(1/\sqrt{R_{\text{CPD}}})$ with respect to the variable P_{CPD}, we can establish the following corollary by substituting R_{CPD} as a function of P_{CPD}.

Corollary 5.3.3 (Convergence rate for CPD)**.** *Under the conditions of Theorem 5.3.1, the function $f_{\text{CPD}}(\boldsymbol{x})$ converges in distribution to the* GP *defined by Eq.* (5.3)*. The convergence rate is given by:*

$$f_{\text{CPD}}(\boldsymbol{x}) \to \mathcal{O}\left(\left(\frac{MQ}{P_{\text{CPD}}}\right)^{\frac{1}{2}}\right).$$

Due to their hierarchical structure, TT models are a composition of R_{TT}^{Q-1} variables, but can be represented in a quadratic number of model parameters in R_{TT}, since $P_{\text{TT}} = \mathcal{O}(MQR_{\text{TT}}^2)$. Expressing then the CLT convergence rate of $\mathcal{O}(1/\sqrt{R_{\text{TT}}^{Q-1}})$ as a function of P_{TT} yields the following corollary.

Corollary 5.3.4 (Convergence rate for TT)**.** *Under the conditions of Theorem 5.3.2, the function* $f_{\mathtt{TT}}(\boldsymbol{x})$ *converges in distribution to the* GP *defined by Eq.* (5.3)*. The convergence rate is given by:*

$$f_{\mathtt{TT}}(\boldsymbol{x}) \to \mathcal{O}\left(\left(\frac{MQ}{P_{\mathtt{TT}}}\right)^{\frac{Q-1}{4}}\right).$$

Therefore, when dealing with identical models in terms of the number of basis functions (M), product kernel terms (Q), and the number of model parameters ($P_{\mathtt{CPD}} = P_{\mathtt{TT}}$), $f_{\mathtt{TT}}(\boldsymbol{x})$ will converge at a polynomially faster rate than $f_{\mathtt{CPD}}(\boldsymbol{x})$, thus exhibiting GP behavior with a reduced number of model parameters. In particular, based on Corollaries 5.3.3 and 5.3.4 we expect the GP convergence rate of TT models to be faster for $Q \geq 3$.

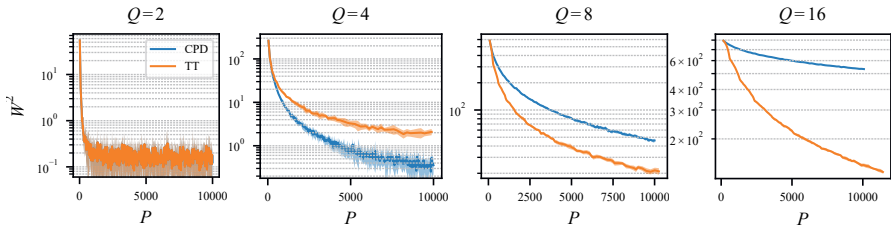


Figure 5.2.: Mean and standard deviation of the Cramér–von Mises statistic W^2 evaluated between the empirical *Cumulative Density Function* (CDF) of CPD and TT models specified in Theorems 5.3.1 and 5.3.2 evaluated at $N = 10$ random points as a function of model parameters P for $Q = 2, 4, 8, 16$. The two models are equivalent for $Q = 2$. Notice how TT converges faster to the GP as the dimensionality of the inputs Q increases.

These insights are relevant for practitioners engaged with TN-constrained kernel machines, as they shed light on the balance between the GP and (deep) neural network behavior inherent in these models. Notably, when using the priors of Theorems 5.3.1 and 5.3.2 CPD and TT-constrained models, akin to shallow and DNNs respectively, have the capacity to craft additional nonlinearities beyond the provided basis functions. This characteristic can result in more expressive model when dealing with a limited number of parameters. However, as the parameter count increases, we expect these models to transition towards GP behavior, characterized by a fixed feature representation and static in comparison.

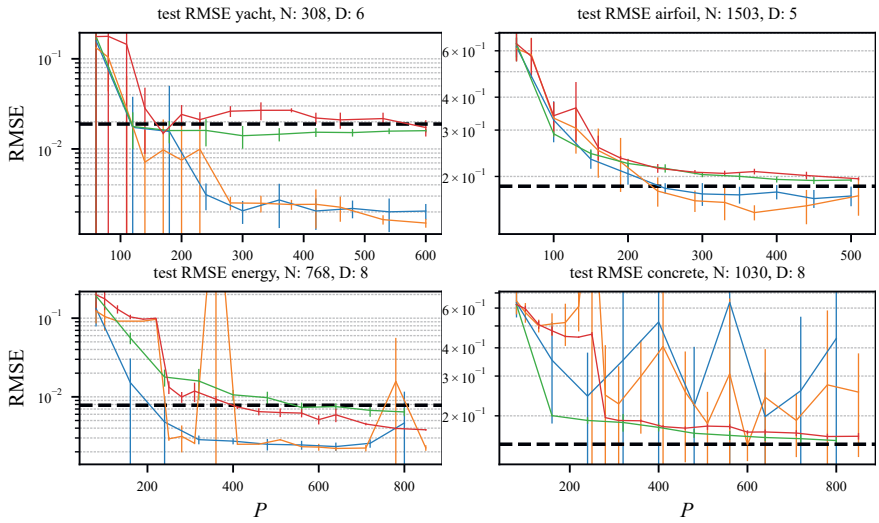


Figure 5.3.: Mean and standard deviation of the test *Root Mean Squared Error* (RMSE) of CPD and TT models for regularization Eqs. (5.17) and (5.18) (green and red curves respectively) as a function of model parameters P as well as their target *Kernel Ridge Regression* (KRR) (dotted line). In the plots, the probabilistic regularization of Eq. (5.18) and its TT counterpart are denoted by a blue and orange line respectively. The dotted line corresponds to the KRR (GP posterior mean) baseline. The proposed regularization which stems from Theorems 5.3.1 and 5.3.2 achieves lower test RMSE with fewer parameters, with the notable exception of the concrete datasets where it leads to overfitting.

5.3.2 CONSEQUENCES FOR MAP ESTIMATION

As discussed in Section 5.2.4, TN-constrained kernel machines are typically trained in the ML or MAP framework by constraining the weights $\boldsymbol{w}$ in the log-likelihood or log-posterior to be a TN. In said MAP context, and e.g. when specifying a normal prior on the model weights $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda})$, the resulting regularization term Ω (log-prior) is approximated by $\Omega_{\mathtt{TN}}$ as

$$\Omega := ||\boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{w}||^2 \approx \Omega_{\mathtt{TN}} := ||\mathtt{TN}(\mathrm{ten}\left(\boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{w}\right))||^2,$$

where $\boldsymbol{\Lambda} = \otimes_{q=1}^{Q} \boldsymbol{\Lambda}^{(q)}$. For example, in case of CPD-constrained models we have

$$\Omega_{\texttt{CPD}} = ||\odot_{q=1}^{Q} \left(\boldsymbol{W}^{(q)\mathrm{T}} \boldsymbol{\Lambda}^{(q)-1} \boldsymbol{W}^{(q)} \right)||^2. \tag{5.17}$$

This form of regularization is considered for TT by Novikov, Oseledets, and Trofimov [12], Wahls *et al.* [35], and Chen *et al.* [37] and for CPD by Wesel and Batselier [13, 15]. It provides a Frobenius norm approximation of the regularization term which recovers the original MAP estimate as the hyperparameters of $\texttt{TN}(\text{ten}(\Lambda \boldsymbol{w}))$ are chosen such that $\texttt{TN}(\text{ten}(\boldsymbol{\Lambda} \boldsymbol{w})) = \text{ten}(\boldsymbol{\Lambda} \boldsymbol{w})$. If we now consider the log-posterior of Theorem 5.3.1 we end up with

$$\Omega_{\texttt{CPD}} \coloneqq R^{\frac{1}{Q}} \sum_{q=1}^{Q} ||\boldsymbol{\Lambda}^{(q)-\frac{1}{2}} \boldsymbol{W}^{(q)}||^2. \tag{5.18}$$

This regularization has been employed without the scaling factor $R^{\frac{1}{Q}}$ and with $\boldsymbol{\Lambda}^{(q)} = \boldsymbol{I} M_q$ in the work of Kargas and Sidiropoulos [34], who may not have been aware of the underlying connection with GPs at that time. Contrary to the regularization $\Omega_{\texttt{TN}}$ in Eq. (5.17), it provides an approximation which recovers the log-prior Ω and thus the MAP, which in combination with a Gaussian likelihood and squared-loss is equivalent to the GP posterior mean *in the limit* of large ranks. These considerations point to the fact that if the practitioner is interested only in a MAP estimate which recovers the GP posterior mean as faithfully as possible given the computational budget at hand, he might be more interested in the established regularization of Eq. (5.17). On the contrary, the choice of Eq. (5.18) in combination with squared-loss recovers the GP MAP *in the limit*, yielding models that can fit the data more closely albeit with an increased possibility of overfitting with respect to the associated GP baseline. Furthermore, sampling the priors in Theorems 5.3.1 and 5.3.2 provide a sensible initial guess for gradient-based optimization which adjusts to the dimensionality of the inputs and the choice of rank hyperparameters [40].

5.4 NUMERICAL EXPERIMENTS

We setup two numerical experiments in order to respectively empirically observe the claims in Theorems 5.3.1 and 5.3.2 by evaluating the convergence to the prior GP in Eq. (5.3), and to evaluate the GP behavior of such models at prediction in the finite rank case. In all experiments we made use of the *Hilbert–space Gaussian*

Process (HGP) [22] BFs which approximate stationary tensor product kernels of the form $\prod_{q=1}^{D} k(x_q, x'_q)$, and opt for $M_q = 10$ basis functions per dimension. We ran all experiments on a Dell Inc. Latitude 7410 laptop computer with 16 GB of RAM. The Python implementation is available at github.com/fwesel/tensorGP.

5.4.1 GP CONVERGENCE

In order to empirically verify the convergence to the GP of Eq. (5.3) we sample 10000 instances of the CPD and TT models specified in Theorems 5.3.1 and 5.3.2 for increasing CPD and TT ranks yielding up to $P = 10000$ model parameters. Since the target distribution is Gaussian with known moments, we record the Cramér–von Mises statistic W^2 [41] which gives a metric of closeness between the target and our sampled empirical CDF. We repeat this for 10 randomly sampled data points and for $Q = 2, 4, 8, 16$ and report the mean and standard deviation of the results in Fig. 5.2. Therein it can be observed that for the same number of model parameters, TT converges more rapidly than CPD as the dimensionality of the inputs grows. Both approaches however need exponentially more parameters to converge at the same rate for increasing dimensionality of the inputs. Note that for $Q = P = 4$ CPD, contrary to what stated in Section 5.3.1 still converges faster due to the approximation made when considering $P_{\text{TT}} = QMR^2$. Histograms of the empirical CDF for one datapoint are shown in Fig. 5.1. This behavior stems from the fact that for a fixed combination of Q, M and P, TT captures an exponential R^{Q-1} range of model interactions in contrast to the R linear interactions exhibited by CPD.

5.4.2 GP BEHAVIOR AT PREDICTION

To investigate whether CPD and TT-constrained kernel machines trained with the priors of Theorems 5.3.1 and 5.3.2 indeed exhibit less GP behavior compared to the standard CPD and TT-constrained prior we tackle four small *University of California, Irvine* (UCI) regression problems [42]. We consider 70 % of the data for training and the remaining for test and train a KRR model (equivalent to the GP posterior mean) on the training data and chose its kernel and regularization hyperparameters by 3-fold cross-validation. With the found KRR hyperparameters we then train two CPD-constrained kernel machines with *Alternating Least-Squares* (ALS) for an increasing number of ranks and thus of parameters, one such model with the standard regularization (Eq. (5.17)) and one with the regularization that follows from Theorem 5.3.1 (Eq. (5.18)). We repeat the same procedure

for TT-constrained kernel machines. We report the RMSE on test data in Fig. 5.3 and on train in Fig. 4 in the appendix. In Fig. 5.3 one can observe that on all datasets the predictions on unseen data of both CPD and TT models trained with the standard regularization (green and red curves respectively) converge to the KRR baseline (dotted line) for $P \ll N$. On the contrary, the CPD and TT models trained with the regularization term of Eq. (5.18) (blue and orange respectively) with the exception of the concrete dataset fare better in terms of test error, as they have been trained with a regularization that recover the KRR baseline in the limit. Plots of the training errors can be found in Fig. 4 in Section C.1, where it can be seen that the regularization enforced by Theorems 5.3.1 and 5.3.2 yields overall models that fit the data better and, with the exception of the concrete dataset, generalize better.

5.5 RELATED WORK

Our contribution is closely tied to the links between Bayesian neural networks and GPs, first established for single-layer single-output neural networks [43, 44] having sigmoidal [45], Gaussian [46] and rectified linear unit [47] as activation function. This idea was extended to DNNs by Lee *et al.* [48] and Matthews *et al.* [49] for the most common activation functions. Further extensions have been proposed to CNNs where the number of channels tends to infinity [50, 51], to RNNs [52] and to DNNs having low-rank constraints on the weight matrices [53]. In particular Theorem 5.3.1 resembles the results of Neal [43, 44] and Williams [46] which relate infinite-width single layer neural networks to GPs. The CPD rank corresponds exactly to the width of the neural network. The crucial difference lies however in the Kronecker product structure, which is not present in neural networks and introduces a nonlinearity of different kind than the activation function. TTs on the other hand resemble DNNs as they map the output of each core to the next one. However, in contrast to DNNs, the inputs are processed over the depth of the network. For a more in depth discussion we refer the reader to [54]. Likewise Theorem 5.3.2 is the TN counterpart to the works of Lee *et al.* [48] and Matthews *et al.* [49] which relate finite depth neural networks to GPs. The priors we propose are also used in practice as a sensible initial guess for gradient-based optimization of TN-constrained models [40]. The results related to TT-constrained kernel machines in Theorem 5.3.2 were also derived by Guo and Draper [55, 56] from a quantum mechanical perspective, though a theoretical and experimental comparison with CPD-constrained kernel machines was not provided.

5.6 CONCLUSION

In this paper we proved that CPD and TT-constrained kernel machines are product kernel GPs in the limit of large TN ranks when placing suitable priors on their parameters. We characterized the target GP and showed that compared to CPD, TT-based models converge faster to the GP when dealing with higher-dimensional inputs. The proposed priors can be used in case of finite rank to train more flexible models that better fit the data compared to the standard approach which seeks instead to approximate the posterior with the addition of a TN constraint. One important limitation is that the ensuing models are more susceptible to overfitting and have thus to be tuned with more care. We empirically demonstrated these observations by means of numerical experiments.

REFERENCES

[1] F. Wesel and K. Batselier. "Tensor Network-Constrained Kernel Machines as Gaussian Processes". In: *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2025, pp. 2161–2169.

[2] A. Cichocki. "Era of Big Data Processing: A New Approach via Tensor Networks and Tensor Decompositions". In: *arXiv:1403.2048 [cs]* (Aug. 2014). arXiv: 1403.2048 [cs].

[3] 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". In: *Foundations and Trends® in Machine Learning* 9.4-5 (2016), pp. 249–429.

[4] A. Cichocki, A.-H. Phan, Q. Zhao, N. Lee, I. V. Oseledets, M. Sugiyama, and D. Mandic. "Tensor Networks for Dimensionality Reduction and Large-Scale Optimizations. Part 2 Applications and Future Perspectives". In: *Foundations and Trends® in Machine Learning* 9.6 (2017), pp. 249–429. arXiv: 1708.09165.

[5] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov. "Tensorizing Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc., 2015.

[6] M. Jaderberg, A. Vedaldi, and A. Zisserman. "Speeding up Convolutional Neural Networks with Low Rank Expansions". In: *Proceedings of the British Machine Vision Conference 2014* (2014).

[7] V. Lebedev, Y. Ganin, M. Rakhuba, I. V. Oseledets, and V. S. Lempitsky. "Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition". In: *International Conference on Learning Representations*. Jan. 2015.

[8] J. Ye, L. Wang, G. Li, D. Chen, S. Zhe, X. Chu, and Z. Xu. "Learning Compact Recurrent Neural Networks With Block-Term Tensor Decomposition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9378–9387.

[9] C. Hua, G. Rabusseau, and J. Tang. "High-Order Pooling for Graph Neural Networks with Tensor Decomposition". In: *Advances in Neural Information Processing Systems* 35 (Dec. 2022), pp. 6021–6033.

[10] X. Ma, P. Zhang, S. Zhang, N. Duan, Y. Hou, M. Zhou, and D. Song. “A Tensorized Transformer for Language Modeling”. In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019.

[11] E. M. Stoudenmire and D. J. Schwab. “Supervised Learning with Tensor Networks”. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. Curran Associates Inc., Dec. 2016, pp. 4806–4814.

[12] A. Novikov, I. Oseledets, and M. Trofimov. “Exponential Machines”. In: *Bulletin of the Polish Academy of Sciences: Technical Sciences; 2018; 66; No 6 (Special Section on Deep Learning: Theory and Practice); 789-797* (2018).

[13] F. Wesel and K. Batselier. “Large-Scale Learning with Fourier Features and Tensor Decompositions”. In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 17543–17554.

[14] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press, 2006.

[15] F. Wesel and K. Batselier. “Tensor-Based Kernel Machines with Structured Inducing Points for Large and High-Dimensional Data”. In: *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2023, pp. 8308–8320.

[16] M. Titsias. “Variational Learning of Inducing Variables in Sparse Gaussian Processes”. In: *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2009, pp. 567–574.

[17] J. Hensman, N. Fusi, and N. D. Lawrence. “Gaussian Processes for Big Data”. In: *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*. UAI’13. AUAI Press, Aug. 2013, pp. 282–290.

[18] A. Wilson and H. Nickisch. “Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP)”. In: *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, June 2015, pp. 1775–1784.

[19] J. Quiñonero-Candela and C. E. Rasmussen. “A Unifying View of Sparse Approximate Gaussian Process Regression”. In: *Journal of Machine Learning Research* 6.65 (2005), pp. 1939–1959.

[20] J. Hensman, N. Durrande, and A. Solin. “Variational Fourier Features for Gaussian Processes”. In: *The Journal of Machine Learning Research* 18.1 (Jan. 2017), pp. 5537–5588.

[21] M. Yadav, D. Sheldon, and C. Musco. “Faster Kernel Interpolation for Gaussian Processes”. In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. PMLR, Mar. 2021, pp. 2971–2979.

[22] A. Solin and S. Särkkä. “Hilbert Space Methods for Reduced-Rank Gaussian Process Regression”. In: *Statistics and Computing* 30.2 (Mar. 2020), pp. 419–446.

[23] A. Rahimi and B. Recht. “Random Features for Large-Scale Kernel Machines”. In: *Proceedings of the 20th International Conference on Neural Information Processing Systems*. Curran Associates Inc., Dec. 2007, pp. 1177–1184.

[24] M. Lázaro-Gredilla, J. Quiñnero-Candela, C. E. Rasmussen, and b. R. Figueiras-Vidal. “Sparse Spectrum Gaussian Process Regression”. In: *Journal of Machine Learning Research* 11.63 (2010), pp. 1865–1881.

[25] L. Csató and M. Opper. “Sparse On-Line Gaussian Processes”. In: *Neural Computation* 14.3 (Mar. 2002), pp. 641–668.

[26] M. W. Seeger, C. K. I. Williams, and N. D. Lawrence. “Fast Forward Selection to Speed Up Sparse Gaussian Process Regression”. In: *International Workshop on Artificial Intelligence and Statistics*. PMLR, Jan. 2003, pp. 254–261.

[27] E. Snelson and Z. Ghahramani. “Sparse Gaussian Processes Using Pseudo-inputs”. In: *Advances in Neural Information Processing Systems*. Vol. 18. MIT Press, 2006.

[28] J. Hensman, A. Matthews, and Z. Ghahramani. “Scalable Variational Gaussian Process Classification”. In: *Artificial Intelligence and Statistics*. PMLR, Feb. 2015, pp. 351–360.

[29] D. K. Duvenaud, H. Nickisch, and C. Rasmussen. “Additive Gaussian Processes”. In: *Advances in Neural Information Processing Systems* 24 (2011), pp. 226–234.

[30] X. Lu, A. Boukouvalas, and J. Hensman. “Additive Gaussian Processes Revisited”. In: *Proceedings of the 39th International Conference on Machine Learning*. PMLR, June 2022, pp. 14358–14383.

[31] T. Hastie, J. Friedman, and R. Tibshirani. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY: Springer, 2001.

[32] F. L. Hitchcock. “The Expression of a Tensor or a Polyadic as a Sum of Products”. In: *Journal of Mathematics and Physics* 6.1-4 (1927), pp. 164–189.

[33] I. V. Oseledets. “Tensor-Train Decomposition”. In: *SIAM Journal on Scientific Computing* 33.5 (Jan. 2011), pp. 2295–2317.

[34] N. Kargas and N. D. Sidiropoulos. “Supervised Learning and Canonical Decomposition of Multivariate Functions”. In: *IEEE Transactions on Signal Processing* (2021), pp. 1–1.

[35] S. Wahls, V. Koivunen, H. V. Poor, and M. Verhaegen. “Learning Multidimensional Fourier Series with Tensor Trains”. In: *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. Dec. 2014, pp. 394–398.

[36] K. Batselier, Z. Chen, and N. Wong. “Tensor Network Alternating Linear Scheme for MIMO Volterra System Identification”. In: *Automatica* 84 (Oct. 2017), pp. 26–35.

[37] Z. Chen, K. Batselier, J. A. K. Suykens, and N. Wong. “Parallelized Tensor Train Learning of Polynomial Classifiers”. In: *IEEE Transactions on Neural Networks and Learning Systems* 29.10 (Oct. 2018), pp. 4621–4632.

[38] J. A. Reyes and E. M. Stoudenmire. “Multi-Scale Tensor Network Architecture for Machine Learning”. In: *Machine Learning: Science and Technology* 2.3 (July 2021), p. 035036.

[39] P. Izmailov, A. Novikov, and D. Kropotov. “Scalable Gaussian Processes with Billions of Inducing Inputs via Tensor Train Decomposition”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR, Mar. 2018, pp. 726–735.

[40] F. Barratt, J. Dborin, and L. Wright. “Improvements to Gradient Descent Methods for Quantum Tensor Network Machine Learning”. In: *Second Workshop on Quantum Tensor Networks in Machine Learning*. May 2021.

[41] R. B. D’Agostino and M. A. Stephens. *Goodness-of-Fit Techniques*. CRC Press, Jan. 1986.

[42] D. Dua and C. Graff. *UCI Machine Learning Repository*. 2017.

[43] R. M. Neal. *Bayesian Learning for Neural Networks*. Springer Science & Business Media, Jan. 1996.

[44] R. M. Neal. “Priors for Infinite Networks”. In: *Bayesian Learning for Neural Networks*. Ed. by R. M. Neal. Lecture Notes in Statistics. New York, NY: Springer, 1996, pp. 29–53.

[45] C. Williams. “Computing with Infinite Networks”. In: *Advances in Neural Information Processing Systems*. Vol. 9. MIT Press, 1996.

[46] C. Williams. “Computing with Infinite Networks”. In: *Advances in Neural Information Processing Systems 9* (1997), pp. 295–301.

[47] Y. Cho and L. Saul. “Kernel Methods for Deep Learning”. In: *Advances in Neural Information Processing Systems*. Vol. 22. Curran Associates, Inc., 2009.

[48] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. “Deep Neural Networks as Gaussian Processes”. In: *International Conference on Learning Representations*. Feb. 2018.

[49] A. G. d. G. Matthews, M. Rowland, J. Hron, R. E. Turner, and Z. Ghahramani. *Gaussian Process Behaviour in Wide Deep Neural Networks*. Aug. 2018. arXiv: 1804.11271 [cs, stat].

[50] R. Novak, L. Xiao, Y. Bahri, J. Lee, G. Yang, J. Hron, D. A. Abolafia, J. Pennington, and J. Sohl-dickstein. “Bayesian Deep Convolutional Networks with Many Channels Are Gaussian Processes”. In: *International Conference on Learning Representations*. Sept. 2018.

[51] A. Garriga-Alonso, C. E. Rasmussen, and L. Aitchison. “Deep Convolutional Networks as Shallow Gaussian Processes”. In: *International Conference on Learning Representations*. Sept. 2018.

[52] X. Sun, S. Kim, and J.-I. Choi. “Recurrent Neural Network-Induced Gaussian Process”. In: *Neurocomputing* 509 (Oct. 2022), pp. 75–84.

[53] T. Nait-Saada, A. Naderi, and J. Tanner. "Beyond IID Weights: Sparse and Low-Rank Deep Neural Networks Are Also Gaussian Processes". In: *The Twelfth International Conference on Learning Representations*. Oct. 2023. arXiv: 2310.16597 [cs, stat].

[54] N. Cohen, O. Sharir, and A. Shashua. "On the Expressive Power of Deep Learning: A Tensor Analysis". In: *Conference on Learning Theory*. PMLR, June 2016, pp. 698–728.

[55] E. Guo and D. Draper. *Infinitely Wide Tensor Networks as Gaussian Process*. Jan. 2021. arXiv: 2101.02333 [stat].

[56] E. Guo and D. Draper. *Neural Tangent Kernel of Matrix Product States: Convergence and Applications*. Nov. 2021. arXiv: 2111.14046 [stat].

A INTRODUCTION

A.1 NOTATION

Throughout this paper we denote scalars in both capital and non-capital italics w, W, vectors in non-capital bold $\boldsymbol{w}$, matrices in capital bold $\boldsymbol{W}$ and tensors in capital italic bold font $\mathcal{W}$. The m-th entry of a vector $\boldsymbol{w} \in \mathbb{R}^M$ is indicated as w_m and the $m_1 m_2 \dots m_Q$-th entry of a Q-dimensional tensor $\mathcal{W} \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_Q}$ as $w_{m_1,m_2,\dots,m_Q}$. We employ the column notation to indicate a set of elements of tensor given a set of indices, e.g. $\boldsymbol{W}_{m_1,:,m_2}$ and $\boldsymbol{W}_{m_1,1:3,m_2}$ represent respectively all elements and the first three elements along the second dimension of tensor $\mathcal{W}$ with fixed indices m_1 and m_2. The Kronecker product is denoted by $\otimes$ and the Hadamard (elementwise) by $\odot$. We employ one-based indexing for all tensors. The Frobenius inner product between two Q-dimensional tensors $\mathcal{V}, \mathcal{W} \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_Q}$ is

$$\langle \mathcal{V}, \mathcal{W} \rangle_{\mathrm{F}} \coloneqq \sum_{m_1=1}^{M_1} \sum_{m_2=1}^{M_2} \cdots \sum_{m_Q=1}^{M_Q} v_{m_1,m_2,\dots,m_Q} w_{m_1,m_2,\dots,m_Q},$$

and the Frobenius norm of $\mathcal{W} \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_Q}$ is denoted and defined as

$$||\mathcal{W}||^2 \coloneqq \langle \mathcal{W}, \mathcal{W} \rangle_{\mathrm{F}}.$$

We define the vectorization operator as $\mathrm{vec}\,(\cdot) : \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_Q} \to \mathbb{R}^{M_1 M_2 \cdots M_Q}$ such that

$$\mathrm{vec}\,(\mathcal{W})_m = w_{m_1,m_2,\dots,m_Q},$$

with $m = m_1 + \sum_{q=2}^{Q}(m_q - 1)\prod_{k=1}^{q-1} M_k$. Likewise, its inverse, the tensorization operator $\text{ten}(\cdot) : \mathbb{R}^{M_1 M_2 \cdots M_Q} \to \mathbb{C}^{M_1 \times M_2 \times \ldots M_Q}$ is defined such that

$$\text{ten}(\boldsymbol{w})_{m_1, m_2, \cdots, m_Q} = w_m.$$

B TN-CONSTRAINED KERNEL MACHINES AS GPS

B.1 GP OF CPD-CONSTRAINED KERNEL MACHINE

Theorem 5.3.1 (CPD-constrained kernel machine as GP)**.** *Consider the* CPD*-constrained kernel machine*

$$f_{\texttt{CPD}}(\boldsymbol{x}) \coloneqq \langle \texttt{R}_1(\text{ten}\left(\boldsymbol{\varphi}(\boldsymbol{x})\right)), \texttt{CPD}(\text{ten}(\boldsymbol{w}))\rangle_{\text{F}}.$$

If each of the R columns $\boldsymbol{w}^{(q)}{}_{:,r} \in \mathbb{R}^{M_q}$ *of each* CPD *core is an i.i.d. random variable such that*

$$\mathbb{E}\left[\boldsymbol{w}^{(q)}_{:,r}\right] = \mathbf{0},$$
$$\mathbb{E}\left[\boldsymbol{w}^{(q)}_{:,r}\boldsymbol{w}^{(q)\text{T}}_{:,r}\right] = R^{-\frac{1}{Q}}\boldsymbol{\Lambda}^{(q)},$$

then $f_{\texttt{CPD}}(\boldsymbol{x})$ *converges in distribution as* $R \to \infty$ *to the* GP

$$f_{\texttt{CPD}}(\boldsymbol{x}) \sim \mathcal{GP}\left(0, \prod_{q=1}^{Q} \boldsymbol{\varphi}^{(q)}(\boldsymbol{x})^{\text{T}}\boldsymbol{\Lambda}^{(q)}\boldsymbol{\varphi}^{(q)}(\cdot)\right).$$

Proof. Consider the R intermediate functions h_r of Eq. (5.13) which constitute the CPD-constrained model of Eq. (5.11). Due to the i.i.d. assumption on $\boldsymbol{w}^{(q)}{}_{:,r}$ each addend is the same function of i.i.d. random variables and thus is itself i.i.d.The mean of each addend is

$$\mathbb{E}[h_r(\boldsymbol{x})] = \mathbb{E}\left[\prod_{q=1}^{Q} \boldsymbol{\varphi}^{(q)}(\boldsymbol{x})^{\text{T}}\boldsymbol{w}^{(q)}{}_{:,r}\right] = 0, \tag{.19}$$

due to the i.i.d. assumption and the linearity of expectation. Its covariance is

$$\begin{aligned}
&\mathbb{E}\left[h_r(\boldsymbol{x})h_r(\boldsymbol{x}')\right] && \text{(.20a)}\\
=&\mathbb{E}\left[\prod_{q=1}^{Q}\boldsymbol{\varphi}^{(q)}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{w}^{(q)}{}_{:,r}\prod_{q=1}^{Q}\boldsymbol{\varphi}^{(q)}(\boldsymbol{x}')^{\mathrm{T}}\boldsymbol{w}^{(q)}{}_{:,r}\right] && \text{(.20b)}\\
=&\mathbb{E}\left[\prod_{q=1}^{Q}\boldsymbol{\varphi}^{(q)}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{w}^{(q)}{}_{:,r}\boldsymbol{w}^{(q)}{}_{:,r}{}^{\mathrm{T}}\boldsymbol{\varphi}^{(q)}(\boldsymbol{x}')\right] && \text{(.20c)}\\
=&\prod_{q=1}^{Q}\boldsymbol{\varphi}^{(q)}(\boldsymbol{x})^{\mathrm{T}}\mathbb{E}\left[\boldsymbol{w}^{(q)}{}_{:,r}\boldsymbol{w}^{(q)}{}_{:,r}{}^{\mathrm{T}}\right]\boldsymbol{\varphi}^{(q)}(\boldsymbol{x}') && \text{(.20d)}\\
=&\frac{1}{R}\prod_{q=1}^{Q}\boldsymbol{\varphi}^{(q)}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\Lambda}^{(q)}\boldsymbol{\varphi}^{(q)}(\boldsymbol{x}').
\end{aligned}$$

Here the step from Eq. (.20b) to Eq. (.20c) exploits the fact that the transpose of a scalar is equal to itself, the step from Eq. (.20c) to Eq. (.20d) is due to the linearity of expectation. As the variances of each intermediate function h_r are appropriately scaled, by the CLT the partial sum $f_{\mathtt{CPD}}(\boldsymbol{x})$ converges in distribution to a multivariate normal distribution, which is fully specified by its first two moments

$$\begin{aligned}
\mathbb{E}\left[f_{\mathtt{CPD}}(\boldsymbol{x})\right] &= 0,\\
\mathbb{E}\left[f_{\mathtt{CPD}}(\boldsymbol{x})f_{\mathtt{CPD}}(\boldsymbol{x}')\right] &= \prod_{q=1}^{Q}\boldsymbol{\varphi}^{(q)}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\Lambda}^{(q)}\boldsymbol{\varphi}^{(q)}(\boldsymbol{x}').
\end{aligned}$$

Since any finite collection of $\{f_{\mathtt{CPD}}(\boldsymbol{x}),\ldots,f_{\mathtt{CPD}}(\boldsymbol{x}')\}$ will have a joint multivariate normal distribution with the aforementioned first two moments, we conclude that $f_{\mathtt{CPD}}(\boldsymbol{x})$ is the Gaussian process

$$f_{\mathtt{CPD}}(\boldsymbol{x}) \sim \mathcal{GP}\left(0, \prod_{q=1}^{Q}\boldsymbol{\varphi}^{(q)}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\Lambda}^{(q)}\boldsymbol{\varphi}^{(q)}(\cdot)\right).$$

□

B.2 GP OF TT-CONSTRAINED KERNEL MACHINE IN THE SEQUENTIAL LIMIT OF THE TT RANKS

Theorem 5.3.2 (TT-constrained kernel machine as GP)**.** *Consider the* TT*-constrained kernel machine*

$$f_{\mathrm{TT}}(\boldsymbol{x}) \coloneqq \langle \mathrm{R}_1(\mathrm{ten}\left(\boldsymbol{\varphi}(\boldsymbol{x})\right)), \mathrm{TT}(\mathrm{ten}\left(\boldsymbol{w}\right))\rangle_{\mathrm{F}}$$

If each of the $R_{q-1}R_q$ *fibers* ${\boldsymbol{W}^{(q)}}_{r_{q-1},:,r_q} \in \mathbb{R}^{M_q}$ *of each* TT *core is an i.i.d. random variable such that*

$$\mathbb{E}\left[\boldsymbol{W}^{(q)}_{r_{q-1},:,r_q}\right] = \mathbf{0},$$
$$\mathbb{E}\left[\boldsymbol{W}^{(q)}_{r_{q-1},:,r_q}{\boldsymbol{W}^{(q)}_{r_{q-1},:,r_q}}^{\mathrm{T}}\right] = \frac{1}{\sqrt{R_{q-1}R_q}}\boldsymbol{\Lambda}^{(q)},$$

then $f_{\mathrm{TT}}(\boldsymbol{x})$ *converges in distribution as sequentially* $R_1 \to \infty$, $R_2 \to \infty, \ldots, R_{Q-1} \to \infty$ *to the Gaussian process*

$$f_{\mathrm{TT}}(\boldsymbol{x}) \sim \mathcal{GP}\left(0, \prod_{q=1}^{Q} \boldsymbol{\varphi}^{(q)}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\Lambda}^{(q)}\boldsymbol{\varphi}^{(q)}(\cdot)\right).$$

Proof. Define the vector of intermediate function $\boldsymbol{h}^{(q+1)} \in \mathbb{R}^{R_{q+1}}$ recursively as

$$h^{(q+1)}_{r_{q+1}} \coloneqq \sum_{r_q=1}^{R_q} z^{(q+1)}_{r_q,r_{q+1}}(x_{q+1})h^{(q)}_{r_q},$$

with $h^{(0)} \coloneqq 1$. Note that the first two moments of intermediate variable $z^{(q+1)}_{r_q,r_{q+1}}(x_{q+1})$ are

$$\mathbb{E}\left[z^{(q+1)}_{r_q,r_{q+1}}(\boldsymbol{x})\right] = 0,$$
$$\mathbb{E}\left[z^{(q+1)}_{r_q,r_{q+1}}(\boldsymbol{x})z^{(q+1)}_{r_q,r_{q+1}}(\boldsymbol{x'})\right]$$
$$= \frac{1}{\sqrt{R_qR_{q+1}}}\boldsymbol{\varphi}^{(q)}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\Lambda}^{(q)}\boldsymbol{\varphi}^{(q)}(\boldsymbol{x'}).$$

We proceed by induction. For the induction step suppose that $h^{(q)}_{r_q}$ is a GP, iden-

tical and independent for every r_q such that

$$h_{r_q}^{(q)} \sim \mathcal{GP}\left(0, \frac{1}{\sqrt{R_q}} \prod_{p=1}^{q} \boldsymbol{\varphi}^{(p)}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{\Lambda}^{(p)} \boldsymbol{\varphi}^{(p)}(\cdot)\right).$$

The scalar $h_{r_{q+1}}^{(q+1)}$ is the sum of R_q i.i.d. terms having mean

$$\mathbb{E}\left[h_{r_{q+1}}^{(q+1)}\right] = \mathbb{E}\left[z_{r_q,r_{q+1}}^{(q+1)}(\boldsymbol{x}) h_{r_q}^{(q)}\right] = 0,$$

and covariance

$$\begin{aligned}
&\mathbb{E}\left[h_{r_{q+1}}^{(q+1)} h_{r_{q+1}}^{(q+1)}\right] \\
=&\mathbb{E}\left[z_{r_q,r_{q+1}}^{(q+1)}(\boldsymbol{x}) h_{r_q}^{(q)} z_{r_q,r_{q+1}}^{(q+1)}(\boldsymbol{x}') h_{r_q}^{(q)}\right] \\
=&\mathbb{E}\left[z_{r_q,r_{q+1}}^{(q+1)}(\boldsymbol{x}) z_{r_q,r_{q+1}}^{(q+1)}(\boldsymbol{x}')\right] \mathbb{E}\left[h_{r_q}^{(q)} h_{r_q}^{(q)}\right] \\
=&\frac{1}{\sqrt{R_{q+1}}} \prod_{p=1}^{q+1} \boldsymbol{\varphi}^{(p)}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{\Lambda}^{(p)} \boldsymbol{\varphi}^{(p)}(\boldsymbol{x}').
\end{aligned}$$

Since the assumptions of the CLT are satisfied the partial sum $h_{r_{q+1}}^{(q+1)}$ converges in distribution to the normal distribution, fully specified by the above mentioned first two moments. Since any finite collection of $\{h_{r_{q+1}}^{(q+1)}(\boldsymbol{x}_{1:q+1}), \ldots, h_{r_{q+1}}^{(q+1)}(\boldsymbol{x}'_{1:q+1})\}$ will have a joint multivariate normal distribution with the aforementioned first two moments, we conclude that $h_{r_{q+1}}^{(q+1)}(\boldsymbol{x}_{1:q+1})$ is the GP

$$h_{r_{q+1}}^{(q+1)} \sim \mathcal{GP}\left(0, \frac{1}{\sqrt{R_{q+1}}} \prod_{p=1}^{q+1} \boldsymbol{\varphi}^{(p)}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{\Lambda}^{(p)} \boldsymbol{\varphi}^{(p)}(\cdot)\right).$$

For the base case, consider the R_1 outputs of the first hidden function $h_{r_1}^{(1)}$. They are i.i.d. with mean

$$\mathbb{E}\left[h_{r_1}^{(1)}(\boldsymbol{x})\right] = 0.$$

and covariance

$$\mathbb{E}\left[h_{r_1}^{(1)}(\boldsymbol{x}) h_{r_1}^{(1)}(\boldsymbol{x}')\right] = \frac{1}{\sqrt{R_1}} \boldsymbol{\varphi}^{(1)}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{\Lambda}^{(1)} \boldsymbol{\varphi}^{(1)}(\boldsymbol{x}').$$

We now consider the R_2 outputs of the second hidden function $h_{r_2}^{(2)}$

$$h_{r_2}^{(2)} = \sum_{r_1=1}^{R_1} z_{r_1,r_2}^{(2)}(\boldsymbol{x}) h_{r_1}^{(1)},$$

which are i.i.d. as they are the same function of the R_1 i.i.d. outputs of $h_{r_1}^{(1)}(\boldsymbol{x})$. More specifically, their mean and covariance are

$$\begin{aligned}
&\mathbb{E}\left[h_{r_2}^{(2)}\right] = 0, \\
&\mathbb{E}\left[h_{r_2}^{(2)}(\boldsymbol{x}) h_{r_2}^{(2)}(\boldsymbol{x}')\right] \\
=&\frac{1}{\sqrt{R_2}} \prod_{q=1}^{2} \boldsymbol{\varphi}^{(q)}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{\Lambda}^{(q)} \boldsymbol{\varphi}^{(d)}(\boldsymbol{x}').
\end{aligned}$$

Once more by the CLT, the partial sum $h_{r_2}^{(2)}$ converges in distribution to the normal distribution with the above first two moments. Since any finite collection of $\{h_{r_2}^{(2)}(\boldsymbol{x}), \ldots, h_{r_2}^{(2)}(\boldsymbol{x}'\}$ will have a joint multivariate normal distribution with the aforementioned first two moments, we conclude that $h_2^{(2)}(\boldsymbol{x})$ is the GP

$$h_{r_2}^{(2)} \sim \mathcal{GP}\left(0, \frac{1}{\sqrt{R_2}} \prod_{q=1}^{2} \boldsymbol{\varphi}^{(q)}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{\Lambda}^{(q)} \boldsymbol{\varphi}^{(q)}(\cdot)\right),$$

which is our base case. Hence by induction $f_{\mathrm{TT}}(\boldsymbol{x}) = h^{(Q)}$ converges in distribution as $R_1 \to \infty$, $R_2 \to \infty$, $\ldots$, $R_{Q-1} \to \infty$ to the GP

$$f_{\mathrm{TT}}(\boldsymbol{x}) \sim \mathcal{GP}\left(0, \prod_{q=1}^{Q} \boldsymbol{\varphi}^{(q)}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{\Lambda}^{(q)} \boldsymbol{\varphi}^{(q)}(\cdot)\right).$$

□

B.3 GP OF TT-CONSTRAINED KERNEL MACHINE IN THE SIMULTANEOUS LIMIT OF THE TT RANKS

In Theorem 5.3.2 we prove by induction that the TT-constrained kernel machine converges to a GP by taking successive limits of the TT ranks. This result is analogous to the work of Lee *et al.* [48], who prove that for the DNNs, taking sequentially the limit of each layer. A more practically useful result consists in the convergence in the *simultaneous* limit of TT ranks.

In deep learning Matthews *et al.* [49, theorem 4] prove convergence in the context of DNNs over the widths of all layers simultaneously. Said theorem has been employed to prove GP convergence in the context of convolutional neural networks [51] and in the context of DNNs where each weight matrix is of low rank [53].

Seeing the similarity between TT-constrained kernel machines (Eq. (5.14)) and DNNs and the technicality of the proof, similarly to [51, 53] we draw a one-to-one map between the TT-constrained kernel machines and the DNNs considered in Matthews *et al.* [49, theorem 4]. Convergence in the simultaneous limit is then guaranteed by Matthews *et al.* [49, theorem 4].

We begin by restating the definitions of linear envelope property, DNNs, linear envelope property and normal recursion as found in Matthews *et al.* [49]. To make the comparison easier for the reader, we change the indexing notation to match the one in this paper.

Definition B.1 (Linear envelope property for nonlinearities [49])**.** A nonlinearity $t : \mathbb{R} \to \mathbb{R}$ is said to obey the linear envelope property if there exist $c, l \geq 0$ such that the following inequality holds

$$|t(u)| < c + l|u| \ \forall u \in \mathbb{R}. \tag{.25}$$

Definition B.2 (Fully connected DNN [49])**.** A fully connected deep neural with one-dimensional output and inputs $\boldsymbol{x} \in \mathbb{R}^{R_0}$ is defined recursively such that the initial step is

$$h^{(1)}_{r_1}(\boldsymbol{x}) = \sum_{r_0=1}^{R_0} z^{(1)}_{r_1,r_0} x_{r_0} + b^{(1)}_{r_1}, \tag{.26}$$

the activation step by nonlinear activation function t is given by

$$g^{(q)}_{r_q} = t(f^{(q)}_{r_q}), \tag{.27}$$

and the subsequent layers are defined by the recursion

$$h^{(q+1)}_{r_{q+1}} = \sum_{r_q=1}^{R_q} z^{(q+1)}_{r_{q+1},r_d} g^{(q)}_{r_q} + b^{q+1}_{r_{q+1}}, \tag{.28}$$

so that $h^{(Q)}$ is the output of the network. In the above, $\boldsymbol{Z}^{(q)} \in \mathbb{R}^{R_{q-1} \times R_q}$ and $\boldsymbol{b}^{(q)} \in \mathbb{R}^{R_q}$ are respectively the weights and biases of the q-th layer.

Definition B.3 (Width function [49])**.** For a given fixed input $n \in \mathbb{N}$, a width func-

tion $v^{(q)} : \mathbb{N} \to \mathbb{N}$ at depth q specifies the number of hidden units R_q at depth q.

Lemma B.4 (Normal recursion [49]). *Consider $z^{(q)}_{r_{q-1},r_q} \sim \mathcal{N}(0, C^{(q)}_w)$ and $b^{(q)}_{r_q} \sim \mathcal{N}(0, C^{(q)}_b)$. If the activations of the q-th layer are normally distributed with moments*

$$\mathbb{E}\left[h^{(q)}_{r_q}\right] = 0 \tag{.29}$$

$$\mathbb{E}\left[h^{(q)}_{r_q} h^{(q)}_{r_q}\right] = K(x, x'), \tag{.30}$$

then under recursion Eqs. (.27) *and* (.28)*, as $R_{q-1} \to \infty$, the activations of the next layer converge in distribution to a normal distribution with moments*

$$\mathbb{E}\left[h^{(q+1)}_{r_{q+1}}\right] = 0 \tag{.31}$$

$$\mathbb{E}\left[h^{(q+1)}_{r_{q+1}} h^{(q+1)}_{r_{q+1}}\right] = C^{(q+1)}_w \mathbb{E}_{(\epsilon_1,\epsilon_2)\sim\mathcal{N}(0,K)}\left[t(\epsilon_1)\, t(\epsilon_2)\right] + C^{(q+1)}_b. \tag{.32}$$

We can now state the major result in Matthews *et al.* [49].

Theorem B.5 (GP in the simultaneous limit of fully connected DNNs [49]). *Consider a random* DNN *of the form of Definition B.2 obeying the linear envelope condition of Definition B.1. Then for all sets of strictly increasing width functions $v^{(q)}$ and for any countable input set $\{\boldsymbol{x}, \ldots, \boldsymbol{x'}\}$, the distribution of the output of the network converges in distribution to a* GP *as $n \to \infty$. The* GP *has mean and covariance functions given by the recursion in Lemma B.4.*

Corollary B.6 (GP in the simultaneous limit of TT-constrained kernel machines). *Consider a random* TT*-constrained kernel machine of the form of Definition 5.2.6 obeying the linear envelope condition of Definition B.1. Then for all sets of strictly increasing width functions $v^{(q)}$ and for any countable input set $\{\boldsymbol{x}, \ldots, \boldsymbol{x'}\}$, the distribution of the output of the network converges in distribution to a* GP *as $P \to \infty$. The* GP *has mean and covariance functions given by the recursion in Lemma B.4 and stated in Theorem 5.3.2.*

Proof. When examining Definition B.2 and comparing it with Definition 5.2.6 it becomes clear that both models are similar. In the special case of involving linear activation function and zero biases, the models are structurally identical if one considers unit inputs $x = 1$ in Eq. (.26). The normal recursion in Lemma B.4 is

satisfied by TT-constrained kernel machines, as we have that

$$
\begin{aligned}
t(u) &:= u \;\forall u \in \mathbb{R}, \\
C_b^{(q+1)} &:= 0, \\
C^{(q+1)} &:= \frac{1}{\sqrt{R_q R_{q+1}}} \boldsymbol{\varphi}^{(q)}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{\Lambda}^{(q)} \boldsymbol{\varphi}^{(q)}(\boldsymbol{x}'), \\
K &:= \frac{1}{\sqrt{R_q}} \prod_{p=1}^{q} \boldsymbol{\varphi}^{(p)}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{\Lambda}^{(p)} \boldsymbol{\varphi}^{(p)}(\boldsymbol{x}') \\
\mathbb{E}_{(\epsilon_1,\epsilon_2)\sim\mathcal{N}(0,K)}\left[t(\epsilon_1)\, t(\epsilon_2)\right] &:= K.
\end{aligned}
$$

Hence by Theorem B.5, for all sets of strictly increasing width functions $v^{(q)}$ and for any countable input set $\{\boldsymbol{x}, \ldots, \boldsymbol{x}'\}$, the distribution of the output of the network converges in distribution to a GP, fully specified by the output of the normal recursion in Lemma B.4, which equals the GP in Theorem 5.3.2. □

5

C NUMERICAL EXPERIMENTS

C.1 GP BEHAVIOR AT PREDICTION

We provide the training RMSE related to Section 5.4.2 in Fig. 4, where it can be seen that the new priors yield model that provide a better fit on all datasets.

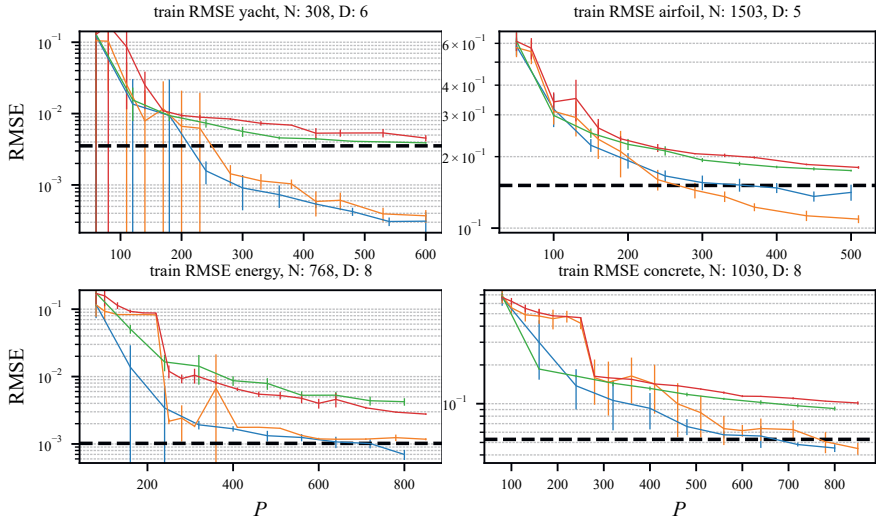


Figure 4.: Mean and standard deviation of the training RMSE of CPD and TT models for regularization Eqs. (5.17) and (5.18) (green and red curves respectively) as a function of model parameters P as well as their target KRR (dotted line). In the plots, the probabilistic regularization of Eq. (5.18) and its TT counterpart are denoted by a blue and orange line respectively. The dotted line corresponds to the KRR (GP posterior mean) baseline. The proposed regularization which stems from Theorems 4.2.3 and 5.3.2 achieves lower test RMSE with fewer parameters, with the notable exception of the concrete datasets where it leads to overfitting.

6

A KERNELIZABLE PRIMAL-DUAL FORMULATION OF THE MLSVD

The ability to express a learning task in terms of a primal and a dual optimization problem lies at the core of a plethora of machine learning methods. For example, Support Vector Machine *(*SVM*),* Least-Squares Support Vector Machine *(*LS-SVM*),* Ridge Regression *(*RR*),* Lasso Regression *(*LR*),* Principal Component Analysis *(*PCA*), and more recently* Singular Value Decomposition *(*SVD*) have all been defined either in terms of primal weights or in terms of dual Lagrange multipliers. The primal formulation is computationally advantageous in the case of large sample size while the dual is preferred for high-dimensional data. Crucially, said learning problems can be made nonlinear through the introduction of a feature map in the primal problem, which corresponds to applying the kernel trick in the dual. In this paper we derive a primal-dual formulation of the* Multilinear Singular Value Decomposition *(*MLSVD*), which recovers as special cases both* PCA *and* SVD*. Besides enabling computational gains through the derived primal formulation, we propose a nonlinear extension of the* MLSVD *using feature maps, which results in a dual problem where a kernel tensor arises. We discuss potential applications in the context of signal analysis and deep learning.*

This chapter has been submitted to the Journal of Machine Learning Research (JMLR) and archived as:
F. Wesel and K. Batselier. *A Kernelizable Primal-Dual Formulation of the Multilinear Singular Value Decomposition.* Oct. 2024. arXiv: 2410.10504

6.1 INTRODUCTION

The linear *Support Vector Machine* (SVM) was for the first time extended to the nonlinear case by Boser, Guyon, and Vapnik [2], giving rise to modern SVM theory. Central to this extension is the primal-dual formulation of the learning problem, which allows for a nonlinear extension in terms of a primal feature map $\boldsymbol{\phi}(\cdot) : \mathbb{R}^{N_1} \rightarrow \mathbb{R}^M$ that maps an inputs $\boldsymbol{x} \in \mathbb{R}^{N_1}$ to a higher (possibly infinite) dimensional space $\mathbb{R}^M$. Solving the corresponding dual problem requires then the evaluation of all pairwise *kernel* evaluations $\kappa(\boldsymbol{x}_n, \boldsymbol{x}_{n'}) := \boldsymbol{\phi}(\boldsymbol{x}_n)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_{n'})$. The so-called *kernel trick* ensures that these kernel evaluations can be computed without ever explicitly mapping the inputs to the higher-dimensional space.

This approach has been applied to a plethora of methods [3], e.g. *Least-Squares Support Vector Machine* (LS-SVM) [4], *Ridge Regression* (RR) [5], *Lasso Regression* (LR) [6], *Principal Component Analysis* (PCA) [7, 8] and more recently to the *Singular Value Decomposition* (SVD) [9] in order to yield their *kernelized* variants.

Among those, PCA is an ubiquitous unsupervised learning approach which seeks an orthogonal subspace that maximizes the covariance between the samples of data. Its kernelized counterpart, *Kernel Principal Component Analysis* (KPCA), seeks an orthogonal subspace which maximizes the covariance of samples of data mapped into a higher dimensional space. By construction, KPCA does not provide any information regarding the row subspace of the data matrix, meaning that if there is any asymmetry in the data, it will not be captured. A related but different method is the SVD, which factors a data matrix in terms of orthogonal row and column subspaces linked by a positive diagonal matrix of so-called *singular values*. The SVD has been cast in the primal-dual framework by Suykens [9], who also proposed a *Kernel Singular Value Decomposition* (KSVD) extension in terms of feature maps of both rows and columns of the data matrix, coupled together by a *compatibility matrix*. In contrast with KPCA, said construction gives rise to kernel functions that can be asymmetric and non-positive and thus are arguably better suited to model real-life data, which often is nonlinear and asymmetric [10]. The *Multilinear Singular Value Decomposition* (MLSVD) [11] extends the concept of the SVD to higher-order arrays, also known as tensors. In simple terms, the MLSVD factors a data tensor in terms of orthogonal subspaces corresponding to each mode coupled by a *core tensor*, which unlike in the SVD case, does not need to be diagonal. The research stream of the MLSVD and the closely associated *Tucker decomposition*, which relaxes the orthogonality constraint, follows the trends of most other tensor decompositions. Early literature focuses on applications in chemometrics and psychometrics. Landmarks in the literature explored applications in facial [12] and gait recognition [13] and explored the

method as an extension of PCA to tensorial data. More recently, the decomposition has been research as a tool for the compression of layers of deep learning models [14, 15, 16, 17], in the setting of tensors completion [18] knowledge graph completion [19], graph classification [20]. The decomposition has also been used as a theoretical tool for the study of deep learning models [21, 22, 23]. Other recent areas of active research tackle the issues of scalability [24, 25, 26, 27, 28] and Bayesian extensions [29, 30].

In this paper we extend the Lanczos decomposition theorem of matrices [31] to the tensor case. This allows us to derive a primal-dual formulation of the MLSVD, which recovers as special cases both PCA and the SVD. The newly established primal formulation can be used to attain computational gains in the large sample regime, and allows us to kernelize the MLSVD by means of feature maps. These feature maps define the construction of a kernel tensor in the dual as opposed to a kernel matrix in the case of PCA and SVD. Similarly to the SVD case, the tensor kernel does not need to be symmetric or positive. We discuss possible choices of kernel functions and applications, which range from signal analysis to deep learning.

The remainder of the paper is structured as follows. In Section 6.2 we provide the background related to the MLSVD as well as a theorem that will be useful to prove our main result in Section 6.3. We discuss related work in Section 6.5 and formulate recommendations for future work in Section 6.6.

6.2 BACKGROUND

In the remainder of this paper we denote tensors with uppercase calligraphic bold e.g. $\boldsymbol{\mathcal{X}}$, matrices in uppercase bold e.g. $\boldsymbol{X}$, vectors in lowecase bold $\boldsymbol{x}$ and scalars in lowercase, e.g. x. We denote the mode-d unfolding of a tensor $\boldsymbol{\mathcal{X}}$ [32] with $\boldsymbol{X}_{(d)}$, the Kronecker product with $\otimes$ and column-major vectorization with $\text{vec}(\cdot)$.

The compact SVD of a rank-R matrix $\boldsymbol{X} \in \mathbb{R}^{N_1 \times N_2}$ can be written as $\text{vec}(\boldsymbol{X}) = (\boldsymbol{U}_2 \otimes \boldsymbol{U}_1)\,\text{vec}(\boldsymbol{S})$, with semi-orthogonal matrices $\boldsymbol{U}_1 \in \mathbb{R}^{N_1 \times R}$ and $\boldsymbol{U}_2 \in \mathbb{R}^{N_2 \times R}$ such that $\boldsymbol{U}_1^{\mathrm{T}}\boldsymbol{U}_1 = \boldsymbol{I}_{R_1}$, $\boldsymbol{U}_2^{\mathrm{T}}\boldsymbol{U}_2 = \boldsymbol{I}_{R_2}$ and a square diagonal matrix $\boldsymbol{S} \in \mathbb{R}^{R \times R}$ of singular values. The MLSVD, also known as the *Higher-Order Singular Value Decomposition* (HOSVD), is one way to generalize the SVD of matrices to higher-order tensors [11], which expresses a Dth-order tensor in terms of D coupled orthogonal subspaces. In order to simplify notation and increase the readability we will consider from now on the case $D = 3$ without any loss of generality. For the general case we refer the reader to Appendix A.

Definition 6.2.1 (*Multilinear Singular Value Decomposition* (MLSVD) [11])**.** The rank-(R_1, R_2, R_3) MLSVD of a 3rd-order tensor $\mathscr{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ is

$$\text{vec}(\mathscr{X}) = (\boldsymbol{U}_3 \otimes \boldsymbol{U}_2 \otimes \boldsymbol{U}_1)\,\text{vec}(\mathscr{S}), \tag{6.1}$$

where $\boldsymbol{U}_1 \in \mathbb{R}^{N_1 \times R_1}, \boldsymbol{U}_2 \in \mathbb{R}^{N_2 \times R_2}, \boldsymbol{U}_3 \in \mathbb{R}^{N_3 \times R_3}$ are semi-orthogonal factor matrices and $\mathscr{S} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ is the *core* tensor such that the matrices $\boldsymbol{S}_{(1)}\boldsymbol{S}_{(1)}^{\mathrm{T}}$, $\boldsymbol{S}_{(2)}\boldsymbol{S}_{(2)}^{\mathrm{T}}$, $\boldsymbol{S}_{(3)}\boldsymbol{S}_{(3)}^{\mathrm{T}}$ are positive diagonal.

In practice, the MLSVD of a 3-rd order tensor can be computed by 3 SVD factorizations as elucidated in De Lathauwer, De Moor, and Vandewalle [11]. Each SVD provides $\boldsymbol{U}_1$, $\boldsymbol{U}_2$ and $\boldsymbol{U}_3$ as the left-singular values of the respective unfoldings $\boldsymbol{X}_{(1)}$, $\boldsymbol{X}_{(2)}$, $\boldsymbol{X}_{(3)}$. The core tensor is then computed by solving Eq. (6.1) for $\mathscr{S}$.

6.3 A PRIMAL-DUAL FORMULATION FOR THE MLSVD

Before presenting our main result, i.e. a primal-dual formulation of the MLSVD, we need to generalize an important theorem by Lanczos [31], who defines *shifted* eigenvalue problems which are equivalent to the SVD. Similarly, the MLSVD of a 3rd-order tensor $\mathscr{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ can then be uniquely defined as a set of 3 coupled matrix equations. A generalization of the proof to tensors of higher order is straightforward and can be found in Theorem A.1.

Theorem 6.3.1 (Generalized Lanczos decomposition theorem)**.** *An arbitrary rank-(R_1, R_2, R_3) tensor $\mathscr{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ can be written in* MLSVD *form, i.e. as in Eq.* (6.1) *with core tensor $\mathscr{S} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ and semi-orthogonal factor matrices $\boldsymbol{U}_1 \in \mathbb{R}^{N_1 \times R_1}$, $\boldsymbol{U}_2 \in \mathbb{R}^{N_2 \times R_2}$ and $\boldsymbol{U}_3 \in \mathbb{R}^{N_3 \times R_3}$ defined by the following set of equations*

$$\begin{aligned} \boldsymbol{U}_1\,\boldsymbol{S}_{(1)} &= \boldsymbol{X}_{(1)}\,(\boldsymbol{U}_3 \otimes \boldsymbol{U}_2), \\ \boldsymbol{U}_2\,\boldsymbol{S}_{(2)} &= \boldsymbol{X}_{(2)}\,(\boldsymbol{U}_3 \otimes \boldsymbol{U}_1), \\ \boldsymbol{U}_3\,\boldsymbol{S}_{(3)} &= \boldsymbol{X}_{(3)}\,(\boldsymbol{U}_2 \otimes \boldsymbol{U}_1), \end{aligned} \tag{6.2}$$

with the additional constraint that $\boldsymbol{S}_{(1)}\boldsymbol{S}_{(1)}^{\mathrm{T}}$, $\boldsymbol{S}_{(2)}\boldsymbol{S}_{(2)}^{\mathrm{T}}$, $\boldsymbol{S}_{(3)}\boldsymbol{S}_{(3)}^{\mathrm{T}}$ are positive diagonal matrices.

Proof. The proof is divided in two steps, first we show that the factor matrices $\boldsymbol{U}_1$, $\boldsymbol{U}_2$, $\boldsymbol{U}_3$ are semi-orthogonal, second we show that indeed Eq. (6.2) implies the MLSVD i.e. Eq. (6.1). We begin by left-multiplying each side of Eq. (6.2) respec-

tively with $\boldsymbol{U}_1^{\mathrm{T}}$, $\boldsymbol{U}_2^{\mathrm{T}}$, $\boldsymbol{U}_3^{\mathrm{T}}$, resulting in three equations of the form

$$\begin{aligned}
\boldsymbol{U}_1^{\mathrm{T}}\boldsymbol{U}_1\boldsymbol{S}_{(1)} &= \boldsymbol{U}_1^{\mathrm{T}}\boldsymbol{X}_{(1)}\,(\boldsymbol{U}_3 \otimes \boldsymbol{U}_2) =: \boldsymbol{D}_{(1)}, \\
\boldsymbol{U}_2^{\mathrm{T}}\boldsymbol{U}_2\boldsymbol{S}_{(2)} &= \boldsymbol{U}_2^{\mathrm{T}}\boldsymbol{X}_{(2)}\,(\boldsymbol{U}_3 \otimes \boldsymbol{U}_1) =: \boldsymbol{D}_{(2)}, \\
\boldsymbol{U}_3^{\mathrm{T}}\boldsymbol{U}_3\boldsymbol{S}_{(3)} &= \boldsymbol{U}_3^{\mathrm{T}}\boldsymbol{X}_{(3)}\,(\boldsymbol{U}_2 \otimes \boldsymbol{U}_1) =: \boldsymbol{D}_{(3)}.
\end{aligned} \tag{6.3}$$

By construction, all three right-hand sides of Eq. (6.3) are different unfoldings of the same tensor $\mathscr{D} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ for any choice of $\mathscr{X}$ and $\boldsymbol{U}_1$, $\boldsymbol{U}_2$, $\boldsymbol{U}_3$. Vectorizing both sides of the equations yields

$$\begin{aligned}
(\boldsymbol{I}_{R_3} \otimes \boldsymbol{I}_{R_2} \otimes \boldsymbol{U}_1^{\mathrm{T}}\boldsymbol{U}_1)\,\mathrm{vec}\,(\mathscr{S}) &= \mathrm{vec}\,(\mathscr{D}), \\
(\boldsymbol{I}_{R_3} \otimes \boldsymbol{U}_2^{\mathrm{T}}\boldsymbol{U}_2 \otimes \boldsymbol{I}_{R_1})\,\mathrm{vec}\,(\mathscr{S}) &= \mathrm{vec}\,(\mathscr{D}), \\
(\boldsymbol{U}_3^{\mathrm{T}}\boldsymbol{U}_3 \otimes \boldsymbol{I}_{R_2} \otimes \boldsymbol{I}_{R_1})\,\mathrm{vec}\,(\mathscr{S}) &= \mathrm{vec}\,(\mathscr{D}),
\end{aligned}$$

Equating any two out of the $\binom{3}{2}$ pairs of equations e.g. the first one with the second one results in

$$\begin{aligned}
&(\boldsymbol{I}_{R_3} \otimes \boldsymbol{I}_{R_2} \otimes \boldsymbol{U}_1^{\mathrm{T}}\boldsymbol{U}_1)\,\mathrm{vec}\,(\mathscr{S}) \\
=&(\boldsymbol{I}_{R_3} \otimes \boldsymbol{U}_2^{\mathrm{T}}\boldsymbol{U}_2 \otimes \boldsymbol{I}_{R_1})\,\mathrm{vec}\,(\mathscr{S}).
\end{aligned}$$

This equality holds if $\boldsymbol{U}_1^{\mathrm{T}}\boldsymbol{U}_1 = \boldsymbol{U}_2^{\mathrm{T}}\boldsymbol{U}_2 = \boldsymbol{0}$ (trivial solution), which we do not consider. If $\boldsymbol{U}_2^{\mathrm{T}}\boldsymbol{U}_2$ is full-rank and thus invertible, the right-hand side is invertible. Left-multiplying by the inverse of the right-hand side yields

$$\begin{aligned}
&(\boldsymbol{I}_{R_3} \otimes \boldsymbol{I}_{R_2} \otimes \boldsymbol{I}_{R_1})\,\mathrm{vec}\,(\mathscr{S}) \\
=&(\boldsymbol{I}_{R_3} \otimes \boldsymbol{U}_2^{\mathrm{T}}\boldsymbol{U}_2 \otimes \boldsymbol{I}_{R_1})^{-1}(\boldsymbol{I}_{R_3} \otimes \boldsymbol{I}_{R_2} \otimes \boldsymbol{U}_1^{\mathrm{T}}\boldsymbol{U}_1)\,\mathrm{vec}\,(\mathscr{S}) \\
=&(\boldsymbol{I}_{R_3} \otimes (\boldsymbol{U}_2^{\mathrm{T}}\boldsymbol{U}_2)^{-1} \otimes \boldsymbol{U}_1^{\mathrm{T}}\boldsymbol{U}_1)\,\mathrm{vec}\,(\mathscr{S}),
\end{aligned}$$

where the second equality follows from the mixed-product property, see Loan [33]. The equality holds if and only if $(\boldsymbol{U}_2^{\mathrm{T}}\boldsymbol{U}_2)^{-1} = \boldsymbol{I}_{R_2}$ and $\boldsymbol{U}_1^{\mathrm{T}}\boldsymbol{U}_1 = \boldsymbol{I}_{R_1}$. Repeating the argument with at least $\lceil \frac{3}{2} \rceil$ unique pairs out of the $\binom{3}{2}$ pairs of equations yields, apart from the trivial $\boldsymbol{U}_1 = \boldsymbol{U}_2 = \boldsymbol{U}_3 = \boldsymbol{0}$ solution,

$$\begin{aligned}
\boldsymbol{U}_1^{\mathrm{T}}\boldsymbol{U}_1 &= \boldsymbol{I}_{R_1}, \\
\boldsymbol{U}_2^{\mathrm{T}}\boldsymbol{U}_2 &= \boldsymbol{I}_{R_2}, \\
\boldsymbol{U}_3^{\mathrm{T}}\boldsymbol{U}_3 &= \boldsymbol{I}_{R_3},
\end{aligned} \tag{6.4}$$

which implies that $\boldsymbol{U}_1$, $\boldsymbol{U}_2$ and $\boldsymbol{U}_3$ are semi-orthogonal. Right-multiplying both sides of Eq. (6.2) by respectively $\boldsymbol{S}_{(1)}^{\mathrm{T}}\boldsymbol{U}_1^{\mathrm{T}}$, $\boldsymbol{S}_{(2)}^{\mathrm{T}}\boldsymbol{U}_2^{\mathrm{T}}$, $\boldsymbol{S}_{(3)}^{\mathrm{T}}\boldsymbol{U}_3^{\mathrm{T}}$ yields

$$\begin{aligned}
\boldsymbol{U}_1\,\boldsymbol{S}_{(1)}\boldsymbol{S}_{(1)}^{\mathrm{T}}\boldsymbol{U}_1^{\mathrm{T}} &= \boldsymbol{X}_{(1)}\,(\boldsymbol{U}_3\otimes\boldsymbol{U}_2)\,\boldsymbol{S}_{(1)}^{\mathrm{T}}\boldsymbol{U}_1^{\mathrm{T}},\\
\boldsymbol{U}_2\,\boldsymbol{S}_{(2)}\boldsymbol{S}_{(2)}^{\mathrm{T}}\boldsymbol{U}_2^{\mathrm{T}} &= \boldsymbol{X}_{(2)}\,(\boldsymbol{U}_3\otimes\boldsymbol{U}_1)\,\boldsymbol{S}_{(2)}^{\mathrm{T}}\boldsymbol{U}_2^{\mathrm{T}},\\
\boldsymbol{U}_3\,\boldsymbol{S}_{(3)}\boldsymbol{S}_{(3)}^{\mathrm{T}}\boldsymbol{U}_3^{\mathrm{T}} &= \boldsymbol{X}_{(3)}\,(\boldsymbol{U}_2\otimes\boldsymbol{U}_1)\,\boldsymbol{S}_{(3)}^{\mathrm{T}}\boldsymbol{U}_3^{\mathrm{T}}.
\end{aligned} \tag{6.5}$$

The left-hand side is the eigendecomposition of the right-hand side of Eq. (6.5) due to the assumption that $\boldsymbol{S}_{(1)}\boldsymbol{S}_{(1)}^{\mathrm{T}}$, $\boldsymbol{S}_{(2)}\boldsymbol{S}_{(2)}^{\mathrm{T}}$, $\boldsymbol{S}_{(3)}\boldsymbol{S}_{(3)}^{\mathrm{T}}$ are positive diagonal matrices (eigenvalues) and the above proof that $\boldsymbol{U}_1$, $\boldsymbol{U}_2$, $\boldsymbol{U}_3$ are semi-orthogonal matrices (eigenvectors). From Eq. (6.5) it follows that $\boldsymbol{U}_1$ is an orthogonal basis for the column space of $\boldsymbol{X}_{(1)}$, and likewise for the other unfoldings. We can therefore write

$$\begin{aligned}
\boldsymbol{U}_1\,\boldsymbol{S}_{(1)}\boldsymbol{S}_{(1)}^{\mathrm{T}}\boldsymbol{U}_1^{\mathrm{T}} &= \boldsymbol{U}_1\boldsymbol{R}_1\,(\boldsymbol{U}_3\otimes\boldsymbol{U}_2)\,\boldsymbol{S}_{(1)}^{\mathrm{T}}\boldsymbol{U}_1^{\mathrm{T}},\\
\boldsymbol{U}_2\,\boldsymbol{S}_{(2)}\boldsymbol{S}_{(2)}^{\mathrm{T}}\boldsymbol{U}_2^{\mathrm{T}} &= \boldsymbol{U}_2\boldsymbol{R}_2\,(\boldsymbol{U}_3\otimes\boldsymbol{U}_1)\,\boldsymbol{S}_{(2)}^{\mathrm{T}}\boldsymbol{U}_2^{\mathrm{T}},\\
\boldsymbol{U}_3\,\boldsymbol{S}_{(3)}\boldsymbol{S}_{(3)}^{\mathrm{T}}\boldsymbol{U}_3^{\mathrm{T}} &= \boldsymbol{U}_3\boldsymbol{R}_3\,(\boldsymbol{U}_2\otimes\boldsymbol{U}_1)\,\boldsymbol{S}_{(3)}^{\mathrm{T}}\boldsymbol{U}_3^{\mathrm{T}},
\end{aligned} \tag{6.6}$$

where $\boldsymbol{R}_1\in\mathbb{R}^{R_1\times N_2N_3}$, $\boldsymbol{R}_2\in\mathbb{R}^{R_2\times N_1N_3}$ and $\boldsymbol{R_3}\in\mathbb{R}^{R_3\times N_1N_2}$ are general coefficient matrices. From Eq. (6.6) follows that

$$\begin{aligned}
\boldsymbol{S}_{(1)} &= \boldsymbol{R}_1\,(\boldsymbol{U}_3\otimes\boldsymbol{U}_2),\\
\boldsymbol{S}_{(2)} &= \boldsymbol{R}_2\,(\boldsymbol{U}_3\otimes\boldsymbol{U}_1),\\
\boldsymbol{S}_{(3)} &= \boldsymbol{R}_3\,(\boldsymbol{U}_2\otimes\boldsymbol{U}_1),
\end{aligned} \tag{6.7}$$

which by the semi-orthogonality of $\boldsymbol{U}_1$, $\boldsymbol{U}_2$ and $\boldsymbol{U}_3$ is satisfied if and only if

$$\begin{aligned}
\boldsymbol{R}_1 &= \boldsymbol{S}_{(1)}\left(\boldsymbol{U}_3^{\mathrm{T}}\otimes\boldsymbol{U}_2^{\mathrm{T}}\right),\\
\boldsymbol{R}_2 &= \boldsymbol{S}_{(2)}\left(\boldsymbol{U}_3^{\mathrm{T}}\otimes\boldsymbol{U}_1^{\mathrm{T}}\right),\\
\boldsymbol{R}_3 &= \boldsymbol{S}_{(3)}\left(\boldsymbol{U}_2^{\mathrm{T}}\otimes\boldsymbol{U}_1^{\mathrm{T}}\right).
\end{aligned} \tag{6.8}$$

Substitution of Eq. (6.8) into $\boldsymbol{X}_{(1)}=\boldsymbol{U}_1\boldsymbol{R}_1$, $\boldsymbol{X}_{(2)}=\boldsymbol{U}_2\boldsymbol{R}_2$, $\boldsymbol{X}_{(3)}=\boldsymbol{U}_3\boldsymbol{R}_3$ we conclude that $\mathrm{vec}(\mathscr{X})=(\boldsymbol{U}_3\otimes\boldsymbol{U}_2\otimes\boldsymbol{U}_1)\,\mathrm{vec}(\mathscr{S})$, which is the defining Eq. (6.1) of the MLSVD as in Definition 6.2.1. □

Equipped with Theorem 6.3.1, we can now formulate the MLSVD as a primal-dual optimization problem by defining the *primal* MLSVD optimization problem

in its most general context.

Definition 6.3.2 (Primal MLSVD optimization problem)**.** Given three feature matrices $\boldsymbol{\Phi}_1 \in \mathbb{R}^{N_1 \times M_1}, \boldsymbol{\Phi}_2 \in \mathbb{R}^{N_2 \times M_2}, \boldsymbol{\Phi}_3 \in \mathbb{R}^{N_3 \times M_3}$, a compatibility tensor $\mathscr{C} \in \mathbb{R}^{M_1 \times M_2 \times M_3}$ and regularization parameters $\mathscr{S} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ such that $\boldsymbol{S}_{(1)}\boldsymbol{S}_{(1)}^{\mathrm{T}}$, $\boldsymbol{S}_{(2)}\boldsymbol{S}_{(2)}^{\mathrm{T}}$ and $\boldsymbol{S}_{(3)}\boldsymbol{S}_{(3)}^{\mathrm{T}}$ are positive diagonal matrices, we define the primal optimization problem as

$$
\begin{aligned}
&\max_{\boldsymbol{W}_1,\boldsymbol{W}_2,\boldsymbol{W}_3,\boldsymbol{E}_1,\boldsymbol{E}_2,\boldsymbol{E}_3} J(\boldsymbol{W}_1,\boldsymbol{W}_2,\boldsymbol{W}_3,\boldsymbol{E}_1,\boldsymbol{E}_2,\boldsymbol{E}_3) := \\
&\frac{1}{2}\sum_{d=1}^{3} \mathrm{Tr}\left(\boldsymbol{E}_d \left(\boldsymbol{S}_{(d)}\boldsymbol{S}_{(d)}^{\mathrm{T}}\right)^{-1} \boldsymbol{E}_d^{\mathrm{T}}\right) \\
&- 2\,\mathrm{vec}\left(\mathscr{C}\right)^{\mathrm{T}} \left(\boldsymbol{W}_3 \otimes \boldsymbol{W}_2 \otimes \boldsymbol{W}_1\right) \mathrm{vec}\left(\mathscr{S}\right) \\
&+ \frac{1}{2}\mathrm{vec}\left(\mathscr{C}\right)^{T} \left(\boldsymbol{\Phi}_3^{\mathrm{T}}\boldsymbol{\Phi}_3 \otimes \boldsymbol{\Phi}_2^{\mathrm{T}}\boldsymbol{\Phi}_2 \otimes \boldsymbol{\Phi}_1^{\mathrm{T}}\boldsymbol{\Phi}_1\right) \mathrm{vec}\left(\mathscr{C}\right) \\
\text{s.t.: } &\boldsymbol{E}_1 = \boldsymbol{\Phi}_1 \boldsymbol{C}_{(1)} \left(\boldsymbol{W}_3 \otimes \boldsymbol{W}_2\right) \boldsymbol{S}_{(1)}^{\mathrm{T}}, \\
&\boldsymbol{E}_2 = \boldsymbol{\Phi}_2 \boldsymbol{C}_{(2)} \left(\boldsymbol{W}_3 \otimes \boldsymbol{W}_1\right) \boldsymbol{S}_{(2)}^{\mathrm{T}}, \\
&\boldsymbol{E}_3 = \boldsymbol{\Phi}_3 \boldsymbol{C}_{(3)} \left(\boldsymbol{W}_2 \otimes \boldsymbol{W}_1\right) \boldsymbol{S}_{(3)}^{\mathrm{T}}.
\end{aligned}
\tag{6.9}
$$

In Eq. (6.9) we seek weights matrices $\boldsymbol{W}_1 \in \mathbb{R}^{M_1 \times R_1}$, $\boldsymbol{W}_2 \in \mathbb{R}^{M_2 \times R_2}$, $\boldsymbol{W}_3 \in \mathbb{R}^{M_3 \times R_3}$ and error matrices $\boldsymbol{E}_1 \in \mathbb{R}^{N_1 \times R_1}$, $\boldsymbol{E}_2 \in \mathbb{R}^{N_2 \times R_2}$, $\boldsymbol{E}_3 \in \mathbb{R}^{N_3 \times R_3}$ that maximize an objective function J composed of a term that maximizes the variance associated with each feature matrix, a regularization term that acts on the weights and an optional constant term that ensures that the cost at the optimum is zero. For now we assume that the features $\boldsymbol{\Phi}$, compatibility tensor $\mathscr{C}$ and the regularization parameter $\mathscr{S}$ are given and not necessarily data-dependent, we will later examine relevant choices. We now establish a link between the *primal* MLSVD optimization problem presented in Definition 6.3.2 and the *dual* MLSVD optimization problem.

Theorem 6.3.3. *The dual optimization problem associated with the primal optimization problem of Definition 6.3.2 is the* MLSVD *of the kernel tensor* $\mathscr{K} \in \mathbb{C}^{N_1 \times N_2 \times N_3}$*. The kernel tensor* $\mathscr{K}$ *is defined as*

$$
\mathrm{vec}\left(\mathscr{K}\right) := \left(\boldsymbol{\Phi}_3 \otimes \boldsymbol{\Phi}_2 \otimes \boldsymbol{\Phi}_1\right) \mathrm{vec}\left(\mathscr{C}\right). \tag{6.10}
$$

Proof. Consider the Lagrangian $\mathscr{L}$ associated with the primal optimization prob-

lem of Definition 6.3.2,

$$\begin{aligned}
&\mathcal{L}(\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_3, \boldsymbol{E}_1, \boldsymbol{E}_2, \boldsymbol{E}_3, \boldsymbol{U}_1, \boldsymbol{U}_2, \boldsymbol{U}_3) \\
&= J(\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_3, \boldsymbol{E}_1, \boldsymbol{E}_2, \boldsymbol{E}_3) \\
&- \operatorname{Tr}\left(\left(\boldsymbol{E}_1 - \boldsymbol{\Phi}_1 \boldsymbol{C}_{(1)} (\boldsymbol{W}_3 \otimes \boldsymbol{W}_2) \boldsymbol{S}_{(1)}^{\mathrm{T}}\right) \boldsymbol{U}_1^{\mathrm{T}}\right) \\
&- \operatorname{Tr}\left(\left(\boldsymbol{E}_2 - \boldsymbol{\Phi}_2 \boldsymbol{C}_{(2)} (\boldsymbol{W}_3 \otimes \boldsymbol{W}_1) \boldsymbol{S}_{(2)}^{\mathrm{T}}\right) \boldsymbol{U}_2^{\mathrm{T}}\right) \\
&- \operatorname{Tr}\left(\left(\boldsymbol{E}_3 - \boldsymbol{\Phi}_3 \boldsymbol{C}_{(3)} (\boldsymbol{W}_2 \otimes \boldsymbol{W}_1) \boldsymbol{S}_{(3)}^{\mathrm{T}}\right) \boldsymbol{U}_3^{\mathrm{T}}\right),
\end{aligned}$$

where $\boldsymbol{U}_1 \in \mathbb{R}^{N_1 \times R_1}$, $\boldsymbol{U}_2 \in \mathbb{R}^{N_2 \times R_2}$, $\boldsymbol{U}_3 \in \mathbb{R}^{N_3 \times R_3}$ are Lagrange multiplier matrices. The *Karush-Kuhn-Tucker* (KKT) conditions result in

$$\begin{aligned}
&\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_1} = \boldsymbol{0} \iff 2\boldsymbol{C}_{(1)}(\boldsymbol{W}_3 \otimes \boldsymbol{W}_2)\boldsymbol{S}_{(1)}^{\mathrm{T}} = \boldsymbol{C}_{(1)}\left(\boldsymbol{W}_3 \otimes \boldsymbol{\Phi}_2^{\mathrm{T}}\boldsymbol{U}_2 + \boldsymbol{\Phi}_3^{\mathrm{T}}\boldsymbol{U}_3 \otimes \boldsymbol{W}_2\right)\boldsymbol{S}_{(1)}^{\mathrm{T}}, \\
&\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_2} = \boldsymbol{0} \iff 2\boldsymbol{C}_{(2)}(\boldsymbol{W}_3 \otimes \boldsymbol{W}_1)\boldsymbol{S}_{(2)}^{\mathrm{T}} = \boldsymbol{C}_{(2)}\left(\boldsymbol{W}_3 \otimes \boldsymbol{\Phi}_1^{\mathrm{T}}\boldsymbol{U}_1 + \boldsymbol{\Phi}_3^{\mathrm{T}}\boldsymbol{U}_3 \otimes \boldsymbol{W}_1\right)\boldsymbol{S}_{(2)}^{\mathrm{T}}, \\
&\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_3} = \boldsymbol{0} \iff 2\boldsymbol{C}_{(3)}(\boldsymbol{W}_2 \otimes \boldsymbol{W}_1)\boldsymbol{S}_{(3)}^{\mathrm{T}} = \boldsymbol{C}_{(3)}\left(\boldsymbol{W}_2 \otimes \boldsymbol{\Phi}_1^{\mathrm{T}}\boldsymbol{U}_1 + \boldsymbol{\Phi}_2^{\mathrm{T}}\boldsymbol{U}_2 \otimes \boldsymbol{W}_1\right)\boldsymbol{S}_{(3)}^{\mathrm{T}}, \\
&\frac{\partial \mathcal{L}}{\partial \boldsymbol{E}_1} = \boldsymbol{0} \iff \boldsymbol{E}_1 = \boldsymbol{U}_1\boldsymbol{S}_{(1)}\boldsymbol{S}_{(1)}^{\mathrm{T}}, \\
&\frac{\partial \mathcal{L}}{\partial \boldsymbol{E}_2} = \boldsymbol{0} \iff \boldsymbol{E}_2 = \boldsymbol{U}_2\boldsymbol{S}_{(2)}\boldsymbol{S}_{(2)}^{\mathrm{T}}, \\
&\frac{\partial \mathcal{L}}{\partial \boldsymbol{E}_3} = \boldsymbol{0} \iff \boldsymbol{E}_3 = \boldsymbol{U}_3\boldsymbol{S}_{(3)}\boldsymbol{S}_{(3)}^{\mathrm{T}}, \\
&\frac{\partial \mathcal{L}}{\partial \boldsymbol{U}_1} = \boldsymbol{0} \iff \boldsymbol{E}_1 = \boldsymbol{\Phi}_1 \boldsymbol{C}_{(1)} (\boldsymbol{W}_3 \otimes \boldsymbol{W}_2) \boldsymbol{S}_{(1)}^{\mathrm{T}}, \\
&\frac{\partial \mathcal{L}}{\partial \boldsymbol{U}_2} = \boldsymbol{0} \iff \boldsymbol{E}_2 = \boldsymbol{\Phi}_2 \boldsymbol{C}_{(2)} (\boldsymbol{W}_3 \otimes \boldsymbol{W}_1) \boldsymbol{S}_{(2)}^{\mathrm{T}}, \\
&\frac{\partial \mathcal{L}}{\partial \boldsymbol{U}_3} = \boldsymbol{0} \iff \boldsymbol{E}_3 = \boldsymbol{\Phi}_3 \boldsymbol{C}_{(3)} (\boldsymbol{W}_2 \otimes \boldsymbol{W}_1) \boldsymbol{S}_{(3)}^{\mathrm{T}}.
\end{aligned}$$

The equality of the first three KKT conditions holds for the trivial solution $\boldsymbol{W}_1 = \boldsymbol{0}, \boldsymbol{W}_2 = \boldsymbol{0}, \boldsymbol{W}_3 = \boldsymbol{0}$ or when

$$\boldsymbol{\Phi}_1^{\mathrm{T}}\boldsymbol{U}_1 = \boldsymbol{W}_1,\ \boldsymbol{\Phi}_2^{\mathrm{T}}\boldsymbol{U}_2 = \boldsymbol{W}_2,\ \boldsymbol{\Phi}_3^{\mathrm{T}}\boldsymbol{U}_3 = \boldsymbol{W}_3.$$

These expressions can be used to eliminate $\boldsymbol{W}_1$ and $\boldsymbol{E}_1$ to obtain

$$\begin{aligned}\boldsymbol{U}_1\boldsymbol{S}_{(1)}\boldsymbol{S}_{(1)}^{\mathrm{T}} &= \boldsymbol{\Phi}_1\,\boldsymbol{C}_{(1)}\left(\boldsymbol{\Phi}_3^{\mathrm{T}}\otimes\boldsymbol{\Phi}_2^{\mathrm{T}}\right)(\boldsymbol{U}_3\otimes\boldsymbol{U}_2)\,\boldsymbol{S}_{(1)}^{\mathrm{T}}\\ &= \boldsymbol{K}_{(1)}\,(\boldsymbol{U}_3\otimes\boldsymbol{U}_2)\,\boldsymbol{S}_{(1)}^{\mathrm{T}}.\end{aligned}$$

The definition of $\mathscr{K}$ in Eq. (6.10) was used to write the second equality. Similarly we obtain expressions for the Lagrange multipliers $\boldsymbol{U}_2$ and $\boldsymbol{U}_3$ leading to

$$\begin{aligned}\boldsymbol{U}_1\boldsymbol{S}_{(1)} &= \boldsymbol{K}_{(1)}\,(\boldsymbol{U}_3\otimes\boldsymbol{U}_2)\,,\\ \boldsymbol{U}_2\boldsymbol{S}_{(2)} &= \boldsymbol{K}_{(2)}\,(\boldsymbol{U}_3\otimes\boldsymbol{U}_1)\,,\\ \boldsymbol{U}_3\boldsymbol{S}_{(3)} &= \boldsymbol{K}_{(3)}\,(\boldsymbol{U}_2\otimes\boldsymbol{U}_1)\,,\end{aligned}$$

which by Theorem 6.3.1 is the MLSVD of the tensor $\mathscr{K}\in\mathbb{R}^{N_1\times N_2\times N_3}$. □

Theorem 6.3.3 establishes that instead of computing the MLSVD of a tensor $\mathscr{K}$, one can alternatively solve the optimization problem in Definition 6.3.2. Equation (6.10) can be understood as a generalization of the conventional kernel equations to asymmetric kernel functions of more than two arguments. In particular, it can be interpreted as the dot product of features of three different feature spaces. Likewise, e.g. the primal $\boldsymbol{E}_1$ matrix can be interpreted as containing the dot product between the features of the first feature space with the primal weights in the second and third feature spaces. In contrast, KPCA considers only one feature space, where kernel functions are dot products within this single feature space. The $\boldsymbol{E}_1$ score variables in KPCA are the dot product between the features and their own corresponding primal weights. Our results generalize these notions to multiple data sources and feature spaces. We discuss these connections more in depth in the remainder of the paper. We now prove that the objective function J is equal to zero in the MLSVD solution.

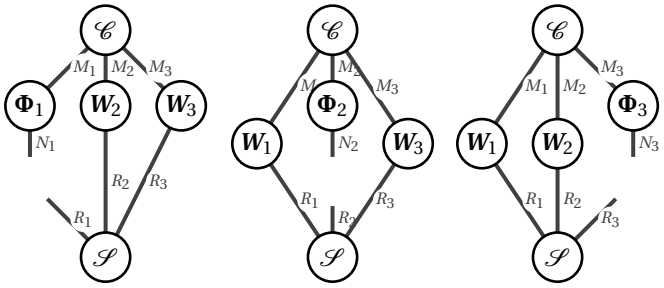
Corollary 6.3.4. *The* MLSVD *solution of the dual problem in Theorem 6.3.3 result in a zero objective function* ($J=0$) *in the primal optimization problem of Definition 6.3.2.*

Proof. Plugging in the KKT conditions for $\boldsymbol{E}_1,\boldsymbol{E}_2,\boldsymbol{E}_3$ and $\boldsymbol{W}_1,\boldsymbol{W}_2,\boldsymbol{W}_3$ into the pri-

mal objective function J results in

$$\frac{1}{2}\sum_{d=1}^{3}\mathrm{Tr}\left(\boldsymbol{E}_d\left(\boldsymbol{S}_{(d)}\boldsymbol{S}_{(d)}^{\mathrm{T}}\right)^{-1}\boldsymbol{E}_d^{\mathrm{T}}\right)-2\,\mathrm{vec}\left(\mathscr{C}\right)^{\mathrm{T}}\left(\boldsymbol{W}_3\otimes\boldsymbol{W}_2\otimes\boldsymbol{W}_1\right)\mathrm{vec}\left(\mathscr{S}\right)+\frac{1}{2}\mathrm{vec}\left(\mathscr{K}\right)^{T}\mathrm{vec}\left(\mathscr{K}\right),$$
$$=\frac{1}{2}\sum_{d=1}^{3}\mathrm{Tr}\left(\boldsymbol{U}_d\,\boldsymbol{S}_{(d)}\boldsymbol{S}_{(d)}^{\mathrm{T}}\,\boldsymbol{U}_d^{\mathrm{T}}\right)-2\,\mathrm{vec}\left(\mathscr{C}\right)^{\mathrm{T}}\left(\boldsymbol{\Phi}_3^{\mathrm{T}}\boldsymbol{U}_3\otimes\boldsymbol{\Phi}_2^{\mathrm{T}}\boldsymbol{U}_2\otimes\boldsymbol{\Phi}_1^{\mathrm{T}}\boldsymbol{U}_1\right)\mathrm{vec}\left(\mathscr{S}\right)+\frac{1}{2}\mathrm{vec}\left(\mathscr{K}\right)^{T}\mathrm{vec}\left(\mathscr{K}\right)$$
$$=\left(\frac{3}{2}-2+\frac{1}{2}\right)\mathrm{vec}\left(\mathscr{K}\right)^{T}\mathrm{vec}\left(\mathscr{K}\right)=0.$$

The first three variance terms are simplified using the cyclic permutation invariance of the Frobenius trace norm. The regularization term is simplified using the definition of the kernel tensor $\mathscr{K}$ and its MLSVD. The third term in the objective function is also simplified using the definition of the kernel tensor $\mathscr{K}$. □



(a) Primal formulation (P), from left to right of $\boldsymbol{E}_1$, $\boldsymbol{E}_2$ and $\boldsymbol{E}_3$.

Figure 6.1.: Primal formulation (Fig. 6.1a) and dual formulation (Fig. 6.1b) in tensor network diagram notation. In these diagrams, each circle represent a tensor and each edge departing from a circle represents an index of the corresponding tensor. A connecting edge denotes then a summation along the corresponding index, an unconnected edge denotes a free index, see Cichocki *et al.* [34] for a more in-depth explanation.

The primal optimization problem of Definition 6.3.2 defines explicitly a model-based approach in terms of primal weights, which is equivalent to the MLSVD but operates in the vector space $\mathbb{R}^{M_1\times M_2\times M_3}$ instead of in the usual vector space $\mathbb{R}^{N_1\times N_2\times N_3}$. Just like other learning problems that admit a primal-dual formula-

tion, each representation has its own advantages in terms of computational complexity.

Remark 6.3.5 (Primal and dual model representation). The MLSVD is characterized by a primal (P) representation in terms of weights $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, $\boldsymbol{W}_3$ and feature maps $\boldsymbol{\Phi}_1$, $\boldsymbol{\Phi}_2$, $\boldsymbol{\Phi}_3$ and a dual (D) representation in terms of a kernel tensor $\mathscr{K}$ defined in Eq. (6.10) and Lagrange multipliers $\boldsymbol{U}_1$, $\boldsymbol{U}_2$, $\boldsymbol{U}_3$:

$$
\begin{aligned}
\boldsymbol{E}_1 &= \boldsymbol{\Phi}_1 \boldsymbol{C}_{(1)} (\boldsymbol{W}_3 \otimes \boldsymbol{W}_2) \boldsymbol{S}_{(1)}^{\mathrm{T}}, \\
\boldsymbol{E}_2 &= \boldsymbol{\Phi}_2 \boldsymbol{C}_{(2)} (\boldsymbol{W}_3 \otimes \boldsymbol{W}_1) \boldsymbol{S}_{(2)}^{\mathrm{T}}, \\
\boldsymbol{E}_3 &= \boldsymbol{\Phi}_3 \boldsymbol{C}_{(3)} (\boldsymbol{W}_2 \otimes \boldsymbol{W}_1) \boldsymbol{S}_{(3)}^{\mathrm{T}},
\end{aligned} \tag{P}
$$

$$
\begin{aligned}
\boldsymbol{E}_1 &= \boldsymbol{K}_{(1)} (\boldsymbol{U}_3 \otimes \boldsymbol{U}_2) \boldsymbol{S}_{(1)}^{\mathrm{T}}, \\
\boldsymbol{E}_2 &= \boldsymbol{K}_{(2)} (\boldsymbol{U}_3 \otimes \boldsymbol{U}_1) \boldsymbol{S}_{(2)}^{\mathrm{T}}, \\
\boldsymbol{E}_3 &= \boldsymbol{K}_{(3)} (\boldsymbol{U}_2 \otimes \boldsymbol{U}_1) \boldsymbol{S}_{(3)}^{\mathrm{T}}.
\end{aligned} \tag{D}
$$

An alternative representation of Eqs. (P) and (D) in tensor networks diagram notation is presented in Fig. 6.1. After precomputing $\boldsymbol{\Phi}_1 \boldsymbol{C}_{(1)}$, $\boldsymbol{\Phi}_2 \boldsymbol{C}_{(2)}$ and $\boldsymbol{\Phi}_3 \boldsymbol{C}_{(3)}$, the primal formulation in Eq. (P) is more convenient in terms of storage and computation when the number of *samples* is larger than the feature space one operates in, i.e. $N \gg M$, requiring a computational and storage complexity of $\mathcal{O}(NM^2)$ with $N := \max(N_1, N_2, N_3)$, $M := \max(M_1, M_2, M_3)$. Alternatively, when the feature space is larger than the number of samples, the dual formulation in Eq. (D) is more attractive, requiring a computational and storage complexity of $\mathcal{O}(N^3)$. In particular, the dual formulation allows to operate implicitly in an infinite-dimensional feature space, as long as the kernel tensor $\mathscr{K}$ can be computed in closed form. We illustrate this trade-off through a numerical example.

Example 6.3.6. Suppose we have a data tensor $\mathscr{X} \in \mathbb{R}^{100 \times 100 \times 100}$ and the goal is to compute its rank-$(5,5,5)$ MLSVD. The storage complexity of the MLSVD in terms of the dual variables is then $5^3 + 3 \cdot 100 \cdot 5 = 1625$. As discussed in detail in section 6.4.1 we have that $M_1 = M_2 = M_3 = 100^2$. The storage complexity of the MLSVD in terms of the primal weights is then $5^3 + 3 \cdot 100^2 \cdot 5 = 150125$, which is 92 times more expensive than the dual form.

Now suppose we have 3 datasets $\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2, \boldsymbol{\Phi}_3$, each being a 100×5 matrix. Choosing the compatibility tensor $\mathscr{C} \in \mathbb{R}^{5 \times 5 \times 5}$ a unit diagonal tensor allows us to compute the corresponding rank-$(5,5,5)$ kernel tensor $\mathscr{K} \in \mathbb{R}^{100 \times 100 \times 100}$. Its dual MLSVD incurs a storage cost of $5^3 + 3 \cdot 100 \cdot 5 = 1625$, whereas the dual representa-

tion requires an 8-times smaller storage cost of $5^3 + 3 \cdot 5 \cdot 5 = 200$.

The situation becomes a bit more involved when the data tensor is non-cubical. Suppose we want to compute the rank-$(5,5,5)$ MLSVD of $\mathscr{X} \in \mathbb{R}^{100 \times 10 \times 5}$. Then $M_1 = 50, M_2 = 500, M_3 = 1000$. The primal representation would then have a storage cost of $5^3 + 5(50+500+1000) = 7875$, while the dual $5^3 + 5(100+10+5) = 700$. A mixed representation in terms of $\boldsymbol{W}_1, \boldsymbol{U}_2, \boldsymbol{U}_3$ would incur an even smaller storage cost of $5^3 + 5(50+10+5) = 450$. Such a mixed representation would be obtained by removing the $\boldsymbol{E}_1$ constraint from Definition (6.3.2) and solving the corresponding optimization problem.

6.4 CONNECTION WITH OTHER METHODS

We will now examine possible choices of features and compatibility tensors $\mathscr{C}$, in particular such that the output of Definition 6.3.2 is equivalent to the MLSVD of a known data tensor $\mathscr{X}$, as well as nonlinear extensions and the relationships with other methods.

6.4.1 LINEAR MLSVD OF A DATA TENSOR

Theorem 6.3.3 requires that the data that is fed into the primal optimization problem comes in the form of matrices. Its dual optimization problem is then the MLSVD of $\mathscr{K}$ in Eq. (6.10). We will now see how the MLSVD of a general data tensor $\mathscr{X}$ can be obtained with suitable choices of features and compatibility tensor. The *linear* MLSVD of a general data tensor $\mathscr{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ follows from Theorem 6.3.3 if one considers as features the mode-d unfoldings of said tensor.

Theorem 6.4.1 (Linear MLSVD of a data tensor)**.** *Consider the data tensor $\mathscr{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ with linear feature maps*

$$\boldsymbol{\Phi}_d = \boldsymbol{X}_{(d)}, \tag{6.11}$$

which correspond to each unfolding of $\mathscr{X}$. If the compatibility tensor $\mathscr{C} \in \mathbb{C}^{N_2 N_3 \times N_1 N_3 \times N_1 N_2}$ satisfies

$$\operatorname{vec}(\mathscr{X}) = \left(\boldsymbol{X}_{(3)} \otimes \boldsymbol{X}_{(2)} \otimes \boldsymbol{X}_{(1)}\right) \operatorname{vec}(\mathscr{C}), \tag{6.12}$$

then solving the primal problem in Definition 6.3.2 yields the MLSVD *of data tensor $\mathscr{X} \in \mathbb{C}^{N_1 \times N_2 \times N_3}$.*

Proof. The compatibility condition of Eq. (6.12) associated with the linear features of Eq. (6.11) satisfies the assumptions of Theorem 6.3.3, yielding thus the

MLSVD of data tensor $\mathscr{X}$. □

The linear compatibility condition of Eq. (6.12) can always be satisfied when the dimensionalty of the data tensor $D \geq 3$, as the right-hand-side is then underdetermined and thus yields infinitely many solutions. A standard choice is then to choose the $\mathscr{C}$ with minimal Frobenius norm. In the $D = 2$ case, the compatibility condition reduces to $\text{vec}(\boldsymbol{X}) = (\boldsymbol{X}^{\mathrm{T}} \otimes \boldsymbol{X})\,\text{vec}(\boldsymbol{C})$, recovering the pseudoinverse-based compatibility condition of the primal-dual formulation of the SVD identified by Suykens [9].

6.4.2 KERNEL MLSVD

The *Kernel Multilinear Singular Value Decomposition* (KMLSVD) of a data tensor $\mathscr{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ is defined by means of a set of general *nonlinear* feature maps

$$\boldsymbol{\Phi}_d = \boldsymbol{\Phi}_d(\boldsymbol{X}_d) \in \mathbb{R}^{N_d \times M_d}, \tag{6.13}$$

which map each dataset or unfolding to a higher-dimensional nonlinear space $\mathbb{R}^{M_d}$. Solving the primal optimization problem of Definition 6.3.2 is then equivalent to the MLSVD of the *kernel tensor* $\mathscr{K} \in \mathbb{C}^{N_1 \times N_2 \times N_3}$ in Theorem 6.3.3, whose defining equation we provide once more:

$$\text{vec}(\mathscr{K}) := (\boldsymbol{\Phi}_3 \otimes \boldsymbol{\Phi}_2 \otimes \boldsymbol{\Phi}_1)\ \text{vec}(\mathscr{C}).$$

The compatibility tensor $\mathscr{C}$ determines whether the kernel tensor function (and tensor kernel) are subject to any kind of symmetry or permutational invariances and together with the choice of feature map, positivity. The tensor kernel does not need to be symmetric or positive-definite, meaning that the MLSVD can represent asymmetric relationships that arise between the orthogonal subspaces where the high-dimensional data lives in terms of the core tensor.

The explicit computation of the kernel tensor scales exponentially in the dimensionality of the original data tensor $\mathscr{X}$ when carried out explicitly. This limitation can instead be bypassed by means of the so-called *kernel trick*, which carries out the computations implicitly in the features spaces. Two examples of kernel functions whose inputs live in the same space are the tensor-variate generalizations of the polynomial and exponential kernels described first by Salzo, Rosasco, and Suykens [35] and Salzo and Suykens [36] in the context of L_p-regularized learning problems. We report here their slightly adjusted definition.

Example 6.4.2 (Polynomial and exponential kernel [35, 36]). The polynomial tensor kernel function of degree $p \geq 1$ is defined as

$$\operatorname{vec}\left(\mathcal{K}^{p}_{\text{polynomial}}\right) := ((\boldsymbol{X}_3 \otimes \boldsymbol{X}_2 \otimes \boldsymbol{X}_1)\operatorname{vec}(\mathcal{I}))^{p}$$

where $\boldsymbol{X}_1 \in \mathbb{R}^{N\times M}$, $\boldsymbol{X}_2 \in \mathbb{R}^{N\times M}$, $\boldsymbol{X}_3 \in \mathbb{R}^{N\times M}$ and $\mathcal{I} \in \mathbb{C}^{M\times M\times M}$ is the 3rd-order diagonal identity tensor. It is implicitly assumed that the inputs live in the same space and can hence be coupled by means of the identity tensor. The exponential tensor kernel is then defined as

$$\mathcal{K}_{\text{exponential}} := \exp(\mathcal{K}^{1}_{\text{polynomial}}).$$

When p is odd then the kernels are not positive-definite. The exponential kernel is defined implicitly by an infinite-dimensional power series feature map [35].

Following Suykens [9], it is straightforward to define the features and compatibility tensor which yields a kernel tensor of elementwise nonlinearities.

Example 6.4.3 (Elementwise nonlinear kernel). Elementwise nonlinear kernels of a data tensor $\mathcal{X} \in \mathbb{R}^{N_1\times N_2\times N_3}$ are defined as

$$\mathcal{K}_{\text{elementwise}} := f(\mathcal{X}),$$

where $f(\cdot)$ is any elementwise nonlinear function. The features are the linear features of Theorem 6.4.1 and the compatibility tensor $\mathcal{C} \in \mathbb{R}^{M_1\times M_2\times M_3}$ satisfies Eq. (6.12).

The primal optimization problem of Definition 6.3.2 encompasses many decompositions. In what follows we provide a brief overview of some examples.

6.4.3 KERNEL ORTHOGONAL CPD

The kernel *Canonical Polyadic Decomposition* (CPD) can be interpreted as a special case of the MLSVD where the core $\mathcal{S} \in \mathbb{R}^{R\times R\times R}$ is a cubical diagonal tensor with nonzero entries and the factor matrices are not orthogonal. The orthogonal CPD [37, 38] retains the orthogonality of the factor matrices and is obtained from Theorem 6.4.1 by choosing linear feature maps and a compatibility tensor such that the compatibility equation Eq. (6.12) is satisfied.

6.4.4 KERNEL SVD

The KMLSVD generalizes the KSVD to higher-order tensors. It is therefore straightforward to obtain the KSVD of a matrix $\boldsymbol{K}$ from Theorem 6.3.3 by considering two data sources $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$ and a positive diagonal regularization matrix $\boldsymbol{S}$. Said primal KSVD optimization problem coincides with the one identified by Suykens [9]. Theorem 6.3.3 then results in the shifted eigenvalue problems

$$\begin{aligned}\boldsymbol{U}_1 \boldsymbol{S} &= \boldsymbol{K}\boldsymbol{U}_2,\\ \boldsymbol{U}_2 \boldsymbol{S}^{\mathrm{T}} &= \boldsymbol{K}^{\mathrm{T}}\boldsymbol{U}_1,\end{aligned}$$

where $\boldsymbol{K} = \boldsymbol{\Phi}_1 \boldsymbol{C} \boldsymbol{\Phi}_2^{\mathrm{T}}$ is the kernel matrix, which is not required to be symmetric or positive-definite. The solution of the shifted eigenvalue problem is then the SVD of $\boldsymbol{K}$, i.e. $\boldsymbol{K} = \boldsymbol{U}_1 \boldsymbol{S} \boldsymbol{U}_2^{\mathrm{T}}$ [31]. Similarly, the *linear* SVD [9] is also recovered by additionally considering as features rows and columns of the data matrix $\boldsymbol{X} \in \mathbb{R}^{N_1 \times N_2}$ and as compatibility matrix $\boldsymbol{C} \in \mathbb{R}^{M_1 \times M_2}$ as defined in Section 6.4.2. Notably in contrast with the KMLSVD case detailed in Section 6.4.1, in the 2-dimensional KSVD case it is possible to easily define an asymmetric kernel function $\kappa(\boldsymbol{x}_1, \boldsymbol{C}\boldsymbol{x}_2)$ based on a predefined symmetric (positive-definite) kernel function $\kappa(\cdot,\cdot)$, exploiting the fact that the compatibility matrix maps one input space to the other, as proposed by Tao *et al.* [10].

6.4.5 KERNEL PCA

Consider a feature matrix $\boldsymbol{\Phi} \in \mathbb{R}^{N \times M}$. In the primal problem of Definition 6.3.2 we are now interested in finding two weight matrices $\boldsymbol{W}_1, \boldsymbol{W}_2$ that project the data to score variables $\boldsymbol{\Phi}\boldsymbol{W}_1, \boldsymbol{\Phi}\boldsymbol{W}_2$ with maximal variance. The compatibility matrix $\boldsymbol{C}$ is chosen to be a positive diagonal matrix, and the regularization parameter matrix $\boldsymbol{S}$ is chosen diagonal with positive entries. Theorem 6.3.3 then results in the shifted eigenvalue problems

$$\begin{aligned}\boldsymbol{U}_1 \boldsymbol{S} &= \boldsymbol{K}\boldsymbol{U}_2,\\ \boldsymbol{U}_2 \boldsymbol{S} &= \boldsymbol{K}\boldsymbol{U}_1,\end{aligned}$$

where $\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{C}\boldsymbol{\Phi}^{\mathrm{T}}$ is a symmetric positive-definite kernel matrix. The symmetry of the kernel matrix implies that $\boldsymbol{U}_1 = \boldsymbol{U}_2$, ensuring that the dual problem is the eigenvalue decomposition of the kernel matrix i.e. $\boldsymbol{K} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{U}^{\mathrm{T}}$. Linear PCA is recovered by choosing $\boldsymbol{\Phi} = \boldsymbol{X}$.

6.4.6 HIGHER-ORDER KPCA

Theorem 6.3.3 makes it possible to define a higher-order KPCA which considers higher-order interactions between the (mapped) data. Consider e.g. three identical features $\boldsymbol{\Phi}_1 = \boldsymbol{\Phi}_2 = \boldsymbol{\Phi}_3 = \boldsymbol{\Phi}$ and akin to KPCA a supersymmetric compatibility tensor $\mathscr{C}$ and positive superdiagonal regularization $\mathscr{S}$. Then by Theorem 6.3.3 the dual optimization problem is

$$\begin{aligned}\boldsymbol{U}_1\boldsymbol{S} &= \boldsymbol{K}(\boldsymbol{U}_3 \otimes \boldsymbol{U}_2),\\ \boldsymbol{U}_2\boldsymbol{S} &= \boldsymbol{K}(\boldsymbol{U}_3 \otimes \boldsymbol{U}_1),\\ \boldsymbol{U}_3\boldsymbol{S} &= \boldsymbol{K}(\boldsymbol{U}_2 \otimes \boldsymbol{U}_1),\end{aligned}$$

where $\boldsymbol{K} = \boldsymbol{K}_{(1)} = \boldsymbol{K}_{(2)} =: \boldsymbol{K}_{(3)}$ and $\boldsymbol{S}_{(1)} = \boldsymbol{S}_{(2)} = \boldsymbol{S}_{(3)} =: \boldsymbol{S}$ by construction. Consequently by the orthogonality of $\boldsymbol{U}_1$, $\boldsymbol{U}_2$ and $\boldsymbol{U}_3$ it follows that $\boldsymbol{U}_1 = \boldsymbol{U}_2 = \boldsymbol{U}_3 =: \boldsymbol{U}$, resulting in $\boldsymbol{U}\boldsymbol{S} = \boldsymbol{K}(\boldsymbol{U} \otimes \boldsymbol{U})$. Examples of kernel tensors that encode said higher-order interactions are the polynomial and exponential kernels in Example 6.4.2 [35, 36].

6.5 RELATED WORK

As already mentioned, the development of a primal-dual formulation of PCA was carried out by Suykens *et al.* [8], but the idea to kernelize the method is older and generally attributed to Mika *et al.* [7].

The SVD was cast in the primal-dual framework by Suykens [9], who proposed to kernelize the approach. This idea was recently further developed by Tao *et al.* [10], who proposed to use the compatibility matrix to build asymmetric kernels departing from standard symmetric and positive-definite kernels. Tao *et al.* [10] also extended the Nyström method in order to efficiently approximate said asymmetric kernels, enabling to fully exploit the computational advantages that stem from the primal KSVD optimization problem. Chen *et al.* [39, 40] proposed to decompose the self-attention kernel matrix in *Transformer* networks [41] using the primal formulation of the KSVD. This enables to fully capture its asymmetric nature in contrast with the existing alternatives, which consider only the row or column-space and are therefore effectively discarding information. He *et al.* [42] considered asymmetric kernels in the LS-SVM primal-dual formulation and consider applications in the context of directed graphs, where asymmetry is naturally present. He *et al.* [43] extended the celebrated *Random Fourier Features* (RFF) [44] kernel approximation framework to handle asymmetric kernels.

The MLSVD and the related Tucker decomposition [45, 46], which relaxes the orthogonality constraint on the factor matrices, has applications in signal and image processing, computer vision, chemometrics, finance, human motion analysis, data mining, machine learning and deep learning. We redirect the interested reader to the survey papers of Kolda and Bader [32], Cichocki *et al.* [34], Cichocki *et al.* [47], and Panagakis *et al.* [48]. Kernelizing the MLSVD was attempted by Li, Du, and Lin [49] and Zhao *et al.* [50] who proposed to map each unfolding of a data tensor to the *same* feature space, whose left-singular vectors are the factor matrices of the decomposition. The approach can be easily encompassed in Definition 6.3.2, which is more general as it allows for asymmetry and permutational variance by selecting the core tensor appropriately.

6.6 CONCLUSIONS, FURTHER RESEARCH AND APPLICATIONS

In this paper we extend the Lanczos decomposition theorem to the tensor case, enabling us to cast the MLSVD as a primal-dual optimization problem. This allows for a straightforward nonlinear extension in terms of feature maps in the primal, whose associated dual optimization problem is then the MLSVD of a kernel tensor. Importantly, the presented optimization framework recovers as special cases both SVD and PCA. Besides the nonlinear extension, the benefits of having the MLSVD defined in terms of a primal and a dual optimization problem are computational in nature. In particular, the practitioner can opt for solving the cheapest optimization problem given the circumstances. The primal formulation, as typically the case in kernel methods, is more convenient when dealing with large sample sizes and relatively small features, while the dual allows to tackle the case where the features are large w.r.t the number of samples, or even infinite-dimensional.

In our opinion, further research is needed, specifically focusing on the theory, analysis, and design of kernel functions with more than two vector inputs. This research could explore the design of generally applicable kernel functions, such as the e.g. the ubiquitous Gaussian kernel in the two input case, or be driven by specific applications. An interesting research direction would then also consist in investigating the possible decomposition or approximation of said kernels using feature maps and core tensors, in order to fully leverage the computational advantages of the primal formulation. The proposed approach can in principle be utilized as a nonlinear extension of MLSVD and thus be employed whenever its

linear counterpart is used, which means e.g. for feature extraction, signal analysis, image processing and computer vision. An notable area of possible application is deep learning, where the primal formulation of the MLSVD could be applied to approximate nonlinear kernel tensors which often arise. Two examples are the activated convolution kernel in *Convolutional Neural Network* (CNN), which thus far has been trained decomposed in Tucker form before being activated, or generalizations of the self-attention mechanism in *Transformer* having multiple attention vectors instead of only queries, values and keys.

REFERENCES

[1] F. Wesel and K. Batselier. *A Kernelizable Primal-Dual Formulation of the Multilinear Singular Value Decomposition*. Oct. 2024. arXiv: 2410.10504.

[2] B. E. Boser, I. M. Guyon, and V. N. Vapnik. "A Training Algorithm for Optimal Margin Classifiers". In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT '92. New York, NY, USA: Association for Computing Machinery, July 1992, pp. 144–152.

[3] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press, 2002.

[4] J. Suykens and J. Vandewalle. "Least Squares Support Vector Machine Classifiers". In: *Neural Processing Letters* 9.3 (June 1999), pp. 293–300.

[5] C. Saunders, A. Gammerman, and V. Vovk. "Ridge Regression Learning Algorithm in Dual Variables". In: *Proceedings of the Fifteenth International Conference on Machine Learning*. ICML '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., July 1998, pp. 515–521.

[6] V. Roth. "The Generalized LASSO". In: *IEEE Transactions on Neural Networks* 15.1 (Jan. 2004), pp. 16–28.

[7] S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. "Kernel PCA and De-Noising in Feature Spaces". In: *Advances in Neural Information Processing Systems*. Vol. 11. MIT Press, 1998.

[8] J. Suykens, T. Van Gestel, J. Vandewalle, and B. De Moor. "A Support Vector Machine Formulation to PCA Analysis and Its Kernel Version". In: *IEEE Transactions on Neural Networks* 14.2 (Mar. 2003), pp. 447–450.

[9] J. A. K. Suykens. "SVD Revisited: A New Variational Principle, Compatible Feature Maps and Nonlinear Extensions". In: *Applied and Computational Harmonic Analysis* 40.3 (May 2016), pp. 600–609.

[10] Q. Tao, F. Tonin, P. Patrinos, and J. A. K. Suykens. *Nonlinear SVD with Asymmetric Kernels: Feature Learning and Asymmetric Nyström Method*. June 2023. arXiv: 2306.07040 [cs].

[11] L. De Lathauwer, B. De Moor, and J. Vandewalle. "A Multilinear Singular Value Decomposition". In: *SIAM Journal on Matrix Analysis and Applications* 21.4 (Jan. 2000), pp. 1253–1278.

[12] M. A. O. Vasilescu and D. Terzopoulos. “Multilinear Analysis of Image Ensembles: TensorFaces”. In: *Computer Vision — ECCV 2002*. Ed. by A. Heyden, G. Sparr, M. Nielsen, and P. Johansen. Berlin, Heidelberg: Springer, 2002, pp. 447–460.

[13] M. Vasilescu. “Human Motion Signatures: Analysis, Synthesis, Recognition”. In: *2002 International Conference on Pattern Recognition*. Vol. 3. Aug. 2002, 456–460 vol.3.

[14] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin. “Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications”. In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2016.

[15] A.-H. Phan, K. Sobolev, K. Sozykin, D. Ermilov, J. Gusak, P. Tichavský, V. Glukhov, I. Oseledets, and A. Cichocki. “Stable Low-Rank Tensor Decomposition for Compression of Convolutional Neural Network”. In: *Computer Vision – ECCV 2020*. Ed. by A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm. Cham: Springer International Publishing, 2020, pp. 522–539.

[16] S. Jie and Z.-H. Deng. “FacT: Factor-Tuning for Lightweight Adaptation on Vision Transformer”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 37.1 (June 2023), pp. 1060–1068.

[17] E. Zangrando, S. Schotthöfer, G. Ceruti, J. Kusch, and F. Tudisco. “Geometry-Aware Training of Factorized Layers in Tensor Tucker Format”. In: *Advances in Neural Information Processing Systems* 37 (Jan. 2025), pp. 129743–129773.

[18] M. Filipović and A. Jukić. “Tucker Factorization with Missing Data with Application to Low-$$n$$-Rank Tensor Completion”. In: *Multidimensional Systems and Signal Processing* 26.3 (July 2015), pp. 677–692.

[19] I. Balažević, C. Allen, and T. M. Hospedales. “TuckER: Tensor Factorization for Knowledge Graph Completion”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 5184–5193. arXiv: 1901.09590 [cs].

[20] T. Wen, E. Chen, and Y. Chen. “Tensor-View Topological Graph Neural Network”. In: *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2024, pp. 4330–4338.

[21] N. Cohen, O. Sharir, and A. Shashua. “On the Expressive Power of Deep Learning: A Tensor Analysis”. In: *Conference on Learning Theory*. PMLR, June 2016, pp. 698–728.

[22] P. Milanesi, H. Kadri, S. Ayache, and T. Artières. “Implicit Regularization in Deep Tensor Factorization”. In: *2021 International Joint Conference on Neural Networks (IJCNN)*. July 2021, pp. 1–8.

[23] K. Hariz, H. Kadri, S. Ayache, M. Moakher, and T. Artières. “Implicit Regularization in Deep Tucker Factorization: Low-Rankness via Structured Sparsity”. In: *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2024, pp. 2359–2367.

[24] O. A. Malik and S. Becker. “Low-Rank Tucker Decomposition of Large Tensors Using TensorSketch”. In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018.

[25] A. Traore, M. Berar, and A. Rakotomamonjy. “Singleshot : A Scalable Tucker Tensor Decomposition”. In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019.

[26] K. Ghalamkari and M. Sugiyama. “Fast Tucker Rank Reduction for Non-Negative Tensors Using Mean-Field Approximation”. In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 443–454.

[27] T. Heo and C. Bajaj. “Sample Efficient Learning of Factored Embeddings of Tensor Fields”. In: *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2024, pp. 4591–4599.

[28] J. Hood and A. J. Schein. “The AL0CORE Tensor Decomposition for Sparse Count Data”. In: *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2024, pp. 4654–4662.

[29] S. Fang, R. M. Kirby, and S. Zhe. “Bayesian Streaming Sparse Tucker Decomposition”. In: *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*. PMLR, Dec. 2021, pp. 558–567.

[30] S. Fang, A. Narayan, R. Kirby, and S. Zhe. “Bayesian Continuous-Time Tucker Decomposition”. In: *Proceedings of the 39th International Conference on Machine Learning*. PMLR, June 2022, pp. 6235–6245.

[31] C. Lanczos. “Linear Systems in Self-Adjoint Form”. In: *The American Mathematical Monthly* (Nov. 1958).

[32] T. G. Kolda and B. W. Bader. “Tensor Decompositions and Applications”. In: *SIAM Review* 51.3 (Aug. 2009), pp. 455–500.

[33] C. F. V. Loan. “The Ubiquitous Kronecker Product”. In: *Journal of Computational and Applied Mathematics*. Numerical Analysis 2000. Vol. III: Linear Algebra 123.1 (Nov. 2000), pp. 85–100.

[34] 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”. In: *Foundations and Trends® in Machine Learning* 9.4-5 (2016), pp. 249–429.

[35] S. Salzo, L. Rosasco, and J. Suykens. “Solving Lp-Norm Regularization with Tensor Kernels”. In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. PMLR, Mar. 2018, pp. 1655–1663.

[36] S. Salzo and J. A. K. Suykens. “Generalized Support Vector Regression: Duality and Tensor-Kernel Representation”. In: *Analysis and Applications* 18.01 (Jan. 2020), pp. 149–183.

[37] T. G. Kolda. “Orthogonal Tensor Decompositions”. In: *SIAM Journal on Matrix Analysis and Applications* 23.1 (Jan. 2001), pp. 243–255.

[38] M. Sørensen, L. D. Lathauwer, P. Comon, S. Icart, and L. Deneire. “Canonical Polyadic Decomposition with a Columnwise Orthonormal Factor Matrix”. In: *SIAM Journal on Matrix Analysis and Applications* 33.4 (Jan. 2012), pp. 1190–1213.

[39] Y. Chen, Q. Tao, F. Tonin, and J. Suykens. “Primal-Attention: Self-attention through Asymmetric Kernel SVD in Primal Representation”. In: *Advances in Neural Information Processing Systems* 36 (Dec. 2023), pp. 65088–65101.

[40] Y. Chen, Q. Tao, F. Tonin, and J. A. K. Suykens. *Self-Attention through Kernel-Eigen Pair Sparse Variational Gaussian Processes*. May 2024. arXiv: 2402.01476 [cs, stat].

[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. ukasz Kaiser, and I. Polosukhin. “Attention Is All You Need”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[42] M. He, F. He, L. Shi, X. Huang, and J. A. K. Suykens. “Learning With Asymmetric Kernels: Least Squares and Feature Interpretation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.8 (Aug. 2023), pp. 10044–10054.

[43] M. He, F. He, F. Liu, and X. Huang. “Random Fourier Features for Asymmetric Kernels”. In: *Machine Learning* (Sept. 2024).

[44] A. Rahimi and B. Recht. “Random Features for Large-Scale Kernel Machines”. In: *Proceedings of the 20th International Conference on Neural Information Processing Systems*. Curran Associates Inc., Dec. 2007, pp. 1177–1184.

[45] L. R. Tucker. “Implications of Factor Analysis of Three-Way Matrices for Measurement of Change”. In: *Problems in measuring change* 15.122-137 (1963), p. 3.

[46] L. R. Tucker. “Some Mathematical Notes on Three-Mode Factor Analysis”. In: *Psychometrika* 31.3 (Sept. 1966), pp. 279–311.

[47] A. Cichocki, A.-H. Phan, Q. Zhao, N. Lee, I. V. Oseledets, M. Sugiyama, and D. Mandic. “Tensor Networks for Dimensionality Reduction and Large-Scale Optimizations. Part 2 Applications and Future Perspectives”. In: *Foundations and Trends® in Machine Learning* 9.6 (2017), pp. 249–429. arXiv: 1708.09165.

[48] Y. Panagakis, J. Kossaifi, G. G. Chrysos, J. Oldfield, M. A. Nicolaou, A. Anandkumar, and S. Zafeiriou. “Tensor Methods in Computer Vision and Deep Learning”. In: *Proceedings of the IEEE* 109.5 (May 2021), pp. 863–890.

[49] Y. Li, Y. Du, and X. Lin. “Kernel-Based Multifactor Analysis for Image Synthesis and Recognition”. In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. Vol. 1. Oct. 2005, 114–119 Vol. 1.

[50] 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". In: *IEEE Signal Processing Magazine* 30.4 (July 2013), pp. 137–148.

A A PRIMAL-DUAL FORMULATION FOR THE MLSVD

For completeness we present the generalized Lanczos and primal MLSVD optimization problem for any tensor of order D.

Theorem A.1 (Generalized Lanczos decomposition theorem)**.** *An arbitrary rank-$(R_1, R_2, \ldots, R_D)$ tensor $\mathscr{X} \in \mathbb{R}^{N_1 \times N_2 \times \cdots \times N_D}$ can be written in* MLSVD *form, i.e. as in Eq.* (6.1) *with core tensor $\mathscr{S} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_D}$ and D semi-orthogonal factor matrices $\boldsymbol{U}_d \in \mathbb{R}^{N_d \times R_d}$ defined by the following set of equations*

$$\boldsymbol{U}_d \boldsymbol{S}_{(d)} = \boldsymbol{X}_{(d)} \left(\boldsymbol{U}_D \otimes \boldsymbol{U}_{D-1} \otimes \boldsymbol{U}_{d+1} \otimes \boldsymbol{U}_{d-1} \otimes \cdots \otimes \boldsymbol{U}_1\right), \tag{.14}$$

with the D additional constraint that $\boldsymbol{S}_{(d)}\boldsymbol{S}_{(d)}^{\mathrm{T}}$ are positive diagonal matrices.

Proof. The proof is divided in two steps, first we show that the D factor matrices $\boldsymbol{U}_d$ are semi-orthogonal, second we show that indeed Eq. (.14) implies the MLSVD i.e. Eq. (6.1). We begin by left-multiplying each side of Eq. (.14) respectively with $\boldsymbol{U}_d^{\mathrm{T}}$, resulting in equations of the form

$$\boldsymbol{U}_d^{\mathrm{T}}\boldsymbol{U}_d \boldsymbol{S}_{(d)} = \boldsymbol{U}_d^{\mathrm{T}} \boldsymbol{X}_{(d)} \left(\boldsymbol{U}_D \otimes \boldsymbol{U}_{D-1} \otimes \boldsymbol{U}_{d+1} \otimes \boldsymbol{U}_{d-1} \otimes \cdots \otimes \boldsymbol{U}_1\right), \tag{.15}$$

By construction, the right-hand sides of Eq. (.15) are different unfoldings of the same tensor $\mathscr{D} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_D}$ for any choice of $\mathscr{X}$ and $\boldsymbol{U}_d$. Vectorizing both sides of the equation yields

$$\left(\boldsymbol{I}_{R_D} \otimes \boldsymbol{I}_{R_{D-1}} \otimes \cdots \otimes \boldsymbol{U}_d^{\mathrm{T}}\boldsymbol{U}_d \otimes \cdots \otimes \boldsymbol{I}_{R_1}\right) \operatorname{vec}(\mathscr{S}) = \operatorname{vec}(\mathscr{D}),$$

Equating any two left-hand sides out of the $\binom{D}{2}$ pairs of equations results in $\boldsymbol{U}_d^{\mathrm{T}}\boldsymbol{U}_d = \boldsymbol{d}$ (trivial solution), which we do not consider, or if $\boldsymbol{U}_d^{\mathrm{T}}\boldsymbol{U}_d$ is full-rank and thus invertible, and by from the mixed-product property, see Loan [33] the equality holds if and only if $(\boldsymbol{U}_d^{\mathrm{T}}\boldsymbol{U}_d)^{-1} = \boldsymbol{I}_{R_d}$. Repeating the argument with at least $\lceil \frac{D}{2} \rceil$ unique pairs out of the $\binom{D}{2}$ pairs of equations yields, apart from the trivial $\boldsymbol{U}_d = \boldsymbol{0}$ solution,

$$\boldsymbol{U}_d^{\mathrm{T}}\boldsymbol{U}_d = \boldsymbol{I}_{R_d}, \tag{.16}$$

which implies that any $\boldsymbol{U}_d$ is semi-orthogonal. Right-multiplying both sides of

Eq. (.14) by $\boldsymbol{S}_{(d)}^{\mathrm{T}}\boldsymbol{U}_d^{\mathrm{T}}$ yields

$$\boldsymbol{U}_d\,\boldsymbol{S}_{(d)}\boldsymbol{S}_{(d)}^{\mathrm{T}}\boldsymbol{U}_d^{\mathrm{T}} = \boldsymbol{X}_{(d)}\,(\boldsymbol{U}_D \otimes \boldsymbol{U}_{D-1} \otimes \cdots \otimes \boldsymbol{U}_{d+1} \otimes \boldsymbol{U}_{d-1} \otimes \cdots \boldsymbol{U}_1)\,\boldsymbol{S}_{(d)}^{\mathrm{T}}\boldsymbol{U}_d^{\mathrm{T}}. \tag{.17}$$

The left-hand side is the eigendecomposition of the right-hand side of Eq. (.17) due to the assumption that $\boldsymbol{S}_{(d)}\boldsymbol{S}_{(d)}^{\mathrm{T}}$ are positive diagonal matrices (eigenvalues) and the above proof that $\boldsymbol{U}_d$ are semi-orthogonal matrices (eigenvectors). From Eq. (.17) it follows that $\boldsymbol{U}_d$ is an orthogonal basis for the column space of $\boldsymbol{X}_{(d)}$, and likewise for the other unfoldings. We can therefore write

$$\boldsymbol{U}_d\,\boldsymbol{S}_{(d)}\boldsymbol{S}_{(d)}^{\mathrm{T}}\boldsymbol{U}_d^{\mathrm{T}} = \boldsymbol{U}_d\boldsymbol{R}_d\,(\boldsymbol{U}_D \otimes \boldsymbol{U}_{D-1} \otimes \cdots \otimes \boldsymbol{U}_{d+1} \otimes \boldsymbol{U}_{d-1} \otimes \cdots \boldsymbol{U}_1)\,\boldsymbol{S}_{(d)}^{\mathrm{T}}\boldsymbol{U}_d^{\mathrm{T}}. \tag{.18}$$

where $\boldsymbol{R}_d \in \mathbb{R}^{R_d \times N_D N_{D-1} \cdots N_{d+1} N_{d-1} \cdots N_1}$ are general coefficient matrices. From Eq. (.18) follows that

$$\boldsymbol{S}_{(d)} = \boldsymbol{R}_d\,(\boldsymbol{U}_D \otimes \boldsymbol{U}_{D-1} \otimes \cdots \otimes \boldsymbol{U}_{d+1} \otimes \boldsymbol{U}_{d-1} \otimes \cdots \boldsymbol{U}_1)\,, \tag{.19}$$

which by the semi-orthogonality of $\boldsymbol{U}_d$ is satisfied if and only if

$$\boldsymbol{R}_d = \boldsymbol{S}_{(d)}\,(\boldsymbol{U}_D \otimes \boldsymbol{U}_{D-1} \otimes \cdots \otimes \boldsymbol{U}_{d+1} \otimes \boldsymbol{U}_{d-1} \otimes \cdots \boldsymbol{U}_1)\,, \tag{.20}$$

Substitution of Eq. (.20) into $\boldsymbol{X}_{(d)} = \boldsymbol{U}_d\,\boldsymbol{R}_d$ we conclude that
$\mathrm{vec}\,(\mathscr{X}) = (\boldsymbol{U}_D \otimes \boldsymbol{U}_{D-1} \otimes \cdots \otimes \boldsymbol{U}_1)\,\mathrm{vec}\,(\mathscr{S})$, which is the defining Eq. (6.1) of the MLSVD as in Definition 6.2.1. □

Definition A.2 (Primal MLSVD optimization problem)**.** Given D feature matrices $\boldsymbol{\Phi}_1 \in \mathbb{R}^{N_1 \times M_1}, \boldsymbol{\Phi}_2 \in \mathbb{R}^{N_2 \times M_2}, \ldots, \boldsymbol{\Phi}_D \in \mathbb{R}^{N_D \times M_D}$, a compatibility tensor $\mathscr{C} \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_D}$, and regularization parameters $\mathscr{S} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_D}$, we define the primal optimiza-

tion problem as

$$\max_{\boldsymbol{W}_1,\ldots,\boldsymbol{W}_D,\boldsymbol{E}_1,\ldots,\boldsymbol{E}_D} J(\boldsymbol{W}_1,\ldots,\boldsymbol{W}_D,\boldsymbol{E}_1,\ldots,\boldsymbol{E}_D) :=$$
$$\frac{1}{2}\sum_{d=1}^{D} \operatorname{Tr}\left(\boldsymbol{E}_d \left(\boldsymbol{S}_{(d)}\boldsymbol{S}_{(d)}^{\mathrm{T}}\right)^{-1}\boldsymbol{E}_d^{\mathrm{T}}\right) - (D-1)\operatorname{vec}(\mathscr{C})^{\mathrm{T}}\left(\boldsymbol{W}_D\otimes\cdots\otimes\boldsymbol{W}_1\right)\operatorname{vec}(\mathscr{S})$$
$$+\frac{D-2}{2}\operatorname{vec}(\mathscr{C})^{T}\left(\boldsymbol{\Phi}_D^{\mathrm{T}}\boldsymbol{\Phi}_D\otimes\cdots\otimes\boldsymbol{\Phi}_1^{\mathrm{T}}\boldsymbol{\Phi}_1\right)\operatorname{vec}(\mathscr{C})$$
such that: $\boldsymbol{E}_1 = \boldsymbol{\Phi}_1\boldsymbol{C}_{(1)}\left(\boldsymbol{W}_D\otimes\cdots\otimes\boldsymbol{W}_3\otimes\boldsymbol{W}_2\right)\boldsymbol{S}_{(1)}^{\mathrm{T}}$,
$\boldsymbol{E}_2 = \boldsymbol{\Phi}_2\boldsymbol{C}_{(2)}\left(\boldsymbol{W}_D\otimes\cdots\otimes\boldsymbol{W}_3\otimes\boldsymbol{W}_1\right)\boldsymbol{S}_{(2)}^{\mathrm{T}}$,
$\vdots$
$\boldsymbol{E}_D = \boldsymbol{\Phi}_D\boldsymbol{C}_{(D)}\left(\boldsymbol{W}_{D-1}\otimes\cdots\otimes\boldsymbol{W}_2\otimes\boldsymbol{W}_1\right)\boldsymbol{S}_{(D)}^{\mathrm{T}}$.

Definition A.2 reduces to Definition 6.3.2 when $D = 3$.

Theorem A.3. *The primal optimization problem of Definition A.2 is equivalent to the* MLSVD *of the tensor* $\mathscr{K} \in \mathbb{C}^{N_1\times N_2\times\cdots\times N_D}$. *The tensor* $\mathscr{K}$ *is defined as*

$$\operatorname{vec}(\mathscr{K}) := \left(\boldsymbol{\Phi}_D\otimes\cdots\otimes\boldsymbol{\Phi}_2\otimes\boldsymbol{\Phi}_1\right)\operatorname{vec}(\mathscr{C}). \tag{.21}$$

Proof. The corresponding Lagrangian for the optimization problem in A.2 has D Lagrange multiplier matrices $\boldsymbol{U}_1,\ldots,\boldsymbol{U}_D$. We write out the KKT conditions for $\boldsymbol{W}_1, \boldsymbol{E}_1$ and $\boldsymbol{U}_1$, as the remaining conditions are similar.

$$\frac{\partial\mathscr{L}}{\partial\boldsymbol{W}_1} = \boldsymbol{0} \iff (D-1)\,\boldsymbol{C}_{(1)}(\boldsymbol{W}_D\otimes\cdots\otimes\boldsymbol{W}_2)\,\boldsymbol{S}_{(1)}^{\mathrm{T}} = \boldsymbol{C}_{(1)}\underbrace{\left(\boldsymbol{W}_D\otimes\cdots\otimes\boldsymbol{\Phi}_2^{\mathrm{T}}\boldsymbol{U}_2+\cdots+\boldsymbol{\Phi}_D^{\mathrm{T}}\boldsymbol{U}_D\otimes\cdots\otimes\boldsymbol{W}_2\right)}_{(D-1)\text{ terms}}\boldsymbol{S}_{(1)}^{\mathrm{T}}.$$

The KKT conditions for the remaining weight matrices will have similar form and all equalities are trivially satisfied when $\boldsymbol{W}_d = \boldsymbol{0}$ or $\boldsymbol{W}_d = \boldsymbol{\Phi}_d^{\mathrm{T}}\boldsymbol{U}_d$ $(1 \le d \le D)$. Setting the partial derivative of the Lagrangian with respect to $\boldsymbol{E}_1$ to zero results in the condition

$$\frac{\partial\mathscr{L}}{\partial\boldsymbol{E}_1} = \boldsymbol{0} \iff \boldsymbol{E}_1 = \boldsymbol{U}_1\boldsymbol{S}_{(1)}\boldsymbol{S}_{(1)}^{\mathrm{T}},$$

and likewise for the other error matrices. Substitution of $\boldsymbol{W}_1$ and $\boldsymbol{E}_1$ into the con-

straint results in

$$\begin{aligned}\boldsymbol{U}_1 \boldsymbol{S}_{(1)} \boldsymbol{S}_{(1)}^{\mathrm{T}} &= \boldsymbol{\Phi}_1 \boldsymbol{C}_{(1)} \left(\boldsymbol{W}_D \otimes \cdots \otimes \boldsymbol{W}_3 \otimes \boldsymbol{W}_2\right) \boldsymbol{S}_{(1)}^{\mathrm{T}}, \\ &= \boldsymbol{\Phi}_1 \boldsymbol{C}_{(1)} \left(\boldsymbol{\Phi}_D^{\mathrm{T}} \otimes \cdots \otimes \boldsymbol{\Phi}_2^{\mathrm{T}}\right) \left(\boldsymbol{U}_D \otimes \cdots \otimes \boldsymbol{U}_2\right) \boldsymbol{S}_{(1)}^{\mathrm{T}}, \\ &= \boldsymbol{K}_{(1)} \left(\boldsymbol{U}_D \otimes \cdots \otimes \boldsymbol{U}_2\right) \boldsymbol{S}_{(1)}^{\mathrm{T}},\end{aligned}$$

which is the first equation of the generalized Lanczos theorem. A similar construction applies for the remaining equations. □

Corollary A.4. *The* MLSVD *solution of the dual problem in Theorem A.3 results in a zero objective function* ($J = 0$) *in the primal optimization problem of Definition A.2.*

Proof. Plugging in the KKT conditions for $\boldsymbol{E}_1, \ldots, \boldsymbol{E}_D$ and $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_D$ into the primal objective function J results in

$$\begin{aligned}&\frac{1}{2} \sum_{d=1}^{D} \mathrm{Tr}\left(\boldsymbol{E}_d \left(\boldsymbol{S}_{(d)} \boldsymbol{S}_{(d)}^{\mathrm{T}}\right)^{-1} \boldsymbol{E}_d^{\mathrm{T}}\right) - (D-1) \operatorname{vec}(\mathscr{C})^{\mathrm{T}} \left(\boldsymbol{W}_D \otimes \cdots \otimes \boldsymbol{W}_2 \otimes \boldsymbol{W}_1\right) \operatorname{vec}(\mathscr{S}) \\ &+ \frac{(D-2)}{2} \operatorname{vec}(\mathscr{K})^T \operatorname{vec}(\mathscr{K}) \\ &= \frac{1}{2} \sum_{d=1}^{D} \mathrm{Tr}\left(\boldsymbol{U}_d \boldsymbol{S}_{(d)} \boldsymbol{S}_{(d)}^{\mathrm{T}} \boldsymbol{U}_d^{\mathrm{T}}\right) - (D-1) \operatorname{vec}(\mathscr{C})^{\mathrm{T}} \left(\boldsymbol{\Phi}_D^{\mathrm{T}} \boldsymbol{U}_D \otimes \cdots \otimes \boldsymbol{\Phi}_2^{\mathrm{T}} \boldsymbol{U}_2 \otimes \boldsymbol{\Phi}_1^{\mathrm{T}} \boldsymbol{U}_1\right) \operatorname{vec}(\mathscr{S}) \\ &+ \frac{1}{2} \operatorname{vec}(\mathscr{K})^T \operatorname{vec}(\mathscr{K}) \\ &= \left(\frac{D}{2} - (D-1) + \frac{(D-2)}{2}\right) \operatorname{vec}(\mathscr{K})^T \operatorname{vec}(\mathscr{K}) = 0.\end{aligned}$$

□

F

G

CONCLUSION

This thesis has made contributions to the understanding and application of *Tensor Networks* (TNs) in the context of kernel machines. By exploring the interplay between TNs and kernel machines, we have developed novel methodologies that enhance the scalability and efficiency of kernel-based models, while also uncovering theoretical connections to *Gaussian Processes* (GPs) and optimization.

The first major contribution of this work lies in demonstrating how TNs can be leveraged to accelerate and scale kernel machines. By imposing low-rank TN constraints on the tensorized model weights, we have shown that it is possible to learn kernel machines with linear complexity in the dimensionality of the data *while implicitly approximating kernel functions up to machine precision.* This was achieved through the exploration of Fourier features for the approximation of stationary product kernels in Chapter 2, Nyström approximations for the approximation of arbitrary product kernels in Chapter 3, and quantized models in Chapter 4. The quantized TN-constrained models, in particular, offer a compelling advantage by achieving higher *Vapnik–Chervonenkis* (VC)-dimension bounds with fewer parameters, enabling faster learning without sacrificing any model expressivity.

The second key contribution is the establishment of a theoretical connection between TN-constrained kernel machines and GPs. In Chapter 5, we demonstrated that under appropriate priors, CPD and *Tensor Train* (TT)-constrained models converge to GPs as the ranks of the TNs approach infinity. This result not only provides a deeper understanding of the relationship between these models but also offers insights into their generalization behavior, depending on the training regime.

Finally, in Chapter 6, we characterized the MLSVD in terms of primal and dual optimization problems. This formulation opens new avenues for kernelizing TNs and paves the way for future applications in machine learning and data analysis.

In summary, this thesis has advanced the field by providing scalable and efficient methods for training kernel machines using TNs, uncovering theoretical connections to GPs, and offering new perspectives on optimization problems in tensor decompositions. These contributions not only enhance the practical utility of kernel machines but also deepen our theoretical understanding of their underlying structures.

G.1 FURTHER WORK

Several promising directions for future research emerge from this work, which can further deepen the understanding and expand the applicability of TN-constrained models.

One of the most compelling avenues for future research is the development of probabilistic TN-constrained kernel machines and TN-constrained GPs. While this thesis has established a theoretical connection between TN-constrained models and GPs in the infinite-rank limit, further research could focus on developing different kinds of TN-based probabilistic models e.g. by approximating the full or weight-space GP posterior using tools such as variational inference or *Monte Carlo Markov Chain* (MCMC) in conjunction with the priors developed in this thesis. Importantly, developing such probabilistic TN-constrained kernel machines could enable uncertainty quantification, which is critical for applications in decision-making and risk-sensitive domains. Future work could also explore the integration of hierarchical priors or non-Gaussian likelihoods to enhance the flexibility and expressiveness of these models.

The scalability and efficiency of TN-constrained models, in particular when quantized, make them particularly well-suited for applications where memory and computational resources are limited. An example of such application is the field of seizure detection, where models that can operate with reduced memory and computational requirements are crucial for real-time processing on portable, low-power devices. These devices, such as wearable EEG monitors, must efficiently process large amounts of data without draining the battery or requiring constant access to cloud-based resources. The reduced size and complexity of quantized TN-constrained models make them an ideal candidate for deployment in such resource-constrained environments, allowing for faster response times and improved user experience without compromising accuracy. The early research by De Rooij, Wesel, and Hunyadi [1] provides a strong foundation for further development and optimization of such models for deployment in resource-constrained environments. Future research should focus on identifying and de-

veloping new applications where such simple yet powerful models are necessary. For example, in edge computing, mobile devices, embedded systems, internet of things devices, or real-time systems, the ability to deploy compact models without sacrificing performance is crucial.

The primal-dual formulation of the MLSVD introduced in this thesis opens up exciting possibilities for its application. Future work can focus on developing optimization algorithms that leverage the primal-dual formulation of the MLSVD allowing to fully reap the computational and storage benefits of the primal formulation. Another logical line of research lies in embedding the primal formulation of the MLSVD into deep learning architectures to approximate latent tensorial structures within these models, similarly to what has been done with the SVD [2]. For instance, investigating how the MLSVD can be used to impose low-rank structures on the weights of deep neural networks could lead to more efficient training and inference, while exploring its use for interpretability and compression could reduce the memory and energy footprint of large-scale models.

Finally, further theoretical and algorithmic advancements are needed to fully realize the potential of TN-constrained models. Key areas of focus include automatic rank adaptation, where adaptive methods for selecting the ranks of TN-constrained models could balance expressiveness and efficiency, scalable optimization, where designing algorithms for training TN-constrained models on large-scale datasets could leverage distributed computing or stochastic optimization techniques, and generalization bounds, where deriving tighter bounds compared to the state-of-the-art ones by Khavari and Rabusseau [3] could lead to better understanding of the learning capabilities and limitations of these models. In conclusion, the research presented in this thesis has opened up numerous exciting directions for future work. By exploring these avenues, we can further advance the field of TNs and kernel machines, enabling the development of more efficient, interpretable, and powerful models for a wide range of applications.

REFERENCES

[1] S. J. De Rooij, F. Wesel, and B. Hunyadi. “Efficient Patient Fine-Tuned Seizure Detection with a Tensor Kernel Machine”. In: *2024 32nd European Signal Processing Conference (EUSIPCO)*. IEEE, 2024, pp. 1372–1376.

[2] Y. Chen, Q. Tao, F. Tonin, and J. Suykens. “Primal-Attention: Self-attention through Asymmetric Kernel SVD in Primal Representation”. In: *Advances in Neural Information Processing Systems* 36 (Dec. 2023), pp. 65088–65101.

[3] B. Khavari and G. Rabusseau. “Lower and Upper Bounds on the Pseudo-Dimension of Tensor Network Models”. In: *Advances in Neural Information Processing Systems*. May 2021.

CURRICULUM VITÆ

Frederiek WESEL

2 August 1994 Born in Bordighera, Italy.

EDUCATION

2020–2025 **PhD in Machine Learning**
Delft University of Technology, Delft, The Netherlands
Thesis: Tensor-Based Kernel Methods
Promotors: Dr. ir. K. Batselier
Dr. B. Hunyadi
Prof. dr. ir. J.W. van Wingerden

2016–2019 **Master of Science in Applied Mathematics**
Delft University of Technology, Delft, The Netherlands

2013–2016 **Bachelor of Science in Advanced Technology**
University of Twente, Enschede, The Netherlands

2008–2013 **Diploma di Superamento dell'Esame di Stato Conclusivo**
Liceo Scientifico A. Aprosio, Ventimiglia, Italy

EXPERIENCE

2025–Present **Senior Data Scientist at Equinor, Oslo, Norway**

2023–2023 **Visiting Researcher at Simula Research Laboratories, Oslo, Norway**

2018–2019 **Machine Learning Intern at ABN AMRO, Amsterdam, The Netherlands**

2017–2018 **Software Engineering Intern at DRG, Zoetermeer, The Netherlands**

List of Publications

In this thesis

1. F. Wesel and K. Batselier. "Large-Scale Learning with Fourier Features and Tensor Decompositions". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 17543–17554

2. F. Wesel and K. Batselier. "Tensor-Based Kernel Machines with Structured Inducing Points for Large and High-Dimensional Data". In: *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2023, pp. 8308–8320

3. F. Wesel and K. Batselier. "Quantized Fourier and Polynomial Features for More Expressive Tensor Network Models". In: *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2024, pp. 1261–1269

4. F. Wesel and K. Batselier. *A Kernelizable Primal-Dual Formulation of the Multilinear Singular Value Decomposition*. Oct. 2024. arXiv: 2410.10504

5. F. Wesel and K. Batselier. "Tensor Network-Constrained Kernel Machines as Gaussian Processes". In: *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2025, pp. 2161–2169

Other publications

1. E. Memmel, C. Menzen, J. Schuurmans, F. Wesel, and K. Batselier. "Position: Tensor Networks Are a Valuable Asset for Green AI". in: *Proceedings of the 41st International Conference on Machine Learning*. PMLR, July 2024, pp. 35340–35353

2. F. M. Viset, A. Kullberg, F. Wesel, and A. Solin. "Exploiting Hankel-Toeplitz Structures for Fast Computation of Kernel Precision Matrices". In: *Trans. Mach. Learn. Res.* (Jan. 2024)

3. S. J. De Rooij, F. Wesel, and B. Hunyadi. "Efficient Patient Fine-Tuned Seizure Detection with a Tensor Kernel Machine". In: *2024 32nd European Signal Processing Conference (EUSIPCO)*. IEEE, 2024, pp. 1372–1376