A weighted subspace exponential kernel for support tensor machines

Kirandeep Kour[1*†], Sergey Dolgov[2], Peter Benner[1,3], Martin Stoll[3] and Max Pfeffer[3†]

[1*]Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstraße -1, Magdeburg, 39106, Germany.
[2]Department of Mathematical Sciences, University of Bath, Bath, BA2 7AY, United Kingdom.
[3]Department of Mathematics, TU Chemnitz, Reichenhainer Str. 41, Chemnitz, 09126, Germany.

*Corresponding author(s). E-mail(s): kour@mpi-magdeburg.mpg.de;
Contributing authors: S.Dolgov@bath.ac.uk; benner@mpi-magdeburg.mpg.de; martin.stoll@mathematik.tu-chemnitz.de; max.pfeffer@mathematik.tu-chemnitz.de;
[†]These authors contributed equally to this work.

Abstract

High-dimensional data in the form of tensors are challenging for kernel classification methods. To both reduce the computational complexity and extract informative features, kernels based on low-rank tensor decompositions have been proposed. However, what decisive features of the tensors are exploited by these kernels is often unclear. In this paper we propose a novel kernel that is based on the Tucker decomposition. For this kernel the Tucker factors are computed based on re-weighting of the Tucker matrices with tuneable powers of singular values from the HOSVD decomposition. This provides a mechanism to balance the contribution of the Tucker core and factors of the data. We benchmark support tensor machines with this new kernel on several datasets. First we generate synthetic data where two classes differ in either Tucker factors or core, and compare our novel and previously existing kernels. We show robustness of the new kernel with respect to both classification scenarios. We further test the new method on real-world datasets. The proposed kernel has demonstrated a higher test accuracy than the state-of-the-art tensor train multi-way multi-level kernel, and a significantly lower computational time.

Keywords: Kernel methods, Low-rank Tensor Decompositions, Supervised Learning

1 Introduction

Support Vector Machines (SVM) (Vapnik, 1995, 1998), also known as Support Vector Network, maximum margin classifier, are popular machine learning methods, which allow for soft margins and high-dimensional feature embedding by using kernels. However, the standard SVM (Cortes & Vapnik, 1995) is based on vectors, and may struggle (in terms of both computational complexity and overfitting) for multi-dimensional (tensor) data. Many applications (e.g. in healthcare or signal processing) contain multidimensional data, hence studying kernel methods for tensorial data is an important topic that is addressed in this paper.

The approximation of tensors based on low-rank decompositions has received a lot of attention in scientific computing over recent years (Cichocki, 2011; Cichocki et al., 2016; Kolda & Bader, 2009; Liu, Guo, He, & Yang, 2015). The tensor-based SVM was introduced as Supervised Tensor Learning (STL) in Guo, Kotsia, and Patras (2012); Hao, He, Chen, and Yang (2013); Tao, Li, Hu, Maybank, and Wu (2007); Zhou, Li, and Zhu (2013). Using low-rank tensor approximations such as the Canonical Polyadic (CP) (Hitchcock, 1927; Nion & Lathauwer, 2008), Tucker (Lathauwer, Moor, & Vandewalle, 2000; Tucker, 1966), and Tensor Train formats (I. Oseledets & Tyrtyshnikov, 2010; I.V. Oseledets, 2011) within STL alleviates the *curse of dimensionality*, and allows one to reduce both computational complexity, by computing existing kernels faster, and overfitting, by designing new dedicated kernels using directly the components of the low-rank decomposition (Signoretto, Olivetti, Lathauwer, & Suykens, 2011, 2012; Zhao, Zhou, Adali, Zhang, & Cichocki, 2013a).

In the context of kernel methods, the *Dual Structure-preserving Kernel* (DuSK) for STL, which is particularly tailored to SVM and tensor data, was introduced in He et al. (2014). This kernel is defined using the CP format. Later, further kernelization in factors, specifically the *Kernelized-CP* (KCP) factorization, have been introduced (He, Lu, Ding, et al., 2017; He, Lu, Ma, et al., 2017), and the entire technique has been called the *Multi-way Multi-level Kernel* (MMK) method. Once an accurate CP approximation is available, DuSK and MMK typically deliver an accurate and efficient classification. However, the CP approximation of arbitrary data can be numerically unstable and difficult to compute (de Silva & Lim, 2008). In general, any optimization method (Newton, Steepest Descent or Alternating Least Squares) to obtain the CP decomposition might return only a locally optimal solution, and it is difficult to assess whether this is a local or global optimum.

In contrast, the Tucker approximation problem is well-posed, and a quasi-optimal Tucker approximation can be computed reliably by a sequence of singular value decompositions (SVD) (Lathauwer et al., 2000). Therefore, the Tucker format is also used often in data science. In Kotsia and Patras (2011) the authors have adopted the Tucker decomposition of the weight parameter to retain more structural information, and Zeng, Wang, Shen, and Shi (2017) extended this by using a Genetic Algorithm

(GA) prior to the Support Tucker Machine (STuM) for the contraction of the input feature tensor. In Wolf, Jhuang, and Hazan (2007), the authors proposed to minimize the rank of the weight parameter with the orthogonality constraints on the columns of the weight parameter instead of the classical maximum-margin criterion, and in Pirsiavash, Ramanan, and Fowlkes (2009) the orthogonality constraints are relaxed to further improve Wolf's method.

Further understanding of the KCP approach He, Lu, Ma, et al. (2017) is provided by a kernelized Tucker model, inspired by Signoretto, Tran Dinh, De Lathauwer, and Suykens (2014).

The Tensor Train (TT) decomposition offers a stable approximation similarly to the Tucker format, whereas scaling to higher dimensions like the CP format. A straightforward generalization of DuSK (MMK) to the TT format was proposed in Chen, Batselier, Ko, and Wong (2019).

However, why exactly the kernels based on low-rank decompositions are good for classification remains unclear. Moreover, since any tensor decomposition is a nonlinear parametrization of the tensor, its representation may be not unique. For example, Tucker and TT decompositions are invariant to rotation and scaling of the factors. These formats can also be converted from one to another, albeit with a change of ranks. Eliminating redundancy in rotation, scaling, and TT to CP conversion in the TT-MMK method has significantly improved the classification accuracy Kour, Dolgov, Stoll, and Benner (2023).

Further attempts to understand the key features of tensorial data and design the kernel accordingly include the Tucker *subspace kernel* Zhao, Zhou, Adali, Zhang, and Cichocki (2013b). Here, the kernel compares projectors onto the subspaces spanned by Tucker factors. The latter are known to effectively capture the multilinear structure of the data. For example, Taguchi and Turki (2021) used HOSVD for unsupervised feature extraction. Multiway analysis enables one to effectively capture the multilinear structure of the data, which is usually available as a priori information about the data. In Yan et al. (2007) a subspace learning technique for Face Recognition was introduced. The factor match score, a consistent way of comparing the feature vectors of tensor decompositions, has been introduced in Acar, Kolda, and Dunlavy (2011).

However, the data may contain decisive features not only in the Tucker subspaces, but also in the Tucker core. How to capture both in a computationally efficient way remained largely an open problem. This paper aims to fill this gap by introducing a novel kernel that show high robustness with respect to where the classification information are contained within the tensor.

Novel contributions

The main aim of this paper is to introduce a novel kernel that shows high robustness with respect to where the classification information is contained within the tensor. The main contributions with this novel kernel are summarized as follows:

- We propose a new form of writing the Tucker (HOSVD) decomposition with weighted factors.

- Based on this form, we propose a new kernel for support tensor machines, which admits fast computation, whereas the weighting takes into account both Tucker factors and core in building the nonlinear decision boundary.
- Using synthetic data with class assignment based on either Tucker factors or core, we confirm that the new kernel provides an accurate classification in all cases, in contrast to existing Tucker-based kernels.
- Finally, we test that the new kernel provides higher classification accuracy than state-of-the-art methods also on real datasets.

2 Notation and background

This section sets up and extends notations for tensors and the binary classification problem to multi-dimensional data.

2.1 Tensor Algebra

In context of numerical multi-linear algebra, a multidimensional array is called a tensor. Tensors are a generalization of matrices (2-modes; rows and columns) with a higher number of *dimensions / modes*. We denote all tensors by a calligraphic letter $\mathcal{X}$. We assume that all tensors are real-valued. For a general introduction to tensors and their properties we refer to Kolda and Bader (2009) and the references mentioned therein. We summarize the common notations encountered in this paper in Table 1.

Symbol	Description	Definition
$\overline{i_1,\ldots,i_M}$	multi-index	$\overline{i_1,\ldots,i_M} = 1+\sum_{k=1}^{M}(i_k-1)\prod_{m=1}^{k-1} I_m$
x	scalar value	$x \in \mathbb{R}$
$\mathbf{x}$	vector	$\mathbf{x} \in \mathbb{R}^I$
$\mathbf{X}$	matrix	$\mathbf{X} \in \mathbb{R}^{I\times J}$
$\mathcal{X}$	tensor	$\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_M}$
$x_{i_1 i_2 \ldots i_M}$	$(i_1,i_2,\ldots,i_M)$-entry of a tensor	
$\mathcal{X}_{(m)}$	m-mode matricization	$(x_{(m)})_{i_m,\overline{i_1,\ldots,i_{m-1},i_{m+1},\ldots,i_M}} = x_{i_1,\ldots,i_M}$
$\mathcal{X}\times_m \mathbf{A}$	m-mode product	$(\mathcal{X}\times_m \mathbf{A})_{(m)} = \mathbf{A}\mathcal{X}_{(m)},\ \mathbf{A}\in\mathbb{R}^{J\times I_m}$
$\mathcal{Z}=\mathcal{X}\circ\mathcal{Y}$	outer product	$z_{i_1,\ldots,i_M,j_1,\ldots,j_N} = x_{i_1,\ldots,i_M} y_{j_1,\ldots,j_N}$.
$\mathbf{A}\otimes\mathbf{B}$	Kronecker product	$\begin{bmatrix} a_{i,1}\mathbf{B} & \cdots & a_{i,J}\mathbf{B}\end{bmatrix} \in \mathbb{R}^{IK\times JL}$ $\mathbf{A}\in\mathbb{R}^{I\times J}, \mathbf{B}\in\mathbb{R}^{K\times L}$
$\langle M\rangle$	integer values from 1 to M	$\{1,2,\cdots,M\}$
$\langle\mathcal{X},\mathcal{Y}\rangle$	inner product of tensors $\mathcal{X}$ and $\mathcal{Y}$	$\sum_{i_1}^{I_1}\sum_{i_2}^{I_2}\ldots\sum_{i_m}^{I_M} x_{i_1 i_2\ldots i_m} y_{i_1 i_2 \ldots i_m}$.
$\|\mathcal{X}\|$	Frobenius norm of the tensor $\mathcal{X}$	$\sqrt{\langle\mathcal{X},\mathcal{X}\rangle}$

Table 1 Tensor notation used in this paper.

2.2 Support Tensor Machines for supervised learning

Although tensor objects can be reshaped into vectors, the structural information encoded in the tensorial data are lost. For example, in an fMRI image, the values of adjacent voxels are typically close to each other (He et al., 2014). It is natural to replace the vector-valued SVM by a tensor-valued SVM (cf. Supervised Tensor Learning (STL) introduced in Guo et al. (2012); Tao et al. (2007); Zhou et al. (2013). An extension of the STL using kernelized tensor factorization with maximum-margin criterion (SVM) is given in He, Lu, Ma, et al. (2017). This preserves the nonlinear structure and enhances the overall performance of the STL model, called Kernelized Support Tensor Machines (KSTM).

2.2.1 Kernelized Support Tensor Machine

The KSTM is a binary classification model for N tensor input data points $\{(\mathcal{X}_i, y_i)\}_{i=1}^N$ where each tensor is of the form $\mathcal{X}_i \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ with labels $y_i \in \{0,1\}$ leading to a nonlinear decision boundary. The method follows a maximum margin approach to get the separation hyperplane. Hence, the objective function for a nonlinear boundary can be written as follows (Cai, He, Wen, Han, & Ma, 2006):

$$\begin{aligned} &\min_{w,b} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^N \xi_i \\ &\text{subject to} \quad y_i(\langle \Psi(\mathcal{X}_i), w\rangle + b) \geq 1 - \xi_i \qquad \xi_i \geq 0 \quad \forall i. \end{aligned} \tag{1}$$

The classification setup given in (1) is known as Support Tensor Machine (STM) (Tao, Li, Hu, Maybank, & Wu, 2005). The dual formulation of the corresponding primal problem is given as follows:

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

The nonlinear feature embedding for tensor inputs in a tensor space to a feature space is analogous to working with vector inputs in a vector space. We can define an embedding from low-dimensional tensor-product space to the tensor-product Reproducing Kernel Hilbert Space (He, Lu, Ma, et al., 2017). And by using Mercer's Theorem, i.e. having a kernel that is positive semidefinite, we can construct a feature embedding Ψ such that,

$$\begin{aligned} &\Psi : \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M} \text{ (input space)} \rightarrow \mathbb{F} \text{ (feature space)} \\ &\exists\, K \colon \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M} \times \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M} \mapsto \mathbb{R} \quad \text{s.t.} \quad K\left(\mathcal{X}, \mathcal{X}'\right) = \langle \Psi(\mathcal{X}), \Psi(\mathcal{X}')\rangle_{\mathbb{F}}. \end{aligned}$$

This is not only computationally tractable but also avoids the explicit computation of the function Ψ. The kernel matrix that results from the continued evaluation of the kernel function on the data points is then positive semidefinite. With the help of the kernel, a linear learning algorithm can learn a *nonlinear boundary*, without explicitly knowing the nonlinear function Ψ.

Therefore, by using the kernel trick, KSTM is defined as follows:

$$\begin{aligned} \max_{\alpha_1,\ldots,\alpha_N} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathcal{X}_i, \mathcal{X}_j) \\ \text{subject to} \quad & 0 \le \alpha_i \le C, \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \ \ \forall i. \end{aligned} \tag{3}$$

Once we have the real-valued function (kernel) value for each pair of tensors, we can use the state-of-the-art LIBSVM implementation (Chang & Lin, 2011) , which relies on the *Sequential Minimal Optimization* algorithm to optimize the weights α_i. Hence, the preeminent part is the kernel function $K(\mathcal{X}_i, \mathcal{X}_j)$.

The STM classifier for predicting labels for unseen test data in tensor form is then given by

$$G(\mathcal{X}) = \operatorname{sign}\left(\sum_{i=1}^{N} \alpha_i y_i K(\mathcal{X}_i, \mathcal{X}) + b_0 \right), \tag{4}$$

where

$$b_0 = \frac{1}{N_0} \sum_{i:\alpha_i \in (0,C)} \left(y_i - \sum_{j=1}^{N} \alpha_j K(\mathcal{X}_j, \mathcal{X}_i) \right), \quad \text{with} \quad N_0 = \sum_{i:\alpha_i \in (0,C)} 1. \tag{5}$$

The only task needed for the KSTM is thus to choose a well-suited kernel function. This way, we can work with the input data in a high-dimensional space, while all computations are performed in the original low-dimensional space. We will discuss possible choices for tensor kernels in Sec. 3, where we will also introduce a novel tensor kernel. These kernels are based on low-rank formats for tensors, which we introduce now.

2.3 Low-rank Tensor Decompositions

Given the complexity of storing the full tensor $\mathcal{X}$, it is often desirable to have a different potentially more economic representation. As such, tensor decomposition methods have seen much progress over the last two decades, and they are applied to solve problems of varying computational complexity. The main goal is the linear (or at most polynomial) scaling of the computational complexity in the dimension (order) of a tensor. The key ingredient is the separation of variables via approximate low-rank factorizations.

Canonical Polyadic decomposition

The Canonical Polyadic (CP) decomposition of an $M^{th}-$order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ is a factorization into a sum of rank-one components (Hitchcock, 1927), which is given element-wise as

$$x_{i_1 i_2 \dots i_M} \cong \sum_{r=1}^{R} a_{i_1,r}^{(1)} a_{i_2,r}^{(2)} \cdots a_{i_M,r}^{(M)},$$

or shortly,
$$\mathcal{X} \cong [\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(M)}]\!], \tag{6}$$

where $\mathbf{A}^{(m)} = \left[a_{i_m,r}^{(m)}\right] \in \mathbb{R}^{I_m \times R}$, $m = 1, \dots, M$, are called *factor matrices* of the CP decomposition, see Fig. 1, and R is called the CP-rank. The notation $[\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(M)}]\!]$ is also called the Kruskal representation of the tensor. Despite the simplicity of the CP format, the problem of the best CP approximation is often ill-posed (de Silva & Lim, 2008). A practical CP approximation can be computed via the Alternating Least Squares (ALS) method (Nion & Lathauwer, 2008), but convergence may be slow. It may also be difficult to choose the rank R.

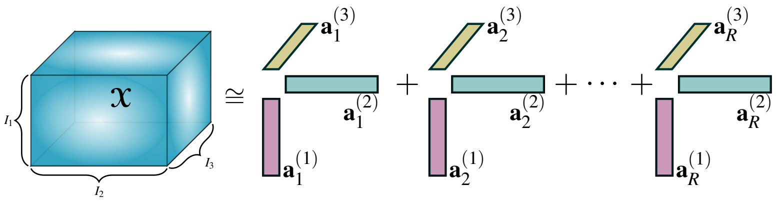

Figure 1 CP decomposition of a 3-way tensor of rank R.

Tucker Decomposition

The Tucker decomposition consists of a decomposition of the tensor into matrices and a core tensor, where the core tensor has smaller dimension compared to the original tensor. For a given tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_M}$ the Tucker decomposition is as then given by,

$$\begin{aligned} \mathcal{X} &\cong \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_M=1}^{R_M} g_{r_1 r_2 \dots r_M} \left(\mathbf{u}_{r_1}^{(1)} \circ \mathbf{u}_{r_2}^{(2)} \circ \dots \circ \mathbf{u}_{r_M}^{(M)}\right) \\ &= [\![\mathcal{G}; \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \dots, \mathbf{U}^{(M)}]\!]. \end{aligned} \tag{7}$$

Here, $\mathcal{G}$ is a tensor of size $\mathbb{R}^{R_1 \times R_2 \times \dots \times R_M}$ and R_m is the Tucker rank in each mode matricization of the tensor $\mathcal{X}$. A crucial advantage of the Tucker format (and all tree tensor networks) is the ability to perform algebraic operations directly on the component tensors, avoiding full tensors. Moreover, we can compute a quasi-optimal Tucker approximation of any given tensor using the SVD. This builds on the fact that the Tucker decomposition constitutes a successive matrix factorization, where each Tucker rank is the matrix rank of the appropriate unfolding of the tensor, and hence the Tucker approximation problem is well-posed (Lathauwer et al., 2000). One unique advantage of the Tucker format is the interpretability of the leaf-components

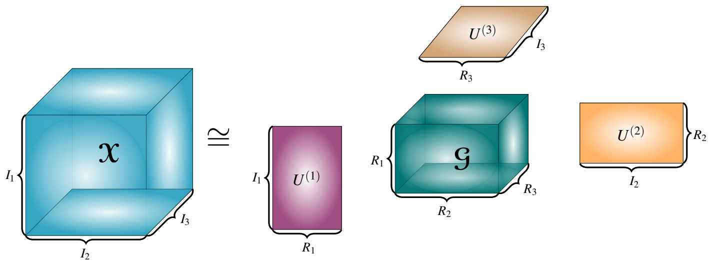

Figure 2 Tucker decomposition of a 3-way tensor.

$\mathbf{U}^{(m)}$: Since they result directly from an SVD of the m-th matricization, their columns constitute an orthonormal basis of the subspace of $\mathbb{R}^{I_m}$ that the data lies in.

Tensor Train decomposition

The Tensor Train (TT) (I.V. Oseledets, 2011) decomposition of an $M^{th}-$order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ is defined element-wise as

$$x_{i_1 i_2 \ldots i_M} \cong \sum_{r_0,\ldots,r_M} c^{(1)}_{r_0,i_1,r_1} c^{(2)}_{r_1,i_2,r_2} \cdots c^{(M)}_{r_{M-1},i_M,r_M},$$
$$\mathcal{X} \cong \langle\!\langle \mathcal{C}^{(1)}, \mathcal{C}^{(2)}, \ldots, \mathcal{C}^{(M)} \rangle\!\rangle, \tag{8}$$

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

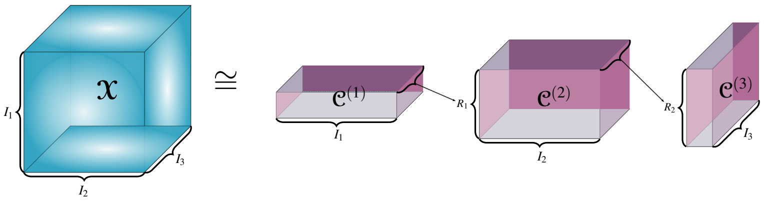

Figure 3 TT decomposition of a 3-way tensor.

composition is also a tree-based tensor format, all above-mentioned advantages of the Tucker format also translate to the TT format. Furthermore, complexity of the TT format is quadratic in the ranks, whereas the Tucker format scales exponentially with the Tucker ranks. This reduction of complexity however comes at the cost of interpretability: There is no straightforward way to interpret the meaning of the TT components.

2.3.1 Converting Tucker and TT into CP

It is easy to see from (6), (7), and (8) that one can convert a tensor in Tucker or TT format into the CP format without too much effort: Summing over all Tucker ranks $R_1, \ldots, R_M$ or all TT-ranks $r_0, \ldots, r_M$ will yield a sum of rank-one tensors as required in the CP format. We emphasize here that this does not result in a *minimal*

CP decomposition but only in a CP *representation* of the tensor. This can however still be useful, as long as the obtained CP representation retains the interpretable qualities of the Tucker or TT decomposition.

The main difficulty with obtaining a meaningful conversion is that none of the decomposition formats is really unique: CP allows for a rescaling of columns of the factor matrices, and for Tucker and TT, one can insert identity matrices $I = QQ^{-1}$ between the modes without changing the tensor.

In a previous work (Kour et al., 2023), the problem of meaningfully converting TT into CP was overcome by enforcing uniqueness in the TT-SVD, then converting into CP, and then equilibrating the column norms of the factor matrices in order to avoid ambiguity in the CP representation.

This technique can be used analogously for the conversion of Tucker into CP: In the Higher Order SVD (HOSVD), we enforce uniqueness by fixing the sign of each singular vector. This is done by finding the element of maximum absolute value and making it positive. The Tucker tensor is then converted to CP by simply summing over all ranks $R_1, \dots, R_M$ and a norm equilibration is performed in order to distribute the scalar $g_{r_1 r_2 \dots r_M}$ in (7) across all factors.

3 Kernels for tensor data in SVM

In this section, we discuss possible choices for tensor kernels. First, we briefly recapitulate existing tensor kernels before we introduce our new kernel. This, together with the numerical study of tensor kernel in Sec. 4, is the main result of this article. At the end of this section, we compare the complexity of computing the different kernels.

3.1 Existing kernels

The Gaussian kernel

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

$$K(\mathcal{X}, \mathcal{Y}) = \exp\left(\frac{-\|\mathcal{X} - \mathcal{Y}\|_F^2}{2g^2}\right), \tag{9}$$

with g being the length scale of the kernel. We compute the distance between the two input tensors using the Frobenius norm. This norm can be computed efficiently in each of the tensor formats introduced above (Table 2 shows the leading terms of the complexity estimates). However, by treating the tensor as a simple vector, we lose valuable information about the different tensor modes. It has been observed (e.g. in He et al. (2014); Kour et al. (2023)) that this straightforward idea yields suboptimal classification results and that it can be improved by introducing more sophisticated tensor kernels.

Dual Structure-preserving Kernel

The Dual Structure-preserving Kernel (**DuSK**) was introduced first in He et al. (2014) for a rank-one tensor factorization and was later extended for the Kernelized CP decomposition in He, Lu, Ding, et al. (2017). For given tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ and $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ and their corresponding CP decomposition given by $[\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(M)}]\!]$ and $[\![\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(M)}]\!]$, the formulation of the kernel approximation by DuSK is given as follows:

$$\begin{aligned}\langle \Psi(\mathcal{X}), \Psi(\mathcal{Y}) \rangle &= K(\mathcal{X}, \mathcal{Y}) \\ &= K\left(\sum_{i=1}^{R} \mathbf{a}_i^{(1)} \otimes \mathbf{a}_i^{(2)} \otimes \cdots \otimes \mathbf{a}_i^{(M)}, \sum_{j=1}^{R} \mathbf{b}_j^{(1)} \otimes \mathbf{b}_j^{(2)} \otimes \cdots \otimes \mathbf{b}_j^{(M)} \right) \\ &= \sum_{i,j=1}^{R} k(\mathbf{a}_i^{(1)}, \mathbf{b}_j^{(1)}) k(\mathbf{a}_i^{(2)}, \mathbf{b}_j^{(2)}) \cdots k(\mathbf{a}_i^{(M)}, \mathbf{b}_j^{(M)}),\end{aligned} \tag{10}$$

where,

$$k(\mathbf{a}, \mathbf{b}) = \exp\left(\frac{-\|\mathbf{a} - \mathbf{b}\|^2}{2g^2} \right). \tag{11}$$

In short, we evaluate the kernel function $k(\cdot,\cdot)$ on the individual factors of the CP decomposition. The motivation of DuSK is simple: Since the CP decomposition is often unique (up to norm equilibration), comparing the feature vectors in each mode directly will most likely improve classification. However, this kernel is inherently designed for CP tensors, which is why we have to convert other tensor formats into the CP format first (see Sec. 2.3.1 and Kour et al. (2023)).

The subspace kernel

Instead of comparing the feature vectors in the CP decomposition, one can use a similar approach for the Tucker format. Here, the feature vectors are stored in the leaf-components $\mathbf{U}^{(m)}$ and they span the column spaces of the m-th matricizations. Thus, it makes sense to compare the *projections* onto these subspaces. This kernel was introduced in Zhao et al. (2013b).

Let $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ denote an M^{th}-order tensor, when the SVD is applied on the mode-m unfolding as $\mathcal{X}_{(m)} = \mathbf{U}_{\mathcal{X}}^{(m)} \Sigma_{\mathcal{X}}^{(m)} \mathbf{V}_{\mathcal{X}}^{(m)^\top}$ and similarly for $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$, $\mathcal{Y}_{(m)} = \mathbf{U}_{\mathcal{Y}}^{(m)} \Sigma_{\mathcal{Y}}^{(m)} \mathbf{V}_{\mathcal{Y}}^{(m)^\top}$, then the Chordal distance-based kernel is defined as,

$$K(\mathcal{X}, \mathcal{Y}) = \prod_{m=1}^{M} \exp\left(-\frac{1}{2g^2} \left\| \mathbf{U}_{\mathcal{X}}^{(m)} \mathbf{U}_{\mathcal{X}}^{(m)^\top} - \mathbf{U}_{\mathcal{Y}}^{(m)} \mathbf{U}_{\mathcal{Y}}^{(m)^\top} \right\|_F^2 \right) \tag{12}$$

This kernel provides us with rotation and reflection invariance for elements on the Grassmann manifold. Furthermore, the kernel does not see the core tensor $\mathcal{G}$ and is invariant under rescaling of the feature vectors (i.e., the columns of the leaf matrices).

3.2 The weighted subspace exponential kernel

The subspace kernel performs well when the information about the classification is stored in the subspaces $\mathbf{U}_{\mathcal{X}}^{(m)}$ and it performs poorly when the information is mostly contained in the core tensor $\mathcal{G}$. This is confirmed in our synthetic numerical experiments below. The DuSK uses a similar strategy as the subspace kernel: Here, we compare all the feature vectors in the CP decomposition. DuSK therefore also performs better if most of the information is in the subspaces, i.e., in the feature vectors.

The main contribution of this article is an improved tensor kernel for Tucker tensors that also includes information of the core tensor $\mathcal{G}$, combining the strengths of the subspace kernel and DuSK, and that outperforms both of them in common scenarios, and can be computed more efficiently than DuSK.

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

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

for any $p \in \mathbb{R}$ (assuming no zero singular values, or defining $0^0 = 1$). Using this, we can distribute singular values over the Tucker factors in the HOSVD (see Algorithm 1) and we use the resulting *rescaled* features $\bar{\mathbf{U}}^{(m)} = \mathbf{U}^{(m)}\left(\Sigma^{(m)}\right)^p$ for the computation of the kernel. We note that by default, we choose $p = \frac{1}{M}$ as this corresponds to distributing the norm $\|\mathcal{X}\| = \|\Sigma^{(m)}\|_F$ equally over the M feature matrices, which provides accurate classification in practice.

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

$$K(\mathcal{X},\mathcal{Y}) = \prod_{m=1}^{M}\sum_{i,j=1}^{R_m} k((\mathbf{u}_{\mathcal{X}}^{(m)})_i,(\mathbf{u}_{\mathcal{Y}}^{(m)})_j) = \prod_{m=1}^{M}\sum_{i,j=1}^{R_m}\exp\left(-\frac{1}{2g^2}\left\|\bar{\mathbf{u}}_{\mathcal{X},i}^{(m)} - \bar{\mathbf{u}}_{\mathcal{Y},j}^{(m)}\right\|_F^2\right), \tag{13}$$

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

Algorithm 1 Weighted HOSVD

```
Require: Given tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$, Tucker ranks $R_1, \ldots, R_M$, weighting power
    $p$ (default $p = 1/M$).
Ensure: Tucker factors $\bar{\mathbf{U}}^{(1)}, \ldots, \bar{\mathbf{U}}^{(M)}$ and core $\mathcal{G}$.
  for $m = 1$ to $M$ do
      Step 1: Computing uniqueness-enforced HOSVD
      Compute SVD $\left[\mathbf{U}^{(m)}, \Sigma^{(m)}, (\mathbf{V}^{(m)})^\top\right] = \mathrm{svd}(\mathcal{X}_{(m)})$,
      where $\Sigma^{(m)} = \mathrm{diag}(\sigma_1^{(m)}, \sigma_2^{(m)}, \ldots, \sigma_{I_m}^{(m)})$
      for $r_m = 1$ to $R_m$ do
          $i^*_{r_m} = \arg\max_{i=1,\ldots,I_m} |u_{i,r_m}^{(m)}|$
          $\hat{\mathbf{u}}_{r_m}^{(m)} := \mathbf{u}_{r_m}^{(m)} / \mathrm{sign}(u_{i^*_{r_m}, r_m}^{(m)})$
      end for
      $\hat{\mathbf{U}}^{(m)} = [\hat{\mathbf{u}}_1^{(m)}, \hat{\mathbf{u}}_2^{(m)}, \ldots, \hat{\mathbf{u}}_{R_m}^{(m)}]$
      Step 2: Computing norm weighted factors
      $\bar{\mathbf{U}}^{(m)} = \hat{\mathbf{U}}^{(m)} \Sigma_p^{(m)}$
      where $\Sigma_p^{(m)} = \mathrm{diag}((\sigma_1^{(m)})^p, (\sigma_2^{(m)})^p, \ldots, (\sigma_{R_m}^{(m)})^p)$
      $\mathcal{G} \leftarrow \mathtt{ttm}(\mathcal{X}, \hat{\mathbf{U}}^{(m)} (\Sigma_p^{(m)})^{-1}, m)$      (Vannieuwenhoven, Vandebril, & Meerbergen, 2012)
  end for
```

3.3 Computational complexity of the different kernels

For a large number N of data points, for large tensor order M or large mode dimensions I_m, computing the different tensor kernels can be very time consuming. If the data input is already given in CP format, computing the DuSK is not too expensive. But most often, the data is given as a full tensor. In these cases, we prefer to compute first a TT or Tucker decomposition of the tensor and then convert it into CP, in order to circumvent the aforementioned numerical issues with the computation of the CP decomposition. Here, we will however only note the complexity of the kernel computation with respect to the given ranks. We denote the maximal dimensions or ranks by $I = \max_{m=1,\ldots,M} I_m$, $R_{Tucker} = \max_{m=1,\ldots,M} R_m$, and $R_{TT} = \max_{m=1,\ldots,M-1} r_m$. The ranks are also to be understood as the maximal respective rank of all data inputs.

Table 2 summarizes the computational complexity for a single entry of the kernel matrix. We note that all kernels except the Gaussian kernel can only be computed if

Table 2 Theoretical complexity of computing a single entry of the different kernel matrices from data given in different formats. Note that some kernel-format combinations are not defined.

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

the tensor is in a low-rank format. Also, we interpret a tensor in CP format to be a Tucker tensor with diagonal core tensor $\mathcal{G}$ and thus the subspace kernel and WSEK are computed using the factor matrices. Furthermore, in the cases of Tucker and TT tensors, we report on the complexity using naive matrix multiplication ($\mathcal{O}(n^3)$).

We observe that for large tensor order M, computation of the Gauss kernel is prohibitive if it is not done in a low-rank format. If the CP rank R is small, all kernels can be computed efficiently. However, if the CP decomposition has to be obtained by conversion from Tucker or TT, these ranks can be large and DuSK suffers from the curse of dimensionality. WSEK is even more efficient than the subspace kernel. We will report on the CPU times for our numerical experiments in Sec. 5.

4 A numerical study on synthetic data

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

4.1 Interpreting CP and Tucker

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

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

This observation can be exploited when designing a tensor kernel for KSTM. The subspace kernel introduced in Section 3 can only see the subspaces, i.e. the leafs of the Tucker tensor. DuSK includes the core data via the norm equilibration. The Gaussian kernel uses the whole tensor but it is less sensible in spotting the subspaces, because the tensor is simply vectorized and this information is hidden. Our proposed WSEK includes information of the core tensor because the feature vectors are weighted with the p-th power of the corresponding singular values.

4.2 A Synthetic Experiment

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

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

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

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

4.3 Results and Interpretation

In Fig. 4 and Fig. 5, we can see the results for the two experimental scenarios. In both cases, we plotted the test error (i.e. the classification accuracy on the test set) for the ranks $r_{approx} = 1,3,5,10$. For each rank, we plotted the accuracies of the different kernels for all noise levels in one picture. The computation of DuSK was too expensive in the case $r_{approx} = 10$ as this requires the computation of norms of tensors with CP rank 1000 many times.

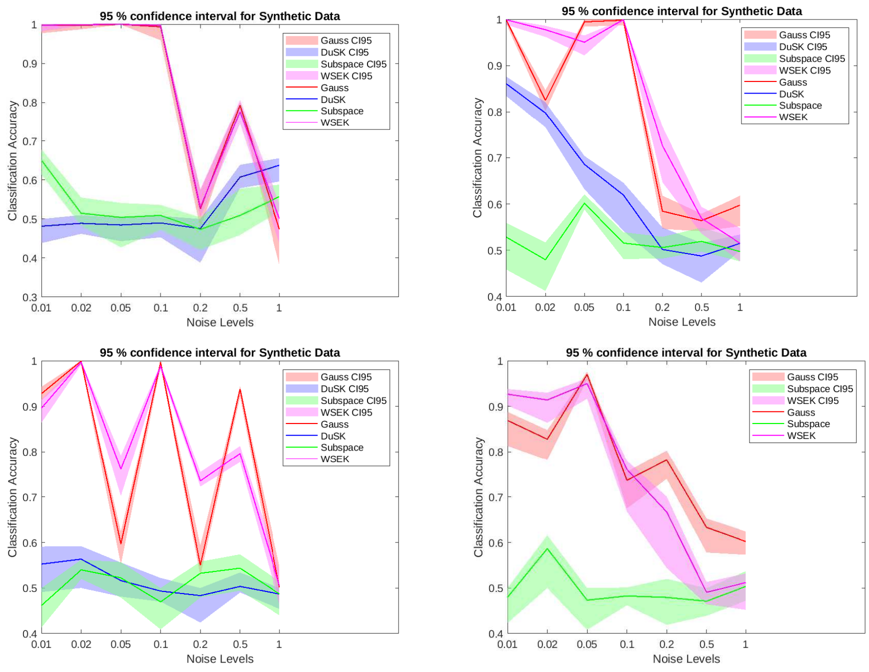

Figure 4 Classification Accuracy with 95% confidence interval for different noise levels in the *core*-scenario of the generated synthetic data with different rank trucation $r_{approx} = 1, 3, 5, 10$ (DuSK not included for $r_{approx} = 10$)

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

These results are not surprising following the considerations in Sec. 4.1: The Subspace kernel sees only the information in the leafs and it can therefore not perform well in the *core*-scenario. The DuSK kernel includes extra information so that it performs better in the *core*-scenario (especially when we guessed the rank correctly, $r_{approx} = r_{exact} = 3$) but still not as good as the Gaussian kernel. In the *leaf*-scenario, all the information is in the subspaces and therefore both DuSK and the Subspace kernel perform very well. Here, the Gaussian kernel performed much worse than all other kernels. The WSEK was designed to do well in both scenarios and this is shown also in these experiments.

We will see in the next chapter that especially the *leaf*-scenario is realistic: In the ADNI dataset explored below, the subspace kernel performs better than the Gaussian kernel and it even outperforms DuSK. We conclude that including information on the m-mode subspaces is crucial for the design of a tensor kernel. This is also a possible explanation why DuSK has performed well in many settings. Our new kernel however outperforms DuSK in all our experiments.

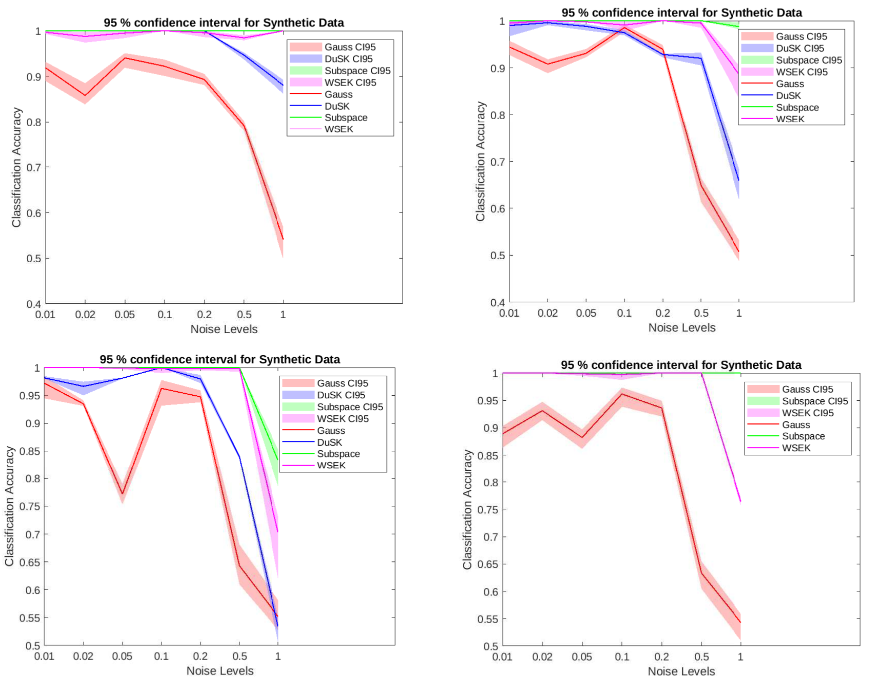

Figure 5 Classification Accuracy with 95% confidence interval for different noise levels in the *leaf*-scenario of the generated synthetic data with different rank trucation $r_{approx} = 1,3,5,10$ (DuSK not included for $r_{approx} = 10$)

5 Classification of real datasets

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

All numerical experiments have been done in `MATLAB 2019b`. Low-rank tensor approximations are computed using `TT-Toolbox` (TT-Toolbox, 2023) and `tensor toolbox` (tensortoolbox Version 3.4, 2022). We have run all experiments on a compute cluster which is equipped with `2 TB NVMe SSD` Harddisk, 2×`Intel Xeon Skylake Silver 4210R` CPUs with 10 cores per CPU, and `768 GB DDR4 ECC` of RAM.The hyperparameters in KSTM (3) are tuned similarly to Synthetic experiments (see Section 4.2), the only difference is that we report results for real data experiments for each of the rank $R \in \{1,2,\cdots,10\}$, where $R_1 = R_2 = R_3 = R$. The SVM problem (3) is solved using the `svmtrain` function from LIBSVM (Chang & Lin, 2011) library, and `svmpredict` computes the classification accuracy using (4) and (5).

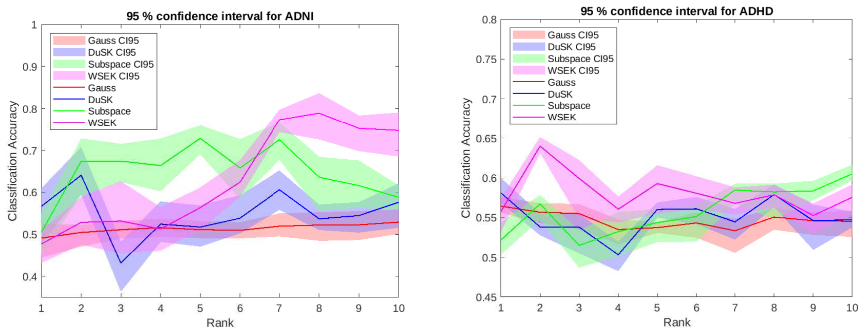

Figure 6 (left) Comparison of mean classification accuracy with variance for the different kernels with different rank truncation for ADNI dataset, (right) Comparison of mean classification accuracy with variance for the different kernels with different rank truncation for ADHD dataset.

5.1 Resting-state fMRI data collection

- **Alzheimer Disease (ADNI)**: The ADNI[1] stands for Alzheimer Disease Neuroimaging Initiative. It contains the resting state fMRI images of 33 subjects. The dataset was collected from the authors of He, Lu, Ding, et al. (2017). The images belong to either Mild Cognitive Impairment (MCI) with Alzheimer Disease (AD), or normal controls. Each image is a tensor of size $61 \times 73 \times 61$, containing 271633 elements in total. The AD+MCI images are labeled with -1, and the normal control images are labeled with 1. Preprocessing of the data sets is explained in He et al. (2014).
- **Attention Deficit Hyperactivity Disorder (ADHD):** The ADHD dataset is collected from the ADHD-200 global competition dataset[2]. It is a publicly available preprocessed fMRI dataset from eight different institutes. The original dataset is unbalanced, so we have chosen 200 subjects randomly, ensuring that 100 of them are ADHD patients (assigned the classification label -1) and the 100 other subjects are healthy (denoted with label 1). Each of the 200 resting state fMRI samples contains $49 \times 58 \times 47 = 133574$ voxels.

Remark 1 The dataset taken here is exactly the same as that used in Kour et al. (2023), so the TT-MMK results can be compared one to one. However, the particular indices of the collected data are not similar to those selected in He et al. (2014), so the accuracy of DuSK reported below is not directly comparable to that in He et al. (2014).

5.2 Numerical results

In this section, we summarize the results for the two fMRI datasets:

- **Classification accuracy:** In Fig. 6, we show the average classification accuracy resulting from the cross validation. In Table 3, we show the best classification accuracy between rank $[1, 10]$. On both datasets the proposed WSEK gives the

[1] http://adni.loni.usc.edu/
[2] http://neurobureau.projects.nitrc.org/ADHD200/Data.html

best average classification accuracy (79% for ADNI and 64% for ADHD) compared to other state of the art tensor kernels. We note that the accuracy of the subspace kernel improves for higher ranks in the ADHD dataset and is then similar to that of WSEK but this choice of rank is very high for a low-rank truncation method. On the other hand, our proposed kernel gives good classification accuracy already at rank 2. The Gaussian kernel was computed for different Tucker approximations of the full tensor. Using the full tensor in the computation of the kernel did not improve the accuracy. As in Kour et al. (2023), the DuSK kernel was computed using a CP approximation of the full tensor (CP-DuSK), because computing the Tucker decomposition and then converting to CP yielded high CP ranks and DuSK was too slow.

Table 3 Maximum average classification accuracy in percentage ± standard deviation for different methods, data sets, and rank $R \in [1, 10]$. The values for TTCP-DuSK are taken from Kour et al. (2023) for comparison.

Methods	ADNI	ADHD
Gauss	53	50
CP-DuSK	64 ± 0.05 (R = 5)	58 ± 0.02 (R = 6)
TTCP-DuSK	73 ± 0.03 **(R = 4)**	63 ± 0.01 (R = 5)
Tucker Subspace	73 ±0.04 (R = 7)	61 ±0.02 (R = 10),
WSEK	**79** ±0.03 (R = 8)	**64** ±**0.01 (R = 2)**

- **Running Time:** Table 4 shows the running times for the computation of the different kernels on the ADNI dataset. For small tensors, computing the Gauss kernel is fast. Both the subspace kernel and WSEK can be computed efficiently. Computation of DuSK however quickly becomes prohibitive and takes a long time in these experiments.

Table 4 Comparison of Kernel computation time for $R \in [1, 10]$. The values for TTCP-DuSK are taken from Kour et al. (2023) for comparison.

Kernel	Format	Parameters	CPUtime for ADNI (# run = 1)
Gaussian	Ktensor	C, g	20 seconds
DuSK	CP (2.3)	$C, g,\ R$	17 minutes
DuSK	TTCP (2.3.1)	$C, g,\ R_{TT}$	3.5 hours
Subspace	Tucker (2.3)	$C, g,\ R_{Tucker}$	25 seconds
WSEK	SqrtmHOSVD (1)	$C, g,\ R_{Tucker}$	30 seconds

- **Statistic comparison:** In Tabel 3 and Figure 6, the variance for corresponding mean accuracy is shown. The WSEK STM shows a good trade-off between classification accuracy and variance. For the ADHD dataset, it even gives the best classfication accuracy with the lowest variance value for small ranks.

5.3 Conclusion

Our real world experiments show superior performance of the WSEK both in terms of classification accuracy and running time. We conclude that the classification information of the datasets is hidden mostly in the subspaces of the Tucker decomposition (hence the previously observed good performance of DuSK) but that classification can be improved by taking the singular values into account (as done in WSEK). Furthermore, computing the Tucker decomposition of tensor inputs is straightforward and efficient, resulting in an all-around very robust tensor kernel.

Author contributions

All authors contributed to the conception and design of the tensor kernel method. Implementation and analysis of the new kernel were performed by Kirandeep Kour and Max Pfeffer. The first draft of the manuscript was written by Max Pfeffer and Kirandeep Kour. All authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Acknowledgments

K.K. is supported by the International Max Planck Research School for Advanced Methods in Process and System Engineering-**IMPRS ProEng**, Magdeburg. M.P. is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Projektnummer 448293816.

Statements and Declaration

Conflict of interest The authors declare that they have no conflict of interests. Also, there are no financial and non-financial interest to disclose.

References

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

Bader, B.W., Kolda, T.G., et al. (2022). *Tensor toolbox for matlab version 3.4.* Retrieved from https://www.tensortoolbox.org/

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

Chang, C.-C., & Lin, C.-J. (2011, May). Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, *2*(3). Retrieved from https://doi.org/10.1145/1961189.1961199

Chen, C., Batselier, K., Ko, C.-Y., Wong, N. (2019). A support tensor train machine. *2019 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8).

Cichocki, A. (2011). Tensor decompositions: new concepts in brain data analysis? *Journal of the Society of Instrument and Control Engineers*, *50*(7), 507–516.

Cichocki, A., Lee, N., Oseledets, I., Phan, A.-H., Zhao, Q., Mandic, D.P. (2016). Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *FNT in Machine Learning*, *9*(4-5), 249-429.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*, 273-297.

de Silva, V., & Lim, L.-H. (2008). Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications*, *30*(3), 1084–1127.

Guo, W., Kotsia, I., Patras, I. (2012). Tensor learning for regression. *IEEE Transactions on Image Processing*, *21*(2), 816-827.

Hao, Z., He, L., Chen, B., Yang, X. (2013). A linear support higher-order tensor machine for classification. *IEEE Transactions on Image Processing*, *22*(7), 2911-2920.

He, L., Kong, X., Yu, P.S., Yang, X., Ragin, A.B., Hao, Z. (2014). Dusk: A dual structure-preserving kernel for supervised tensor learning with applications to neuroimages. *Proceedings of the 2014 SIAM International Conference on Data Mining (SDM)*, 127-135. Retrieved from https://epubs.siam.org/doi/abs/10.1137/1.9781611973440.15

He, L., Lu, C.-T., Ding, H., Wang, S., Shen, L., Yu, P.S., Ragin, A.B. (2017). Multi-way multi-level kernel modeling for neuroimaging classification. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6846-6854.

He, L., Lu, C.-T., Ma, G., Wang, S., Shen, L., Yu, P.S., Ragin, A.B. (2017). Kernelized support tensor machines. *Proceedings of the 34th International*

Conference on Machine Learning-Volume 70, 1442–1451.

Hitchcock, F.L. (1927). The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, *6*(1-4), 164-189.

Kolda, T.G., & Bader, B.W. (2009). Tensor decompositions and applications. *SIAM Review*, *51*(3), 455–500.

Kotsia, I., & Patras, I. (2011, June). Support Tucker Machines. *CVPR 2011*, 633-640.

Kour, K., Dolgov, S., Stoll, M., Benner, P. (2023). Efficient structure-preserving support tensor train machine. *Journal of Machine Learning Research*, *24*(4), 1–22. Retrieved from http://jmlr.org/papers/v24/20-1310.html

Lathauwer, L.D., Moor, B.D., Vandewalle, J. (2000). A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, *21*(4), 1253-1278.

Liu, X., Guo, T., He, L., Yang, X. (2015). A low-rank approximation-based transductive support tensor machine for semisupervised classification. *IEEE Transactions on Image Processing*, *24*(6), 1825-1838.

Nion, D., & Lathauwer, L.D. (2008). Fast communication: An enhanced line search scheme for complex-valued tensor decompositions. *Signal Processing*, *88*(3), 749–755.

Oseledets, I., & Dolgov, S. (2023). *Tt-toolbox.* GitHub. Retrieved from https://github.com/oseledets/TT-Toolbox

Oseledets, I., & Tyrtyshnikov, E. (2010). Tt-cross approximation for multidimensional arrays. *Linear Algebra and its Applications*, *432*(1), 70-88. Retrieved from https://www.sciencedirect.com/science/article/pii/S0024379509003747

Oseledets, I.V. (2011). Tensor-train decomposition. *SIAM Journal on Scientific Computing*, *33*(5), 2295–2317.

Pirsiavash, H., Ramanan, D., Fowlkes, C.C. (2009). Bilinear classifiers for visual recognition. *Advances in Neural Information Processing Systems 22*, 1482–1490.

Signoretto, M., Olivetti, E., Lathauwer, L.D., Suykens, J.A.K. (2011). A kernel-based framework to tensorial data analysis. *Neural Networks*, *24*(8), 861 - 874.

Signoretto, M., Olivetti, E., Lathauwer, L.D., Suykens, J.A.K. (2012). Classification of multichannel signals with cumulant-based kernels. *IEEE Transactions on Signal Processing*, *60*(5), 2304-2314.

Signoretto, M., Tran Dinh, Q., De Lathauwer, L., Suykens, J.A.K. (2014). Learning with tensors: a framework based on convex optimization and spectral regularization. *Machine Learning*, *94*(3), 303–351. Retrieved from https://doi.org/10.1007/s10994-013-5366-3

Taguchi, Y.-H., & Turki, T. (2021). Mathematical formulation and application of kernel tensor decomposition based unsupervised feature extraction. *Knowledge-Based Systems*, *217*, 106834. Retrieved from https://doi.org/10.1016/j.knosys.2021.106834

Tao, D., Li, X., Hu, W., Maybank, S., Wu, X. (2005). Supervised tensor learning. *Proceedings of the Fifth IEEE International Conference on Data Mining*, 450-457. Retrieved from https://doi.org/10.1109/ICDM.2005.139

Tao, D., Li, X., Hu, W., Maybank, S., Wu, X. (2007). Supervised tensor learning. *Knowledge and Information Systems*, 1-42.

Tucker, L.R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, *31*, 279-311. Retrieved from https://doi.org/10.1007/BF02289464

Vannieuwenhoven, N., Vandebril, R., Meerbergen, K. (2012). A new truncation strategy for the higher-order singular value decomposition. *SIAM Journal on Scientific Computing*, *34*(2), A1027-A1052.

Vapnik, V. (1995). *The nature of statistical learning theory.* New York: Springer-Verlag.

Vapnik, V. (1998). *Statistical learning theory*. Wiley-Interscience.

Wolf, L., Jhuang, H., Hazan, T. (2007). Modeling appearances with low-rank SVM. *IEEE Conference on Computer Vision and Pattern Recognition*.

Yan, S., Xu, D., Yang, Q., Zhang, L., Tang, X., Zhang, H.-J. (2007). Multilinear discriminant analysis for face recognition. *IEEE Transactions on Image Processing*, *16*(1), 212-220.

Zeng, D., Wang, S., Shen, Y., Shi, C. (2017). A GA-based feature selection and parameter optimization for support tucker machine. *Procedia Computer Science*, *111*, 17 - 23.

Zhao, Q., Zhou, G., Adali, T., Zhang, L., Cichocki, A. (2013a). Kernelization of tensor-based models for multiway data analysis: Processing of multidimensional structured data. *IEEE Signal Processing Magazine*, *30*(4), 137-148.

Zhao, Q., Zhou, G., Adali, T., Zhang, L., Cichocki, A. (2013b). Kernelization of tensor-based models for multiway data analysis: Processing of multidimensional structured data. *IEEE Signal Processing Magazine*, *30*(4), 137-148.

Zhou, H., Li, L., Zhu, H. (2013). Tensor regression with applications in neuroimaging data analysis. *Journal of the American Statistical Association*, *108*(502), 540-552.