
Efficient Structure-preserving Support Tensor Train Machine

Kirandeep Kour¹ Sergey Dolgov² Martin Stoll³ Peter Benner^{1,3}

Abstract

Deploying the multi-relational tensor structure of a high dimensional feature space, more efficiently improves the performance of machine learning algorithms. One encounters the *curse of dimensionality*, and working with vectorized data fails to preserve the data structure. To mitigate the nonlinear relationship of tensor data more economically, we propose the *Tensor Train Multi-way Multi-level Kernel (TT-MMK)*. This technique combines kernel filtering of the initial input data (*Kernelized Tensor Train (KTT)*), stable reparametrization of the KTT in the Canonical Polyadic (CP) format, and the Dual Structure-preserving Support Vector Machine (SVM) Kernel for revealing nonlinear relationships. We demonstrate numerically that the TT-MMK method is more reliable computationally, is less sensitive to tuning parameters, and gives higher prediction accuracy in the SVM classification compared to similar tensorised SVM methods.

1. Introduction

In many real world applications, resulted data are high-dimensional and stored as tensors. It is also often very expensive to generate or collect the data and we assume that we are not necessarily given a large number of test and training data. Nevertheless, it remains crucial to classify the obtained, and, in the cases considered here, tensorial data points. A prototypical example of this type are fMRI brain images (Glover, 2011) that come with voxel valued rather than pixel valued images and a temporal dimension.

Among the most popular methods for classifying data points are support vector machines (SVM). These are based on a margin maximization and the computation of the corres-

ponding weights via an optimization framework, typically the SMO algorithm (Platt, 1999). These methods often show outstanding performance, but the standard SVM model (Cortes & Vapnik, 1995) is designed for vector-valued rather than tensor-valued data. Although tensor objects can be reshaped into vectors, much of the information inherent in the data is lost. For example, in an fMRI image, adjacent voxels usually show a similarity in the data (He et al., 2014). As a result, the transition of vector-valued SVM to tensor-valued SVM has been proposed under the name supervised tensor learning (STL) (Tao et al., 2007; Zhou et al., 2013; Guo et al., 2012). In (Wolf et al., 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 (Pirsiavash et al., 2009) relaxed the orthogonality constraints to further improve Wolf's method. (Hao et al., 2013) consider rank-one tensor factorization of each input tensor, while (Kotsia & Patras, 2011) adopted the Tucker decomposition of the weight parameter instead of rank-one tensor decompositions to retain more structural information. (Zeng et al., 2017) extended this by using a Genetic Algorithm (GA) prior to the Support Tucker Machine for contraction of the input feature tensor. Along with these rank-one and Tucker representations, recently the weight tensor of STL has been approximated using a Tensor Train (TT) decomposition (Chen et al., 2018). We point out that these methods are mainly focusing on a linear representation of the data. It is well known for standard classification that a linear decision boundary is often not suitable for separation of the given data.

Naturally, the goal is to design a nonlinear transformation of the data and we refer to (Signoretto et al., 2011; 2012; Zhao et al., 2013) where kernel methods have been used for tensor data. These methods have in common that their approaches are based on MLSVD/HOSVD, which rely on the flattening of the tensor data. Therefore, the resulting vector and matrix dimensions are so high that the methods are prone to overfitting. Moreover, the intrinsic tensor structure is typically lost. Thus, other approaches are desired.

Throughout scientific computing and data science the approximation of tensors based on low-rank tensor formats has received much attention over recent years (Cichocki et al., 2016; Kolda & Bader, 2009; Cichocki, 2013; Liu et al.,

¹Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg, Germany. ²Department of Mathematical Sciences, University of Bath, Bath, United Kingdom. ³Faculty of Mathematics, Technische Universität Chemnitz, Chemnitz, Germany. Correspondence to: Kirandeep Kour <kour@mpi-magdeburg.mpg.de>.

2015). Particularly tailored to SVM and tensor data is the method introduced in (He et al., 2014), where the authors construct structure-preserving kernels for STL. The mapping is defined on a rank-one tensor. The authors consider a rank-one tensor decomposition coming from the Canonical Polyadic/ PARAFAC (CP) format (Hitchcock, 1927; 1928). This kernel is known as Dual Structure-preserving Kernel (DuSK) and we discuss its properties in more detail in Section 3.6. The CP approximation results in an accurate and efficient scheme but the CP factorization can be numerically unstable (de Silva & Lim, 2008), and choosing the best rank for CP is an NP hard problem (Håstad, 1989). Lately, kernelized tensor factorizations, specifically a kernelized CP (KCP) factorization, have been used in (He et al., 2017a), which is called the Multi-way Multi-level Kernel (MMK) method. Further elaborating and understanding the approach of KCP (He et al., 2017b) provides a kernelized Tucker model, inspired from (Signoretto et al., 2013).

Going further into tensor decomposition, recently, kernel approximation for TT has been introduced in (Chen et al., 2020). We had pursued similar ideas, but the results obtained for the datasets we will later present were not satisfactory. Hence, we tried to come up with a better exploitation of the data structure, which led to the results presented in this paper.

Tensor decompositions have become an indispensable tool in many learning tasks. For example, (Novikov et al., 2016) uses the TT decomposition for both the input tensor and the corresponding weight parameter in generalized linear models of machine learning. A Kernel Principal Component Analysis (KPCA), a kernel based nonlinear feature extraction technique, was proposed in (Wu & Farquhar, 2007). The paper (Lebedev et al., 2014) proposes a way to speed up Convolutional Neural Networks (CNN) by applying a low-rank CP decomposition on a kernel projection tensor.

1.1. Main Novelty

As can be seen from our previous discussion, tensor methods bring great potential for dealing with high-dimensional data. Nevertheless, several gaps remain, and we address these challenges through the following contributions.

- We are expanding the use of efficient learning from rank-one decomposition to a general framework by using a TT decomposition. This is more robust, stable and computationally efficient.
- We are proposing an exact expansion of a TT to the CP format, where we add two more constraints, namely the norm equilibration and the uniqueness of Singular Value Decomposition (SVD). The norm equilibration ensures equal chances of each TT core to be important and allows using same tuning parameters for all TT

cores. The uniqueness of SVD provides a stable TT decomposition, in a sense that close tensors yield close TT cores, which is in general not true. We have observed that this is the most crucial part of the proposed classification model, which increases the classification accuracy of STL.

- For STL to learn effectively, we regularize the input data by means of defining a best low-rank approximation via kernel projection over latent factors of the tensor input.
- We also provide a kernel approximation for the best low-rank input data by defining a nonlinear mapping into the tensor product Hilbert space. This way we extract information from the low-rank tensor factors.

In order to achieve this, we structure the paper as follows. In Section 2, we set the stage introducing basic definitions and important tools. The extension to the tensor format SVM is explained in Section 2.3 and we also introduce the Kernelized Support Tensor Machine (KSTM) via the kernel trick (Sec. 2.2.1). In Section 3 we explain the entire proposed algorithm. In particular, In Section 3 we introduce the TT-CP expansion, the norm equilibrium and discuss the uniqueness of SVD along with the proposed algorithm. Section 4 shows a range of numerical results for standard datasets and in comparison to a variety of competing approaches.

2. Preliminaries

2.1. Tensor decompositions

2.1.1. CANONICAL POLYADIC DECOMPOSITION

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

$$\mathcal{X}_{i_1, \dots, i_M} \cong \sum_{r=1}^R \mathbf{a}_{i_1, r}^{(1)} \mathbf{a}_{i_2, r}^{(2)} \dots \mathbf{a}_{i_M, r}^{(M)}, \quad \text{or shortly}$$

$$\mathcal{X} \cong \llbracket \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \dots, \mathbf{A}^{(M)} \rrbracket, \quad (1)$$

where $\mathbf{A}^{(m)} = [\mathbf{a}_{i_m, r}^{(m)}] \in \mathbb{R}^{I_m \times R}$, $m = 1, \dots, M$, are called factor matrices of the CP decomposition, see Figure 1, and R is called the CP-rank. CP can be computed via the Alternating Least Squares (ALS) method (Nion & Lathauwer, 2008), but the convergence can be slow, also it is unclear how to choose R .

2.1.2. TENSOR TRAIN DECOMPOSITION

The Tensor Train (TT) (Oseledets, 2011) decomposition is the basis for our proposed algorithm. The TT approximation

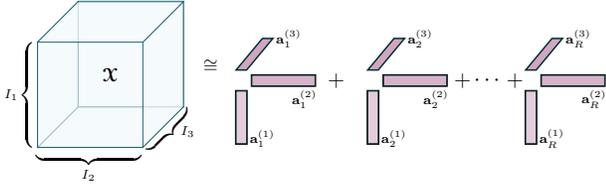


Figure 1. CP decompositions for a 3-way tensor.

of an M^{th} -order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_M}$ is defined as

$$\mathcal{X}_{i_1, \dots, i_M} \cong \sum_{r_0, \dots, r_M} \mathfrak{G}_{r_0, i_1, r_1}^{(1)} \mathfrak{G}_{r_1, i_1, r_2}^{(2)} \dots \mathfrak{G}_{r_{M-1}, i_M, r_M}^{(M)},$$

$$\mathcal{X} \cong \langle\langle \mathfrak{G}^{(1)}, \mathfrak{G}^{(2)}, \dots, \mathfrak{G}^{(M)} \rangle\rangle, \quad (2)$$

where $\mathfrak{G}^{(m)} \in \mathbb{R}^{R_{m-1} \times I_m \times R_m}$, $m = 1, \dots, M$, are 3rd-order tensors called *TT-cores* (see Figure 2), and R_0, \dots, R_M with $R_0 = R_M = 1$ are called *TT-ranks*.

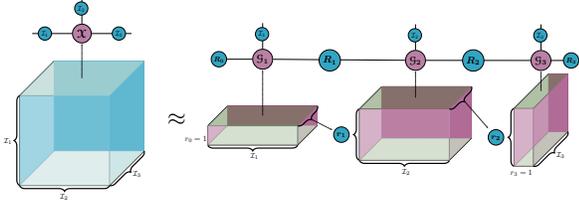


Figure 2. TT decompositions for a 3-way tensor.

The alluring capability of the TT format is its ability to perform algebraic operations directly on TT-cores avoiding full tensors. Moreover, we can compute a quasi-optimal TT approximation of any given tensor using SVD. For more details on TT foundations we refer to (Oseledets, 2011).

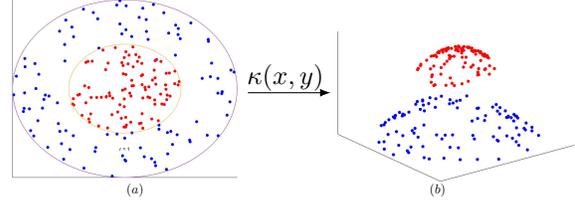
2.2. Support Vector Machine

In this section, we review the SVM method. For a given training dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, with *input data* $\mathbf{x}_i \in \mathbb{R}^m$ and *labels* $y_i \in \{-1, 1\}$, the dual-optimization problem can be defined as follows,

$$\max_{\alpha} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \quad (3)$$

$$\text{subject to } 0 \leq \alpha_i \leq C, \quad \sum_{i=1}^M \alpha_i y_i = 0. \quad (4)$$

where, the unknown function ϕ defines a nonlinear decision boundary with $\phi: \mathbf{x}_i \rightarrow \phi(\mathbf{x}_i)$. Efficient evaluation of $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ is done by the *Kernel Trick* approximation.


 Figure 3. Nonlinear mapping using kernel trick. (a) Nonlinear classification of data in \mathbb{R}^2 . (b) Linear classification in high-dimension (\mathbb{R}^3).

2.2.1. FEATURE MAP AND KERNEL TRICK

The **feature map** is a map $\phi: \mathbb{X} \rightarrow \mathbb{F}$, where \mathbb{F} is a Hilbert Space (HS), the feature space. Also, every feature map is defined via a kernel such that $\kappa_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathbb{F}}$. Employing properties of inner products, $[\kappa_{i,j}]$ is a symmetric positive definite matrix. This approach is known as the *kernel trick*, it is used in order to get a linear learning algorithm to learn a *nonlinear boundary*, without explicitly knowing the nonlinear function ϕ . In case of SVM, the only task needed is to choose a legitimate kernel function. That is how we work with the multi-way input data in the high-dimensional space while doing all the computation in the original low dimensional space. Figure 3 illustrates the linear separation in a higher dimensional space.

2.3. Kernelized Support Tensor Machine

In our case, we have a dataset $\{(\mathcal{X}_i, y_i)\}_{i=1}^N$ with input data in the form of a tensor $\mathcal{X}_i \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_M}$. Hence, the tensor extension of the dual nonlinear SVM from (3) can written as follows:

$$\max_{\alpha} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \langle \phi(\mathcal{X}_i), \phi(\mathcal{X}_j) \rangle$$

$$\text{subject to } 0 \leq \alpha_i \leq C, \quad \sum_{i=1}^N \alpha_i y_i = 0. \quad (5)$$

The classification setup given in (5) is known as Support Tensor Machine (STM) (Tao et al., 2007). Also, the nonlinear feature mapping $\phi: \mathcal{X} \rightarrow \phi(\mathcal{X})$ takes tensor input data to a higher dimensional space, similar to the vector case.

Therefore, by using the kernel trick, explained in Section

2.2.1, STM can defined as follows:

$$\max_{\alpha} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \kappa(\mathcal{X}_i, \mathcal{X}_j) \quad (6)$$

$$\text{subject to } 0 \leq \alpha_i \leq C, \quad \sum_{i=1}^M \alpha_i y_i = 0. \quad (7)$$

We call this setup, the *Kernelized STM (KSTM)*. For the learning algorithm KSTM, the essential term is the kernel $\kappa(\mathcal{X}_i, \mathcal{X}_j)$. Therefore this is the preeminent part for computational aspects. We are proposing a computationally efficient kernel κ , containing all the dominant features required for training and testing of the algorithm next.

2.4. Tensor Product Reproducing Kernel Hilbert Space

Assume $(\mathcal{H}_m, \langle \cdot, \cdot \rangle_m, \kappa^{(m)})$ is a Reproducing Kernel Hilbert Space (RKHS) of a function $\kappa^{(m)}$ defined on the set \mathcal{X}_m , where $\kappa^{(m)}: \mathcal{X}_m \times \mathcal{X}_m \rightarrow \mathbb{R}$ is a reproducing kernel (Berlinet & Thomas-Agnan, 2004) of HS \mathcal{H}_m , for any $m = 1, \dots, M$, and $\langle \cdot, \cdot \rangle_m$ defines an inner product on the same space. Following the definition provided in (Signoretto et al., 2013), the space $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \dots \otimes \mathcal{H}_M$ is called a Tensor product RKHS (TP-RKHS) and is denoted by $(\mathcal{H}, \langle \cdot, \cdot \rangle, \kappa)$ with κ as a *reproducing kernel*.

We are interested in representing functions $f(x) \in \mathcal{H}$ of the tuple $x = (x^{(1)}, x^{(2)}, \dots, x^{(M)}) \in \mathcal{X}$, where $\mathcal{X} := \mathcal{X}_1 \times \mathcal{X}_2 \times \dots \times \mathcal{X}_M$. This allows us to express any $f \in \mathcal{H}$, and hence to form the tensor product space using linear combination of tensor product (“rank-one”) functions:

$$f(x) = \sum_j \alpha_j f_{j_1}^{(1)}(x_1) \otimes \dots \otimes f_{j_M}^{(M)}(x_M),$$

where $j = (j_1, \dots, j_M)$ is a tuple index, and $f_{j_m}^{(m)} \in \mathcal{H}_m$, $m = 1, \dots, M$. Additionally, from the definition of RKHS on each set \mathcal{X}_m it holds that

$$f(x) = \sum_j \alpha_j \prod_{m=1}^M \langle f_{j_m}^{(m)}, \kappa_{x_m}^{(m)} \rangle_m, \quad (8)$$

where $\kappa_{x_m}^{(m)} = \kappa^{(m)}(x_m, \cdot)$ is the reproducing kernel evaluated at the target point x_m . This gives the tensor product reproducing kernel

$$\kappa = \kappa^{(1)} \otimes \dots \otimes \kappa^{(M)}.$$

The concept of RKHS can be used to associate a given tensor input data to a function, which allows us to filter irrelevant components from the tensor, such as the measurement noise. This is explained in Section 3.5.

3. The proposed algorithm

In this paper, the whole research idea evolves around solving the basic classification problem. Further, we have given a training dataset $\{\mathcal{X}_i \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_M}, y_i\}_{i=1}^N \subset \mathbb{X} \times \mathbb{Y}$, where \mathbb{X} is an input tensor space and \mathbb{Y} is a set of class labels containing $\{-1, 1\}$. The main accomplishment is to find a function $f: \mathbb{X} \rightarrow \mathbb{Y}$, which can approximate and predict the corresponding class labels for any test input data, while maintaining the Bias-Variance trade-off (Hastie et al., 2001) of the predicted model.

3.1. TT-CP Expansion

Despite the numerical difficulties outlined in the introduction, the simplicity of the CP decomposition makes it a convenient and intuitive tool for structuring and analyzing the input tensor data. The TT decomposition is easy to compute, but different TT blocks might have different scales; besides, they may be not unique. This complicates the TT data classification with SVM. In this section, we propose a combined representation to elucidate both of these obstacles. First, we compute a TT decomposition of the data in order to get a robust approximation. Second, the resulting TT-cores are converted into the CP format keeping the structure of the original data. This means the data is basically coming from TT cores, while being structured in CP format. This can be done in the following manner.

Let the TT approximation of the input $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ be

$$\mathcal{X}_{i_1, i_2, i_3} = \sum_{r_1, r_2} \mathcal{G}_{i_1, r_1}^{(1)} \mathcal{G}_{r_1, i_2, r_2}^{(2)} \mathcal{G}_{r_2, i_3}^{(3)}$$

then we can merge r_1, r_2 into one index $r = r_1 + (r_2 - 1)R_1$ and write a CP decomposition,

$$\mathcal{X}_{i_1, i_2, i_3} = \sum_{r=1}^{R_1 R_2} H_{i_1, r}^{(1)} H_{i_2, r}^{(2)} H_{i_3, r}^{(3)},$$

where the CP factors are defined as

$$\begin{aligned} H_{i_1, r}^{(1)} &= \mathcal{G}_{i_1, r_1}^{(1)} \mathbf{1}_{r_2}, \\ H_{i_2, r}^{(2)} &= \mathcal{G}_{r_1, i_2, r_2}^{(2)}, \\ H_{i_3, r}^{(3)} &= \mathbf{1}_{r_1} \mathcal{G}_{r_2, i_3}^{(3)}, \end{aligned}$$

using $\mathbf{1}$ for a vector of all ones. This is simply rearranging of the data using permutation and reshaping from one form of a tensor decomposition to another (TT to CP). Notice that this expansion is exact and actually does not need any new computations (only copying of data). In particular, the CP factors can be computed in Matlab using the following commands:

```
H1 = reshape(permute(G1, [2, 1, 3]), I1, R1);
H1 = kron(ones(1, R2), H1);
H2 = reshape(permute(G2, [2, 1, 3]), I2, R1*R2);
H3 = reshape(permute(G3, [2, 1, 3]), I3, R2);
H3 = kron(H3, ones(1, R1));
```

3.2. Norm Equilibration

The above explained TT-CP expansion is used in our preliminary version of the kernel for solving the optimization problem (6). However, this did not lead to better classification results. Notice that (He et al., 2017a) used the CP-ALS algorithm (Nion & Lathauwer, 2008) to compute a CP decomposition, in which the norms of all CP vectors $\mathbf{a}_r^{(1)}, \dots, \mathbf{a}_r^{(M)}$ for the same r were equal. In contrast, the plain TT-SVD algorithm (Oseledets, 2011) does not enforce the norm equilibration itself. Therefore, we add the norm equilibrium constraint to the TT-CP expansion. We have found it to be one key ingredient for the successful application of the proposed TT-SVM method.

For each given input tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, the TT-CP expansion is $\llbracket H^{(1)}, H^{(2)}, H^{(3)} \rrbracket$ with rank r , where we use the so-called Kruskal notation from (1). The norm equilibration of the first core of the TT-CP expansion is completed in the following way,

$$\begin{aligned} \|\mathcal{N}_r\| &= \left\| H_r^{(1)} \right\| * \left\| H_r^{(2)} \right\| * \left\| H_r^{(3)} \right\|, \\ H_r^{(1)} &:= \frac{H_r^{(1)}}{\left\| H_r^{(1)} \right\|} * \|\mathcal{N}_r\|^{1/3}, \\ H_r^{(1)} &:= H_r^{(1)} * \frac{\left(\left\| H_r^{(2)} \right\| * \left\| H_r^{(3)} \right\| \right)^{1/3}}{\left\| H_r^{(1)} \right\|^{2/3}}. \end{aligned}$$

A similar approach has been used for each core of each input tensor. The motivation behind norm equilibrium is to distribute equal chances to each core to be important enough in terms of accumulating the original data in it.

3.3. Uniqueness of SVD

Since the TT decomposition is computed using the SVD (Oseledets, 2011), the particular factors $\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \mathcal{G}^{(3)}$ are defined only up to a sign indeterminacy. For example, in the first step we compute the SVD of the 1-mode unfolding,

$$\mathcal{X}_{(1)} = \sigma_1 u_1 v_1^\top + \dots + \sigma_{I_1} u_{I_1} v_{I_1}^\top,$$

followed by truncating the expansion at rank R_1 or according to the accuracy threshold ε , choosing R_1 such that $\sigma_{R_1+1} < \varepsilon$. However, any set of vectors $\{u_{r_1}, v_{r_1}\}$ can be replaced by $\{-u_{r_1}, -v_{r_1}\}$ without changing the whole tensor. While this is not an issue for data compression, classification using TT factors can be affected significantly by this indeterminacy. For example, tensors that are close to each other should likely produce the same label. In contrast, even a small difference in the original data may lead to a different sign of the singular vectors, and the SVM might assign different labels to such factors.

To circumvent this issue, we fix the signs of the singular vectors as follows. For each $r_1 = 1, \dots, R_1$, we find the position of the maximum in modulus element in the left singular vector, $i_* = \arg \max_i |u_{r_1}(i)|$, and make this element positive,

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

Finally, we collect \bar{u}_{r_1} into the first TT core, $\mathcal{G}^{(1)}(i_1, r_1) = \bar{u}_{r_1}(i_1)$, and continue with the TT-SVD algorithm using \bar{v}_{r_1} as the right singular vectors. In contrast to the sign, the whole dominant singular terms $u_{r_1} v_{r_1}^\top$ depend continuously on the input data, and so do the maximum absolute elements. This ensures that close tensors produce close TT cores.

3.4. Noisy Data and Threshold

Generally, data coming from real world applications are affected by measurement or preprocessing noise. This can affect both computational and modeling aspects, increasing the TT ranks (since a tensor of noise lacks any meaningful TT decomposition), and spoiling the classification if the noise is too large. However, SVD can serve as a de-noising algorithm automatically: the dominant singular vectors are often “smooth” and hence represent a useful image, while the latter singular vectors are more oscillating and capture primarily the noise. Therefore, it is actually beneficial to compute the TT approximation with deliberately low TT ranks / large truncation threshold. On the other hand, the TT rank must not be too low in order to approximate the features of the tensor with sufficient accuracy. Since the precise magnitude of the noise is unknown, we carry out a cross-validation test to find the optimal TT rank.

3.5. Kernelized Tensor Train (KTT)

Tensor decompositions can provide a compact form, while containing the multi-way array structure and keeping the meaning of tensor input data. However, we may need to regularize the underlying input tensor data, so that information can be captured well enough by the learning model (STL). This regularization for the learning model has been proposed for rank-one decomposition of data in (Signoretto et al., 2013), and is based on RKHS.

Therefore, for getting most out of all the latent structural features of a tensor, we treat an M -th order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_M}$ as an element of the TP-RKHS \mathcal{H} , and assume that it has low-rank structure in space \mathcal{H} , such that the following fitting criterion holds (He et al., 2017b):

$$\begin{aligned} f^* &= \underset{f_{j_m}^{(m)}}{\text{argmin}} \sum_i \left(\mathcal{X}_i - \sum_j \prod_{m=1}^M \left\langle f_{j_m}^{(m)}, \kappa_{x_{i_m}}^{(m)} \right\rangle_m \right)^2 \\ &\implies \min_{F^{(m)}} \left\| \mathcal{X} - \llbracket K^{(1)} F^{(1)}, \dots, K^{(M)} F^{(M)} \rrbracket_F \right\|_F^2, \end{aligned} \quad (9)$$

where $K^{(m)} \in \mathbb{R}^{I_m \times I_m}$ and $F^{(m)} \in \mathbb{R}^{I_m \times J_m}$ consist of samples of $\kappa^{(m)}$ and $f_{j_m}^{(m)}$, respectively. This formulation is called Kernelized Tensor Factorization (KTF).

In particular, we are using the TT-CP expansion as a tensor factorization for KTF and we call it Kernelized Tensor Train (KTT). The formulation of a kernelized projection on tensor input data \mathcal{X} was demonstrated in (Lebedev et al., 2014) for the CP-decomposition, which achieves considerable speedups with minimal loss in accuracy. Acknowledging it and using (Bazerque et al., 2012), we extend the rank-one idea to TT-CP and work towards getting a best KTT approximation.

For keeping it simple, without loss of generality, let the input is a 3-dimensional tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, with the TT-CP expansion $\llbracket H^{(1)}, H^{(2)}, H^{(3)} \rrbracket$. In order to get the kernel version of the tensor factorization cores, we define a mapping from input tensor to TT-CP expansion cores. Following the (Bazerque et al., 2012), a nonparametric function $f: \mathbf{X} \times \mathbf{Y} \times \mathbf{Z} \rightarrow \mathbb{R}$ is introduced such that

$$\mathcal{F}_R = \left\{ f: f(x, y, z) \mapsto \sum_{r=1}^R \hat{h}_r^{(1)}(x) \hat{h}_r^{(2)}(y) \hat{h}_r^{(3)}(z), \right. \\ \left. \text{s.t. } \hat{h}_r^{(1)} \in \mathcal{H}_1, \hat{h}_r^{(2)} \in \mathcal{H}_2, \hat{h}_r^{(3)} \in \mathcal{H}_3 \right\},$$

where $\mathcal{H}_1, \mathcal{H}_2$ and \mathcal{H}_3 are HS constructed with respect to kernels $\kappa^{(1)}, \kappa^{(2)}, \kappa^{(3)}$, respectively. From the definition of RKHS,

$$\hat{h}_r^{(1)}(x) = \sum_{i=1}^I h_{i,r}^{(1)} \kappa^{(1)}(x_i, x), \\ \hat{h}_r^{(2)}(y) = \sum_{j=1}^J h_{j,r}^{(2)} \kappa^{(2)}(y_j, y), \\ \hat{h}_r^{(3)}(z) = \sum_{k=1}^K h_{k,r}^{(3)} \kappa^{(3)}(z_k, z),$$

therefore, the interpolation of TT-CP expansion into a continuous function is,

$$\mathcal{F}^* = \hat{f}(x, y, z) = \sum_{r=1}^R \hat{h}_r^{(1)}(x) \hat{h}_r^{(2)}(y) \hat{h}_r^{(3)}(z).$$

Hence, the optimization problem mentioned in (9) can be written in the following parametric way (He et al., 2017a):

$$f^* = \underset{f \in \mathcal{F}_R}{\operatorname{argmin}} \sum_{i=1}^I \sum_{j=1}^J \sum_{k=1}^K \left(x_{i,j,k} - \hat{f}(x_i, y_j, z_k) \right)^2 \quad (11)$$

$$\implies \min_{H^{(m)}} \left\| \mathcal{X} - \llbracket K^{(1)} H^{(1)}, K^{(2)} H^{(2)}, K^{(3)} H^{(3)} \rrbracket \right\|_F^2, \quad (12)$$

with kernel matrices

$$K^{(1)} = \left[\kappa^{(1)}(x_i, x_j) \right] \in \mathbb{R}^{I \times I}, \\ K^{(2)} = \left[\kappa^{(2)}(y_i, y_j) \right] \in \mathbb{R}^{J \times J}, \quad \text{and} \\ K^{(3)} = \left[\kappa^{(3)}(z_i, z_j) \right] \in \mathbb{R}^{K \times K}.$$

These kernel matrices work as a filter over latent structure along each dimension (mode) of a tensor factorization. Consequently, this helps to reveal the pattern of an input image data, similarly to an affine transformation of the first layer of a CNN. To simplify the presentation, we will assume that KTT form of two tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I \times J \times K}$ denoted by using the same notation $\llbracket H^{(1)}, H^{(2)}, H^{(3)} \rrbracket, \llbracket P^{(1)}, P^{(2)}, P^{(3)} \rrbracket$.

3.6. Nonlinear Mapping

The optimal features or the best low-rank approximation of the input tensor object have been obtained via the KTT. After this, the next step is to get the nonlinear classification boundary using KSTM. This leads to the formulation of the most expensive and prominent term in (6). Therefore, the corresponding kernel approximation will be explained further in this section.

In order to get a nonlinear classification boundary for the input tensor data. The downstream computation is that the extracted optimal features (KTT) should reach to the learning model (KSTM) efficiently. As we have seen, the KTT factorization provides factor matrices. Each of these matrices is associated with each of the tensor modes. These factor matrices can be defined in a tensor fashion by using the outer product. Therefore, we extend the idea of the MMK (He et al., 2017a) method for KTT. Hence, a nonlinear mapping from the original feature space for the KTT factors to a third-order HS can be defined in the following way

$$\Psi: \mathbf{X} \times \mathbf{Y} \times \mathbf{Z} \mapsto \mathbb{R}^{\mathcal{H}_1 \times \mathcal{H}_2 \times \mathcal{H}_3}$$

$$\Psi: \sum_{r=1}^R h_r^{(1)} \otimes h_r^{(2)} \otimes h_r^{(3)} \mapsto \sum_{r=1}^R \phi(h_r^{(1)}) \otimes \phi(h_r^{(2)}) \otimes \phi(h_r^{(3)}). \quad (13)$$

Similarly to the standard SVM, applying the kernel trick in equation (13) gives us a practically computable kernel inspired by DuSK (He et al., 2014; 2017a):

$$\langle \Psi(\mathcal{X}), \Psi(\mathcal{Y}) \rangle \approx \kappa(\mathcal{X}, \mathcal{Y})$$

$$\approx \kappa \left(\sum_{r=1}^R h_r^{(1)} \otimes h_r^{(2)} \otimes h_r^{(3)}, \sum_{r=1}^R p_r^{(1)} \otimes p_r^{(2)} \otimes p_r^{(3)} \right),$$

$$= \langle \Psi \left(\sum_{r=1}^R h_r^{(1)} \otimes h_r^{(2)} \otimes h_r^{(3)} \right), \Psi \left(\sum_{r=1}^R p_r^{(1)} \otimes p_r^{(2)} \otimes p_r^{(3)} \right) \rangle$$

$$\mapsto \sum_{i,j=1}^R \langle \phi(h_i^{(1)}) \phi(p_j^{(1)}) \rangle \langle \phi(h_i^{(2)}) \phi(p_j^{(2)}) \rangle \langle \phi(h_i^{(3)}) \phi(p_j^{(3)}) \rangle$$

$$\approx \sum_{i,j=1}^R \kappa(h_i^{(1)}, p_j^{(1)}) \kappa(h_i^{(2)}, p_j^{(2)}) \kappa(h_i^{(3)}, p_j^{(3)}).$$

$$\implies \kappa(\mathcal{X}, \mathcal{Y}) \approx \sum_{i,j=1}^R \prod_{k=1}^3 \exp \left(- \frac{\|h_i^{(k)} - p_j^{(k)}\|^2}{2\sigma^2} \right). \quad (14)$$

Algorithm 1 Kernel approximation as an input of STM

Input: data $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, rank $R_1 = R_2 = R$.
Output: $\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \mathcal{G}^{(3)}$
 (Optional) Set kernel matrices $K^{(1)}, K^{(2)}, K^{(3)}$.
for $i, j = 1$ **to** N **do**
 Compute $\langle\langle \mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \mathcal{G}^{(3)} \rangle\rangle \cong \mathcal{X}_i$ (TT-SVD)
 Compute $\langle\langle \mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \mathbf{u}^{(3)} \rangle\rangle \cong \mathcal{X}_j$ (TT-SVD)
 $\langle\langle H^{(1)}, H^{(2)}, H^{(3)} \rangle\rangle = \langle\langle \mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \mathcal{G}^{(3)} \rangle\rangle$ (TT-CP)
 $\langle\langle P^{(1)}, P^{(2)}, P^{(3)} \rangle\rangle = \langle\langle \mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \mathbf{u}^{(3)} \rangle\rangle$ (TT-CP)
 for $k = 1, 2, 3$ **do**
 (Optional) $H^{(k)} \leftarrow (K^{(k)})^T H^{(k)}$, KTT filtering
 (Optional) $P^{(k)} \leftarrow (K^{(k)})^T P^{(k)}$, KTT filtering
 end for
 $\kappa(\mathcal{X}_i, \mathcal{X}_j) \approx$
 $\sum_{i,j=1}^R \kappa(h_i^{(1)}, p_j^{(1)}) \kappa(h_i^{(2)}, p_j^{(2)}) \kappa(h_i^{(3)}, p_j^{(3)})$.
end for

This kernel approximation (14) performs on each pair of tensor input data. This means on each pair of the best low-rank structure (KTT). The width parameter $\sigma > 0$ needs to be chosen judiciously to ensure accurate learning.

As an input of the KSTM learning model, we are using the TT-CP factors. Therefore, we call this proposed model the Tensor Train Multi-way Multi-level Kernel (TT-MMK). It fulfills the objectives of extracting optimal low-rank features, and of building a more accurate and efficient classification model. Plugging the kernel values (14) into the SVM optimizer (6) completes the algorithm. The performance of this model is shown by numerical experiments in the next section. Along with this, the overall idea is summarized in Algorithm 1, where the computation of $H^{(k)} \leftarrow (K^{(k)})^T H^{(k)}$ is optional and can be switched on and off, depending on the input dataset.

4. Numerical Tests

4.1. Experimental Settings

All the experimental implementation has been done in MATLAB 2016b. In the first step, we compute the TT format of an input tensor using the TT-Toolbox¹, where we modified the function `@tt_tensor/round.m` to enforce the uniqueness of SVD (Sec. 3.3). Moreover, we have implemented the TT-CP conversion, together with the norm equilibration. For the training of the TT-MMK model, we have used the `svmtrain` function available in the LIBSVM² library. The kernel filtering matrices are chosen from *random*, *identity* or *covariance* matrices by

¹<https://github.com/oseledets/TT-Toolbox>

²<https://www.csie.ntu.edu.tw/~cjlin/libsvm/>

comparing the cross-validation errors and choosing the matrix providing the best accuracy. We have run all experiments on a machine equipped with Ubuntu release 16.04.6 LTS (Xenial Xerus) 64-bit, 7.7 GiB of memory, and an Intel Core i5-6600 CPU @ 3.30GHz×4 CPU.



Figure 4. 3D tensor corresponding to the fMRI brain image.

4.1.1. PARAMETER TUNING

The ensemble designed model has three various parameters. First, to simplify tuning of the model to noise, we take all TT ranks equal to the same value $R \in \{1, 2, \dots, 10\}$. Another parameter is the width of the Gaussian Kernel σ . Finally, the third parameter is trade-off C for the KSTM optimization technique (7). Both σ and C are chosen from $\{2^{-8}, 2^{-7}, \dots, 2^7, 2^8\}$. For tuning R, σ and C to the best classification accuracy we use the *k-fold cross validation*. More specifically, we use five fold cross-validation. Along with this, we run all computations 50 times and average the accuracy over these runs. This ensures confident and reproducible comparison of different techniques.

4.2. Data Collection

Alzheimer Disease (ADNI): The ADNI stands for Alzheimer Disease Neuroimaging Initiative. This dataset is collected from ADNI³. It contains the resting state fMRI images of 33 subjects. The dataset was collected from the authors of the paper (He et al., 2017a). The data contain two labels, one stands for Mild Cognitive Impairment (MCI) or Alzheimer Disease (AD), and normal controls. The sample size is $61 \times 73 \times 61 = 271633$. The AD+MCI is labeled with -1 , and the normal control is labeled with 1 . Preprocessing of datasets is explained in (He et al., 2014).

Attention Deficit Hyperactivity Disorder (ADHD): The ADHD dataset is collected from ADHD-200 global competition dataset⁴. It is a publicly available preprocessed fMRI dataset from eight different institutes, collected at one place. The dataset is unbalanced, so we have chosen 200 subjects randomly. The classification labels are ADHD patient (-1)

³<http://adni.loni.usc.edu/>

⁴<http://neurobureau.projects.nitrc.org/ADHD200/Data.html>

and healthy person (1). Each of the 200 resting state fMRI samples contains $49 \times 58 \times 47 = 133574$ voxels.

4.3. Methods Compared

We compare the classification accuracy of the proposed TT-MMK method with the accuracy of the following existing approaches.

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

STuM: The Support Tucker Machine (STuM) (Kotsia & Patras, 2011) uses the Tucker decomposition. The weight parameters of SVM are computed for optimization into Tucker factorization form.

DuSK: The idea of DuSK (He et al., 2014) is based on defining the kernel approximation for the rank-one decomposition. This is one of the first methods in this direction.

MMK: This method is an extension of DuSK to the KCP input. It uses the covariance/random matrix projection over the CP factor matrices, to get a KCP for the given input tensor (He et al., 2017a).

Improved MMK: This is a slightly improved MMK, where identity matrices are used for projecting CP onto KCP instead of the covariance/random matrices. This we found out and named it as Improved MMK.

TT-MMK: This is Algorithm 1 proposed in this paper.

4.4. Results

We used original codes provided by the authors of the paper (He et al., 2017a) for comparing the DuSK and MMK methods to our methods. Our key observations are as follows.

(In)sensitivity to the TT rank selection: Figure 5, show that the proposed method gives almost the same accuracy for different TT ranks. For some samples, even the TT rank 2 gives good classification. Note that this is not the case for MMK, which requires a careful selection of the CP rank.

Kernel filtering: the best accuracy is achieved with the identity filter. This may indicate that the TT decomposition captures already a resolution that is just about right, such that no further filtering is needed. We anticipate that pre-filtering might be necessary for larger noise levels, though.

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

Classification accuracy: the proposed method gives the best average classification accuracy (over 73%), compared to five other state of the art techniques.

In addition to the TT-MMK method as described in Algorithm 1, we tried also a straightforward implementation

Table 1. Average classification accuracy for different methods and datasets

| METHODS | ADNI | ADHD |
|--------------|-------------|-------------|
| SVM | 49 % | 52 % |
| STuM | 51 % | 54 % |
| DuSK | 55 % | 58 % |
| MMK | 69 % | 60 % |
| IMPROVED MMK | 71 % | 61 % |
| TT-MMK | 73 % | 64 % |

of the DuSK Kernel (14) for the TT cores obtained directly from the TT-SVD approximation, without enforcing the uniqueness of SVD, the TT-CP conversion and the norm equilibration. A similar approach was recently proposed in (Chen et al., 2020). However, for the ADNI dataset this “simplified” TT-MMK gives only an accuracy of 60%, which is lower than the accuracy of the MMK (He et al., 2017a) method. This demonstrates that all steps of the unification procedure proposed in Section 3 are important for a reliable TT-SVM classification.

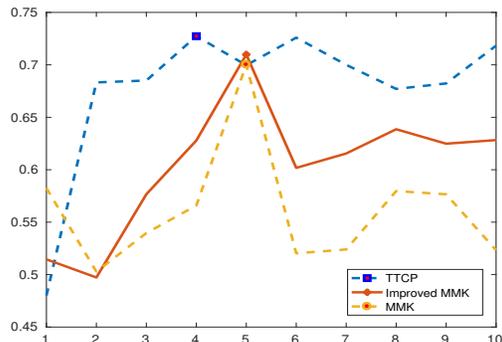


Figure 5. Accuracy of TT-MMK (TTCP), MMK and Improved MMK methods for the ADNI dataset using different TT/CP ranks respectively. TT-MMK and Improved MMK methods use identity kernel filtering while the MMK uses random kernel filtering.

5. Conclusion

We have proposed a new kernel model for SVM classification of tensor input data. Our kernel extends the DuSK approach (He et al., 2017a) to the TT decomposition of the input tensor with enforced uniqueness. The TT decomposition can be computed more reliably than the CP decomposition used in the original DuSK kernel. Using classical fMRI benchmark datasets, we have demonstrated that the new TT-DuSK method provides higher classification accuracy for an unsophisticated choice of the TT ranks. We have also found out that the uniqueness constraints are

crucial for achieving this accuracy. Further research will consider improving the computational complexity of the current scheme, as well as a joint optimization of TT cores and SVM weights. Similarly to the neural network compression in the TT format (Novikov et al., 2015), such a targeted iterative refinement of the TT decomposition may improve the prediction accuracy.

Acknowledgment

This work is supported by the International Max Planck Research School for Advanced Methods in Process and System Engineering-**IMPRS ProEng**, Magdeburg. The authors are grateful to Lifang He for providing codes for the purpose of comparison.

References

- Bazerque, J. A., Mateos, G., and Giannakis, G. B. Nonparametric low-rank tensor imputation. In *IEEE Statistical Signal Processing Workshop (SSP)*, pp. 876–879, 2012.
- Berlinet, A. and Thomas-Agnan, C. *Reproducing Kernel Hilbert Space in Probability and Statistics*. 01 2004.
- Chen, C., Batselier, K., Ko, C.-Y., and Wong, N. A support tensor train machine. *arXiv preprint arXiv: 1804.06114*, 2018.
- Chen, C., Batselier, K., Yu, W., and Wong, N. Kernelized support tensor train machines. *arXiv preprint arXiv:2001.00360*, 2020.
- Cichocki, A. Tensor decompositions: A new concept in brain data analysis. *arXiv preprint arXiv:1305.0395*, 2013.
- Cichocki, A., Lee, N., Oseledets, I., Phan, A.-H., Zhao, Q., and Mandic, D. P. Tensor networks for dimensionality reduction and large-scale optimization: Part I low-rank tensor decompositions. *FNT in Machine Learning*, 9(4-5): 249–429, 2016.
- Cortes, C. and Vapnik, V. Support-vector networks. *Machine Learning*, 20:273–297, 1995.
- de Silva, V. and Lim, L.-H. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM J. Matrix Anal. Appl.*, 30(3):1084–1127, 2008.
- Glover, G. H. Overview of functional magnetic resonance imaging. *Neurosurgery Clinics of North America*, 22(2): 133 – 139, 2011.
- Guo, W., Kotsia, I., and Patras, I. Tensor learning for regression. *IEEE Transactions on Image Processing*, 21(2): 816–827, 2012.
- Hao, Z., He, L., Chen, B., and Yang, X. A linear support higher-order tensor machine for classification. *IEEE Transactions on Image Processing*, 22(7):2911–2920, 2013.
- Håstad, J. Tensor rank is np-complete. In *Automata, Languages and Programming*, pp. 451–460, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg.
- Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, USA, 2001.
- He, L., Kong, X., Yu, P. S., Ragin, A. B., Hao, Z., and Yang, X. Dusk: A dual structure-preserving kernel for supervised tensor learning with applications to neuroimages. *CoRR*, abs/1407.8289, 2014.
- He, L., Lu, C.-T., Ding, H., Wang, S., Shen, L., Yu, P. S., and Ragin, A. B. Multi-way multi-level kernel modeling for neuroimaging classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6846–6854, 2017a.
- He, L., Lu, C.-T., Ma, G., Wang, S., Shen, L., Yu, P. S., and Ragin, A. B. Kernelized support tensor machines. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1442–1451, 2017b.
- Hitchcock, F. L. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.
- Hitchcock, F. L. Multiple invariants and generalized rank of a p-way matrix or tensor. *Journal of Mathematics and Physics*, 7(1-4):39–79, 1928.
- Kolda, T. G. and Bader, B. W. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.
- Kotsia, I. and Patras, I. Support tucker machines. *CVPR 2011*, pp. 633–640, June 2011.
- Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I. V., and Lempitsky, V. S. Speeding-up convolutional neural networks using fine-tuned CP-decomposition. *CoRR*, abs/1412.6553, 2014.
- Liu, X., Guo, T., He, L., and Yang, X. A low-rank approximation-based transductive support tensor machine for semisupervised classification. *IEEE Transactions on Image Processing*, 24(6):1825–1838, 2015.
- Nion, D. and Lathauwer, L. D. Fast communication: An enhanced line search scheme for complex-valued tensor decompositions. Application in DS-CDMA. *Signal Processing*, 88(3):749–755, 2008.

- Novikov, A., Podoprikin, D., Osokin, A., and Vetrov, D. P. Tensorizing neural networks. In *Advances NIPS* 28, pp. 442–450. 2015.
- Novikov, A., Trofimov, M., and Oseledets, I. V. Exponential machines. *arXiv preprint arXiv:1605.03795*, 2016.
- Oseledets, I. V. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.
- Pirsiavash, H., Ramanan, D., and Fowlkes, C. C. Bilinear classifiers for visual recognition. In *Advances in Neural Information Processing Systems* 22, pp. 1482–1490. Curran Associates, Inc., 2009.
- Platt, J. C. *Fast Training of Support Vector Machines Using Sequential Minimal Optimization*, pp. 185–208. MIT Press, 1999.
- Signoretto, M., Olivetti, E., Lathauwer, L. D., and Suykens, J. A. K. A kernel-based framework to tensorial data analysis. *Neural Networks*, 24(8):861 – 874, 2011. Artificial Neural Networks: Selected Papers from ICANN 2010.
- Signoretto, M., Olivetti, E., Lathauwer, L. D., and Suykens, J. A. K. Classification of multichannel signals with cumulant-based kernels. *IEEE Transactions on Signal Processing*, 60(5):2304–2314, 2012.
- Signoretto, M., Lathauwer, L. D., and Suykens, J. A. Learning tensors in reproducing kernel Hilbert spaces with multilinear spectral penalties. *ArXiv*, abs/1310.4977, 2013.
- Tao, D., Li, X., Hu, W., Stephen, M., and Wu, X. Supervised tensor learning. *Knowledge and Information Systems*, pp. 1–42, 2007.
- Wolf, L., Jhuang, H., and Hazan, T. Modeling appearances with low-rank SVM. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.
- Wu, M. and Farquhar, J. A subspace kernel for nonlinear feature extraction. *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, 1125-1130 (2007), 2007.
- Zeng, D., Wang, S., Shen, Y., and Shi, C. A GA-based feature selection and parameter optimization for support tucker machine. *Procedia Computer Science*, 111:17 – 23, 2017. The 8th International Conference on Advances in Information Technology.
- Zhao, Q., Zhou, G., Adali, T., Zhang, L., and Cichocki, A. Kernelization of tensor-based models for multiway data analysis: Processing of multidimensional structured data. *IEEE Signal Processing Magazine*, 30(4):137–148, 2013.
- Zhou, H., Li, L., and Zhu, H. Tensor regression with applications in neuroimaging data analysis. *Journal of the American Statistical Association*, 108(502):540–552, 2013. PMID: 24791032.