Journal of Machine Learning Research 24 (2023) 1-22 Submitted 11/20; Published 1/23

Efficient Structure-preserving Support Tensor Train Machine

Kirandeep Kour KOUR@MPI-MAGDEBURG.MPG.DE

Peter Benner BENNER@MPI-MAGDEBURG.MPG.DE

Computational Methods in Systems and Control Theory
Max Planck Institute for Dynamics of Complex Technical Systems
Magdeburg, D-39106, Germany.

Sergey Dolgov S.DOLGOV@BATH.AC.UK

Department of Mathematical Sciences
University of Bath
Bath BA2 7AY, United Kingdom.

Martin Stoll MARTIN.STOLL@MATHEMATIK.TU-CHEMNITZ.DE

Faculty of Mathematics
Technische Universität Chemnitz
Chemnitz, D-09107, Germany.

Editor: Isabelle Guyon

Abstract

An increasing amount of the collected data are high-dimensional multi-way arrays (tensors), and it is crucial for efficient learning algorithms to exploit this tensorial structure as much as possible. The ever present curse of dimensionality for high dimensional data and the loss of structure when vectorizing the data motivates the use of tailored low-rank tensor classification methods. In the presence of small amounts of training data, kernel methods offer an attractive choice as they provide the possibility for a nonlinear decision boundary. We develop the Tensor Train Multi-way Multi-level Kernel (TT-MMK), which combines the simplicity of the Canonical Polyadic decomposition, the classification power of the Dual Structure-preserving Support Vector Machine, and the reliability of the Tensor Train (TT) approximation. We show by experiments that the TT-MMK method is usually more reliable computationally, less sensitive to tuning parameters, and gives higher prediction accuracy in the SVM classification when benchmarked against other state-of-the-art techniques.

Keywords: Tensor Decomposition, Support Vector Machine, Kernel Approximation, High-dimensional Data, Classification

1. Introduction

In many real world applications, data often emerges in the form of high-dimensional tensors. It is typically very expensive to generate or collect such data, and we assume that we might be given a rather small amount of test and training data. Nevertheless, it remains crucial to be able to classify tensorial data points. A prototypical example of this type is fMRI brain images (Glover, 2011), which consist of three-dimensional tensors of voxels, and may also be equipped with an additional temporal dimension, in contrast to traditional two-dimensional pixel images.

One of the most popular methods for classifying data points are Support Vector Machines (SVM) (Vapnik, 1995, 1998). These are based on margin maximization and the computation of the corresponding weights via an optimization framework, typically the SMO algorithm (Platt, 1998). These methods often show outstanding performance, but the standard SVM model (Cortes and 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 tensorial data is lost. For example, in an fMRI image, the values of adjacent voxels are often close to each other (He et al., 2014). As a result, it was proposed to replace the vector-valued SVM by a tensor-valued SVM. This area was called 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 the Wolf's method. Hao et al. (2013) consider an R-sum rank-one tensor factorization of each input tensor, while Kotsia and Patras (2011) adopted the Tucker decomposition of the weight parameter to retain more structural information. Zeng et al. (2017) extended this by using a Genetic Algorithm (GA) prior to the Support Tucker Machine (STuM) for the contraction of the input feature tensor. Along with these R-sum rank-one tensor and Tucker representations, recently the weight tensor of STL has been approximated using the 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 that a linear decision boundary is often not suitable for the separation of complicated real world data (Hastie et al., 2001).

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. All these methods are based on the Multi-linear Singular Value Decomposition/Higher Order Singular Value Decomposition, 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 over-fitting. Moreover, the intrinsic tensor structure is typically lost. Thus, other approaches are desired.

The approximation of tensors based on low-rank decompositions has received a lot of attention in scientific computing over recent years (Cichocki et al., 2016; Kolda and Bader, 2009; Cichocki, 2013; Liu et al., 2015). A 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 on the Canonical Polyadic (CP) tensor format, also known as Parallel Factor Analysis, or PARAFAC (Hitchcock, 1927, 1928). Once the CP format is available, DuSK delivers an accurate and efficient classification, but the CP approximation of arbitrary data can be numerically unstable and difficult to compute (de Silva and Lim, 2008). In general, any optimization method (Newton, Steepest Descent or Alternating Least Squares) might return only a locally optimal solution, and it is difficult to assess whether this is a local or global optimum. Later on, kernelized tensor factorizations, specifically a Kernelized-CP (KCP) factorization, have been introduced in He et al. (2017a), and the entire technique has been called the Multi-way Multi-level Kernel (MMK) method. Further elaboration and understanding of the KCP approach (He et al., 2017b) is provided by a kernelized Tucker model, inspired by Signoretto et al. (2013).

Recently, kernel approximations in the TT format have been introduced in Chen et al. (2022). Initially, we had pursued a similar idea for fMRI data sets, but we observed that the nonlinear SVM classification using directly the TT factors leads to poor accuracy, since different TT factors have different dimensions and scales, making the feature space more complicated. Hence, we have come up with a better exploitation of the data structure, as we explain in this paper.

Tensor decompositions and kernel-based methods 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 in machine learning. A Kernel Principal Component Analysis (KPCA), a kernel-based nonlinear feature extraction technique, was proposed in Wu and Farquhar (2007). The authors of Lebedev et al. (2014) propose a way to speed up Convolutional Neural Networks (CNN) by applying a low-rank CP decomposition on the kernel projection tensor.

1.1 Main Novelty

In this paper, we develop an efficient structure-preserving nonlinear kernel function for SVM classification of tensorial data, by computing a reliable CP approximation for DuSK. We start with the TT approximation of the data points, which can be computed reliably by the TT-SVD algorithm. Moreover, we enforce uniqueness of the SVD factors, such that “close” tensors yield “close” TT factors. Second, we perform an exact expansion of the TT decomposition into the CP format. This unifies the dimensions of the data used in classification. Finally, we redistribute the norms of the CP factors to equilibrate the actual scales of the data elements. This yields a CP decomposition that is free from scaling indeterminacy, while being a reliable approximation of the original data. We have observed that using this decomposition in DuSK significantly increases the classification accuracy and stability of the STL.

The paper is structured as follows. In Section 2, we set the stage introducing basic definitions and important tools. An extension to the tensor format SVM is explained in Section 2.4, where we also introduce the Kernelized Support Tensor Machine (KSTM) via the kernel trick (Section 2.3.1). In Section 3 we explain the entire proposed algorithm step by step. In particular, we introduce the uniqueness enforcing TT-SVD algorithm (Section 3.1), the TT-CP expansion (Section 3.2) and the norm equilibration (Section 3.3). In Section 4 we benchmark the different steps of the proposed algorithm and compare it to a variety of competing methods using two data sets each from two different fields with a limited amount of training data, which are known to be challenging for classification.

2. Preliminaries

This section introduces terminology and definitions used throughout the paper.

2.1 Tensor Algebra

A tensor is a multidimensional array (Kolda and Bader, 2009) which is a higher order generalization of vectors and matrices. We denote an M^{th}-order tensor ($M \geq 3$) by a calligraphic letter $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$, its entries by $x_{i_1 i_2 \ldots i_M}$, a matrix by a boldface upper case letter $\mathbf{X} \in \mathbb{R}^{I \times J}$, and a vector by a boldface lower case letter $\mathbf{x} \in \mathbb{R}^{I}$. Matrix and vector elements are denoted by $x_{ij} = \mathbf{X}(i,j)$ and $x_i = \mathbf{x}(i)$, respectively. The order of a tensor is the number of its *dimensions*, *ways* or *modes*. The *size* of a tensor stands for the maximum index value in each mode. For example, $\mathcal{X}$ is of order M and the size in each mode is I_m, where $m \in \langle M \rangle := \{1, 2, \ldots, M\}$. For simplicity, we assume that all tensors are real valued.

Definition 1 *An m-mode matricization $\mathcal{X}_{(m)} \in \mathbb{R}^{I_m \times I_1 \ldots I_{m-1} I_{m+1} \ldots I_M}$ for $m \in \langle M \rangle$ is the unfolding (or flattening) of an M^{th}-order tensor into a matrix in the appropriate order of elements, i.e. a tensor element $(i_1, i_2, \ldots i_M)$ maps to an element (i_m, j) of a matrix as follows (Kolda and Bader, 2009):*

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

Definition 2 *An m-mode product $\mathcal{X} \times_m \mathbf{A} \in \mathbb{R}^{I_1 \times \ldots \times I_{m-1} \times J \times I_{m+1}, \times \ldots \times I_M}$, given $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ and $\mathbf{A} \in \mathbb{R}^{J \times I_m}$, is defined as a tensor-matrix product in m^{th} way:*

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

Definition 3 *A mode-(M,1) contracted product $\mathcal{Z} = \mathcal{X} \times_M^1 \mathcal{Y} = \mathcal{X} \times^1 \mathcal{Y} \in \mathbb{R}^{I_1 \times \ldots \times I_{M-1} \times J_2 \times \ldots \times J_M}$, for given tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ and $\mathcal{Y} \in \mathbb{R}^{J_1 \times J_2 \times \ldots \times J_M}$, with $I_M = J_1$, yields a tensor $\mathcal{Z}$ with entries*

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

Definition 4 *The inner product of given tensors $\mathcal{X}$, $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ is defined as*

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

Definition 5 *The outer product of given tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ and $\mathcal{Y} \in \mathbb{R}^{J_1 \times J_2 \times \ldots \times J_N}$ generates an $(M+N)^{th}$ – order tensor $\mathcal{Z} = \mathcal{X} \circ \mathcal{Y}$ with entries*

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

Definition 6 *The Kronecker Product of matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$, $\mathbf{B} \in \mathbb{R}^{K \times L}$ is defined as usual by*

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

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

Moreover, the Khatri-Rao product is a column-wise Kronecker product,

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

These notations are summarized in Table 1.

Table 1: Tensor Notations.

Symbol	Description
x	Lower case letter for scalar value
$\mathbf{x}$	Lower case bold letter for vector
$\mathbf{X}$	Upper case bold letter for matrix
$\boldsymbol{\mathcal{X}}$	Calligraphic bold letter for tensor
$\boldsymbol{\mathcal{X}}_{(m)}$	Calligraphic bold letter with subscript m for m-mode matricization
$\circ$	Outer product
$\otimes$	Kronecker product
$\odot$	Khatri-Rao product
$\times_M^1$	Mode-$(M,1)$ contracted product
$\langle M \rangle$	Integer values from 1 to M
$\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle$	Inner product for tensors $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$

2.2 Tensor Decompositions

Tensor decomposition methods have been significantly enhanced during the last two decades, and applied to solve problems of varying computational complexity. The main goal is the linear (or at most polynomial) scaling of the computational complexity in the dimension (order) of a tensor. The key ingredient is the separation of variables via approximate low-rank factorizations. In this paper we consider two of these decompositions.

2.2.1 CANONICAL POLYADIC DECOMPOSITION

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

$$x_{i_1 i_2 \ldots i_M} \cong \sum_{r=1}^{R} \mathbf{a}_{i_1,r}^{(1)} \mathbf{a}_{i_2,r}^{(2)} \cdots \mathbf{a}_{i_M,r}^{(M)},$$

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

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

2.2.2 TENSOR TRAIN DECOMPOSITION

To alleviate the difficulties of the CP decomposition mentioned above, we build our proposed algorithm on the Tensor Train (TT) (Oseledets, 2011) decomposition. The TT approximation of an

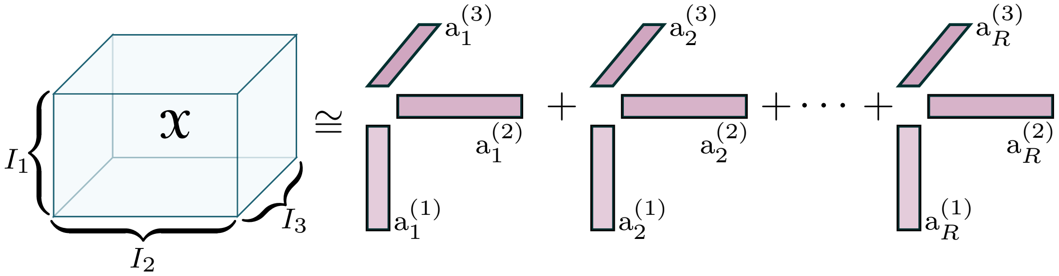

Figure 1: CP decomposition of a 3-way tensor.

$M^{th}-$order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ is defined element-wise as

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

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

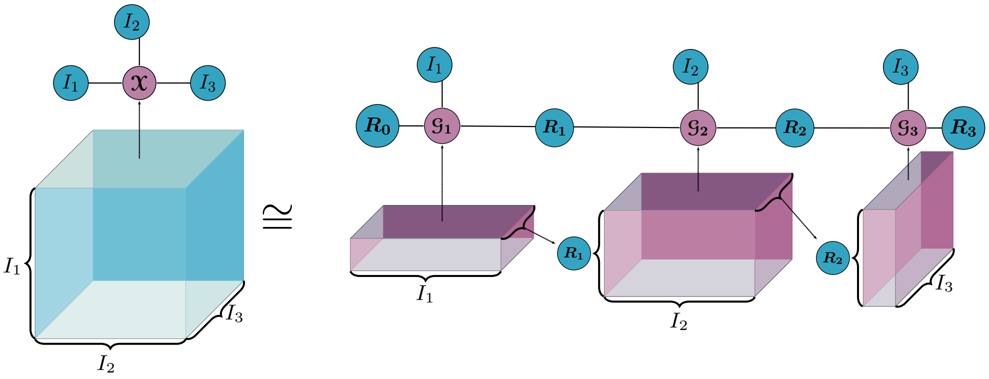

Figure 2: TT decomposition of a 3-way tensor.

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 the SVD. This builds on the fact that the TT decomposition constitutes a recursive matrix factorization, where each TT-rank is the matrix rank of the appropriate unfolding of the tensor, and hence the TT approximation problem is well-posed (Oseledets, 2011).

2.3 Support Vector Machine

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

$$\max_{\alpha_1,\ldots,\alpha_N} \quad \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \langle\phi(\mathbf{x}_i),\phi(\mathbf{x}_j)\rangle$$
$$\text{subject to} \quad 0 \le \alpha_i \le C, \quad \sum_{i=1}^{N}\alpha_i y_i = 0, \tag{3}$$

where a tuning function ϕ defines the nonlinear decision boundary with $\phi\colon \mathbf{x}_i \to \phi\left(\mathbf{x}_i\right)$. In practice, we compute directly $\langle\phi\left(\mathbf{x}_i\right), \phi\left(\mathbf{x}_j\right)\rangle$ using the so-called *Kernel Trick* (Schölkopf et al., 2001).

2.3.1 Feature Map and Kernel Trick

The function $\phi\colon \mathbb{R}^m \to \mathbb{F}$ is called **feature map**, and the *feature space* $\mathbb{F}$ is a Hilbert Space (HS). Every feature map is defined via a kernel such that $k_{i,j} = k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle_{\mathbb{F}}$. Employing the properties of the inner product, we conclude that $[k_{i,j}]$ is a symmetric positive semi-definite matrix. The *kernel trick* lies in defining and computing directly $k\left(\mathbf{x}_i, \mathbf{x}_j\right)$ instead of $\phi(\mathbf{x})$. It is used to get a linear learning algorithm to learn a *nonlinear boundary*, without explicitly knowing the nonlinear function ϕ. The only task needed for the SVM is thus to choose a legitimate kernel function. That is how we work with the input data in the high-dimensional space while doing all the computation in the original low dimensional space. Figure 3 illustrates the linear separation in a higher dimensional space.

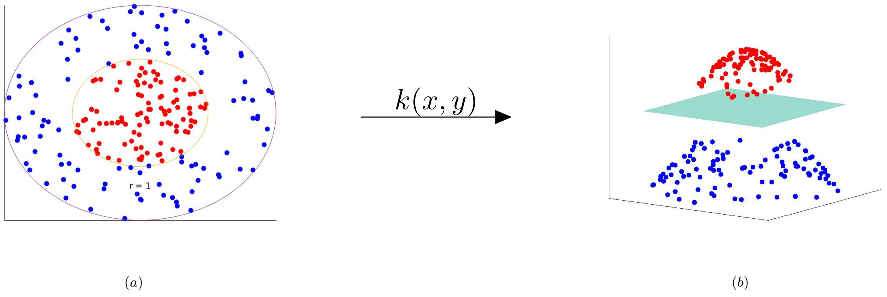

Figure 3: Nonlinear mapping using kernel trick: (a) Nonlinear classification of data in $\mathbb{R}^2$, (b) Linear classification in higher dimension ($\mathbb{R}^3$).

2.4 Kernelized Support Tensor Machine

In our case, we have a data set $\{(\mathcal{X}_i, y_i)\}_{i=1}^{N}$ with input data in the form of a tensor $\mathcal{X}_i \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$. We take the maximum margin approach to get the separation hyperplane. Hence, the objective function for a nonlinear boundary in the tensor space can be written as follows (Cai et al., 2006):

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \tag{4}$$
$$\text{subject to} \quad y_i(\langle\Psi(\mathcal{X}_i), w\rangle + b) \ge 1 - \xi_i \qquad \xi_i \ \ge 0 \ \ \forall i.$$

The classification setup given in (4) is known as Support Tensor Machine (STM) (Tao et al., 2005). The dual formulation of the corresponding primal problem can be given as follows:

$$\begin{aligned} \max_{\alpha_1,\dots,\alpha_N} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j y_i y_j \langle \Psi(\mathcal{X}_i), \Psi(\mathcal{X}_j)\rangle \\ \text{subject to} \quad & 0 \le \alpha_i \le C, \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \ \ \forall i. \end{aligned} \tag{5}$$

The nonlinear feature mapping $\Psi\colon \mathbb{R}^{I_1 \times I_2 \times \dots \times I_M} \to \mathbb{F}$ takes tensorial input data to a higher dimensional space similarly to the vector case. Therefore, by using the kernel trick, explained in Section 2.3.1, STM can be defined as follows:

$$\begin{aligned} \max_{\alpha_1,\dots,\alpha_N} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j y_i y_j K(\mathcal{X}_i, \mathcal{X}_j) \\ \text{subject to} \quad & 0 \le \alpha_i \le C, \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \ \ \forall i. \end{aligned} \tag{6}$$

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

$$G(\mathcal{X}) = \operatorname{sign}\left(\sum_{i=1}^{N} \alpha_i y_i \langle \Psi(\mathcal{X}_i), \Psi(\mathcal{X})\rangle + b_0 \right). \tag{7}$$

By using the *kernel trick* (Schölkopf et al., 2001), this becomes,

$$G(\mathcal{X}) = \operatorname{sign}\left(\sum_{i=1}^{N} \alpha_i y_i K(\mathcal{X}_i, \mathcal{X}) + b_0 \right). \tag{8}$$

The value of b_0 is given as follows,

$$\begin{aligned} b_0 &= \frac{1}{N_0} \sum_{i:\alpha_i \in (0,C)} \left(y_i - \sum_{j=1}^{N} \alpha_j \langle \Psi(\mathcal{X}_j), \Psi(\mathcal{X}_i)\rangle \right), \\ &= \frac{1}{N_0} \sum_{i:\alpha_i \in (0,C)} \left(y_i - \sum_{j=1}^{N} \alpha_j K(\mathcal{X}_j, \mathcal{X}_i) \right), \quad \text{with} \quad N_0 = \sum_{i:\alpha_i \in (0,C)} 1. \end{aligned} \tag{9}$$

We call this setup the *Kernelized STM (KSTM)*. Once we have the real-valued function (kernel) value for each pair of tensors, we can use state-of-the-art LIBSVM (Chang and Lin, 2011), which relies on the *Sequential Minimal Optimization* algorithm to optimize the weights α_i and provides optimal parameter values α_i and b_0. Hence, the preeminent part is the kernel function $K(\mathcal{X}_i, \mathcal{X}_j)$. However, the direct treatment of large tensors can be both numerically expensive and inaccurate due to overfitting. Therefore, we need to choose a kernel that exploits the tensor decomposition. In the next section we propose a particular choice of the kernel for tensor data.

3. The Proposed Algorithm

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

3.1 Uniqueness Enforcing TT- SVD

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

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

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 pair of vectors $\{u_{r_1}, v_{r_1}\}$ can be replaced by $\{-u_{r_1}, -v_{r_1}\}$ without changing the whole expansion. While this is not an issue for data compression, classification using TT factors can be affected significantly by this indeterminacy. For example, tensors that are close to each other should likely produce the same label. In contrast, even a small difference in the original data may lead to a different sign of the singular vectors, and consequently, significantly different values in the kernel matrix $K(\mathcal{X}_j, \mathcal{X}_i)$ and the predicted label (8). As it will be explained in Section 3.5, the kernels are functions of the left singular values u_i only (Algorithm 2).

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

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

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

Lemma 7 *Assume that the singular values $\sigma_1^{(m)}, \ldots, \sigma_{R_m}^{(m)}$ are simple for each $m = 1, \ldots, M-1$. Then Algorithm 1 produces the unique TT decomposition.*

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

Algorithm 1: Uniqueness Enforcing TT-SVD

```
Input: $M$-dimensional tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$, relative error threshold $\epsilon$.
Ensure: Cores $\mathcal{G}^{(\mathbf{1})}, \mathcal{G}^{(\mathbf{2})}, \cdots, \mathcal{G}^{(\boldsymbol{M})}$ of the TT-approximation $\mathcal{X}'$ to $\mathcal{X}$ in the TT-format with TT-rounding ranks $r_m$ equal to the $\delta$-ranks of the unfoldings $\mathcal{X}_{(m)}$ of $\mathcal{X}$, where $\delta = \sqrt{\frac{\epsilon}{M-1}} \|\mathbf{A}\|_F$.
Initialize $\hat{\mathbf{Z}}_1 = \mathcal{X}_{(1)}, R_0 = 1$.
for $m = 1$ to $M - 1$ do
    $\mathbf{Z}_m :=$ reshape $\left(\hat{\mathbf{Z}}_m, [R_{m-1} I_m,\ I_{m+1} \cdots I_M]\right)$
    Compute $\delta$-truncated SVD: $\mathbf{Z}_m = \mathbf{U}_m \mathbf{S}_m \mathbf{V}_m^T + \mathbf{E}_m$, $\|\mathbf{E}_m\|_F \leq \delta$, where
    $\mathbf{U}_m = [u_1^{(m)}, u_2^{(m)}, \ldots, u_{R_m}^{(m)}]$, $\mathbf{S}_m = \text{diag}(\sigma_1^{(m)}, \sigma_2^{(m)}, \ldots, \sigma_{R_m}^{(m)})$, $\mathbf{V}_m = [v_1^{(m)}, v_2^{(m)}, \ldots, v_{R_m}^{(m)}]$
    for $r_m = 1$ to $R_m$ do
        $i^*_{r_m} = \arg\max_{i=1,\ldots,R_{m-1}I_m} |u^{(m)}_{i,r_m}|$ (with ties broken to first element)
        $\bar{u}^{(m)}_{r_m} := u^{(m)}_{r_m} / \text{sign}(u^{(m)}_{i^*_{r_m}, r_m})$,   $\bar{v}^{(m)}_{r_m} := v^{(m)}_{r_m} \cdot \text{sign}(u^{(m)}_{i^*_{r_m}, r_m})$
        $\mathcal{G}^{(m)}_{r_{m-1}, i_m, r_m} = \bar{u}^{(m)}_{r_{m-1} + (I_m - 1) R_{m-1},\ r_m}$,   $\bar{\mathbf{V}}_m = [\bar{v}_1^{(m)}, \bar{v}_2^{(m)}, \ldots, \bar{v}_{R_m}^{(m)}]$
    end for
    $\hat{\mathbf{Z}}_{m+1} := \mathbf{S}_m \bar{\mathbf{V}}_m^T$
end for
$\mathcal{G}^{(M)} = \hat{\mathbf{Z}}_M$
```

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

Remark 8 *Most of the data featuring in machine learning are noisy. Therefore, the singular values of the corresponding matricizations are simple almost surely, and hence the TT decomposition delivered by Algorithm 1 is unique almost surely.*

3.2 TT-CP Expansion

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

Proposition 9 *For a given TT decomposition (2), we can obtain a CP decomposition*

$$\sum_{r_0,\ldots,r_M} \mathcal{G}^{(1)}_{r_0,i_1,r_1} \mathcal{G}^{(2)}_{r_1,i_2,r_2} \cdots \mathcal{G}^{(M)}_{r_{M-1},i_M,r_M} = \sum_{r=1}^{R} \hat{H}^{(1)}_{i_1,r} \hat{H}^{(2)}_{i_2,r} \cdots \hat{H}^{(M)}_{i_M,r}, \tag{10}$$

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

$$\hat{H}^{(m)}_{i_m,r} = \mathcal{G}^{(m)}_{r_{m-1},i_m,r_m}, \quad m = 1, \ldots, M.$$

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

Note that the number of terms R in the CP decomposition (10) can be larger than the minimal CP rank of the exact CP decomposition of the given tensor. However, the nonlinear kernel function is more sensitive to the features of the data rather than the number of CP terms *per se*. In the numerical tests, we observe that the expansion (10) gives actually a better classification accuracy than an attempt to compute an optimal CP approximation using an ALS method.

3.3 Norm Equilibration

In our preliminary experiments, we tried using directly the TT-CP expansion as above with the CP kernel from (He et al., 2017a). However, this did not lead to better classification results. The DuSK kernel (He et al., 2017a) introduces the same width parameter for all CP factors. This requires all CP factors to have identical (or at least close) magnitudes. In contrast, different TT cores have different norms in the plain TT-SVD algorithm (Oseledets, 2011). Here, we rescale the TT-CP expansion to ensure that the columns of the CP factors have equal norms, and hence produce the same kernel features with the same width parameter. We have found this to be a key ingredient for the successful TT-SVM classification.

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

$$n_r = \left\| \hat{H}_r^{(1)} \right\| \cdots \left\| \hat{H}_r^{(M)} \right\|, \tag{11}$$

and distribute this norm equally among the factors,

$$H_r^{(m)} := \frac{\hat{H}_r^{(m)}}{\left\| \hat{H}_r^{(m)} \right\|} \cdot n_r^{1/M}, \qquad m = 1, 2, \cdots, M. \tag{12}$$

3.4 Noise-aware Threshold and Rank Selection

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, the SVD can serve as a de-noising algorithm automatically: the dominant singular vectors are often “smooth” and hence represent a useful signal, 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. Cross-validation is a technique to evaluate the effectiveness of the model, which is done by re-sampling the data into training-testing data sets. Since the precise magnitude of the noise is unknown, we carry out a k-fold cross-validation test ($k = 5$) to find the optimal TT rank.

3.5 Nonlinear Mapping

Equipped with the homogenized TT-CP decompositions of the input tensors, we are ready to define a nonlinear kernel function. We follow closely the rationale behind DuSK proposed in He et al. (2014, 2017a) and express its generalized form for tensors of arbitrary dimension. We assume that the feature map function from the space of tensors to a *tensor product Reproducing Kernel Hilbert*

Space (Signoretto et al., 2013) $\Psi\colon \mathbb{R}^{I_1} \times \cdots \times \mathbb{R}^{I_M} \mapsto \mathbb{F}$ consists of separate feature maps acting on different CP factors,

$$\Psi\colon \sum_{r=1}^{R} H_r^{(1)} \otimes H_r^{(2)} \otimes \cdots \otimes H_r^{(M)} \mapsto \sum_{r=1}^{R} \phi(H_r^{(1)}) \otimes \phi(H_r^{(2)}) \otimes \cdots \otimes \phi(H_r^{(M)}). \tag{13}$$

This allows us to exploit the fact that the data is given in the CP format to aid the classification. However, the feature function $\phi(\mathbf{a})$ is to be defined implicitly through a kernel function. Similarly to the standard SVM, applying the kernel trick to (13) gives us a practically computable kernel. Given CP approximations of two tensors $\mathcal{X} = [x_{i_1,\dots,i_M}]$ and $\mathcal{Y} = [y_{i_1,\dots,i_M}]$,

$$x_{i_1,\dots,i_M} \approx \sum_{r=1}^{R} H_{i_1,r}^{(1)} H_{i_2,r}^{(2)} \cdots H_{i_M,r}^{(M)}, \qquad y_{i_1,\dots,i_M} \approx \sum_{r=1}^{R} P_{i_1,r}^{(1)} P_{i_2,r}^{(2)} \cdots P_{i_M,r}^{(M)},$$

we compute

$$\begin{aligned}
\langle \Psi(\mathcal{X}), \Psi(\mathcal{Y}) \rangle &= K(\mathcal{X}, \mathcal{Y}) \\
&= K\left(\sum_{r=1}^{R} H_r^{(1)} \otimes H_r^{(2)} \otimes \cdots \otimes H_r^{(M)}, \sum_{r=1}^{R} P_r^{(1)} \otimes P_r^{(2)} \otimes \cdots \otimes P_r^{(M)} \right), \\
&= \langle \Psi(\sum_{r=1}^{R} H_r^{(1)} \otimes H_r^{(2)} \otimes \cdots \otimes H_r^{(M)}), \Psi(\sum_{r=1}^{R} P_r^{(1)} \otimes P_r^{(2)} \otimes \cdots \otimes P_r^{(M)}) \rangle, \\
&= \sum_{i,j=1}^{R} \langle \phi(H_i^{(1)}), \phi(P_j^{(1)}) \rangle \langle \phi(H_i^{(2)}), \phi(P_j^{(2)}) \rangle \cdots \langle \phi(H_i^{(M)}), \phi(P_j^{(M)}) \rangle, \\
&= \sum_{i,j=1}^{R} k(H_i^{(1)}, P_j^{(1)}) k(H_i^{(2)}, P_j^{(2)}) \cdots k(H_i^{(M)}, P_j^{(M)}), \qquad (14)
\end{aligned}$$

where

$$k(\mathbf{h}, \mathbf{p}) = \exp\left(-\frac{\|\mathbf{h}-\mathbf{p}\|^2}{2\sigma^2} \right).$$

This kernel approximation is computed for each pair of the tensor input data, represented by its CP factors. The width parameter $\sigma > 0$ needs to be chosen judiciously to ensure accurate learning.

Since the entire calculation starts from the TT decomposition, 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 STM optimizer (6) completes the algorithm. The overall idea is summarized in Algorithm 2.

4. Numerical Tests

- **Experimental Settings**

  All numerical experiments have been done in `MATLAB 2016b`. In the first step, we compute

```
Algorithm 2: TT-CP approximation of the STM Kernel
    Input: data $\{\mathcal{X}_n\}_{n=1}^{N}\mathbb{R}^{I_1\times I_2\times\ldots\times I_M}$, TT-rank $R$.
    Output: Kernel matrix approximation $\big[K(\mathcal{X}_u,\mathcal{X}_v)\big]\in\mathbb{R}^{N\times N}$
    for $n=1$ to $N$ do
        Compute TT approximation $\mathcal{X}_n \cong \langle\!\langle \mathcal{G}^{(1,n)},\mathcal{G}^{(2,n)},\cdots,\mathcal{G}^{(M,n)}\rangle\!\rangle$ using Algorithm 1.
        Compute TT-CP expansion $[\![H^{(1,n)},H^{(2,n)},\cdots,H^{(M,n)}]\!] = \langle\!\langle \mathcal{G}^{(1,n)},\mathcal{G}^{(2,n)},\cdots,\mathcal{G}^{(M,n)}\rangle\!\rangle$ as
        (10) with equilibrated norms as (12).
    end for
    for $u,v=1$ to $N$ do
        $K\left(\mathcal{X}_u,\mathcal{X}_v\right)\approx\sum_{i,j=1}^{R}k(H_i^{(1,u)},H_j^{(1,v)})k(H_i^{(2,u)},H_j^{(2,v)})\cdots k(H_i^{(M,u)},H_j^{(M,v)})$ as (14).
    end for
```

the TT format of an input tensor using the `TT-Toolbox`[1], where we modified the function `@tt_tensor/round.m` to enforce the uniqueness enforcing TT-SVD (Section 3.1). 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 `LIBSVM`[2] library. We have run all experiments on a machine equipped with Ubuntu release 16.04.6 LTS 64-bit, 7.7 GiB of memory, and an Intel Core i5-6600 CPU @ 3.30GHz×4 CPU. The codes are available publicly on `GitHub`[3].

- **Parameter Tuning**
  The entire TT-SVM model depends on three parameters. First, to simplify the selection of TT ranks, we take all TT ranks equal to the same value $R \in \{1, 2, \ldots 10\}$. Another parameter is the width of the Gaussian Kernel σ. Finally, the third parameter is a trade-off constant C for the KSTM optimization technique (6). Both σ and C are chosen from $\{2^{-8}, 2^{-7}, \ldots, 2^{7}, 2^{8}\}$. For tuning R, σ and C to the best classification accuracy, we use the *k-fold cross validation* with $k = 5$. Along with this, we repeat all computations 20 times and compute statistics (average, standard deviation, and numerical quantiles) over these runs. This ensures a confident and reproducible comparison of different techniques.

4.1 Data Collection

1. Resting-state fMRI Datasets

   - **Alzheimer Disease (ADNI):** The ADNI[4] stands for Alzheimer Disease Neuroimaging Initiative. It contains the resting state fMRI images of 33 subjects. The data set was collected from the authors of the paper (He et al., 2017a). 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).

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

- **Attention Deficit Hyperactivity Disorder (ADHD):** The ADHD data set is collected from the ADHD-200 global competition data set[5]. It is a publicly available preprocessed fMRI data set from eight different institutes, collected at one place. The original data set is unbalanced, so 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.

  Note: As mentioned in the MMK paper (He et al., 2017a), the exact indices of the collected data are not mentioned. Hence, our collected dataset might not be exactly the same. Therefore, accuracy percentages are not directly comparable to the MMK paper.

2. Hyperspectral Image (HSI) Datasets: We have taken the `mat` file for both the datasets and their corresponding labels[6]. The following datasets have three dimensional tensor structure of different sizes, where each tensor data point represents a pixel value. Therefore, for our experiment we have taken a patch of size 5×5 for two different pixel values, in order to get a binary classification dataset.

   - **Indian Pines:** The HSI images were collected via the Aviris Sensor[7] over the Indian Pines test site. The size of the dataset is 145×145 pixels over 224 spectral values. Hence, the size of the tensor data is $145 \times 145 \times 224$. The mat file we have collected for our experiment has reduced band size 200. This excludes bands covering the region of water absorption: [104-108], [150-163]. The original dataset contains 16 different labels to identify different corps and living areas. We have taken only 50 datapoints for each of the two labels 11 (Soybean-mintill) and 7 (Grass-pasture-mowed).
   - **Salinas:** This HSI images data was collected by 224 band Aviris Sensor over Salinas valley, California. Similar to Indian Pines, in this case, we have also collected samples for two GroundTruths, namely 9 (Soil-vinyard-develop) and 15 (Vinyard-untrained) each with 50 datapoints. The size of the dataset is 512×217 pixels over 224 spectral values. Hence, the size of the tensor data is $512 \times 217 \times 224$.

4.2 Influence of Individual Algorithmic Steps

In the first test we investigate the impact of each individual transformation of the TT decomposition, outlined in Section 3.1–Section 3.3. Firstly, we can apply a counterpart of the DuSK kernel (14) directly to the initial TT approximation of the data tensors. Given TT decompositions

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

we compute a separable kernel similarly to (14) via

$$k(\mathcal{X},\mathcal{Y}) = \sum_{r_1,t_1=1}^{R_1}\sum_{r_2,t_2=1}^{R_2} k(\mathcal{G}^{(1)}_{r_1},\mathcal{S}^{(1)}_{t_1})k(\mathcal{G}^{(2)}_{r_1,r_2},\mathcal{S}^{(2)}_{t_1,t_2})k(\mathcal{G}^{(3)}_{r_2},\mathcal{S}^{(3)}_{t_2}). \tag{15}$$

[5] `http://neurobureau.projects.nitrc.org/ADHD200/Data.html`

[6] `http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes`

[7] `https://aviris.jpl.nasa.gov/`

A similar approach was also proposed recently in Chen et al. (2022). We compare two versions of this TT-DuSK kernel: “uTT” and “TT”, which correspond to the TT-SVD algorithm with and without uniqueness enforcement (Algorithm 1), respectively.

Next, we expand the TT format without uniqueness enforcement into the CP format as described in Section 3.2 and (10), but without equilibrating the norms, and apply the DuSK kernel (14). The corresponding classifier is called “TTCP”. Note that for a *given* TT decomposition and its exact TT-CP expansion the values of the kernels (15), and (14) coincide. However, different runs of the classification algorithm may produce different signs of the singular vectors in the TT-SVD algorithm, different initial guesses in the SVM, and different splitting of the data into training and test sets during the cross validation.

Finally, we make the norms of the CP factors equilibrated as described in Section 3.3 and (12), followed by the DuSK kernel (14). Depending on using or not using the uniqueness enforcement during the initial TT computation, the corresponding classifiers are called “uTTCPe” and “TTCPe”, respectively.

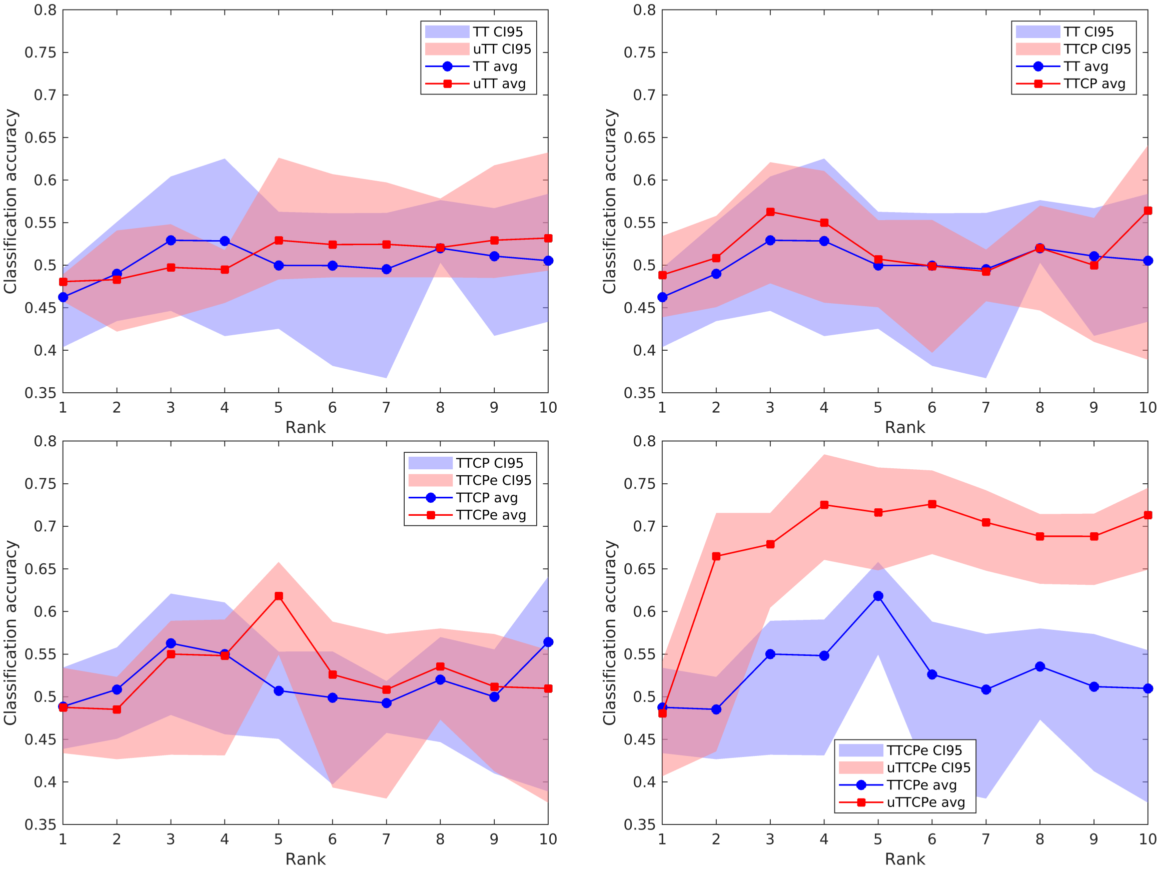

Figure 4: Test classification accuracy of different versions of the TT-MMK algorithm: “TT” vs “uTT” (left top), “TT” vs “TTCP” (right top), “TTCP” vs “TTCPe” (left bottom), and “TTCPe” vs the final algorithm “uTTCPe” (right bottom) for the ADNI dataset. Lines denote averages, and shaded areas denote 95% confidence intervals over 20 runs.

In Figure 4 we compare these versions of the algorithm pairwise to ensure clarity of overlapping confidence intervals. Top left plot of Figure 4 shows that the direct TT counterpart of the DuSK kernel (15) gives a poor test accuracy, although the uniqueness enforcement can improve it slightly for higher ranks.

Next, we compare TT and TTCP DuSK kernels (top right of Figure 4). This is merely a sanity check, since deterministic algorithms would give the same results. Indeed, randomized algorithms give results that are statistically indistinguishable.

In the bottom left plot of Figure 4 we compare TTCP DuSK kernels with and without norm equilibration, but without uniqueness of the underlying TT decomposition. We see that the norm equilibration gives a higher test accuracy at rank 5 which is statistically significant. Nevertheless, the mean accuracy is still below 65%.

Finally, when we plug in both the unique TT format and its norm-equilibrated TTCP expansion (Figure 4, bottom right), we boost the test accuracy above 70%, with the best average accuracy of 73% achieved for rank 4. This shows that all steps of the TT-MMK classifier are important.

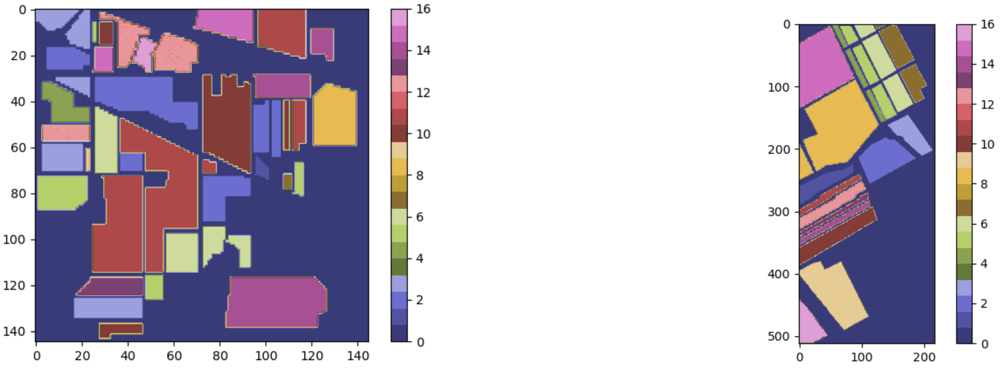

Figure 5: Hyperspectral images with different labels: (a) GroundTruth of Indian Pines dataset, (b) GroundTruth of Salinas dataset.

4.3 Comparison to Other Methods

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

SVM: the standard SVM with Gaussian Kernel. This is the most used optimization method for vector input based on the maximum margin technique. The objective function mentioned in (3) has been optimized using LIBSVM using the *kernel trick* (Schölkopf et al., 2001).

STuM: The Support Tucker Machine (STuM) (Kotsia and Patras, 2011) uses the Tucker decomposition. The weight parameters of the 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. He et al. (2014) solves the STM (6), with kernel approximation using the DuSK format similar to (14).

MMK: This method is an extension of DuSK to the KCP input. The latter is the CP format with factor matrices (1) projected onto a covariance or random matrix (He et al., 2017a). We used the original DuSK and MMK codes provided by the authors of the paper (He et al., 2017a).

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

KSTTM: This method is applied directly on the TT-cores with two different types of kernel computations, namely K-STTM prod and K-STTM sum (Chen et al., 2022). In our experiments, this method ran out of memory for the ADHD dataset during the computation of the kernel matrix.

TT-MMK: This is our proposed method.

Our key observations from the results shown in Table 2 and Figure 7 are as follows.

(In)sensitivity to the TT Rank Selection: Figures 7 and Figure 4 (bottom right) show that the proposed method gives almost the same accuracy for different TT ranks. For some samples, even the TT rank of 2 gives a good classification. Note that this is not the case for MMK, which requires a careful selection of the CP rank.

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

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

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

Running Time: The time taken by the MMK and TT-MMK experiments for the ADNI data with $C, \sigma \in \left[2^{-8}, 2^{-7}, \ldots, 2^{7}, 2^{8}\right]$ are ≈ 17 minutes and ≈ 3.5 hours, respectively, for the entire range of $R \in \{1, 2, \ldots 10\}$. However, if we look closer at Figure 6, the TT-MMK achieves nearly the best accuracy for any rank starting from 2. This means that even though the TT-MMK process takes more time than MMK for the same TT ranks due to the higher CP rank induced by (10), the higher test accuracy is a reasonable reward for the larger CPU time. In particular, if we reduce the range of R to $\{1, \ldots, 5\}$, which is sufficient to discover the best classifiers for both methods, the timings are closer: MMK needs about 1 minute for its best variant (CP rank 5), while the TT rank 4 solution of a better accuracy is computed in about 3 minutes. This slightly higher runtime is acceptable for a better classification accuracy.

Reproducibility: Figure 7 shows that the MMK method has a higher empirical standard deviation (0.05 for the ADNI dataset, 0.02 for the ADHD dataset) compared to the TT-

MMK method with standard deviations of 0.03 and 0.01 for the ADNI and ADHD datasets, respectively. This shows that TT-MMK is more predictable.

Generalization: Top accuracy (see Table 2) in datasets from two different areas (fMRI and HSI) shows that the method is suitable for a wide range of binary tensor classification problems.

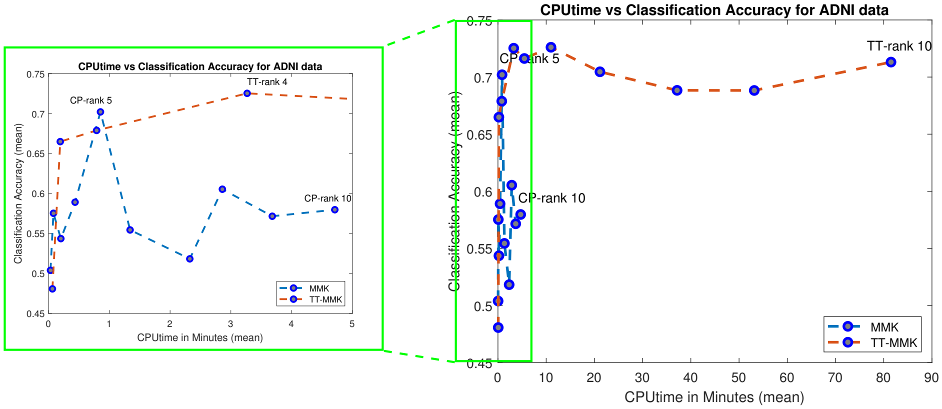

Figure 6: CPU time vs classification accuracy for ADNI data with truncation rank from 1 to 10.

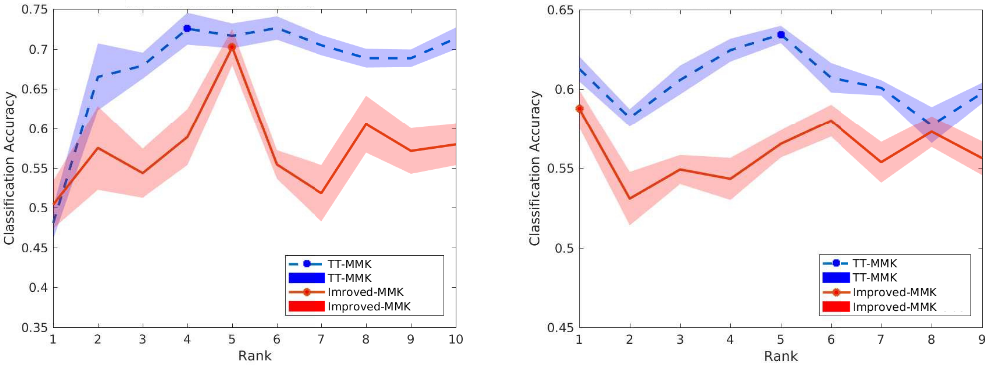

Figure 7: Classification accuracy, average (lines) ± one standard deviation (shaded areas) over 20 runs. Left: ADNI dataset. Right: ADHD dataset.

5. Conclusions

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 and norm distribution. The TT decomposition can be computed more reliably than the CP decomposition used in the original DuSK kernel. Using fMRI and Hyperspectral Image data sets,

Table 2: Average classification accuracy in percentage for different methods and data sets

METHODS	ADNI	ADHD	INDIAN PINES	SALINAS
SVM	49	52	46	47
STUM	51	54	57	74
DUSK	55	57	60	92
MMK	69	58	93	98
IMPROVED MMK	70	58	94	98
K-STTM PROD	60	-	76	**100**
K-STTM SUM	60	-	73	**100**
TT-MMK	**73**	**63**	**99**	99

we have demonstrated that the new TT-MMK method provides higher classification accuracy for an unsophisticated choice of the TT ranks for a wide range of classification problems. We have found out that the each component of the proposed scheme (uniqueness enforced TT, TT-CP expansion and norm equilibration) is crucial for achieving this accuracy.

Further research will consider improving the computational complexity of the current scheme, as well as a joint optimization of the 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.

Acknowledgments

This work is supported by the International Max Planck Research School for Advanced Methods in Process and System Engineering-**IMPRS ProEng**, Magdeburg and is a part of Max Planck research network on Big Data Driven Material Science (**BiGmax**) project. The authors are grateful to Lifang He for providing codes for the purpose of comparison.

References

Deng Cai, Xiaofei He, Ji rong Wen, Jiawei Han, and Wei ying Ma. Support tensor machines for text categorization. *the University of Illinois at Urbana-Champaign Computer Science Department*, 2006.

Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3), May 2011. ISSN 2157-6904. doi: 10.1145/1961189.1961199. URL https://doi.org/10.1145/1961189.1961199.

Cong Chen, Kim Batselier, Ching-Yun Ko, and Ngai Wong. A support tensor train machine. *arXiv preprint arXiv: 1804.06114*, 2018.

Cong Chen, Kim Batselier, Wenjian Yu, and Ngai Wong. Kernelized support tensor train machines. *Pattern Recognition*, 122:108337, 2022. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2021.108337. URL https://www.sciencedirect.com/science/article/pii/S0031320321005173.

Andrzej Cichocki. Tensor decompositions: A new concept in brain data analysis. *arXiv preprint arXiv:1305.0395*, 2013.

Andrzej Cichocki, Namgil Lee, Ivan Oseledets, Anh-Huy Phan, Qibin Zhao, and Danilo P. Mandic. Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *FNT in Machine Learning*, 9(4-5):249–429, 2016.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

Vin de Silva and Lek-Heng Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM J. Matrix Anal. Appl.*, 30(3):1084–1127, 2008.

Gary H. Glover. Overview of functional magnetic resonance imaging. *Neurosurgery Clinics of North America*, 22(2):133 – 139, 2011.

Weiwei Guo, Irene Kotsia, and Ioannis Patras. Tensor learning for regression. *IEEE Transactions on Image Processing*, 21(2):816–827, 2012.

Zhifeng Hao, Lifang He, Bingqian Chen, and Xiaowei Yang. A linear support higher-order tensor machine for classification. *IEEE Transactions on Image Processing*, 22(7):2911–2920, 2013.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, USA, 2001.

Lifang He, Xiangnan Kong, Philip S. Yu, Xiaowei Yang, Ann B. Ragin, and Zhifeng Hao. Dusk: A dual structure-preserving kernel for supervised tensor learning with applications to neuroimages. *Proceedings of the 2014 SIAM International Conference on Data Mining (SDM)*, pages 127–135, 2014. doi: 10.1137/1.9781611973440.15. URL `https://epubs.siam.org/doi/abs/10.1137/1.9781611973440.15`.

Lifang He, Chun-Ta Lu, Hao Ding, Shen Wang, Linlin Shen, Philip S. Yu, and Ann B. Ragin. Multi-way multi-level kernel modeling for neuroimaging classification. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6846–6854, 2017a.

Lifang He, Chun-Ta Lu, Guixiang Ma, Shen Wang, Linlin Shen, Philip S. Yu, and Ann B. Ragin. Kernelized support tensor machines. *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1442–1451, 2017b.

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

Frank L Hitchcock. Multiple invariants and generalized rank of a p-way matrix or tensor. *Journal of Mathematics and Physics*, 7(1-4):39–79, 1928.

Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3): 455–500, 2009.

Irene Kotsia and Ioannis Patras. Support tucker machines. *CVPR 2011*, pages 633–640, June 2011.

Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan V. Oseledets, and Victor S. Lempitsky. Speeding-up convolutional neural networks using fine-tuned CP-decomposition. *CoRR*, abs/1412.6553, 2014.

Xiaolan Liu, Tengjiao Guo, Lifang He, and Xiaowei Yang. A low-rank approximation-based transductive support tensor machine for semisupervised classification. *IEEE Transactions on Image Processing*, 24(6):1825–1838, 2015.

Dimitri Nion and Lieven De Lathauwer. Fast communication: An enhanced line search scheme for complex-valued tensor decompositions. application in ds-cdma. *Signal Processing*, 88(3): 749–755, 2008.

Alexander Novikov, Dmitry Podoprikhin, Anton Osokin, and Dmitry P. Vetrov. Tensorizing neural networks. *Advances NIPS 28*, pages 442–450, 2015.

Alexander Novikov, Mikhail Trofimov, and Ivan V. Oseledets. Exponential machines. *arXiv preprint arXiv:1605.03795*, 2016.

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

Hamed Pirsiavash, Deva Ramanan, and Charless C. Fowlkes. Bilinear classifiers for visual recognition. *Advances in Neural Information Processing Systems 22*, pages 1482–1490, 2009.

John Platt. Sequential minimal optimization : A fast algorithm for training support vector machines. *Microsoft Research Technical Report*, 1998.

Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. *Computational Learning Theory*, pages 416–426, 2001.

Marco Signoretto, Emanuele Olivetti, Lieven De Lathauwer, and Johan A. K. Suykens. A kernel-based framework to tensorial data analysis. *Neural Networks*, 24(8):861 – 874, 2011. Artificial Neural Networks: Selected Papers from ICANN 2010.

Marco Signoretto, Emanuele Olivetti, Lieven De Lathauwer, and Johan A. K. Suykens. Classification of multichannel signals with cumulant-based kernels. *IEEE Transactions on Signal Processing*, 60 (5):2304–2314, 2012.

Marco Signoretto, Lieven De Lathauwer, and Johan A.K. Suykens. Learning tensors in reproducing kernel hilbert spaces with multilinear spectral penalties. *ArXiv*, abs/1310.4977, 2013.

Dacheng Tao, Xuelong Li, Weiming Hu, Stephen Maybank, and Xindong Wu. Supervised tensor learning. *Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 450–457, 2005. doi: 10.1109/ICDM.2005.139. URL `https://doi.org/10.1109/ICDM.2005.139`.

Dacheng Tao, Xuelong Li, Weiming Hu, Stephen Maybank, and Xindong Wu. Supervised tensor learning. *Knowledge and Information Systems*, pages 1–42, 2007.

Valdimir Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York., 1995.

Lior Wolf, Hueihan Jhuang, and Tamir Hazan. Modeling appearances with low-rank SVM. *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

Mingrui Wu and J. Farquhar. A subspace kernel for nonlinear feature extraction. *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 1125–1130, 2007.

Dewei Zeng, Shuqiang Wang, Yanyan Shen, and Changhong Shi. 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.

Qibin Zhao, Guoxu Zhou, Tulay Adali, Liqing Zhang, and Andrzej Cichocki. Kernelization of tensor-based models for multiway data analysis: Processing of multidimensional structured data. *IEEE Signal Processing Magazine*, 30(4):137–148, 2013.

Hua Zhou, Lexin Li, and Hongtu Zhu. Tensor regression with applications in neuroimaging data analysis. *Journal of the American Statistical Association*, 108(502):540–552, 2013.