Kernelized Support Tensor Machines

Lifang He [1] Chun-Ta Lu [1] Guixiang Ma [1] Shen Wang [1] Linlin Shen [2] Philip S. Yu [1 3] Ann B. Ragin [4]

Abstract

In the context of supervised tensor learning, preserving the structural information and exploiting the discriminative nonlinear relationships of tensor data are crucial for improving the performance of learning tasks. Based on tensor factorization theory and kernel methods, we propose a novel Kernelized Support Tensor Machine (KSTM) which integrates kernelized tensor factorization with maximum-margin criterion. Specifically, the kernelized factorization technique is introduced to approximate the tensor data in kernel space such that the complex nonlinear relationships within tensor data can be explored. Further, dual structural preserving kernels are devised to learn the nonlinear boundary between tensor data. As a result of joint optimization, the kernels obtained in KSTM exhibit better generalization power to discriminative analysis. The experimental results on real-world neuroimaging datasets show the superiority of KSTM over the state-of-the-art techniques.

1. Introduction

In many real-world applications, data samples intrinsically come in the form of two-dimensional (matrices) or multi-dimensional arrays (tensors). In medical neuroimaging, for instance, a functional magnetic resonance imaging (fMRI) sample is naturally a third-order tensor consisting of 3D voxels. There has been extensive work in supervised tensor learning (STL) recently. For example, (Tao et al., 2007) proposed a STL framework that extends the standard linear support vector machine (SVM) learning framework to tensor patterns by constructing multilinear models. Under this learning framework, several tensor-based linear models (Zhou et al., 2013; Hao et al., 2013) have been developed. These methods assume, explicitly or implicitly, that data are linearly separable in the input space. However, in practice this assumption is often violated and the linear decision boundaries do not adequately separate the classes.

In order to apply kernel methods for tensor data, several works (Signoretto et al., 2011; 2012; Zhao et al., 2013a) have been presented to convert the input tensors into vectors (or matrices), which are then used to construct kernels. This kind of conversion, though, will destroy the structure information of the tensor data. Moreover, the dimensionality of the resulting vector typically becomes very high, which leads to the curse of dimensionality and small sample size problems (Lu et al., 2008; Yan et al., 2007).

Recently, (Hao et al., 2013; He et al., 2014; Ma et al., 2016) employed CANDECOMP/PARAFAC (CP) factorization (Kolda & Bader, 2009) on the input tensor to foster the use of kernel methods for STL problems. However, as indicated in (Rubinov et al., 2009; Luo et al., 2011), the underlying structure of real data is often nonlinear. Although the CP factorization provides a good approximation to the original tensor data, it only concerned with multilinear formulas. Thus, it is difficult to model complex nonlinear relationships within the tensor data. Most recently, (He et al., 2017) extended CP factorization to the nonlinear case through the exploitation of the representer theorem, and then used kernelized CP (KCP) factorization to facilitate kernel learning. To the best of our knowledge, there is no existing work that tackles factorization and prediction as a joint optimization problem over the kernel methods.

This paper focuses on developing kernelized tensor factorization with kernel maximum-margin constraint, referred as Kernelized Support Tensor Machine (KSTM). KSTM includes two principal ingredients. First, inspired by (Signoretto et al., 2013), we introduce a general formulation of kernelized tensor factorization in the tensor product reproducing kernel Hilbert space, namely kernelized Tucker model, which provides a new perspective on understanding the KCP process. Second, we apply kernel trick to embed the compact representations extracted by KCP into the dual structure-preserving kernels (He et al., 2014) in conjunction with a maximum-margin method to solve the

[1]Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA [2]Institute for Computer Vision, Shenzhen University, Shenzhen, China [3]Institute for Data Science, Tsinghua University, Beijing, China [4]Department of Radiology, Northwestern University, Chicago, IL, USA. Correspondence to: Linlin Shen <llshen@szu.edu.cn>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

STL problems. By integrating KCP and classification as a joint optimization problem, KSTM can benefit from label information during the factorization process such that the extracted representations from KCP are more discriminative. Alternately, the kernels obtained in KSTM have greater discriminating power and have potential to enhance the classification performance.

To demonstrate the effectiveness of the proposed KSTM, we conduct experiments on real-life fMRI neuroimaging data for classification problems. The experimental results show that KSTM has significant improvements over other related state-of-the-art classification methods, including vector based, matrix unfolding based, other tensor based kernel methods, and 3D convolutional neural networks.

The remainder of the paper is organized as follows. In Section 2, we give the notation and preliminaries. In Section 3, we review our concerned problem. In Section 4 we present our model and the learning algorithm. In Section 5, we conduct experimental analysis to justify the proposed method. Finally, we conclude the paper in Section 6.

2. Notation and Preliminaries

In this section we introduce some preliminary knowledge on tensor algebra (Kolda & Bader, 2009) and tensor product reproducing kernel Hilbert space (Signoretto et al., 2013), together with notation. Table 1 lists basic symbols that will be used throughout the paper.

Table 1. List of basic symbols.

Symbol	Definition and description
x	each lowercase letter represents a scale
$\mathbf{x}$	each boldface lowercase letter represents a vector
$\mathbf{X}$	each boldface uppercase letter represents a matrix
$\mathcal{X}$	each calligraphic letter represents a tensor, set or space
$[1:M]$	a set of integers in the range of 1 to M inclusively.
$vec(\cdot)$	denotes column stacking operator
$\langle\cdot,\cdot\rangle$	denotes inner product
$\otimes$	denotes tensor product (outer product)
$*$	denotes Hadamard (element-wise) product
$\odot$	denotes Khatri-Rao product
$\delta(\cdot)$	denotes delta function
$\kappa(\cdot,\cdot)$	represents a kernel function

2.1. Tensor Algebra

A tensor is a multi-dimensional array that generalizes matrix representation, whose dimension is called *mode* or *way*. An M-th order tensor is represented as $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_M}$, and its element is denoted as $x_{i_1,\cdots,i_M}$. The m-mode matricization of tensor $\mathcal{X}$ is denoted by $\mathbf{X}_{(m)} \in \mathbb{R}^{I_m\times J}$, where $J = \Pi_{k\neq m}^{m} I_k$. The inner product of two tensors $\mathcal{X},\mathcal{Y} \in \mathbb{R}^{I_1\times\cdots\times I_M}$ is defined by $\langle\mathcal{X},\mathcal{Y}\rangle = \sum_{i_1=1}^{I_1}\cdots\sum_{i_1=1}^{I_M} x_{i_1,\cdots,i_M} y_{i_1,\cdots,i_M}$. The outer product of M vectors $\mathbf{x}^{(m)} \in \mathbb{R}^{I_m}$ for $m \in [1:M]$ is an M-th order tensor and defined elementwise by $\left(\mathbf{x}^{(1)}\otimes\cdots\otimes\mathbf{x}^{(M)}\right)_{i_1,\cdots,i_M} = x_{i_1}^{(1)}\cdots x_{i_M}^{(M)}$ for all values of the indices.

The two most commonly used factorizations are the Tucker model and CP model. For a generic tensor $\mathcal{X} \in \mathbb{R}^{I_1\times\cdots\times I_M}$, its Tucker factorization is defined as

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

where $\mathbf{U}^{(m)} = [\mathbf{u}_1^{(m)},\cdots,\mathbf{u}_R^{(m)}]$ are factor matrices of size $I_m\times R_m$, $\mathcal{G}\in\mathbb{R}^{R_1\times\cdots\times R_N}$ is called the *core tensor*, and $[\![\cdot]\!]$ is used for shorthand. When all the factor matrices have the same number of components, and the core tensor is super-diagonal, Tucker model simplifies to CP model. In general, CP model is considered to be a multilinear low-rank approximation while Tucker model is regarded as a multilinear subspace approximation (Zhao et al., 2013a).

2.2. Tensor Product Reproducing Kernel Hilbert Space

For any $m \in [1:M]$, let $(\mathcal{H}_m, \langle\cdot,\cdot\rangle_m, \kappa^{(m)})$ be a reproducing kernel Hilbert space (RKHS) of functions on a set $\mathcal{X}_m$ with a reproducing kernel $\kappa^{(m)} : \mathcal{X}_m \times \mathcal{X}_m \to \mathbb{R}$ and the inner product operator $\langle\cdot,\cdot\rangle_m$. The space $\boldsymbol{\mathcal{H}} = \mathcal{H}_1\otimes\cdots\otimes\mathcal{H}_M$ is called a tensor product RKHS of functions on domain $\mathcal{X}$, where $\mathcal{X} := \mathcal{X}_1\times\cdots\times\mathcal{X}_M$. In particular, assume x is the generic tuple $x = (x^{(1)},\cdots,x^{(M)}) \in \mathcal{X}$, let define the tensor product space formed by the linear combinations of the functions $f^{(m)}$ for $m \in [1:M]$ as

$$f^{(1)}\otimes\cdots\otimes f^{(M)} : x \mapsto \prod_{m=1}^{M} f^{(m)}(x^{(m)}),\ \ f^{(m)} \in \mathcal{H}_m.$$

It holds that

$$\begin{aligned}&\sum_{\boldsymbol{j}}\alpha_{\boldsymbol{j}}(f_{j_1}^{(1)}\otimes\cdots\otimes f_{j_M}^{(M)})(x) = \sum_{\boldsymbol{j}}\alpha_{\boldsymbol{j}}\prod_{m=1}^{M} f_{j_m}^{(m)}(x^{(m)})\\ &= \sum_{\boldsymbol{j}}\alpha_{\boldsymbol{j}}\prod_{m=1}^{M}\langle f_{j_m}^{(m)}, k_x^{(m)}\rangle_m.\end{aligned} \tag{2}$$

where $\boldsymbol{j} = (j_1,\cdots,j_M)$ is a multi-index, $\alpha_{\boldsymbol{j}}$ is the combination coefficient, and $k_x^{(m)}$ is the function $k^{(m)}(\cdot,x^{(m)}) : t \mapsto k^{(m)}(t,x^{(m)})$.

3. Problem Formulation and Related Work

Assume we are given a collection of training samples $\mathcal{D} = \{\mathcal{X}_i, y_i\}_{i=1}^{N}$, where $\mathcal{X}_i \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_M}$ is the i-th input sample and $y_i \in \{-1,1\}$ is its corresponding class label.

As we have seen, $\mathcal{X}_i$ is represented in tensor form. To fit a classifier, a commonly used approach is to stack $\mathcal{X}_i$ into a vector. Let $\mathbf{x}_i \triangleq vec(\mathcal{X}_i) \in \mathbb{R}^{I_1 I_2 \cdots I_M}$. The soft margin SVM is defined as

$$\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} + C\sum_{i=1}^{N}[1 - y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b)]_+, \tag{3}$$

where $[1-t]_+ = \max(0, 1-t)^p$ and p is 1 or 2. When $p = 1$, Problem (3) is called $L1$-SVM, and when $p = 2$, $L2$-SVM. $\mathbf{w} \in \mathbb{R}^{I_1 I_2 \cdots I_M}$ is a vector of regression coefficients, $b \in \mathbb{R}$ is an offset term, and C is a pre-specified regularization parameter.

When reshaped into vector $vec(\mathcal{X}_i)$, however, the correlation among the tensor is ignored. It would be more reasonable to exploit the correlation information in developing a classifier, because the correlation is useful and beneficial in improving the classification performance. Intuitively, one can consider the following formulation:

$$\arg\min_{\mathcal{W},b} \; L\big(y, \langle \mathcal{W}, \mathcal{X}\rangle + b\big) + P(\mathcal{W}), \tag{4}$$

where $\mathcal{W} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_M}$ is the tensor of regression coefficients, $L(\cdot)$ is a loss function and $P(\mathcal{W})$ is a penalty function defined on $\mathcal{W}$. For example,

$$\min_{\mathcal{W},b} \frac{1}{2}\langle \mathcal{W}, \mathcal{W}\rangle + C\sum_{i=1}^{N}\big[1 - y_i\big(\langle \mathcal{W}, \mathcal{X}_i\rangle + b\big)\big]_+. \tag{5}$$

However, this formulation is essentially equivalent to Problem (3) when $\mathbf{w} = vec(\mathcal{W})$, because $\langle \mathcal{W}, \mathcal{W}\rangle = vec(\mathcal{W})^{\mathrm{T}} vec(\mathcal{W}) = \mathbf{w}^{\mathrm{T}}\mathbf{w}$ and $\langle \mathcal{W}, \mathcal{X}_i\rangle = vec(\mathcal{W})^{\mathrm{T}} vec(\mathcal{X}_i) = \mathbf{w}^{\mathrm{T}}\mathbf{x}_i$. This implies that the formulation (5) cannot directly address our concern.

To capture the multi-way correlation, an alternative approach is to consider the dependency of the regression tensor $\mathcal{W}$. In particular, one can impose a low-rank constraint on $\mathcal{W}$ to leverage the structure information within $\mathcal{X}_i$. For example, (Tao et al., 2007) and (Cao et al., 2014) assumed that $\mathcal{W} \approx \mathbf{w}^{(1)} \otimes \cdots \otimes \mathbf{w}^{(M)}$, where for $m \in [1:M]$, $\mathbf{w}^{(m)} \in \mathbb{R}^{I_m}$. (Kotsia et al., 2012) and (Lu et al., 2017) assumed that $\mathcal{W} \approx [\![\mathbf{W}^{(1)}, \cdots, \mathbf{W}^{(M)}]\!]$. Several researchers (Hao et al., 2013; Liu et al., 2015), however, have pointed out that this type of models might end up with a suboptimal or even worse solution since it is only an approximation of Problems (3) and (5).

On the other hand, some tensor based kernel methods (Signoretto et al., 2011; Zhao et al., 2013b; He et al., 2014; Guo et al., 2014) have been proposed to solve Problem (5) in the dual space, which takes the form:

$$\max_{\boldsymbol{\beta}\in\mathcal{Q}} \; -\frac{1}{2}\boldsymbol{\beta}^{\mathrm{T}}\mathbf{K}\boldsymbol{\beta} + \mathbf{q}^{\mathrm{T}}\boldsymbol{\beta}, \tag{6}$$

where $\boldsymbol{\beta}$ is the vector of dual variables, $\mathcal{Q}$ is the domain of $\boldsymbol{\beta}$, $\mathbf{q}$ is a vector with $q_i = 1/y_i$, and $\mathbf{K} \in \mathbb{R}^{N\times N}$ is a kernel matrix defined by a kernel function $\kappa(\mathcal{X}_i, \mathcal{X}_j) = \langle \Phi(\mathcal{X}_i), \Phi(\mathcal{X}_j)\rangle$. Notice that kernel function becomes the only domain specific module of Problem (6). In this line, it is essential to optimize kernel design and learn directly from data. In general terms, it can be viewed as the problem of finding a mapping on the input data:

$$\arg\min_{\Phi(\cdot)} \; J(\mathcal{X}, \Phi(\mathcal{X})) + \Omega(\mathcal{X}), \tag{7}$$

where $J(\cdot)$ is a certain criterion between $\mathcal{X}$ and $\Phi(\mathcal{X})$, and $\Omega(\mathcal{X})$ is a specific constraint imposed on the priors of $\mathcal{X}$. It is worthwhile to note that $\Phi(\cdot)$ can be implicit or explicit depending on the learning criterion. In the conventional tensor based kernel methods (Signoretto et al., 2011; Zhao et al., 2013b; He et al., 2014), Problem (7) is learned separately without considering any label information. While it is usually beneficial to utilize label information during kernel learning.

It is desirable to consider both label information and multi-way correlations within tensor data. Pursuing this idea, it is natural to study the Problem (4) and the Problem (7) together. Besides, considering $\mathcal{W}$ can be explicitly expressed by $\mathcal{W} = \sum_{i=1}\beta_i\Phi(\mathcal{X}_i)$, we can make a transformation between $\mathcal{W}$ and $\Phi(\mathcal{X})$, thus expressing $J(\mathcal{X}, \Phi(\mathcal{X}))$ through $\mathcal{W}$. Based on this idea, we present the following formulation:

$$\arg\min_{\mathcal{W},b} \; L\big(y, \langle \mathcal{W}, \mathcal{X}\rangle + b\big) + P(\mathcal{W}) + \Omega(\mathcal{X}). \tag{8}$$

Notice that as $L(\cdot)$ is inherently associated with $\mathcal{W}$ and $\mathcal{X}$, thus all three terms in Problem (8) will be dependent on each other.

4. Methodology

Tensor provides a natural representation for multi-way data, but there is no guarantee that it will be effective for learning. Learning will only be successful if the regularities that underlie data can be captured by the learning model. Hence, how to define $\Omega(\mathcal{X})$ is critical to solve Problem (8). In the following, we first introduce a kernelized tensor factorization to define $\Omega(\mathcal{X})$. Then we propose the kernelized support tensor machine (KSTM) to solve the whole Problem (8), followed by the learning algorithm.

4.1. Kernelized Tensor Factorization

It is well-known that tensor factorization provides a means to capture the multi-way correlation of tensor data. However, it only gives a good approximation – rather than the discriminative capacity. Besides, the standard factorization is only concerned with multilinear formulas without considering the nonlinear relationships within the tensor. To overcome these issues, here we present a general formulation of kernelized tensor factorization, on making use of the notion of tensor product RKHS. In particular, given an M-th order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_M}$, we treat $\mathcal{X}$ as an element of the tensor product RKHS $\boldsymbol{\mathcal{H}}$, and assume that it has a low-rank structure in space $\boldsymbol{\mathcal{H}}$, such that the following fitting criterion holds:

$$\begin{aligned}&\underset{\alpha_{\boldsymbol{j}},\, f_{j_m}^{(m)}}{\arg\min} \sum_{\boldsymbol{i}} \Big(x_{\boldsymbol{i}} - \sum_{\boldsymbol{j}} \alpha_{\boldsymbol{j}} \prod_{m=1}^{M} \langle k_x^{(m)}, f_{j_m}^{(m)} \rangle_m \Big)^2 \\ &= \min_{\mathcal{G},\, \mathbf{F}^{(m)}} \|\mathcal{X} - [\![\mathcal{G}; \mathbf{K}^{(1)}\mathbf{F}^{(1)}, \cdots, \mathbf{K}^{(M)}\mathbf{F}^{(M)}]\!]\|_F^2 \quad (9)\end{aligned}$$

where $\mathcal{G} \in \mathbb{R}^{J_1 \times \cdots \times J_M}$ consists of the elements $\alpha_{\boldsymbol{j}}$, *i.e.*, $\alpha_{j_1, \cdots, j_M}$, $\mathbf{K}^{(m)} \in \mathbb{R}^{I_m \times I_m}$ and $\mathbf{F}^{(m)} \in \mathbb{R}^{I_m \times J_m}$ consist of the elements $k_x^{(m)}$ and $f_{j_m}^{(m)}$ respectively.

Eq. (9) can be viewed as a type of kernelized Tucker factorization model, where kernel matrices $\mathbf{K}^{(m)}$ defined on each mode allow to capture the nonlinear relationships within each mode and the major discriminative features between the modes. Specifically, when $\mathcal{G}$ is super-diagonal and the size of each mode of $\mathcal{G}$ is the same, *i.e.*, $J_1 = \cdots = J_M$, it reduces to the kernelized CP (KCP) factorization model in (He et al., 2017), which can be formulated as

$$\min_{\mathbf{U}^{(1)}, \cdots, \mathbf{U}^{(M)}} \|\mathcal{X} - [\![\mathbf{K}^{(1)}\mathbf{U}^{(1)}, \cdots, \mathbf{K}^{(M)}\mathbf{U}^{(M)}]\!]\|_F^2. \quad (10)$$

Note that $\mathcal{G}$ is absorbed into the matrices. Moreover, it should be noted that although Eq. (10) is a special case of Eq. (9), they have different application scenarios that are similar to CP and Tucker models, see *e.g.*, (Cichocki et al., 2015; Wang et al., 2015; Shao et al., 2015; Cao et al., 2017).

4.2. Kernelized Support Tensor Machine

To solve Problem (8), we pursue a discriminative and nonlinear factorization machine by coupling kernelized tensor factorization with a maximum-margin classifier. For simplicity, we focus on the KCP model. We formulate the primal model of KSTM as follows:

$$\begin{aligned}&\min_{\mathbf{U}_i^{(m)}, \mathbf{K}^{(m)}, \mathcal{W}, b} \underbrace{\gamma \sum_{i=1}^{N} \|\mathcal{X}_i - [\![\mathbf{K}^{(1)}\mathbf{U}_i^{(1)}, \cdots, \mathbf{K}^{(M)}\mathbf{U}_i^{(M)}]\!]\|_F^2}_{\Omega(\mathcal{X})} \\ &+ \underbrace{\langle \mathcal{W}, \mathcal{W} \rangle}_{P(\mathcal{W})} + \underbrace{C \sum_{i=1}^{N} \big[1 - y_i\big(\langle \mathcal{W}, \widehat{\mathcal{X}}_i \rangle + b\big)\big]_+}_{L(y, \langle \mathcal{W}, \widehat{\mathcal{X}} \rangle + b)}, \quad (11)\end{aligned}$$

where γ and C are parameters to control the approximate error and prediction loss respectively, and $\widehat{\mathcal{X}}_i \triangleq [\![\mathbf{U}_i^{(1)}, \cdots, \mathbf{U}_i^{(M)}]\!]$, which have the same size as $\mathcal{X}_i$. Recall that the principle of KCP factorization, our KSTM is able to capture the multi-way and nonlinear correlations within the tensor data. On the other hand, by sharing the kernel matrices $\mathbf{K}^{(m)}$ for different data samples $\mathcal{X}_i$, it makes KSTM possible to characterize tensor data taking into account both common and discriminative features.

The traditional solution for SVM classifier is generally obtained in the dual domain (Vapnik, 2013). However, since the weight tensor $\mathcal{W}$ and the latent factor matrices $\mathbf{U}_i^{(m)}$ are inherently coupled in (11), it is complicated to obtain the dual form of Problem (11). Inspired by the idea of primal optimizations of non-linear SVMs (Chapelle, 2007), the well-known *kernel trick* is introduced here to implicitly capture the non-linear structures. Therefore, we replace $\mathcal{W}$ with a functional form $f(\widehat{\mathcal{X}})$ as follows:

$$f(\widehat{\mathcal{X}}) = \sum_{i=1}^{N} \beta_i \kappa(\widehat{\mathcal{X}}_i, \widehat{\mathcal{X}}), \quad (12)$$

where $\kappa(\cdot)$ is a kernel function. After replacing $\mathcal{W}$ by $f(\widehat{\mathcal{X}})$, Problem (11) is transformed as follows:

$$\begin{aligned}&\min_{\mathbf{U}_i^{(m)}, \mathbf{K}^{(m)}, \boldsymbol{\beta}, b} \gamma \sum_{i=1}^{N} \|\mathcal{X}_i - [\![\mathbf{K}^{(1)}\mathbf{U}_i^{(1)}, \cdots, \mathbf{K}^{(M)}\mathbf{U}_i^{(M)}]\!]\|_F^2 \\ &+ \lambda \sum_{i,j=1}^{N} \beta_i \beta_j \kappa(\widehat{\mathcal{X}}_i, \widehat{\mathcal{X}}_j) \\ &+ \sum_{i=1}^{N} \big[1 - y_i\big(\sum_{j=1}^{N} \beta_j \kappa(\widehat{\mathcal{X}}_i, \widehat{\mathcal{X}}_j) + b\big)\big]_+, \quad (13)\end{aligned}$$

where $\lambda = 1/C$ is the weight between the loss function and the margin, and γ is the relative weight between the generative and the discriminative components.

Writing the kernel matrix $\widehat{\mathbf{K}}$, such that $\widehat{k}_{i,j} = \kappa(\widehat{\mathcal{X}}_i, \widehat{\mathcal{X}}_j)$, and $\widehat{\mathbf{k}}_i$ is the i-th column of $\widehat{\mathbf{K}}$, we can rewrite Problem (13) as follows:

$$\begin{aligned}&\min_{\mathbf{U}_i^{(m)}, \mathbf{K}^{(m)}, \boldsymbol{\beta}, b} \gamma \sum_{i=1}^{N} \|\mathcal{X}_i - [\![\mathbf{K}^{(1)}\mathbf{U}_i^{(1)}, \cdots, \mathbf{K}^{(M)}\mathbf{U}_i^{(M)}]\!]\|_F^2 \\ &+ \lambda \boldsymbol{\beta}^{\mathrm{T}} \widehat{\mathbf{K}} \boldsymbol{\beta} + \sum_{i=1}^{N} \big[1 - y_i\big(\widehat{\mathbf{k}}_i^{\mathrm{T}} \boldsymbol{\beta} + b\big)\big]_+. \quad (14)\end{aligned}$$

This can be regarded as the dual form of Problem (11).

4.3. Learning Algorithm

Now we discuss the solution of Problem (14). As an example, we consider the case of $L2$ loss, *i.e.*, $[1 - t]_+ = \max(0, 1 - t)^2$. The objective function is non-convex, and solving for the global minimum is difficult in general. Therefore we derive an efficient iterative algorithm to reach the local optimum, by alternatively minimizing Problem

(14) for each variable while fixing the other. For the sake of simplicity, we let $\widehat{y}_i \triangleq \widehat{\mathbf{k}}_i^{\mathrm{T}}\boldsymbol{\beta} + b$ in the following.

Update $\mathbf{K}^{(m)}$: Since there is no supervised information involving $\mathbf{K}^{(m)}$, we can utilize the original CP factorization optimization technique to find a solution, by solving the following linear system of equation:

$$\mathbf{K}^{(m)} \sum_{i=1}^{N} \left(\mathbf{U}_i^{(m)} \mathbf{W}_i^{(-m)}\right) = \sum_{i=1}^{N} \left(\mathbf{X}_{i(m)} \mathbf{V}_i^{(-m)}\right), \quad (15)$$

where $\mathbf{V}_i^{(-m)} = \odot_{j\neq m}^{M}(\mathbf{K}^{(j)}\mathbf{U}_i^{(j)})$, $\mathbf{W}_i^{(-m)} = \left(\mathbf{V}_i^{(-m)}\right)^{\mathrm{T}}\mathbf{V}_i^{(-m)}$, and $\mathbf{X}_{i(m)}$ is the m-mode matricization of the data $\mathcal{X}_i$.

Update $\boldsymbol{\beta}$: Similar to (Chapelle, 2007), for a given value of the vector $\boldsymbol{\beta}$, we say that a sample $\widehat{\mathcal{X}}_i$ is a support tensor if $y_i\widehat{y}_i < 1$, that is, if the loss on this sample is nonzero. After reordering the samples such that the first N_v samples are support tensors, the first order gradient with respect to $\boldsymbol{\beta}$ is as follows:

$$\nabla_{\boldsymbol{\beta}} = 2(\lambda\widehat{\mathbf{K}}\boldsymbol{\beta} + \widehat{\mathbf{K}}\mathbf{I}^0(\widehat{\mathbf{K}}\boldsymbol{\beta} - \mathbf{y} + b\mathbf{1})), \quad (16)$$

where $\mathbf{y}$ is the class label vector, $\mathbf{1}$ is a vector of all ones of length N, and $\mathbf{I}^0$ is an $N \times N$ diagonal matrix with the first N_s diagonal entries (number of support tensors) being 1 and 0 others, *i.e.*,

$$\mathbf{I}^0 = \begin{bmatrix} \mathbf{I}_{N_s} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \quad (17)$$

Setting Eq. (16) equal to zero, we can find an optimal solution $\boldsymbol{\beta}$.

Update $\mathbf{U}_i^{(m)}$: As the kernel function is inherently coupled to $\mathbf{U}_i^{(m)}$, selecting an appropriate kernel function is a crucial point for optimization design and performance enhancement. Following (He et al., 2014), we consider mapping the tensor data into tensor Hilbert space and then performing the CP factorization in the Hilbert space. The mapping function is defined as:

$$\phi : \sum_{r=1}^{R}\prod_{m=1}^{M} \otimes\mathbf{u}_r^{(m)} \rightarrow \sum_{r=1}^{R}\prod_{m=1}^{M} \otimes\phi\left(\mathbf{u}_r^{(m)}\right). \quad (18)$$

In this respect, it corresponds to mapping the latent representations of tensor data into high-dimensional tensor Hilbert space that retains the multi-way structure. Then the kernel is just the standard inner product of tensors on that space. Formally, we can derive a dual structural preserving kernel function as follows:

$$\begin{aligned}\kappa\left(\widehat{\mathcal{X}}_i, \widehat{\mathcal{X}}_j\right) &= \kappa\left(\sum_{p=1}^{R}\prod_{m=1}^{M}\otimes\mathbf{u}_{ip}^{(m)}, \sum_{q=1}^{R}\prod_{m=1}^{M}\otimes\mathbf{u}_{jq}^{(m)}\right) \\ &= \sum_{p=1}^{R}\sum_{q=1}^{R}\prod_{m=1}^{M}\kappa\left(\mathbf{u}_{ip}^{(m)}, \mathbf{u}_{jq}^{(m)}\right).\end{aligned}$$

By virtue of its derivation, we see that such a kernel function can take the multi-way structure within tensor flexibility into consideration. Different kernel functions specify different hypothesis spaces or even different knowledge embeddings of the data and thus can be viewed as capturing different notions of correlations. Here we consider both linear and nonlinear cases as examples. By using the innear product, the linear kernel can be directly derived as:

$$\begin{aligned}\kappa\left(\widehat{\mathcal{X}}_i, \widehat{\mathcal{X}}_j\right) &= \sum_{p=1}^{R}\sum_{q=1}^{R}\prod_{m=1}^{M}\mathbf{u}_{ip}^{(m)\mathrm{T}}\mathbf{u}_{jq}^{(m)} \\ &= \mathbf{1}^{\mathrm{T}}\Big(\prod_{m=1}^{M} *\big(\mathbf{U}_i^{(m)\mathrm{T}}\mathbf{U}_j^{(m)}\big)\Big)\mathbf{1}. \quad (19)\end{aligned}$$

For the case of Gaussian RBF kernel, it can be formulated in the similar form

$$\begin{aligned}\kappa\left(\widehat{\mathcal{X}}_i, \widehat{\mathcal{X}}_j\right) &= \sum_{p=1}^{R}\sum_{q=1}^{R}\exp\left(-\sigma\sum_{m=1}^{M}\|\mathbf{u}_{ip}^{(m)} - \mathbf{u}_{jq}^{(m)}\|^2\right) \\ &= \mathbf{1}^{\mathrm{T}}\mathbf{H}_{ij}\mathbf{1}. \quad (20)\end{aligned}$$

where σ is used to choose an appropriate kernel width, and $\mathbf{H}_{ij} \in \mathbb{R}^{R\times R}$ is a matrix whose element $h_{ij}^{pq} = \exp(-\sigma\sum_{m=1}^{M}\|\mathbf{u}_{ip}^{(m)} - \mathbf{u}_{jq}^{(m)}\|^2)$.

In general cases, the first order gradient with respect to $\mathbf{U}_i^{(m)}$ can be written as

$$\nabla_{\mathbf{U}_i^{(m)}} = \frac{\partial\Omega(\cdot)}{\partial\mathbf{U}_i^{(m)}} + \frac{\partial P(\cdot)}{\partial\mathbf{U}_i^{(m)}} + \frac{\partial L(\cdot)}{\partial\mathbf{U}_i^{(m)}} \quad (21)$$

with

$$\begin{cases} \dfrac{\partial\Omega(\cdot)}{\partial\mathbf{U}_i^{(m)}} = 2\gamma(\mathbf{K}^{(m)\mathrm{T}}\mathbf{K}^{(m)}\mathbf{U}_i^{(m)}\mathbf{W}_i^{(-m)} - \mathbf{K}^{(m)}\mathbf{X}_{i(m)}\mathbf{V}_i^{(-m)}) \\ \dfrac{\partial P(\cdot)}{\partial\mathbf{U}_i^{(m)}} = 2\lambda\beta_i\displaystyle\sum_{j=1}^{N}\beta_j\frac{\partial\kappa\left(\widehat{\mathcal{X}}_i, \widehat{\mathcal{X}}_j\right)}{\partial\mathbf{U}_i^{(m)}} \\ \dfrac{\partial L(\cdot)}{\partial\mathbf{U}_i^{(m)}} = 2\delta(i)\displaystyle\sum_{j=1}^{N_s}(\widehat{y}_j - y_j)\beta_j\frac{\partial\kappa\left(\widehat{\mathcal{X}}_i, \widehat{\mathcal{X}}_j\right)}{\partial\mathbf{U}_i^{(m)}} \\ \qquad + 2\beta_i\displaystyle\sum_{j=1}^{N_s}(\widehat{y}_j - y_j)\frac{\partial\kappa\left(\widehat{\mathcal{X}}_i, \widehat{\mathcal{X}}_j\right)}{\partial\mathbf{U}_i^{(m)}} \end{cases} \quad (22)$$

where $\delta(\cdot)$ is a delta function indicating that sample is a support tensor.

The partial gradient of $\mathbf{U}_i^{(m)}$ with respect to $\kappa(\widehat{\mathcal{X}}_i, \widehat{\mathcal{X}}_j)$ for the linear kernel is given by

$$\frac{\partial\kappa\left(\widehat{\mathcal{X}}_i, \widehat{\mathcal{X}}_j\right)}{\partial\mathbf{U}_i^{(m)}} = \mathbf{U}_j^{(m)}\Big(\prod_{k\neq m}^{M} *\big(\mathbf{U}_i^{(k)\mathrm{T}}\mathbf{U}_j^{(k)}\big)\Big). \quad (23)$$

Algorithm 1 Learning KSTMs

```
Input: Training data $\mathcal{D}$, rank of factorization $R$, regularization
    parameters $\gamma$ and $\lambda$
Output: Model parameters $\{\mathbf{U}_i^{(m)}\}$, $\{\mathbf{K}^{(m)}\}$, $\boldsymbol{\beta}$, $b$
 1: Initialize $\{\mathbf{U}_i^{(m)}\}$, $\{\mathbf{K}^{(m)}\}$, $\boldsymbol{\beta}$, $b$
 2: repeat
 3:   for $m := 1$ to $M$ do
 4:     Fixing $\{\mathbf{U}_i^{(m)}\}$, update $\mathbf{K}^{(m)}$ by Eq. (15)
 5:   end for
 6:   Compute kernel matrix $\widehat{\mathbf{K}}$ by Eq. (19) or Eq. (20)
 7:   Fixing $\{\mathbf{U}_i^{(m)}\}$, $\{\mathbf{K}^{(m)}\}$ and $b$, update $\boldsymbol{\beta}$ by Eq. (16)
 8:   for $i := 1$ to $N$ do
 9:     for $m := 1$ to $M$ do
10:       Fixing $\{\mathbf{K}^{(m)}\}$, $\boldsymbol{\beta}$ and $b$, update $\mathbf{U}_i^{(m)}$ by Eq. (21)
11:     end for
12:   end for
13:   Fixing $\{\mathbf{U}_i^{(m)}\}$, $\{\mathbf{K}^{(m)}\}$ and $\boldsymbol{\beta}$, update $b$ by Eq. (25)
14: until convergence
```

For Gaussian RBF kernel, it is given by

$$\frac{\partial\kappa\left(\widehat{\mathcal{X}}_i, \widehat{\mathcal{X}}_j\right)}{\partial \mathbf{U}_i^{(m)}} = 2\sigma\left(\mathbf{U}_j^{(m)}\mathbf{H}_{ij}^{\mathrm{T}} - \mathbf{U}_i^{(m)} diag(\mathbf{H}_{ij}\mathbf{1})\right). \tag{24}$$

Update b : The first order gradient with respect to b is:

$$\nabla_b = 2\sum_{i=1}^{N_s}(\widehat{y}_i - y_i). \tag{25}$$

The overall optimization process is given in Algorithm 1.

Convergence Analysis. Although we have to solve Problem (14) in an iterative process due to non-convexity, each of subproblem is convex with respect to one variable. The objective monotonically decreases in each iteration and it has a lower bound (proof can be derived immediately based on the result in (Tao et al., 2007)). Therefore, it guarantees that we can find the optimal solution of each iteration and finally, Algorithm 1 can converge to a local minimum of the objective function in (14).

Computational Analysis. For brevity, we denote $S_1 = \sum_{m=1}^{M} I_m$, $S_2 = \sum_{m=1}^{M}(I_m)^2$, $S_3 = \sum_{m=1}^{M}(I_m)^3$, and $\pi = \Pi_{m=1}^{M} I_m$. Assuming $I_m > R$. In Algorithm 1, solving $\mathbf{K}^{(m)}$ in lines 3-5 requires $\mathcal{O}(N(MRS_2 + MR\pi + R^2S_1 + \pi S_1) + RS_3)$. In line 6, computing $\widehat{\mathbf{K}}$ requires $\mathcal{O}(N^2R^2S_1)$. Line 7 solves β with $\mathcal{O}(N^3)$. Lines 8-12 solve $\mathbf{U}_i^{(m)}$ taking $\mathcal{O}(N^2N_sM + N^2R^2MS_1 + NMR\pi + NS_3 + NR^3S_1)$. Line 13 solves b with $\mathcal{O}(NN_s)$.

Classification Rules. By solving Problem (14) on the training data, we can obtain the shared kernel matrices $\{\mathbf{K}^{(1)}, \cdots, \mathbf{K}^{(M)}\}$. Given a test sample $\mathcal{X}$, according to the relation between CP and KCP, we have $\mathcal{X} = [\![\mathbf{U}^{(1)}, \cdots, \mathbf{U}^{(M)}]\!] \approx [\![\mathbf{K}^{(1)}\mathbf{V}^{(1)}, \cdots, \mathbf{K}^{(M)}\mathbf{V}^{(M)}]\!]$. Therefore, we first compute CP factorization on the test sample $\mathcal{X}$, and then use $\widehat{\mathcal{X}} = [\![\mathbf{V}^{(1)}, \cdots, \mathbf{V}^{(M)}]\!]$ as its KCP factorization, where $\mathbf{V}^{(m)} = \mathbf{K}^{(m)^{-1}}\mathbf{U}^{(m)}$ for $m \in [1 : M]$. Upon solution, we can classify the test sample $\mathcal{X}$ by using the test kernel matrix.

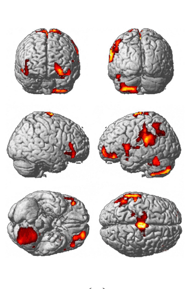
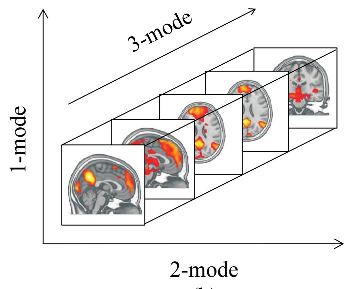

Figure 1. (a) Visualization of an fMRI image from six angles, (b) An illustration of third-order tensor of an fMRI image.

5. Experiments and Results

In order to empirically evaluate the effectiveness of the proposed KSTM, we conduct extensive experiments on real-life neuroimaging (fMRI) data for disease prediction and compare with several state-of-the-art methods. In the following, we introduce the datasets and baselines used and describe how we performed the experiments. Then we present the experimental results as well as the analysis.

5.1. Data Collection and Preprocessing

In the experiments, we consider three resting-state fMRI datasets as follows:

Alzheimer's Disease (ADNI): This dataset is collected from the Alzheimer's Disease Neuroimaging Initiative[1]. It contains the resting-state fMRI images of 33 subjects, including patients with Mild Cognitive Impairment (MCI) or Alzheimer's Disease (AD), and normal controls. We applied SPM8 [2] and REST[3] to preprocess the data. After this, we averaged each individual image over time domain, resulting in 33 samples of size $61 \times 73 \times 61$. We treat AD+MCI as the negative class, and the normal controls as positive class. Finally, we scaled each individual to $[0, 1]$, as the normalization is very important for group analyses among different subjects. A detailed description of preprocessing is available in (He et al., 2014).

Human Immunodeficiency Virus Infection (HIV): This dataset is collected from Chicago Early HIV Infection Study in Northwestern University (Wang et al., 2011),

[1]http://adni.loni.usc.edu/
[2]http://www.l.ion.ucl.ac.uk/spm/software/spm8/
[3]http://resting-fmri.sourceforge.net

Table 2. Summary of compared methods. C is trade-off parameter, σ is kernel width parameter, R is the rank of tensor factorization.

Methods	SVM / SVM+PCA	K_{3rd}	sKL	FK	STuM	DuSK	MMK	3D CNN	KSTM
Type of Input Data	Vectors	Vectors	Matrices	Matrices	Tensor	Tensor	Tensor	Tensor	Tensor
Correlation Exploited	One-way	One-way	One-way	One-way	Multi-way	Multi-way	Multi-way	Multi-way	Multi-way
Kernel Explored	Supervised	Unsupervised	Unsupervised	Unsupervised	——	Unsupervised	Unsupervised	Supervised	Supervised
Nonlinear Factorization	——	——	Unexplored	Unexplored	Unexplored	Unexplored	Explored	——	Explored
Parameters	C, σ	C, σ	C, σ	C, σ	C, σ, R	C, σ, R	C, σ, R	Many*	γ, C, σ, R

Table 3. Classification accuracy comparison (mean $\pm$ standard deviation)

Methods	ADNI	HIV	ADHD
SVM	0.49 ± 0.02	0.70 ± 0.01	0.58 ± 0.00
SVM+PCA	0.50 ± 0.02	0.73 ± 0.03	0.63 ± 0.01
K_{3rd}	0.55 ± 0.01	0.75 ± 0.02	0.55 ± 0.00
sKL	0.51 ± 0.03	0.65 ± 0.02	0.50 ± 0.04
FK	0.51 ± 0.02	0.70 ± 0.01	0.50 ± 0.00
STuM	0.52 ± 0.01	0.66 ± 0.01	0.54 ± 0.03
DuSK	0.75 ± 0.02	0.74 ± 0.00	0.65 ± 0.01
MMK_{best}	0.81 ± 0.01	0.79 ± 0.01	0.70 ± 0.01
MMK_{cov}	0.69 ± 0.01	0.72 ± 0.02	0.66 ± 0.02
3D CNN	0.52 ± 0.03	0.75 ± 0.02	0.68 ± 0.02
KSTM	$\mathbf{0.84 \pm 0.03}$	$\mathbf{0.82 \pm 0.02}$	$\mathbf{0.74 \pm 0.02}$

which contains 83 fMRI brain images of patients with early HIV infection (negative) and normal controls (positive). We used the same preprocessing steps as in ADNI dataset, resulting in 83 samples of size $61 \times 73 \times 61$.

Attention Deficit Hyperactivity Disorder (ADHD): This dataset is collected from ADHD-200 global competition dataset[4], which originally contains 776 subjects, either ADHD patients (negative) or normal controls (positive). Since the dataset is unbalanced, we randomly sampled 100 ADHD patients and 100 normal controls for this study. Finally, we averaged each individual over time domain, resulting in 200 samples of size $58 \times 49 \times 47$.

5.2. Baselines and Metrics

To establish a comparative study, we use the following nine state-of-the-art methods as baselines, each representing a different strategy.

- **SVM**: It is SVM with RBF kernel, which is the most widely used vector method for classification. In the following methods, we use it as the classifier, if not stated explicitly.
- **SVM-PCA**: It is a vector-based subspace learning algorithm, which first uses PCA to reduce the input dimension and then feeds into SVM model. This method is commonly used to deal with high-dimensional classification, in particular fMRI classification (Song et al., 2011; Xie et al., 2009).
- $\mathbf{K}_{3rd}$: It is a vector based tensor kernel method, which exploits the input tensor along each mode to capture structural information and has been used to analyze fMRI data together with RBF kernel (Park, 2011).
- **sKL**: It is a matrix unfolding based tensor kernel method that defined based on the symmetric Kullback-Leibler divergence, and has been used to reconstruct 3D movement (Zhao et al., 2013b).
- **FK**: It is also a matrix unfolding based tensor kernel method, but defined based on multilinear SVD. The constituent kernels are from the class of RBF kernels (Signoretto et al., 2011).
- **STuM**: It is a support tucker machine approach, where the weight tensor parameter is decomposed using the Tucker factorization (Kotsia & Patras, 2011).
- **DuSK**: It is a tensor kernel method based upon CP factorization, which has been used to analyze fMRI data together with RBF kernel (He et al., 2014).
- **MMK**: It is the most recent tensor kernel method based upon KCP factorization (He et al., 2017), which incorporates the KCP into the DuSK. Since the shared kernel matrices involved in KCP are hard to estimate without a prior knowledge, we consider two schemes: first, we randomly generate them and select the best result of 50 repeated times (denoted as MMK_{best}). Second, we perform CP factorization for each data sample and use the covariance matrix of each mode as input (denoted as MMK_{cov}).
- **3D-CNN**: It is a 3D convolutional neural network extended from 2D version (Gupta et al., 2013), which uses the convolution kernel. The convolution kernel is the cubic filters learned from data, which has a small receptive field, but extends through the full depth of the input volume.

Table 2 summarizes the compared methods. We consider eleven methods in total and evaluate their classification performance. For the evaluation where SVM is needed, we apply LibSVM (Chang & Lin, 2011), a widely used implementation of SVM, with RBF kernel as the classifier. We perform 5-fold cross-validation and use classification accuracy as the evaluation measure. This process was repeated 50 times for all methods and the average classification accuracy of each method is reported as the result. The optimal parameters for all methods are determined by grid search. The optimal trade-off parameter is selected

[4]http://neurobureau.projects.nitrc.org/ADHD200/

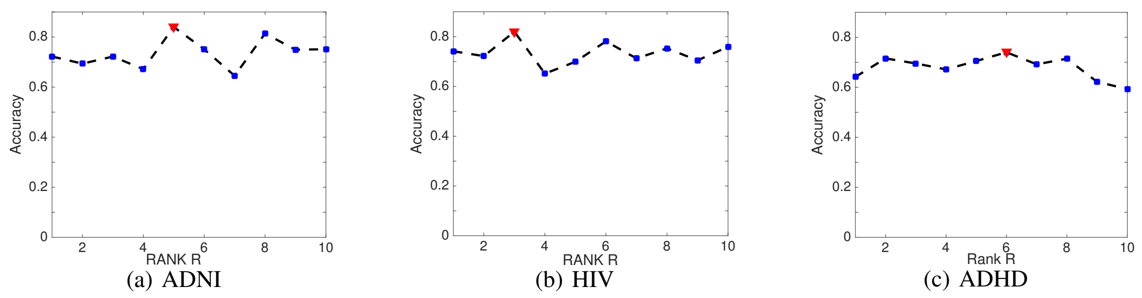

Figure 2. Test accuracy vs. R on (a) ADNI, (b) HIV, and (c) ADHD, where the red triangles indicate the peak positions.

from $C \in \{2^{-8}, 2^{-7}, \cdots, 2^{8}\}$, the kernel width parameter is selected from $\sigma \in \{2^{-8}, 2^{-7}, \cdots, 2^{8}\}$, the optimal rank R is selected from $\{1, 2, \cdots, 10\}$, and the regularized factorization parameter is selected from $\gamma \in \{2^{-8}, 2^{-7}, \cdots, 2^{8}\}$. Note that in the proposed method $\lambda = 1/C$. The optimal parameters for 3D-CNN, *i.e.*, receptive field (R), zero-padding (P), the input volume dimensions (Width $\times$ Height $\times$ Depth, or $W \times H \times D$) and stride length (S) are tuned based on (Gupta et al., 2013).

5.3. Classification Performance

Experimental results in Table 3 shows classification performance of compared methods, where the best result is highlighted in bold type. We can see that the proposed KSTM outperforms all the other methods on three datasets. This is mainly because KSTM can learn the nonlinear relationships embedded within the tensor together with considering prior knowledge across different data samples, while the other methods fail to fully explore the nonlinear relationships or prior knowledge in the tensor object.

Specifically, the kernel methods which unfold the input data into vectors or matrices tend to have a lower performance compared to those who preserve the tensor structure. This indicates that unfolding tensor into vectors or matrices would lose the multi-way structural information within tensor, leading to the degraded performance. Besides, KSTM always performs better than DuSK, which empirically shows the effectiveness of feature extraction in tensor data rather than approximation. Compared to MMK method, we can see that a further improvement can be achieved by benefiting from the use of label information during the factorization procedure. These results demonstrate the effectiveness and considerable advantages of the proposed KSTM method for fMRI classification.

5.4. Parameter Sensitivity

Although the optimal values of the parameters in our proposed KSTM are found by grid search, it is still important to see the sensitivity of KSTM to the rank of factorization R. To this end, we vary $R \in \{1, 2, \cdots, 10\}$, while the other parameters are still selected by grid search. Figure 2 shows the variation of test accuracy over different R on three datasets. We can observe that the rank parameter R has a significant effect on the test accuracy and the optimal value of R depends on the data, while in general the optimal value of R lies in the range $2 \leq R \leq 6$, which may provide a good guidance for selection of the R in advance.

In summary, the classification performance of KSTM relies on parameter R, which is difficult to specify the optimal value in advance. However, in most cases the optimal value of R lies in a small range of values and it is not time-consuming to find it using the grid search strategy in practical applications.

6. Conclusion

In this paper, we have introduced a Kernelized Support Tensor Machine (KSTM), with an application to neuroimaging classification. Different from conventional kernel methods, KSTM is based on the integration of kernelized tensor factorization with kernel maximum-margin classifier. Typically this is done by defining a joint optimization problem, so that the kernels obtained in KSTM have a greater discriminating power. The kernelized tensor factorization is introduced to capture the complex nonlinear relationships within tensor data, by means of the notion of tensor product RKHS, which supplies a new perspective on tensor factorization methods. Empirical studies on three different neurological disorder prediction tasks demonstrated the superiority of KSTM over existing state-of-the-art tensor classification methods.

Acknowledgements

This work is supported in part by NSFC through grants 61503253, 61672357 and 61672313, NSF through grants IIS-1526499 and CNS-1626432, NIH through grant R01-MH080636, and the Science Foundation of Shenzhen through grant JCYJ20160422144110140.

References

Cao, Bokai, He, Lifang, Kong, Xiangnan, Yu, Philip S., Hao, Zhifeng, and Ragin, Ann B. Tensor-based multi-view feature selection with applications to brain diseases. In *ICDM*, pp. 40–49. IEEE, 2014.

Cao, Bokai, He, Lifang, Wei, Xiaokai, Xing, Mengqi, Yu, Philip S., Klumpp, Heide, and Leow, Alex D. t-bne: Tensor-based brain network embedding. In *SDM*, 2017.

Chang, Chih-Chung and Lin, Chih-Jen. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011.

Chapelle, Olivier. Training a support vector machine in the primal. *Neural computation*, 19(5):1155–1178, 2007.

Cichocki, Andrzej, Mandic, Danilo, De Lathauwer, Lieven, Zhou, Guoxu, Zhao, Qibin, Caiafa, Cesar, and Phan, Huy Anh. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE Signal Processing Magazine*, 32(2):145–163, 2015.

Guo, Tengjiao, Han, Le, He, Lifang, and Yang, Xiaowei. A ga-based feature selection and parameter optimization for linear support higher-order tensor machine. *Neurocomputing*, 144:408–416, 2014.

Gupta, Ashish, Ayhan, Murat, and Maida, Anthony. Natural image bases to represent neuroimaging data. In *ICML*, pp. 987–994, 2013.

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

He, Lifang, Kong, Xiangnan, Yu, Philip S, Yang, Xiaowei, Ragin, Ann B, and Hao, Zhifeng. Dusk: A dual structure-preserving kernel for supervised tensor learning with applications to neuroimages. In *SDM*, pp. 127–135, 2014.

He, Lifang, Lu, Chun-Ta, Ding, Hao, Wang, Shen, Shen, Linlin, Yu, Philip S., and Ragin, Ann B. Multi-way multi-level kernel modeling for neuroimaging classification. In *CVPR*, 2017.

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

Kotsia, Irene and Patras, Ioannis. Support tucker machines. In *CVPR*, pp. 633–640. IEEE, 2011.

Kotsia, Irene, Guo, Weiwei, and Patras, Ioannis. Higher rank support tensor machines for visual recognition. *Pattern Recognition*, 45(12):4192–4203, 2012.

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

Lu, Chun-Ta, He, Lifang, Shao, Weixiang, Cao, Bokai, and Yu, Philip S. Multilinear factorization machines for multi-task multi-view learning. In *WSDM*, pp. 701–709. ACM, 2017.

Lu, Haiping, Plataniotis, Konstantinos N, and Venetsanopoulos, Anastasios N. Mpca: Multilinear principal component analysis of tensor objects. *IEEE Transactions on Neural Networks*, 19(1):18–39, 2008.

Luo, Dijun, Nie, Feiping, Huang, Heng, and Ding, Chris H. Cauchy graph embedding. In *ICML*, pp. 553–560, 2011.

Ma, Guixiang, He, Lifang, Lu, Chun-Ta, Yu, Philip S., Shen, Linlin, and Ragin, Ann B. Spatio-temporal tensor analysis for whole-brain fmri classification. In *SDM*, pp. 819–827. SIAM, 2016.

Park, Sung Won. Multifactor analysis for fmri brain image classification by subject and motor task. Electrical and computer engineering technical report, Carnegie Mellon University, 2011.

Rubinov, Mikail, Knock, Stuart A, Stam, Cornelis J, Micheloyannis, Sifis, Harris, Anthony WF, Williams, Leanne M, and Breakspear, Michael. Small-world properties of nonlinear brain activity in schizophrenia. *Human brain mapping*, 30(2):403–416, 2009.

Shao, Weixiang, He, Lifang, and Yu, Philip S. Clustering on multi-source incomplete data via tensor modeling and factorization. In *PAKDD*, pp. 485–497. Springer, 2015.

Signoretto, Marco, De Lathauwer, Lieven, and Suykens, Johan AK. A kernel-based framework to tensorial data analysis. *Neural networks*, 24(8):861–874, 2011.

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

Signoretto, Marco, De Lathauwer, Lieven, and Suykens, Johan AK. Learning tensors in reproducing kernel hilbert spaces with multilinear spectral penalties. *arXiv preprint arXiv:1310.4977*, 2013.

Song, Sutao, Zhan, Zhichao, Long, Zhiying, Zhang, Jiacai, and Yao, Li. Comparative study of svm methods combined with voxel selection for object category classification on fmri data. *PloS one*, 6(2):e17191, 2011.

Tao, Dacheng, Li, Xuelong, Wu, Xindong, Hu, Weiming, and Maybank, Stephen J. Supervised tensor learning. *Knowledge and Information Systems*, 13(1):1–42, 2007.

Vapnik, Vladimir. *The nature of statistical learning theory*. Springer science & business media, 2013.

Wang, Senzhang, He, Lifang, Stenneth, Leon, Yu, Philip S., and Li, Zhoujun. Citywide traffic congestion estimation with social media. In *GIS*, pp. 34. ACM, 2015.

Wang, Xue, Foryt, Paul, Ochs, Renee, Chung, Jae-Hoon, Wu, Ying, Parrish, Todd, and Ragin, Ann B. Abnormalities in resting-state functional connectivity in early human immunodeficiency virus infection. *Brain connectivity*, 1(3):207–217, 2011.

Xie, Song-yun, Guo, Rong, Li, Ning-fei, Wang, Ge, and Zhao, Hai-tao. Brain fmri processing and classification based on combination of pca and svm. In *IJCNN*, pp. 3384–3389. IEEE, 2009.

Yan, Shuicheng, Xu, Dong, Yang, Qiang, Zhang, Lei, Tang, Xiaoou, and Zhang, Hong-Jiang. Multilinear discriminant analysis for face recognition. *IEEE Transactions on Image Processing*, 16(1):212–220, 2007.

Zhao, Qibin, Zhou, Guoxu, Adalı, Tülay, Zhang, Liqing, and Cichocki, Andrzej. Kernel-based tensor partial least squares for reconstruction of limb movements. In *ICASSP*, pp. 3577–3581. IEEE, 2013a.

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

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