STATISTICALLY CONSISTENT SUPPORT TENSOR MACHINE FOR MULTI-DIMENSIONAL DATA

By

Peide Li

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Statistics – Doctor of Philosophy

2021

ABSTRACT

STATISTICALLY CONSISTENT SUPPORT TENSOR MACHINE FOR MULTI-DIMENSIONAL DATA

By

Peide Li

Tensors are generalizations of vectors and matrices for multi-dimensional data representation. Fueled by novel computing technologies, tensors have expanded to various domains, including statistics, data science, signal processing, and machine learning. Comparing to traditional data representation formats, tensor data representation distinguishes itself with its capability of preserving complex structures and multi-way features for multi-dimensional data. In this dissertation, we explore some tensor-based classification models and their statistical properties. In particular, we propose few novel support tensor machine methods for huge-size tensor and multimodal tensor classification problems, and study their classification consistency properties. These methods are applied to different applications for validation.

The first piece of work considers classification problems for gigantic size multi-dimensional data. Although current tensor-based classification approaches have demonstrated extraordinary performance in empirical studies, they may face more challenges such as long processing time and insufficient computer memory when dealing with big tensors. In chapter 3, we combine tensor-based random projection and support tensor machine, and propose a Tensor Ensemble Classifier (TEC) for ultra-high dimensional tensors, which aggregates multiple support tensor machines estimated from randomly projected CANDECOMP/PARAFAC (CP) tensors. This method utilizes Gaussian and spares random projections to compress high-dimensional tensor CP factors, and predicts their class labels with support tensor machine classifiers. With the well celebrated Johnson-Lindenstrauss Lemma and ensemble techniques, TEC methods are shown to be statistically consistent while having high computational efficiencies for big tensor data. Simulation studies and real data applications including Alzheimer's Disease MRI Image classification and Traffic Image classification are provided as empirical evidence to validate the performance of TEC models.

The second piece of work considers classification problems for multimodal tensor data, which are particularly common in neuroscience and brain imaging analysis. Utilizing multimodal data is of great interest for machine learning and statistics research in these domains, since it is believed that integration of features from multiple sources can potentially increase model performance while unveiling the interdependence between heterogeneous data. In chapter 4, we propose a Coupled Support Tensor Machine (C-STM) which adopts Advanced Coupled Matrix Tensor Factorization (ACMTF) and Multiple Kernel Learning (MKL) techniques for coupled matrix tensor data classification. The classification risk of C-STM is shown to be converging to the optimal Bayes risk, making itself a statistically consistent rule. The framework can also be easily extended for multimodal tensors with data modalities greater than two. The C-STM is validated through a simulation study as well as a simultaneous EEG-fMRI trial classification problem. The empirical evidence shows that C-STM can utilize information from multiple sources and provide a better performance comparing to the traditional methods.

To my parents and my grandmother.

ACKNOWLEDGEMENTS

I have received support and assistance from many people throughout the writing of this dissertation and my journey toward PhD. I want to take a moment and thank them.

First, I would like to express my deepest gratitude to my advisor Dr. Tapabrata Maiti, whose expertise is invaluable in exploring research questions. He always provides me with constructive insights and strong supports that sharpen my thinking and bring my work to a higher level. Without his guidance, I would not have made such a progress in this field.

I would also like to thank my dissertation guidance committee members, Dr. Jiayu Zhou, Dr. Ping-shou Zhong, Dr. David Zhu, and Dr. Shrijita Bhattacharya. Their comments and suggestions are extremely beneficial for my research.

I want to extend my appreciation to my collaborators, Dr. Selin Aviyente, Dr. Rejaul Karim, and Mr. Emre Sofuoglu. It is a great pleasure to work with them. Their expertise as well as dedication to scientific research help me to extend my work to a much boarder level.

I am also grateful to the help I obtained from all the professors and staff members in the Department of Statistics and Probability. I really appreciate the wonderful courses as well as the assistance they provided.

During my six years at Michigan State University, I made a lot of friends and met many kind peers. Thanks to them, I did not feel lonely during my PhD journey. I am very grateful to their sincerity and patience. I wish you all have a wonderful future.

Finally, I would like to thank my parents and Miss Jialin Qu. Thank you for your accompany and concerns that support me to go through this journey and overcome difficulties especially under the COVID-19 pandemic.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

With the development of computer technologies, more and more data with complex structures are observed in various research domains. The high-dimensionality as well as the multi-dimensional structure of the data have raised new challenges in data analysis to the communities of engineering, statistics, and data science. Learning multi-dimensional data with traditional statistical learning methods may not be appropriate, since these methods can suffer from the curses of dimensionality. Moreover, traditional methods are not able to preserve the intrinsic structures for multi-dimensional data, and are not able to utilize their multi-way features. Thus, developing novel statistical learning frameworks and data modeling techniques for multi-dimensional data has become popular in contemporary machine learning and statistics analysis.

As a generalization of vectors and matrices for higher-order data, tensor is originally proposed by [66], and becomes an efficient data representation format for multi-dimension data. Fueled by novel computing technologies arises in the past decade, tensors have expanded to many research domains such as statistics, data science, signal processing, and machine learning. Surveys from [78, 70, 18] demonstrate great potentials of using tensor data representation in data mining, statistics, and machine learning. It turns out that using tensor for multi-dimensional data learning can be efficient and appropriate since tensor can help to preserve multi-way structures for the data. Further, advanced operations in tensor algebra can also help to reduce computational cost, and, more importantly, unveil the complex correlation structures for the data. All these benefits make tensor data representation a perfect tool for learning multi-dimensional data.

Similar to the traditional machine learning research, current tensor-based machine learning and data mining techniques can be categorized as supervised learning and unsupervised learning. In the category of supervised learning, there are tensor regression and classification models, which usually take tensors as inputs. Depending on the types of outputs, tensor regression models are separated into tensor-to-scalar regression [58, 155, 148, 152, 130, 93, 62, 91], and tensor-to-tensor regression models [99, 121, 98, 51]. Moreover, there are tensor Bayes regression [57], tensor quantile regression [105], and tensor regression-based deep neural network model [80]. For tensor-based classification models, there are models which based on discriminant [147, 142, 103, 92]. In addition, many variants of support tensor machine models [136, 61, 63, 127, 64, 28] are also developed under the idea of maximum margin classifier. [114] provides a extension of probabilistic tensor discriminant analysis which extends linear discriminant to tensor data.

Comparing to the supervised learning, research on unsupervised learning with tensors are more dominant. The tensor decomposition techniques in [78] and [113] can be applied in many different application fields for multi-way feature extraction, latent factor estimation, and tensor subspace learning. These decomposition methods are later extended to different application fields. For example, [68, 1, 102] use low-rank tensor decomposition and robust tensor principle component analysis for missing data imputation. In spatio-temporal analysis such as traffic or internet data analysis, tensor decomposition are alos adopted in [120, 30, 146] for spatial and temporal feature extraction. Tensor approaches are also widely applied in anomaly detection problems. Survey from [44] reviews multiple anomaly detection algorithms basing on tensor data representation, which include predicting tensor anomalies from multiple tensors [31] and identifying abnormal elements within a single tensor [87, 153, 156]. Since tensor decomposition can be considered as generalizations of spectral decomposition on higher-order data, [67, 150, 131, 16, 132] use various decomposition methods to perform clustering and community detection for mulit-dimensional and heterogeneous data. In graphcial model and network analysis, [40, 138, 12, 111, 149] use tensor to model the correlation structures among different heterogeneous structures. Additionally, tensor decomposition can be applied in research about recommend system such as [17, 154, 104].

Apart from these two major categories, tensor data representation is often time used for the development of efficient algorithms. For example, random projection is a popular dimension reduction technique but is expensive to apply for high dimensional vector data. Saving the projection matrices can also be memory inefficient. [133, 71, 119] show that tensorizing random projection matrices can reduce the memory cost significantly while preserve the asymptotic isometry property of random projection for high dimensional data.

Motivated by these existing work using tensor data representation, this dissertation further investigates the performance of tensor-based machine learning models with a focus on classification problems. Particularly, we explore the statistical property of current tensor classification methods as well as their performance in multiple applications. In addition, we propose novel tensor classifiers for big tensor and multimodal tensor data classification. These become two major contributions in this dissertation.

1.1 Overview

The dissertation is organized as follow. In the rest part of this introduction chapter, we provide a review about tensor algebra, operations, and decomposition methods. We also briefly introduce some important statistical concepts in classification analysis in this chapter. The next three chapters in the dissertation explore tensor classification problems from different aspects. In chapter 2, We provide a survey about few most popular tensor classification methods in statistics and machine learning literature. All the methods are applied to Alzheimer's Disease MRI Image classification and Traffic Image classification problems to benchmark their performances. We further investigate the classification consistency for a certain type of non-parameteric tensor classifier, which is CP Support Tensor Machine (CP-STM). We show that with certain tensor kernel functions, CP-STM is statistically consistent.

Chapter 3 considers a specific tensor classification problem where the input tensors are in high dimension. In contemporary data science research, multi-dimensional observations such as spatial-temporal data, medical imaging data are usually coming with high dimensionality, i.e, the dimension of each mode is high even the data is in tensor shape. This raises extra challenges to the existing tensor-based classification models. To address the issue of high dimensionality, we propose a Tensor Ensemble Classifier (TEC) for ultra-high dimensional tensors, which aggregates multiple support tensor machines estimated from randomly projected CANDECOMP/PARAFAC (CP) tensors. This method utilizes Gaussian and spares random projections to compress high-dimensional tensor

CP factors, and predicts their class labels with support tensor machine classifier. With the well celebrated Johnson-Lindenstrauss Lemma and ensemble techniques, TEC methods are shown to be statistically consistent while having memory efficiency for big tensor data. Simulation studies and real data applications including Alzheimer's Disease MRI Image classification and Traffic Image classification are provided as empirical evidence to validate the performance of TEC models.

In the last chapter, we consider classification problems for multimodal tensor data, which are particularly common in neuroscience and brain imaging analysis. Utilizing multimodal data is of great interest for machine learning and statistics research in these domains, since it is believed that integration of features from multiple sources can potentially increase model performance while unveiling the interdependence between heterogeneous data. In chapter 4, we propose a Coupled Support Tensor Machine (C-STM) which adopted Advanced Coupled Matrix Tensor Factorization (ACMTF) and Multiple Kernel Learning (MKL) techniques for coupled matrix tensor data classification. The excess risk of C-STM is shown to be converging to the optimal Bayes risk, making itself a statistically consistent rule. The framework can also be easily extended for multimodal tensors with data modalities greater than two. The C-STM is validated with in a simulation study as well as in a simultaneous EEG-fMRI trial classification application. The empirical evidence shows that C-STM can utilize information from multiple source and provide a better performance comparing to the traditional methods.

1.2 Tensor Algebra

In this section, we introduce notations, and review some elementary concepts about tensors. Detailed introduction for tensor algebra, tensor decomposition, and tensor product space can be referred from [77] and [59].

1.2.1 Notations

The mathematical notations in the rest part of the thesis are defined as follow. Numbers and scalars are denoted by lowercase and capital letters such as x, N. Vectors are denoted by boldface lowercase

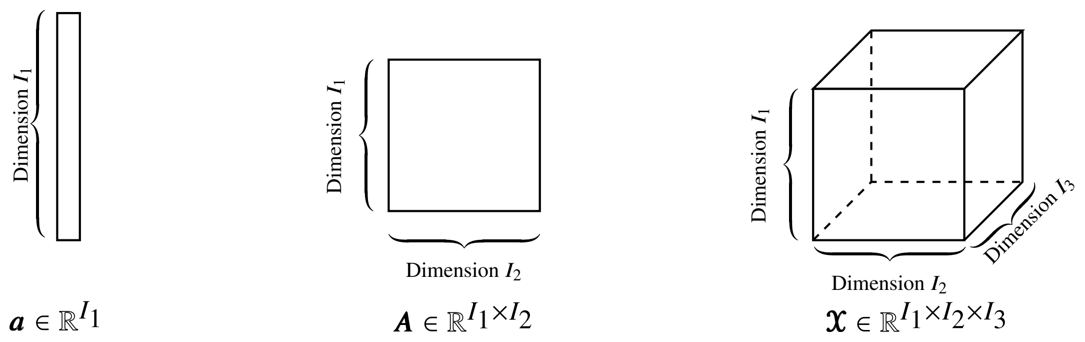

Dimension I_1
Dimension I_1
Dimension I_2
Dimension I_1
Dimension I_2
Dimension I_3
$\boldsymbol{a} \in \mathbb{R}^{I_1}$
$\boldsymbol{A} \in \mathbb{R}^{I_1 \times I_2}$
$\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$

Figure 1.1: Vector, Matrix, Tensor

letters, e.g. $\boldsymbol{a}$. Matrices are denoted by boldface capital letters, e.g. $\boldsymbol{A}, \boldsymbol{B}$. Higher-dimensional tensors are generalization of vector and matrix representations for higher order data, which are denoted by boldface Euler script letters such as $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}$. In general, functions and transformations are also denoted by boldface lowercase letters $\boldsymbol{f}, \boldsymbol{g}$, but with clear description to distinguish from vectors. The only exception is kernel function, which will be denote by $K(\cdot, \cdot)$. Vector spaces, functional spaces, and tensor spaces are denoted by boldface Mathcal font in Latex such as $\boldsymbol{\mathcal{H}}, \boldsymbol{\mathcal{F}}$. Euclidean spaces with one or multiple dimensions are represented by $\mathbb{R}^{I_1}$ and $\mathbb{R}^{I_1 \times I_2}$, where I_1 and I_2 stand for the size of each dimension. In addition to these notations, we use $\mathbb{E}$ and $\mathbb{P}$ to denote the expectation and probability in short. Other notations may also be used and be introduced as needed in the following content.

Tensor generalizes vectors and matrices by including multiple indices in its structure, making it possible to represent multi-dimensional data. Figure 1.1 provides a comparison between vector, matrix, and tensor. Tensor $\boldsymbol{\mathcal{X}}$ can denote three-dimensional data since it provides three indices. The order of a tensor is the number of dimensions, also known as ways or modes. For example, the vector $\boldsymbol{a}$ in figure 1.1 is a one-way tensor, matrix $\boldsymbol{A}$ is a two-way tensor, and $\boldsymbol{\mathcal{X}}$ is a three-way tensor. In general, a tensor can have d modes as long as d is an integer.

The way of indicating entries of tensors is same as we do for vectors and matrices. The i-th entry of a vector $\boldsymbol{x}$ is x_i, the (i, j)-th element of a matrix $\boldsymbol{X}$ is $x_{i,j}$, and the $(i_1, ..., i_d)$-th element of a d-way tensor $\boldsymbol{\mathcal{X}}$ is $x_{i_1,...,i_d}$. The indices of a tensor $i_1, ..., i_d$ range from 1 to their capital version, e.g. $i_k = 1, ..., I_k$ for every mode $k = 1, ...d$.

Sub-arrays of a tensor are formed when a subset of the indices are fixed. Similar to matrices that have rows and columns, high-dimensional tensors have various types of sub-arrays. For example, by fixing every index but one in a d-way tensor, we can get one of its fibers, which are analogue of matrix rows and columns. Another type of frequently used tensor sub-arrays is slice, which is a two dimensional section of a tensor. A slice of a tensor can be defined by fixing all but two indices. We will use $\mathcal{X}_{:i_2 \ldots i_d}$ to denote one fiber of a d-way tensor, and use $\mathcal{X}_{::i_3 \ldots i_d}$ to denote one of its slices.

Like the L2 norm for vectors in Euclidean spaces, the L2 norm, also called Frobenius norm, of a d-way tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_d}$ is the square root of the sum of the squares of all its elements, i.e.

$$||\mathcal{X}||_{\text{Fro}} =< \mathcal{X}, \mathcal{X} >= \sqrt{\sum_{i_1}^{I_1} \cdots \sum_{i_d}^{I_d} x_{i_1,\ldots,i_d}^2} \tag{1.1}$$

where

$$< \mathcal{X}_1, \mathcal{X}_2 >= \sum_{i_1}^{I_1} \cdots \sum_{i_d}^{I_d} x_{1,i_1,\ldots,i_d} \cdot x_{2,i_1,\ldots,i_d} \tag{1.2}$$

is the inner product of two tensors $\mathcal{X}_1$ and $\mathcal{X}_2$. In the following content, we may use different types of inner product induced by kernel functions, and we will specify those inner products as needed.

In many situations, one may need to transform a tensor into a vector or a matrix for computation. Such transformations are called tensor vectroization and unfolding. In the thesis, I denote the vectorization of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_d}$ as $\text{Vec}(\mathcal{X})$, which is in the dimension of $\prod_{j=1}^{d} I_j$. Tensor unfolding reorders a tensor into a matrix, putting mode-k fibers, $\mathcal{X}_{i_1,i_2..,i_{k-1},:,i_{k+1}\ldots i_d}$ as the columns of the matrix. As a result, the matrix is in the shape of $I_k \times \prod_{j=1,j\neq k}^{d} I_j$, and is denoted by $\mathcal{X}_{(k)}$. Although there are multiple ways of performing tensor vectorization and unfolding, the resulting vectors and matrices are equal up to a permutation. As long as the transformations are consistent, algorithms and theoretical analysis are remain intact. We follow the tensor vectorization and unfolding rules from [79] in the thesis.

In addition to the basic concepts, we also need some operations for vectors and matrices in order to construct tensors and present our work. The first one is the outer product of vectors. Let

$\boldsymbol{a} \in \mathbb{R}^p$ and $\boldsymbol{b} \in \mathbb{R}^q$ be two column vectors, the outer product of them is defined by

$$\boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{a} \cdot \boldsymbol{b}^T \tag{1.3}$$

which is a $p \times q$ matrix. If $\boldsymbol{A} \in \mathbb{R}^{p\times t}$ is a p by t matrix and $\boldsymbol{b} \in \mathbb{R}^q$ is a column vector, then

$$\boldsymbol{A} \circ \boldsymbol{b} = [\boldsymbol{A} * b_1, ..., \boldsymbol{A} * b_q] \tag{1.4}$$

which is a $p \times t \times q$ array. "$*$" stands for the element-wise product. The outer product with a vector increase the multiplier by one more dimension.

Another operation is Kronecker Product, which is a version of outer product for matrices. Let $\boldsymbol{A} \in \mathbb{R}^{I\times J}$, $\boldsymbol{B} \in \mathbb{R}^{K\times L}$ be two arbitrary matrices. The Kronecker Product of $\boldsymbol{A}$ and $\boldsymbol{B}$ is $\boldsymbol{A} \otimes \boldsymbol{B} \in \mathbb{R}^{(IK)\times(JL)}$

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}B & ... & a_{1J}B \\ ... & ... & ... \\ a_{I1}B & ... & a_{IJ}B \end{bmatrix} = [a_1 \circ b_1, ..., a_J \circ b_L] \tag{1.5}$$

Compared with the vector outer product, it restricts the resulting product to be matrices. The Khatri-Rao product is the "matching column-wise" Kronecker product between two matrices with same number of columns. Given matrices $\boldsymbol{A} \in \mathbb{R}^{I\times K}$ and $\boldsymbol{B} \in \mathbb{R}^{J\times K}$, the product is defined as:

$$\boldsymbol{A} \odot \boldsymbol{B} = [\boldsymbol{a}_1 \circ \boldsymbol{b}_1, ..., \boldsymbol{a}_K \circ \boldsymbol{b}_K] \tag{1.6}$$

It requires the two multiplier matrices to have the same number of columns, and the resulting products to be matrices as well. The vector outer product, matrix Kroncker product, and matrix Khatri-Rao product can be regarded as tensor product in mathematical analysis. As a result, we may use $\otimes$ to denote general tensor product in part of our theoretical development.

The mode-n product is a product operation defined between a tensor and a matrix. Assume $\mathcal{X} \in \mathbb{R}^{I_1\times...\times I_n\times...\times I_d}$ is a d-way tensor, and $\boldsymbol{U} \in \mathbb{R}^{P_n\times I_n}$ is a matrix. The mode-n product between tensor $\mathcal{X}$ and matrix $\boldsymbol{U}$ is defined as

$$\mathcal{X} \times_n \boldsymbol{U} = \boldsymbol{U} \cdot \mathcal{X}_{(n)} \tag{1.7}$$

where $\mathcal{X}_{(n)}$ is the n-th mode unfolding matrix of tensor $\mathcal{X}$ with shape I_n by $\prod_{j\neq n} I_j$. The resulting product is still a d-way tensor in shape of $I_1 \times ... \times P_n \times ... \times I_d$.

1.2.2 Tensor Decomposition

The notations and mathematical operations introduced above make it possible to represent tensors with their decomposition forms. Tensor decomposition is a way to represent, or approximate, a tensor with various pre-defined forms. With specially designed structure, new representation and approximation makes it more flexible to develop novel machine learning models for tensor data, and optimize the existing frameworks by simplifying computation steps. In this section, we review three most popular tensor decomposition methods, Parafac, Tucker, and Tensor-Train decomposition.

Candecomp / Parafac Decomposition (CP) is an extension of matrix singular value decomposition for higher-order tensors. It represents a tensor as a summation of vector outer products shown in figure 1.2. Each product term in the summation is also known as rank-one tensor. For a d-mode tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \ldots \times I_d}$, its CP decomposition is defined as

$$\mathcal{X} = \sum_{k=1}^{r} \alpha_k \boldsymbol{x}_k^{(1)} \circ \boldsymbol{x}_k^{(2)} \ldots \circ \boldsymbol{x}_k^{(d)} \tag{1.8}$$

where $\boldsymbol{x}_k^{(j)} \in \mathbb{R}^{I_j}$ are called tensor CP components for $j = 1, \ldots, d$. α_k are scalars and are often merged into one of the CP components for simplicity. As a result, CP decomposition can also be written as

$$\mathcal{X} = \sum_{k=1}^{r} \boldsymbol{x}_k^{(1)} \circ \boldsymbol{x}_k^{(2)} \ldots \circ \boldsymbol{x}_k^{(d)} \tag{1.9}$$

In our presentation, we will merge the scalar weights α_k to CP components and use the equation (1.9) for CP decomposition unless we specifically mention the weights. r is known as the CP rank for the tensor, which is the number of different outer products that adds up to the tensor. For a tensor which cannot be well represented by equation (1.9), i.e. the equation (1.9) does not hold, its CP decomposition is defined as

$$\hat{\mathcal{X}} \approx \sum_{k=1}^{r} \boldsymbol{x}_k^{(1)} \circ \boldsymbol{x}_k^{(2)} \ldots \circ \boldsymbol{x}_k^{(d)}$$

where

$$\hat{\mathcal{X}} = \sum_{k=1}^{r} \boldsymbol{x}_k^{(1)} \circ \boldsymbol{x}_k^{(2)} \ldots \circ \boldsymbol{x}_k^{(d)} \quad \text{and} \quad \hat{\mathcal{X}} = \arg\min_{\hat{\mathcal{X}}} ||\mathcal{X} - \hat{\mathcal{X}}||_{Fro}$$

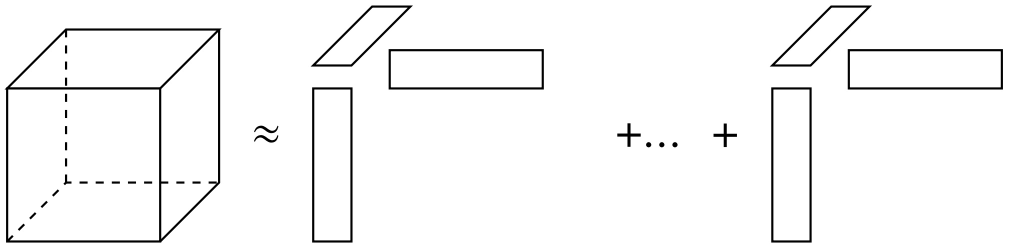


Figure 1.2: Tensor CP Decomposition

For the convenience of notation, we follow [78] and denote tensor CP decomposition (1.9) as

$$\mathcal{X} = [\![\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, ..., \boldsymbol{X}^{(d)}]\!] \quad \text{or} \quad \mathfrak{U}_{\mathcal{X}} = [\![\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, ..., \boldsymbol{X}^{(d)}]\!] \tag{1.10}$$

where $\boldsymbol{X}^{(j)} \in \mathbb{R}^{I_j \times r}$ are called CP factor matrices. The k-th column in $\boldsymbol{X}^{(j)}$ is the vector shape tensor CP factor $x_j^{(j)}$ in equation (1.9). This notation is also called Kruskal tensor. In the paper, we will use either tensor CP decomposition or CP tensor to refer any tensor that in expressed in (1.9) or (1.10).

Tucker Decomposition is a form of Principle Component Analysis for higher-order tensors, often denoted by Higher-order PCA (HOPCA). It factorizes a tensor into the form of a core tensor multiplied by a factor matrix at each one of its modes. The Tucker decomposition of a a d-mode tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 ... \times I_d}$ is defined as

$$\mathcal{X} = \mathcal{G} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} .... \times_d \boldsymbol{U}^{(d)} \tag{1.11}$$

where $\mathcal{G} \in \mathbb{R}^{P_1 \times P_2 ... \times P_d}$ is the core tensor in the shape of $P_1 \times P_2 ... \times P_d$. $\boldsymbol{U}^{(j)} \in \mathbb{R}^{P_j \times I_j}, j = 1, .., d$ are mode-wise factor matrices. In practice, one can restrict the factor matrices to be orthogonal, and thus consider the columns of these matrices as principle components from each mode. The core tensor $\mathcal{G}$ measures the interaction across different components. An example of 3-way tensor Tucker decomposition is demonstrated in figure 1.3. Similar to the CP decomposition, we can define the Tucker decomposition for an arbitrary tensor $\mathcal{X}$ even if the equation (1.11) does not hold. It is defined as

$$\mathcal{X} \approx \mathcal{G} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} .... \times_d \boldsymbol{U}^{(d)}$$

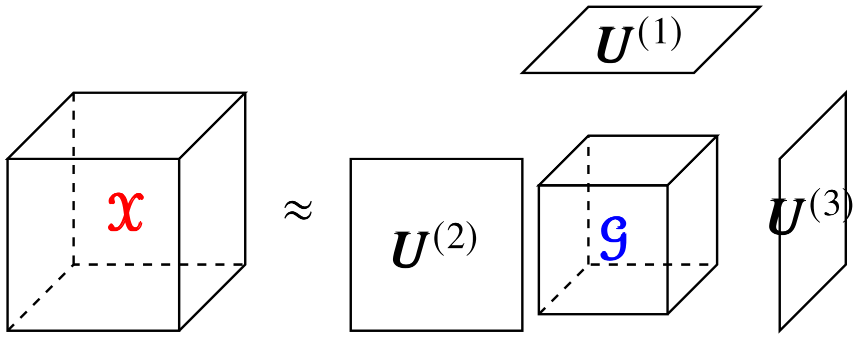


Figure 1.3: Tucker Decomposition

where

$$\hat{\mathcal{X}} = \mathcal{G} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} .... \times_d \boldsymbol{U}^{(d)} \quad \text{and} \quad \hat{\mathcal{X}} = \arg\min_{\hat{\mathcal{X}}} ||\mathcal{X} - \hat{\mathcal{X}}||_{Fro}$$

Notice that CP decomposition is actually a special case of Tucker decomposition, when the core tensor $\mathcal{G}$ in the decomposition is super-diagonal and all $P_1, ..., P_d$ are equal. The estimation of Tucker decomposition can be done with an iterative alternating least square algorithm introduced in [35]. Although Tucker decomposition is not easy to be interpreted comparing to CP decomposition, its mode-wise factor matrices can be regarded as basis for the row space of each tensor mode. Thus, it has been widely applied in problems like image compression and higher-order data feature extraction.

1.2.3 Tensor Product Space

Apart from the algebraic notations and operations for tensors, we also want to include a brief introduction about tensor functional and tensor space. They are essential in the development of universal tensor kernel functions and statistical consistency. We refer [59] for the definition of tensor product spaces and tensor calculus. Since we consider general tensor product in this section, we use $\otimes$ to denote it in our description.

For finite dimensional vector spaces, the space of their tensor product is call algebraic tensor space.

Definition 1.2.1. *Let $\boldsymbol{\mathcal{V}} \subset \mathbb{R}^{I_1}$ and $\boldsymbol{\mathcal{W}} \subset \mathbb{R}^{I_2}$ be two compact subspace of the Euclidean spaces $\mathbb{R}^{I_1}$ and $\mathbb{R}^{I_2}$. $\boldsymbol{\mathcal{V}} = \{\boldsymbol{v} : \boldsymbol{v} = \sum_i \alpha_i \boldsymbol{v}_i\}$, $\boldsymbol{\mathcal{W}} = \{\boldsymbol{w} : \boldsymbol{w} = \sum_j \beta_j \boldsymbol{w}_j\}$, where $\{\boldsymbol{v}_i\}$ and $\{\boldsymbol{w}_j\}$ are the basis of $\boldsymbol{\mathcal{V}}$ and $\boldsymbol{\mathcal{W}}$. The algebraic tensor space of $\boldsymbol{\mathcal{V}}$ and $\boldsymbol{\mathcal{W}}$, denoted by $\boldsymbol{\mathcal{T}}$, is a space spanned by the tensor products of basis.*

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{V}} \otimes \boldsymbol{\mathcal{W}} = \{\boldsymbol{t} : \boldsymbol{t} = \sum_{i,j} \gamma_{i,j} \boldsymbol{v}_i \otimes \boldsymbol{w}_j\} \tag{1.12}$$

The basis function of this algebraic tensor space are $\{\boldsymbol{v}_i \otimes \boldsymbol{w}_j\}$. The characteristic algebraic properties of the tensor space is the bilinearity, meaning that for all $\boldsymbol{t} \in \boldsymbol{\mathcal{T}}$ and $a \in \mathbb{R}$

$$a \cdot \boldsymbol{t} = \sum_{i,j} \gamma_{i,j} (a \cdot \boldsymbol{v}_i) \otimes \boldsymbol{w}_j = \sum_{i,j} \gamma_{i,j} \boldsymbol{v}_i \otimes (a \cdot \boldsymbol{w}_j)$$

We call this algebraic tensor space of a second-order algebraic tensor space since the basis are the tensor products of two vectors. This second-order tensor space is still a vector space, as defined in mathematics. However, if we consider a specific tensor product, outer product "$\circ$", in the definition 1.2.1, it is indeed isometric to a second-order tensor (matrix) space. The bijection connecting two spaces is a specific folding and unfolding rule as we introduced earlier. Notice that the algebraic tensor space measures distance by Euclidean norm, and the norms of multi-dimensional arrays are measured by Frobenuis norm. The equivalence between the Euclidean norm and Frobenuis preserves the distances between points unchanged before and after unfolding, and making two spaces isometric.

Similarly, if the general tensor product is replaced by Kroncker or Khatri-Rao product, the definition 1.2.1 can be extended for the product of matrices spaces. The isometry property also connects this abstract mathematical definition to the more concrete definition of tensors, especially those tensors decomposed into CP forms. As a result, data in forms of multi-dimensional arrays can be considered as points in a tensor product space. In general, we can define dth-order algebraic tensor space as

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{V}}^{(1)} \otimes \boldsymbol{\mathcal{V}}^{(2)} \dots \otimes \boldsymbol{\mathcal{V}}^{(d)} = span\{\otimes_{j=1}^{d} \boldsymbol{v}_k^{(j)}, \boldsymbol{v}_k^{(j)} \in \boldsymbol{\mathcal{V}}^{(j)}, j = 1, \dots d\} \tag{1.13}$$

for d-way tensors. This would make it feasible for us to develop further statistical analysis on tensors.

In the definition of algebraic tensor space, we only consider Euclidean subspaces and connects it to the spaces of multi-dimensional arrays. Indeed, the definition 1.2.1 can be extended to tensor products of any metric spaces such as the products of inner product spaces, the products of functional spaces, and the products of Reproducing Kernel Hilbert spaces. We define

Definition 1.2.2. *Let* $<\cdot,\cdot>_j$ *be a general inner product defined on* $\boldsymbol{\mathcal{V}}^{(j)}$ *such that* $\boldsymbol{\mathcal{V}}^{(j)}$ *is a inner product space.* $\boldsymbol{\mathcal{X}} = \otimes_{j=1}^{d}\mathbf{V}^{(j)}$ *is going to be an inner product space with inner product* $<\cdot,\cdot>_{\boldsymbol{\mathcal{X}}}$.

$$\boldsymbol{\mathcal{X}} = span\{\otimes_{j=1}^{d}\boldsymbol{f}_j^{(k)}, \boldsymbol{f}_k^{(j)} \in \boldsymbol{\mathcal{V}}^{(j)} \text{ are basis functions}, k = 1, ...d\} \tag{1.14}$$

For $\boldsymbol{f} = \sum_k \otimes_{j=1}^{d}\boldsymbol{f}_k^{(j)} \in \boldsymbol{\mathcal{X}}$ *and* $\boldsymbol{g} = \sum_l \otimes_{j=1}^{d}\boldsymbol{g}_l^{(j)} \in \boldsymbol{\mathcal{X}}$, *where* $\boldsymbol{f}_k^{(j)}, \boldsymbol{g}_l^{(j)} \in \mathbf{V}^{(j)}$ *are basis functions, the inner product is*

$$<\boldsymbol{f},\boldsymbol{g}>_{\boldsymbol{\mathcal{X}}} = \sum_{k,l}\prod_{j=1}^{d} <\boldsymbol{f}_k^{(j)},\boldsymbol{g}_l^{(j)}>_j \tag{1.15}$$

This definition generalizes the tensor product spaces to any arbitrary inner products spaces. For example, if each $\boldsymbol{\mathcal{V}}^{(j)}$ is a uni-variate functional space, then $\boldsymbol{\mathcal{X}}$ is a multi-variate functional space whose elements are functions mapping vectors into scalars. Moreover, this definition will help us to construct tensor Reproducing Kernel Hilbert space in chapter 2.

1.3 The Bayes Error and Classification Consistency

Statistical analysis in classification problems often time tries to validate the performance of a model by looking at whether its classification risk is close to the Bayes risk, and if it is statistically consistent. These two components are essential in the evaluation of generalization ability for a specific model. We briefly review the definition of Bayes error and classification consistency in this section. More details can be referred from [36].

1.3.1 The Bayes Problem

Consider a binary classification problem where (X, Y) is a pair of random variables taking their respective values on $\mathbb{R}^d$ and $\{0, 1\}$. Let

$$\boldsymbol{\eta}(\boldsymbol{x}) = \mathbb{P}(Y = 1|\boldsymbol{x}) = \mathbb{E}(Y|\boldsymbol{x}) \tag{1.16}$$

Naturally, any measureable function $\boldsymbol{f} : \mathbb{R}^d \to \{0,1\}$ can be a potential classifier or decision function. Now if we consider the very naive zero one loss function $\mathcal{L}(z,y) = \mathbf{1}\{z \neq y\}$, then the expected loss of a classifier $\boldsymbol{f}$, called risk of $\boldsymbol{f}$, is $\mathcal{R}(\boldsymbol{f}) = \mathbb{E}[\mathcal{L}(\boldsymbol{f}(X),Y)] = \mathbb{P}(\boldsymbol{f}(X) \neq Y)$. Let

$$\boldsymbol{f}^*(\boldsymbol{x}) = \begin{cases} 1 & \boldsymbol{\eta}(\boldsymbol{x}) \geqslant \frac{1}{2} \\ 0 & \text{Otherwise} \end{cases}$$

It is easy to show that among all possible decision functions, $\boldsymbol{f}^*$ has the smallest risk, making itself the best possible classifier. A Bayes problem is to find the optimal classifier $\boldsymbol{f}^*$, and $\boldsymbol{f}^*$ itself is called the Bayes classifier or Bayes rule. The classification risk of the Bayes rule, $\mathcal{R}^* = \mathcal{R}(\boldsymbol{f}^*)$, is defined as the Bayes risk, which is the smallest possible risk one can obtain. Under most circumstances, it is infeasible to estimate the Bayes rule since the distribution of (X,Y) is unknown.

1.3.2 Consistent Classification Rules

Instead of searching for Bayes rule, most of time we construct a classifier from a limited amount of data. Suppose $T_n = \{(\boldsymbol{x}_1, y_1), ...(\boldsymbol{x}_n, y_n)\}$ is a collection of observations for the random variable (X,Y), the empirical estimate of the classification risk for a decision function $\boldsymbol{f}$ is

$$\mathcal{R}_n = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{\boldsymbol{f}(\boldsymbol{x}_i) \neq y_i\}$$

$\mathbf{1}\{\cdot\}$ is an indicator function. A "good" classifier can be constructed from the data T_n by searching for the optimizer that minimizes the empirical risk. Such a procedure is call Empirical Risk Minimization (ERM), and T_n is called training set. If we denote the empirical optimal decision function as $\boldsymbol{f_n}$, it is a function conditional on the training set T_n. However, if we use the same strategy but with different training set, we can get a sequence of decision functions $\{\boldsymbol{f_n}\}$. Such a sequence of functions estimated with the same strategy / rule but different training data is called a classification rule, a way of finding optimal decision functions from a training data. ERM procedure produces a decision rule, and there are more variants of ERM such as regularized ERM in the statistical learning literature.

To show a classification rule is good from mathematical aspect, one possible way is to prove it is statistically consistent.

Definition 1.3.1. *A classification rule is (weakly) consistent for a certain distribution of* (X, Y) *if*

$$\mathcal{R}(\boldsymbol{f_n}) \to \mathcal{R}^* \qquad \textit{in probability}$$

and is strongly consistent if

$$\mathcal{R}(\boldsymbol{f_n}) \to \mathcal{R}^* \qquad \textit{a.s.}$$

as $n \to \infty$.

A consistent rule, not the specific classifier learned from a training data, guarantees that taking sufficiently large samples can reconstruct the unknown data distribution, and finally identify the optimal classifier. This property is like telling a classification rule is learning data in a right way, since it eventually will unveil the whole data distribution. The reconstruction here means the Bayes risk of the classification problem will be eventually the same as the risk of estimated classifier with sufficient training data, and thus will be known. For most classification models, developing statistical consistency is of great interest as it validates that the model is learning the data in a "right" way. In this thesis, our theoretical analysis is also mostly focus on the development of statistical consistency for tensor-based classifiers.

1.3.3 Surrogate Loss Consistency

In binary classification problems, the most intuitive and basic loss function is the "zero-one" loss $\mathcal{L}(z, y) = \mathbf{1}\{z \neq y\}$. However, its non-convexity brings lots of challenges in both computational aspect and statistical properties. Moreover, the zero-one loss may have a worse performance than other surrogate loss in various classification applications. Recent works [13, 122, 151] demonstrate that there are many surrogate loss functions for binary classification which equip with convexity and nice statistical properties, making the estimation procedure more tractable. Another motivation of using well-behaved surrogate loss in classification applications is that there is a general quantitative relationship between the approximation and estimation risks associated with surrogate losses, and those associated with zero-one loss. Here we denote the risk of a decision function $\boldsymbol{f}$ associated with a surrogate loss $\mathcal{L}$ as $\mathcal{R}_{\mathcal{L}}(\boldsymbol{f})$, and the corresponding risk associated with the zero-one loss as $\mathcal{R}(\boldsymbol{f})$. Further, the Bayes risk under loss function $\mathcal{L}$ is denoted as $\mathcal{R}^*_{\mathcal{L}}$, and the Bayes risk under zero-one loss is denoted as $\mathcal{R}^*$. The definition of these risks are

$$\mathcal{R}_{\mathcal{L}}(\boldsymbol{f}) = \mathbb{E}_{\boldsymbol{\mathcal{X}}\times\boldsymbol{\mathcal{Y}}}\left[\mathcal{L}(\boldsymbol{f}(X), Y)\right], \quad \mathcal{R}^*_{\mathcal{L}} = \mathbb{E}_{\boldsymbol{\mathcal{X}}\times\boldsymbol{\mathcal{Y}}}\left[\mathcal{L}(\boldsymbol{f}^*(X), Y)\right]$$

where $\boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}$ is the domain of the random variables (X, Y). The expectation is taken over the joint distribution of (X, Y). $\boldsymbol{f}^*$ is the Bayes classifier such that $\boldsymbol{f}^* = \arg\min \mathcal{R}_{\mathcal{L}}(\boldsymbol{f})$. $\boldsymbol{f}^*$ is the optimal among all measureable functions which map data in $\boldsymbol{\mathcal{X}}$ to labels $\boldsymbol{\mathcal{Y}}$. Results from [151] shows that for any measureable function $\boldsymbol{f}$

$$\boldsymbol{\psi}(\mathcal{R}(\boldsymbol{f}) - \mathcal{R}^*) \leqslant \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}) - \mathcal{R}^*_{\mathcal{L}}$$

for a nondecreasing function $\boldsymbol{\psi} : [0, 1] \to [0, \infty)$. This suggests that the statistical consistency developed under surrogate loss function indicates the consistency under zero-one loss, as long as the surrogate loss is well-behaved. Thanks to this general relationship, one can develop statistical consistency for decision rules using surrogate loss and enjoy their nice mathematical properties like Lipschitz continuity instead of using zero-one loss. This reduces the problem difficulties significantly. Lastly, using surrogate losses may enable us to develop a uniform upper bound on the risk of a function $\boldsymbol{f_n}$ that minimizes the empirical risks. This may help us to further obtain an explicit, uniform bound on the excess risk, $\mathcal{R}_{\mathcal{L}}(\boldsymbol{f_n}) - \mathcal{R}^*_{\mathcal{L}}$, highlighting the convergence rate of a specific decision rule.

The well-behaved losses are sometimes called self-calibrated or classification-calibrated loss. Examples of such loss functions include Hinge loss, Squared Hinges loss, and exponential loss. A more detailed discussion about self-calibrated loss is available in [95] and the Section 2 of [128]. We estimate tensor classification models using Hinge and Squared Hinge loss in this thesis, and develop our theoretical results with surrogate loss functions.

CHAPTER 2

TENSOR CLASSIFICATION MODELS

In this chapter, we provide an introduction to several tensor-based classification models, and compare their performance empirically. Moreover, the statistical consistency of few classifiers are established.

2.1 Introduction

In contemporary machine learning and statistics research, tensor has become a popular tool to model multi-dimensional data such as spatio-temporal data, brain imaging, and multimodal data. Comparing to the traditional vector presentation, tensor preserves the multi-way structures of the data, providing more correlations among different modes for data mining and modeling. In addition, the existing tensor decomposition methods from [78] can help to estimate the low-dimensional structure for tensor data, which reduce the computational complexity significantly for tensor-based models.

As an essential part of supervised tensor learning, tensor classification problems try to predict data labels from tensors. Current literature about tensor classification can be categorized in several groups. First, since distances between tensors can be easily estimated by Frobenious norm, K-nearest neighbour classifiers can be easily established. However, the Frobenious norm of a tensor is equivalent to the L2 norm of its vectorization. Such extensions are indeed equivalent to the vector-based K-nearest neighbour classifiers, and thus have no computational gain from tensor representation. An improvement [92] then has been made on the tensor K-nearest neighbour classifiers by combining it with a Fisher discriminant analysis. Utilizing the multi-way features preserved by tensors, [92] learns multi-linear transformations projecting tensors to lower multi-dimensional spaces where they are easier to be classified. [114] also develops a probabilistic discriminant analysis for tensors, using density instead of distance to discriminate data. Another type of tensor-based classifiers borrow the separating hyperplane from support vector machine, and build support tensor machine models. With different tensor decomposition and kernel functions, there are models like rank-1 CP-STM [136], CP-STMs [63, 64], Tucker STM [127], and support tensor train machine [28]. These models benefit from the distribution-free assumption for tensor data, and are more flexible in real-data applications. Finally, logistic regression model can also be generalized for tensor data. [155] and [93] develop generalized linear regression models with CP and Tucker tensor coefficients, which can be adopted for classification problems.

Although the current approaches have demonstrated impressive performance, not all of them provide theoretical guarantee on their generalization ability. According to [36], a classifier with solid generalization ability should be statistically consistent, having their excess classification risks converge to the optimal Bayes risk. Bayes risk is the minimal risk one can obtain from a classification problem with data confirming a certain type of probability distribution. The difference between the risk of a learned classifier and the optimal Bayes risk quantifies the performance of the classifier theoretically. Such results are well established for traditional statistical classification approaches, however, are not completed for all tensor-based methods.

In this chapter, we introduce few popular tensor-based classifiers in current literature including CP-STM [63], tensor discriminant analysis [92] and CP-GLM [155], and investigate their performance through numerical studies. Further, we discuss their statistical consistency, and provide a theoretical result which establishes the statistical consistency for CP-STM. For other methods, the results are introduced as they are can be easily extended from the existing literature. The rest parts in this chapter are organized as follow: Section 2.2 reviews three major types of tensor-based classifiers and their consistency results. Section 2.3 develop the consistency result for the support tensor machine model. In section 2.4, we compare the performance of all reviewed tensor-based methods with two different real data applications. Section 2.5 concludes the chapter.

2.2 Tensor Classification Algorithms

In this section, we introduce five different tensor-based classifiers which are categorized into three groups depending on their model mechanism.

2.2.1 Support Tensor Machine

Support tensor machine extends the idea of kernel support vector machine (see e.g. [128]), and construct a separating hyperplane with support tensors for classification. In this part, we review the Candecomp/Parafac - Support Tensor Machine (CP-STM) model from [63], and provide two different model estimation algorithms.

Suppose there is a training data $T_n = \{(\mathcal{X}_1, y_1), (\mathcal{X}_2, y_2), ..., (\mathcal{X}_n, y_n)\}$, where $\mathcal{X}_i \in \boldsymbol{\mathcal{X}} \subset \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_d}$ are d-way tensors. $\mathcal{X}$ is a compact tensor space, which is a subspace of $\mathbb{R}^{I_1 \times I_2 \times \ldots \times I_d}$. $y_i \in \{1, -1\}$ are binary labels. CP-STM, like the traditional kernel support vector machine, tries to estimate a decision function $\boldsymbol{f} : \boldsymbol{\mathcal{X}} \to \mathbb{R}$ such that it minimizes the objective function

$$\min \quad \lambda ||\boldsymbol{f}||^2 + \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\boldsymbol{f}(\mathcal{X}_i), y_i) \tag{2.1}$$

$\mathcal{L}$ is a loss function for classification such as Hinge loss, squared Hinge loss, and zero-one loss. λ is a tuning parameter. $||\boldsymbol{f}||^2 =< \boldsymbol{f}, \boldsymbol{f} >= \int \boldsymbol{f}(\mathcal{X})^2 d\mathcal{X}$ is the square of functional norm for $\boldsymbol{f}$. The kernel functions for tensor data are defined on the CP representation of tensors. Assume two d-way tensors with CP rank r are represented as $\mathcal{X}_1 = \sum_{k=1}^{r} \boldsymbol{x}_{1,k}^{(1)} \circ \boldsymbol{x}_{1,k}^{(2)} \ldots \circ \boldsymbol{x}_{1,k}^{(d)}$ and $\mathcal{X}_2 = \sum_{k=1}^{r} \boldsymbol{x}_{2,k}^{(1)} \circ \boldsymbol{x}_{2,k}^{(2)} \ldots \circ \boldsymbol{x}_{2,k}^{(d)}$, a tensor kernel function is defined as

$$K(\mathcal{X}_1, \mathcal{X}_2) = \sum_{l,k=1}^{r} \prod_{j=1}^{d} K^{(j)}(\boldsymbol{x}_{1,l}^{(j)}, \boldsymbol{x}_{2,k}^{(j)}) \tag{2.2}$$

where $K^{(j)}$ are vector-based kernel functions measuring inner products for factors in different tensor modes. The kernel function (2.2) measures the inner products between two tensors by aggregating kernel values of their CP factors across ranks. With the kernel trick and representer's theorem [9], the optimal decision rule for the optimization problem (2.1) has the form of

$$\boldsymbol{f}(\mathcal{X}) = \sum_{i=1}^{n} \alpha_i y_i K(\mathcal{X}_i, \mathcal{X}) = \boldsymbol{\alpha}^T \boldsymbol{D}_y \boldsymbol{K}(\mathcal{X}) \tag{2.3}$$

where $\mathcal{X}$ is a new d-way rank-r tensor with shape $I_1 \times I_2 \times \ldots \times I_d$. $\boldsymbol{\alpha} = [\alpha_1, ..., \alpha_n]^T$ are the coefficients learned by plugging function (2.3) into objective function (2.1) and minimize (2.1). $\boldsymbol{D}_y$ is a diagonal matrix whose diagonal elements are $y_1, .., y_n$. $\boldsymbol{K}(\mathcal{X}) = [K(\mathcal{X}_1, \mathcal{X}), ..., K(\mathcal{X}_n, \mathcal{X})]^T$

is a column vector. If we denote the collections of functions which are in the form of equation (2.3) by $\boldsymbol{\mathcal{H}}$ such that $\boldsymbol{\mathcal{H}} = \{\boldsymbol{f} : \boldsymbol{f} = \boldsymbol{\alpha}^T \boldsymbol{D}_y \boldsymbol{K}(\mathcal{X}), \boldsymbol{\alpha} \in \mathbb{R}^n\}$, then the optimal STM classifier is denoted as

$$\boldsymbol{f_n} = \arg\min_{\boldsymbol{f} \in \boldsymbol{\mathcal{H}}} \lambda ||\boldsymbol{f}||^2 + \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\boldsymbol{f}(\mathcal{X}_i), y_i) \tag{2.4}$$

and the class labels are predicted by Sign$[\boldsymbol{f_n}]$. $\boldsymbol{\mathcal{H}}$ is the Reproducing Kernel Hilbert Space generated by tensor kernel (2.2). Support tensors are the tensors whose corresponding coefficients, α_i in $\boldsymbol{f_n}$, are non-zero.

Estimating $\boldsymbol{f_n}$ from the training data T_n can be accomplished in various ways. Also, the estimated classifiers can be different, and have different excess error with different types of loss function $\mathcal{L}$. We adopt two different loss functions, Hinge loss and squared Hinge loss, and provide different estimation algorithms. Hinge loss $\mathcal{L}(\boldsymbol{f}(\mathcal{X}), y) = \max(0, 1 - y \cdot \boldsymbol{f}(\mathcal{X}))$ is a convex and non-differentiable loss designed for support vector / tensor types of classifiers. Comparing to the regular zero-one loss in binary classification, Hinge loss has the same level of penalty for miss-classified points that are close to the separating hyper-plan, while putting a more severe penalty for those which are far away from the plan. [122] demonstrates the statistical robustness and the fast convergence rate of Hinge loss in binary classification problems. Minimizing the objective function (2.1) with Hinge loss can be shown to be equivalent to the optimization problem

$$\begin{aligned} \min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \quad & \frac{1}{2} \boldsymbol{\alpha}^T \boldsymbol{D}_y \boldsymbol{K} \boldsymbol{D}_y \boldsymbol{\alpha} - \boldsymbol{1}^T \boldsymbol{\alpha} \\ \text{S.T.} \quad & \boldsymbol{\alpha}^T \boldsymbol{y} = 0 \\ & 0 \leq \boldsymbol{\alpha} \leq \frac{1}{2n\lambda} \end{aligned} \tag{2.5}$$

Equation (2.5) is the dual problem of the original STM problem with Hinge loss. The derivation is provided in [25]. Notice that this problem has quadratic objective function and inequality constrains, which can be solved by Quadratic Programming (QP) in [20]. The steps are summarized in the algorithm 1. We use python-style pseudo-code to denote columns of matrices. For example, $\boldsymbol{X}_i^{(m)}[:, k]$ stands for the k-th column of CP factor matrix $\boldsymbol{X}_i^{(m)}$. We will use such notations in the rest part of the thesis.

Algorithm 1 Hinge STM

```
1:  procedure STM Train
2:      Input: Training set $T_n = \{\mathcal{X}_i\}$, $\boldsymbol{y}$, kernel function $K$, tensor rank r, $\lambda$
3:      for i = 1, 2,...n do
4:          $\mathcal{X}_i = [\boldsymbol{X}_i^{(1)}, ..., \boldsymbol{X}_i^{(d)}]$          ▷ CP decomposition by ALS algorithm
5:      Create initial matrix $\boldsymbol{K} \in \mathbb{R}^{n \times n}$
6:      for i = 1,...,n do
7:          for j = 1,...,i do
8:              $\boldsymbol{K}_{i,j} = \sum_{k,l=1}^{r} \prod_{m=1}^{d} K(\boldsymbol{X}_i^{(m)}[:,k], \boldsymbol{X}_j^{(m)}[:,l])$          ▷ Kernel values
9:              $\boldsymbol{K}_{j,i} = \boldsymbol{K}_{i,j}$
10:     Solve the quadratic programming problem (2.5) and find the optimal $\boldsymbol{\alpha}^*$.
11:     Output: $\boldsymbol{\alpha}^*$
```

Another loss function which is commonly used for binary classification is the Squared Hinge loss, which is a convex and differentiable surrogate of Hinge loss. Squared Hinge loss squares the Hinge loss $\mathcal{L}(\boldsymbol{f}(\mathcal{X}), y) = (\max(0, 1 - y \cdot \boldsymbol{f}(\mathcal{X})))^2$, making it differentiable when $y \cdot \boldsymbol{f}(\mathcal{X}) = 1$. Plugging Squared Hinge loss makes the objective function (2.1) differentiable, and can be minimized by letting its derivative to be zero. The objective function (2.1) with Squared Hinge loss, written in the matrix form, is

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \quad \lambda \boldsymbol{\alpha}^T \boldsymbol{D}_y \boldsymbol{K} \boldsymbol{D}_y \boldsymbol{\alpha} + \frac{1}{n} \sum_{i=1}^{n} (\max(0, 1 - y_i \cdot \boldsymbol{\alpha}^T \boldsymbol{D}_y \boldsymbol{k}_i))^2 \tag{2.6}$$

$\boldsymbol{K}$ is the n-by-n kernel matrix whose (i, j)-th element is $K(\mathcal{X}_i, \mathcal{X}_j)$. $\boldsymbol{k}_i = [K(\mathcal{X}_i, \mathcal{X}_1), ..., K(\mathcal{X}_i, \mathcal{X}_n)]^T$ is the i-th column of the kernel matrix $\boldsymbol{K}$. The derivative of (2.6) with respect to $\boldsymbol{\alpha}$ is

$$\begin{aligned} \nabla =& 2\lambda \boldsymbol{D}_y \boldsymbol{K} \boldsymbol{D}_y \boldsymbol{\alpha} + \frac{2}{n} \sum_{i=1}^{n} (-1) y_i \boldsymbol{D}_y \boldsymbol{k}_i \cdot \max(0, 1 - y_i \cdot \boldsymbol{\alpha}^T \boldsymbol{D}_y \boldsymbol{k}_i)^T \\ =& 2\lambda \boldsymbol{D}_y \boldsymbol{K} \boldsymbol{D}_y \boldsymbol{\alpha} + \frac{2}{n} \sum_{i \in \boldsymbol{s}} \left[\boldsymbol{D}_y \boldsymbol{k}_i \boldsymbol{k}_i^T \boldsymbol{D}_y \boldsymbol{\alpha} - \boldsymbol{D}_y \boldsymbol{k}_i y_i \right] \\ =& 2\lambda \boldsymbol{D}_y \boldsymbol{K} \boldsymbol{D}_y \boldsymbol{\alpha} + \frac{2}{n} \boldsymbol{D}_y \boldsymbol{K} \boldsymbol{I}_{\boldsymbol{s}} \left[\boldsymbol{K}^T \boldsymbol{D}_y \boldsymbol{\alpha} - \boldsymbol{y} \right] \\ =& 2\boldsymbol{D}_y \boldsymbol{K} \left[\left[\lambda \boldsymbol{I} + \frac{1}{n} \boldsymbol{I}_{\boldsymbol{s}} \boldsymbol{K}^T \right] \boldsymbol{D}_y \boldsymbol{\alpha} - \frac{1}{n} \boldsymbol{I}_{\boldsymbol{s}} \boldsymbol{y} \right] \end{aligned} \tag{2.7}$$

where $\boldsymbol{y} = [y_1, ..., y_n]^T$ and $\boldsymbol{s}$ is the collection of indices for support tensors. Support tensors are those tenors with labels $(\mathcal{X}_i, y_i)$ such that $y_i \cdot \boldsymbol{\alpha}^T \boldsymbol{D}_y \boldsymbol{k}_i < 1$ when a $\boldsymbol{\alpha}$ is given. $\boldsymbol{I}_{\boldsymbol{s}}$ is a identity matrix with size n whose diagonal elements corresponding to non-support tensors are set to be zero. To estimate the α such that the derivative (2.7) equals to zero, we can use the Gaussian-Newton method (see e.g. [47]). If we denote the second derivative of the objective function (2.6) with respect to $\boldsymbol{\alpha}$ by $\boldsymbol{H}$

$$\boldsymbol{H} = 2\boldsymbol{D}_y\boldsymbol{K}(\lambda\boldsymbol{I} + \frac{1}{n}\boldsymbol{I_s}\boldsymbol{K}^T)\boldsymbol{D}_y \tag{2.8}$$

If $\boldsymbol{\alpha}^*$ is the root of $\nabla = 0$, then Gaussian-Newton algorithm uses the first order Taylor expansion and assumes that $\nabla|_{\boldsymbol{\alpha}^*} = \nabla|_{\boldsymbol{\alpha}} + \boldsymbol{H}|_{\boldsymbol{\alpha}} \cdot (\boldsymbol{\alpha}^* - \boldsymbol{\alpha})$. Since we assume $\nabla|_{\boldsymbol{\alpha}^*} = 0$, the equation reduces to

$$\begin{aligned}
\boldsymbol{\alpha}^* &= \boldsymbol{\alpha} - \boldsymbol{H}^{-1} \nabla |_{\boldsymbol{\alpha}} \\
&= \boldsymbol{\alpha} - \boldsymbol{D}_y(\lambda\boldsymbol{I} + \frac{1}{n}\boldsymbol{I_s}\boldsymbol{K}^T)^{-1} \cdot \left[(\lambda\boldsymbol{I} + \frac{1}{n}\boldsymbol{I_s}\boldsymbol{K}^T)\boldsymbol{D}_y\boldsymbol{\alpha} - \frac{1}{n}\boldsymbol{I_s}\boldsymbol{y}\right] \\
&= \frac{1}{n}\boldsymbol{D}_y(\lambda\boldsymbol{I} + \frac{1}{n}\boldsymbol{I_s}\boldsymbol{K}^T)^{-1}\boldsymbol{I_s}\boldsymbol{y} \\
&= \frac{1}{n}\boldsymbol{D}_y(\lambda\boldsymbol{I} + \frac{1}{n}\boldsymbol{I_s}\boldsymbol{K})^{-1}\boldsymbol{I_s}\boldsymbol{y}
\end{aligned} \tag{2.9}$$

We drop the transpose for convenience in the last step since kernel matrix is symmetric. Notice that the derivation uses the fact $\boldsymbol{D}_y$ is symmetric and orthogonal, $\boldsymbol{D}_y\boldsymbol{D}_y^T = \boldsymbol{I}$, since $y_i^2 = 1$. The algorithm starts with an initial value of $\boldsymbol{\alpha}$, and keeps updating $\boldsymbol{\alpha}^*$ with equation (2.9) iteratively until convergence. During each iteration, the newly estimation $\boldsymbol{\alpha}^*$ will replace $\boldsymbol{\alpha}$ for the next iteration. Although the update rule does not include $\boldsymbol{\alpha}$ in the final explicit form (2.9), the indices of support tensors are updated at each iteration. Thus, $\boldsymbol{I_s}$ will be updated, and making estimate for $\boldsymbol{\alpha}^*$ to be different. The algorithmic steps for Squared Hinge loss STM is summarized in the algorithm 2.

After estimating $\boldsymbol{\alpha}$ from either (2.5) with Hinge loss or (2.6) with Squared Hinge loss, the class label can be predicted by $\text{Sign}[\boldsymbol{f}(\mathfrak{X})] = \text{Sign}[\boldsymbol{\alpha}\boldsymbol{D}_y\boldsymbol{K}(\mathfrak{X})]$. CP representation of tensors in the training data are already available. To calculate $\boldsymbol{K}(\mathfrak{X})$, one has to find the CP representation for the testing data $\mathfrak{X}$ and then calculate the kernel values with equation (2.2).

2.2.2 Tensor Discriminant Analysis

The second types of classification method is tensor-based discriminant analysis (TDA). Tensor discriminant analysis combines tensor-based K-nearest neighbour classifier and multilinear feature

Algorithm 2 Squared Hinge STM

```
1: procedure STM Train
2:     Input: Training set $T = \{\mathcal{X}_i\}$, $\boldsymbol{y}$, kernel function $K$, tensor rank r, $\lambda$, $\eta$, maxiter
3:     for i = 1, 2,...n do
4:         $\mathcal{X}_i = [\boldsymbol{X}_i^{(1)}, ..., \boldsymbol{X}_i^{(d)}]$        ▷ CP decomposition by ALS algorithm
5:     Create initial matrix $\boldsymbol{K} \in \mathbb{R}^{n\times n}$
6:     for i = 1,...,n do
7:         for j = 1,...,i do
8:             $\boldsymbol{K}_{i,j} = \sum_{k,l=1}^{r} \prod_{m=1}^{d} K(\boldsymbol{X}_i^{(m)}[:,k], \boldsymbol{X}_j^{(m)}[:,l])$        ▷ Kernel values
9:             $\boldsymbol{K}_{j,i} = \boldsymbol{K}_{i,j}$
10:    Create $\boldsymbol{\alpha}^* = \boldsymbol{1}_{n\times 1}, \boldsymbol{\alpha} = \boldsymbol{0}_{n\times 1}$        ▷ Initial Value, can be different
11:    Iteration = 0
12:    while $||\boldsymbol{\alpha}^* - \boldsymbol{\alpha}||_2 \geqslant \eta$ & Iteration $\leqslant$ maxiter do
13:        $\boldsymbol{\alpha} = \boldsymbol{\alpha}^*$
14:        Find $\boldsymbol{s} \in \mathbb{R}^{n\times 1}$. $\boldsymbol{s}_i \in \{0, 1\}$ such that $\boldsymbol{s}_i = 1$ if $y_i \boldsymbol{k}_i^T \boldsymbol{\alpha} < 1$        ▷ Indicating support tensors
15:        $\boldsymbol{I}_s = \mathrm{diag}(\boldsymbol{s})$        ▷ Create diagonal matrix with $S$ as diagonal
16:        $\boldsymbol{\alpha}^* = \frac{1}{n}\boldsymbol{D}_y(\lambda \boldsymbol{I} + \frac{1}{n}\boldsymbol{I_s K})^{-1}\boldsymbol{I_s y}$        ▷ Update
17:    Output: $\boldsymbol{\alpha}^*$
```

extraction to improve the classification performance and utilize the multi-way correlations. It seeks a tensor-to-tensor projection transforming tensors into a new tensor subspace which maximizes the data separation. To measure the level of data separation in the new tensor subspace, TDA adopted two criteria from Fisher discriminant analysis [108]: the scatter ratio criterion and the scatter difference criterion. The optimal tensor-to-tensor projection is selected to maximize either one of the criterion. The discriminant analysis with tensor representation (DATER) [147] and multilinear discriminant analysis MDA [103] search the optimal projection utilizing the maximum ratio criteria. However, the algorithms are not stable, and do not converge over iterations. The general tensor discriminant analysis (GTDA) from [135] and MDA from [142] use the maximum scatter difference criterion, and provide two convergent algorithms in tensor subspace learning. However, the model classification performance relies heavily on tuning parameters. [92] proposes Direct General Tensor Discriminant Analysis (DGTDA) which maximizes the scatter difference and estimates the global optimal tensor-to-tensor projection without parameter tuning. In addition, they also propose a Constrained Multilinear Discriminant Analysis (CMDA) by maximizing the scatter ratio and restricting the tensor-to-tensor projection matrices to be orthogonal. In this part, we provide a review on DGTDA and CMDA methods.

In tensor discriminant analysis (TDA), a tensor-to-tensor projection is defined using tensor mode-wise products introduced in section 1.2. Suppose $\mathcal{X}$ is a d-way tensor with size $I_1 \times ... \times I_d$, then a tensor-to-tensor projection transforms $\mathcal{X}$ to $\mathcal{Z} \in \mathbb{R}^{P_1 \times ... \times P_d}$

$$\mathcal{Z} = \mathcal{X} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} ... \times_d \boldsymbol{U}^{(d)} \tag{2.10}$$

where $\boldsymbol{U}^{(j)}$ are $P_j \times I_j$ projection matrices. The projection is defined uniquely by the collection of projection matrices $\{\boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)} ... \boldsymbol{U}^{(d)}\}$. Now let's assume the training data are tensors from binary classes, and are denoted by $\mathcal{X}_{c,i}$. $c = 1, 2$ stands for the class of tensor data, and i stands for the i-th sample from class c. Like traditional statistics, the mean projected tensor for class c is defined as

$$\begin{aligned} \overline{\mathcal{Z}}_c &= \frac{1}{n_c} \sum_{i=1}^{n_c} \mathcal{Z}_{c,i} = \frac{1}{n_c} \sum_{i=1}^{n_c} \mathcal{X}_{c,i} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} ... \times_d \boldsymbol{U}^{(d)} \\ &= \overline{\mathcal{X}}_c \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} ... \times_d \boldsymbol{U}^{(d)} \end{aligned} \tag{2.11}$$

where $\overline{\mathcal{X}}_c = \frac{1}{n_c} \sum\limits_{i=1}^{n_c} \mathcal{X}_{c,i}$ is the class mean of original tensors in class c. n_c is the number of samples in class c. Similarly, the overall mean of tensors from both classes is

$$\begin{aligned} \overline{\mathcal{Z}} &= \frac{1}{n} \sum_{c=1}^{2} n_c \cdot \overline{\mathcal{Z}}_c = \frac{1}{n} \sum_{c=1}^{2} \sum_{i=1}^{n_c} \mathcal{X}_{c,i} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} ... \times_d \boldsymbol{U}^{(d)} \\ &= \overline{\mathcal{X}} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} ... \times_d \boldsymbol{U}^{(d)} \end{aligned} \tag{2.12}$$

where $\overline{\mathcal{X}} = \frac{1}{n} \sum\limits_{c=1}^{2} n_c \cdot \overline{\mathcal{X}}_c$. $n = n_1 + n_2$ is the total number of samples. As an extension of Fisher discriminant analysis, TDA looks at mode-wise between-class scatter matrices and within-class scatter matrices in the projected subspace. For mode j, the between class scatter matrix is defined as

$$
\begin{aligned}
\boldsymbol{B}_j &= \sum_{c=1}^{2} n_c \big[\overline{\mathcal{Z}}_c - \overline{\mathcal{Z}}\big]_{(j)} \cdot \big[\overline{\mathcal{Z}}_c - \overline{\mathcal{Z}}\big]_{(j)}^\top \\
&= \sum_{c=1}^{2} n_c \big[(\overline{\mathcal{X}}_c - \overline{\mathcal{X}}) \prod_k \times_k \boldsymbol{U}^{(k)}\big]_{(j)} \cdot \big[(\overline{\mathcal{X}}_c - \overline{\mathcal{X}}) \prod_k \times_k \boldsymbol{U}^{(k)}\big]_{(j)}^\top \\
&= \boldsymbol{U}^{(j)} \boldsymbol{B}_j^{\bar{j}} \boldsymbol{U}^{(j)\top}
\end{aligned}
\tag{2.13}
$$

$\boldsymbol{B}_j^{\bar{j}} = \sum\limits_{c=1}^{2} n_c \big[(\overline{\mathcal{X}}_c - \overline{\mathcal{X}}) \prod_{k \neq j} \boldsymbol{U}^{(k)}\big]_{(j)} \cdot \big[(\overline{\mathcal{X}}_c - \overline{\mathcal{X}}) \prod_{k \neq j} \times_k \boldsymbol{U}^{(k)}\big]_{(j)}^\top$ is the between-class scatter matrix in the partially projected subspace (all modes excepts j-th mode are projected). $\boldsymbol{B}_j$ is in dimension of $P_j \times P_j$, and $\boldsymbol{B}_j^{\bar{j}}$ is in dimension $P_j \times P_j$ since the j-th mode is not projected. $\big[\overline{\mathcal{Z}}_c - \overline{\mathcal{Z}}\big]_{(j)}$ are the j-th mode unfolding matrices for tensor $\big[\overline{\mathcal{Z}}_c - \overline{\mathcal{Z}}\big]$ which is in dimension $I_j \times \prod_{k \neq j} I_k$. The derivation of equation (2.13) is available in [92]. With the same idea, the mode-j within-class scatter matrices is deinfed as

$$
\begin{aligned}
\boldsymbol{W}_j &= \frac{1}{n} \sum_{c=1}^{2} \sum_{i=1}^{n_c} \big[\mathcal{X}_{c,i} \prod_k \times_k \boldsymbol{U}^{(k)}\big]_{(j)} \big[\mathcal{X}_{c,i} \prod_k \times_k \boldsymbol{U}^{(k)}\big]_{(j)}^\top \\
&= \boldsymbol{U}^{(j)} \boldsymbol{W}_j^{\bar{j}} \boldsymbol{U}^{(j)\top}
\end{aligned}
\tag{2.14}
$$

$\boldsymbol{W}_j^{\bar{j}} = \frac{1}{n} \sum\limits_{c=1}^{2} \sum\limits_{i=1}^{n_c} \big[(\mathcal{X}_{i,c} - \overline{\mathcal{X}}_c) \prod_{k \neq j} \boldsymbol{U}^{(k)}\big]_{(j)} \cdot \big[(\mathcal{X}_{i,c} - \overline{\mathcal{X}}_c) \prod_{k \neq j} \times_k \boldsymbol{U}^{(k)}\big]_{(j)}^\top$ is the within-class scatter matrix in the partially projected subspace (all modes excepts j-th mode are projected). Notice that the within-class matrices (2.14) and between-class scatter matrices (2.13) are analogous to within-class and between-class covariance matrices in traditional statistics. Thus, for each mode, an optimal projection matrix can be estimated by maximizing the ratio of (2.13) and (2.14), or the difference between (2.13) and (2.14).

GDTDA learns the projection matrices by maximizing the scatter difference of (2.13) and (2.14). Additionally, it assumes that all the projection matrices are orthogonal. The objective function for

DGTDA is

$$\begin{aligned}\max_{\boldsymbol{U}^{(j)}} \quad & ||\boldsymbol{B}_j||^2_{\mathbf{Fro}} - \zeta||\boldsymbol{w}_j||^2_{\mathbf{Fro}} \\ & = tr\left\{\boldsymbol{U}^{(j)}\boldsymbol{B}^{\bar{j}}_j\boldsymbol{U}^{(j)\top}\right\} - \zeta tr\left\{\boldsymbol{U}^{(j)}\boldsymbol{W}^{\bar{j}}_j\boldsymbol{U}^{(j)\top}\right\} \\ \text{S.T.} \quad & \boldsymbol{U}^{(j)} \cdot \boldsymbol{U}^{(j)\top} = \boldsymbol{I}\end{aligned} \tag{2.15}$$

For each mode j, the projection matrix can be estimated with singular value decomposition. The whole algorithm estimates $\boldsymbol{U}^{(j)}$ in a single run without multiple iterations and tuning parameter. ζ is selected through singular value decomposition instead of tuning. We summarize this in the algorithm 3. We use SVD in the algorithm to denote ordinary singular value decomposition for matrix. This algorithm is very common and is used directly without introduction.

Algorithm 3 DGTDA Projection Learning

```
1: procedure DGTDA
2:     Input: Tensors $\{\mathcal{X}_{c,i}; i = 1, 2, ..., n_c; c = 1, 2\}$, Target dimension $P_1, P_2..., P_d$
3:
4:     for j = 1 2, ..., d do
5:         Calculate $\overline{\boldsymbol{B}}_j = \sum_{c=1}^{2} n_c\left[(\overline{\mathcal{X}}_c - \overline{\mathcal{X}})\right]_{(j)} \cdot \left[(\overline{\mathcal{X}}_c - \overline{\mathcal{X}})\right]^\top_{(j)}$
6:         Calculate $\overline{\boldsymbol{W}}_j = \frac{1}{n}\sum_{c=1}^{2}\sum_{i=1}^{n_c}\left[\mathcal{X}_{c,i}\prod_k \times_k \boldsymbol{U}^{(k)}\right]_{(j)}\left[\mathcal{X}_{c,i}\prod_k \times_k \boldsymbol{U}^{(k)}\right]^\top_{(j)}$
7:         $[], \boldsymbol{\Sigma}, [] = SVD(\overline{\boldsymbol{W}}_j^{-1} \cdot \overline{\boldsymbol{B}}_j)$    ▹ SVD Decomposition and take singular value only
8:         $\zeta = \max(\text{diag}(\boldsymbol{\Sigma}))$    ▹ Take the maximum singular value
9:         $\boldsymbol{M} = \boldsymbol{B}_j - \zeta \cdot \boldsymbol{W}_j$
10:        $\boldsymbol{U}, [], [] = SVD(\boldsymbol{M})$    ▹ SVD Decomposition and take left singular vectors
11:        $\hat{\boldsymbol{U}}^{(j)} = \boldsymbol{U}[:, 1 : P_j]^\top$    ▹ Takes the first $P_j$ colums and transpose
12:    Return $\left\{\hat{\boldsymbol{U}}^{(j)}, j = 1, ..., d\right\}$
```

Instead of maximizing the scatter difference, CMDA learns the projection matrices by maximizing the ratio of (2.13) and (2.14). Different from the existing work of TDA [147, 103] which also use scatter ratio, CMDA include the orthogonal assumption for projection matrices hoping to make the algorithm converges over iterations. The objective function for CMDA can be defined as

$$\begin{aligned} \max_{\boldsymbol{U}^{(j)}} \quad & \frac{||\boldsymbol{B}_j||^2_{\textbf{Fro}}}{||\boldsymbol{W}_j||^2_{\textbf{Fro}}} = \frac{tr\{\boldsymbol{U}^{(j)}\boldsymbol{B}_j^{\bar{j}}\boldsymbol{U}^{(j)\top}\}}{tr\{\boldsymbol{U}^{(j)}\boldsymbol{W}_j^{\bar{j}}\boldsymbol{U}^{(j)\top}\}} \\ \text{S.T.} \quad & \boldsymbol{U}^{(j)} \cdot \boldsymbol{U}^{(j)\top} = \boldsymbol{I} \end{aligned} \tag{2.16}$$

For each mode j, the projection matrix can be estimated with singular value decomposition as well. The whole algorithm estimates $\boldsymbol{U}^{(j)}$ over iterations until all the estimated projection matrices are approximately orthogonal. We summarize this in the algorithm 4.

Algorithm 4 CMDA Projection Learning

```
1:  procedure CMDA
2:      Input: Tensors $\{\mathcal{X}_{c,i}; i = 1, 2, ..., n_c; c = 1, 2\}$, Target dimension $P_1, P_2..., P_d$, $\eta$, maxitor
3:      Initialize $\boldsymbol{U}_0^{(j)} = \mathbf{1}; j = 1, .., d$        ▹ Initialize value, matrix with values all equal to 1
4:      while $t \leqslant$ maxiter do                                        ▹ Repeat up to max iteration
5:          for j = 1 2, 3, ...,d do
6:              $\boldsymbol{B}_{j,t}^{\bar{j}} = \sum_{c=1}^{2} n_c \left[(\overline{\mathcal{X}}_c - \overline{\mathcal{X}}) \prod_{k \neq j} \boldsymbol{U}^{(k)}\right]_{(j)} \cdot \left[(\overline{\mathcal{X}}_c - \overline{\mathcal{X}}) \prod_{k \neq j} \times_k \boldsymbol{U}^{(k)}\right]_{(j)}^{\top}$
7:              $\boldsymbol{W}_{j,t}^{\bar{j}} = \frac{1}{n} \sum_{c=1}^{2} \sum_{i=1}^{n_c} \left[(\mathcal{X}_{i,c} - \overline{\mathcal{X}}_c) \prod_{k \neq j} \boldsymbol{U}^{(k)}\right]_{(j)} \cdot \left[(\mathcal{X}_{i,c} - \overline{\mathcal{X}}_c) \prod_{k \neq j} \times_k \boldsymbol{U}^{(k)}\right]_{(j)}^{\top}$
8:              $\boldsymbol{U}, [], [] = SVD(\boldsymbol{W}_{j,t}^{\bar{j}-1} \cdot \boldsymbol{B}_{j,t}^{\bar{j}})$                   ▹ SVD
9:              $\hat{U}_t^{(j)} = \boldsymbol{U}[:, 1 : P_j]^{\top}$            ▹ Takes the first $P_j$ colums and transpose
10:         Check $err(t) = \sum_{j=1}^{d} ||\hat{U}_t^{(j)} \cdot \hat{U}_t^{(j)\top} - \boldsymbol{I}||_{\text{Fro}} \leqslant \eta$
11:         if $err(t) \leqslant \eta$ then
12:             Stop Iteration                                             ▹ Quit loop
13:         $t = t + 1$
14:     Return $\{\hat{\boldsymbol{U}}^{(j)}, j = 1, ..., d\}$
```

The last step in the TDA classification is assigning class labels to new test points. For both DGTDA and CMDA, a tensor-based K-nearest neighbour method is adopted for this purpose. Recall the Frobenius norm for tensor in section 1.2, one can define distance between two tensors with Frobenius norm. For any two tensors $\mathcal{X}_1$ and $\mathcal{X}_2$ in the same dimension, the distance is defined as $dis(\mathcal{X}_1, \mathcal{X}_2) = ||\mathcal{X}_1 - \mathcal{X}_2||_{\text{Fro}}$. A K-Nearest Neighbour classifier can use distance and predict class labels for test point. We combine the subspace learning from DGTDA and CMDA algorithm with this KNN classifier, and summarize the whole classification procedure for DGTDA and CMDA in algorithm 5 at the end of this part.

Algorithm 5 Tensor Discriminant Analysis Classification

```
1: procedure TDA
2:     Input: Training set $T_n = \{\mathcal{X}_{c,i}; i = 1, 2, ..., n_c; c = 1, 2\}$, Labels $\{y_1, .., y_n\} \in \{0, 1\}$, Target dimension $P_1, P_2..., P_d$, Test set $\{\mathcal{X}_1^*, ...\mathcal{X}_m^*\}$,$\eta$, maxitor, k
3:     if DGTDA then
4:         $\{\boldsymbol{U}^{(d)}, ..., \boldsymbol{U}^{(d)}\}$ = DGTDA($\{\mathcal{X}_{c,i}; i = 1, 2, ..., n_c; c = 1, 2\}$, $P_1, P_2..., P_d$)
5:     else
6:         $\{\boldsymbol{U}^{(d)}, ..., \boldsymbol{U}^{(d)}\}$ = CMDA($\{\mathcal{X}_{c,i}; i = 1, 2, ..., n_c; c = 1, 2\}$, $P_1, P_2..., P_d, \eta$, maxitor)
7:     for i = 1,..n do
8:         $\mathcal{Z}_{c,i} = \mathcal{X}_{c,i} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} ... \times_d \boldsymbol{U}^{(d)}$
9:     for i = 1,.., m do
10:        $\mathcal{Z}_i^* = \mathcal{X}_i^* \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} ... \times_d \boldsymbol{U}^{(d)}$
11:        $\boldsymbol{d} = [dis(\mathcal{Z}_1, \mathcal{Z}_i^*), ..., dis(\mathcal{Z}_n, \mathcal{Z}_i^*)]$
12:        $\boldsymbol{d'} = \arg sort(d)$        ▹ Sort distance in increasing order, and return the index
13:        $y_i^* = 1\{\frac{1}{k}\sum_{l=1}^{k} y[\boldsymbol{d'}[l]] \geqslant 0\}$
14:    Return $y_1^*, ..., y_m^*$
```

2.2.3 Tensor Regression

The last type of tensor classification model we want to review in this part is tensor regression model. Tensor regression is a collection of statistical models taking tensor-shape predictors. These models can predict tensor or scalar response from the tensor covariates. [155, 93] propose tensor generalized linear regression model by assuming CP and Tucker decomposition structures on regression coefficients. [62] introduces sparse penalty on tensor CP regression models to provide an efficient and scalable model for unit-rank tensor regression problems. More recently, tensor response regression [88, 152, 99], Bayesian tensor regression [57], and tensor regression with variation norm penalty [45] are also developed. In this part, we review the CP tensor generalized linear regression model (CP-GLM) [155] for tensor classification problems.

CP-GLM assumes a regression model with tensor covariates. Let $\boldsymbol{g}(\cdot)$ be the link function, $\mathcal{X} \in \mathbb{R}^{I_1 \times .. \times I_d}$ be a d-way tensor, and $\mathcal{B} \in \mathbb{R}^{I_1 \times .. \times I_d}$ be the tensor coefficient. The CP-GLM with scalar response is defined as

$$\boldsymbol{g}(\mu) = \alpha + \boldsymbol{\gamma}^\top \boldsymbol{z} + < \mathfrak{B}, \mathfrak{X} > \tag{2.17}$$

γ is a scalar, and $\boldsymbol{z} \in \mathbb{R}^p$ is a p dimensional vector predictor. $\boldsymbol{\gamma}$ is a p dimensional vector coefficient as well. If $\mathfrak{B}$ has a low-rank CP decomposition, then we can use Kruskal tensor to denote it as $\mathfrak{U}_{\mathfrak{B}} = [\![\boldsymbol{B}^{(1)}, ..., \boldsymbol{B}^{(d)}]\!]$. Each $\boldsymbol{B}^{(j)}, j = 1, .., d$ is a I_j by r matrix whose columns are CP factors of tensor $\mathfrak{B}$. The rank ofr $\mathfrak{B}$ is assumed to be r. With Khatri-rao product, the inner product between tensor coefficient and predictor can be written as

$$\begin{aligned} < \mathfrak{U}_{\mathfrak{B}}, \mathfrak{X} > &=< \boldsymbol{B}^{(j)} \big(\boldsymbol{B}^{(d)} \odot ..\boldsymbol{B}^{(j+1)} \odot \boldsymbol{B}^{(j-1)} ..., \odot \boldsymbol{B}^{(1)}\big)^\top, \mathfrak{X} > \\ &=< \boldsymbol{B}^{(j)}, \mathfrak{X}\big(\boldsymbol{B}^{(d)} \odot ..\boldsymbol{B}^{(j+1)} \odot \boldsymbol{B}^{(j-1)} ..., \odot \boldsymbol{B}^{(1)}\big) > \end{aligned} \tag{2.18}$$

for j-th mode CP components. If we write the inner product into a vector form, then $\boldsymbol{B}^{(j)}$ can be estimated with regular maximum likelihood estimate (MLE) method by fixing α, $\boldsymbol{\gamma}$, and all other $\boldsymbol{B}^{(k)}, k \neq j$. A iterative MLE can be adopted to estimate all the CP compoent matrices as well as the regular scalar and vector coefficients in model (2.17). If we denote the likelihood function as $\boldsymbol{\ell}(\alpha, \boldsymbol{\gamma}, \mathfrak{U}_{\mathfrak{B}})$, which can be derived in the exactly same ways as ordinary GLM, the iterative MLE algorithm can be summarized in the algorithm 6. In tensor binary classification problems, we can

Algorithm 6 Tensor CP Generalized Linear Model

```
 1: procedure TDA
 2:     Input: $\{\mathfrak{X}_1, ...\mathfrak{X}_n\}, \{\boldsymbol{z}_1, ...\boldsymbol{z}_n\}, \boldsymbol{y}, \eta$, maxitor
 3:     Initialize $\mathfrak{U}_{\mathfrak{B},0} = [\![\mathbf{0}, ..\mathbf{0}]\!]$          ▷ Initialize Kruskal tensor as 0 matrices
 4:     Initialize $(\alpha_0, \boldsymbol{z}_0) = \arg\max \boldsymbol{\ell}(\alpha, \boldsymbol{\gamma}, \mathfrak{U}_{\mathfrak{B},0})$
 5:     while t < maxitor do
 6:         for j = 1, ..., d do
 7:             $\boldsymbol{B}^{(j)}_{t+1} = \arg\max \boldsymbol{\ell}(\alpha_t, \boldsymbol{\gamma}_t, [\![\boldsymbol{B}^{(1)}_{t+1}....\boldsymbol{B}^{(j-1)}_{t+1}, \boldsymbol{B}^{(j)}, \boldsymbol{B}^{(j+1)}_t ..., \boldsymbol{B}^{(d)}_t]\!])$
 8:         $(\alpha_{t+1}, \boldsymbol{z}_{t+1}) = \arg\max \boldsymbol{\ell}(\alpha, \boldsymbol{\gamma}, [\![\boldsymbol{B}^{(1)}_{t+1}....\boldsymbol{B}^{(j-1)}_{t+1}, \boldsymbol{B}^{(j)}_{t+1}, \boldsymbol{B}^{(j+1)}_t ..., \boldsymbol{B}^{(d)}_{t+1}]\!])$
 9:         if $\boldsymbol{\ell}_{t+1} - \boldsymbol{\ell}_t \leqslant \eta$ then
10:             Stop
11:         $t = t + 1$
12:     Output: $\alpha, \boldsymbol{\gamma}, [\![\boldsymbol{B}^{(1)}....\boldsymbol{B}^{(j-1)}, \boldsymbol{B}^{(j)}_{t+1} \boldsymbol{B}^{(j+1)}_t ..., \boldsymbol{B}^{(d)}]\!]$
```

take the link function $\boldsymbol{g}$ as the logit function, and predict the probability that $\mathbb{P}(y = 1|\mathfrak{X})$. The class labels are predicted by doing a threshold on the predicted probability.

2.3 Statistical Analysis

As we mentioned in section 1.3, statistical consistency is one of the most important properties for decision rules as it demonstrates generalization ability of rules from the aspect of prediction risk. The consistency of tensor discriminant analysis can be easy established since DGTDA and CMDA are both converging ([92]) algorithms providing unique tensor-to-tensor projections. In addition, tensor K-nearest neighbour classifier is equivalent to vector K-nearest neighbour, which is consistent ([36]). These two facts make the tensor discriminant analysis a consistent classifier. Tensor CP-GLM models the conditional probabilities for data labels through regression model. Its consistency is guaranteed by the (strong) consistency of regression coefficients ([155]). In this section, we provide few theoretical helping to establish the statistical consistency for CP-STM. The section contains two parts for theory development. One is the universal property establishment for CP-STM kernel functions, the other is consistency proof for CP-STM.

2.3.1 Universal Tensor Kernels

Our first result is about the universal property of tensor kernel functions. Kernel Universal property plays a very important role in kernel learning methods such as support vector machine and kernel regression [107]. Since kernel-based learning methods always estimate optimal solution from Reproducing Kernel Hlibert Space (RKHS), the error of approximating the complete functional space $\{\boldsymbol{f} : \boldsymbol{\mathcal{X}} \to \boldsymbol{\mathcal{Y}}, \boldsymbol{f} \text{ measureable}\}$ with RKHS is critical in the generalization ability of the learned rules. In other word, if the approximation error of RKHS is larger, then the prediction from kernel-based methods will be more biased. Kernel with universal property guarantees that the approximation error of RKHS can be as small as possible on any compact subspace of the input space. To present our result about universal property for tensor kernels, we first provide a formal definition about the universal property.

Definition 2.3.1. *Let $K(\cdot,\cdot)$ be a continuous kernel function defined on $\boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{X}} \to \mathbb{R}$. Given a compact subspace $\boldsymbol{\mathcal{Z}} \subset \boldsymbol{\mathcal{X}}$ from the input space, the kernel section of $K(\cdot,\cdot)$ over $\boldsymbol{\mathcal{Z}}$ is $\boldsymbol{\mathcal{K}}(\boldsymbol{\mathcal{Z}}) :=$*

$\overline{span}\{K_x, x \in \mathcal{Z}\}$, which is a RKHS generated by kernel $K(\cdot,\cdot)$ and subspace $\mathcal{Z}$. If for any continuous function $\boldsymbol{f} : \mathcal{Z} \to \mathbb{R}$, there is a positive number $\epsilon > 0$ and a function $\boldsymbol{g} \in \boldsymbol{\mathcal{K}}(\mathcal{Z})$ such that

$$||\boldsymbol{f} - \boldsymbol{g}||_\infty = \sup_{x \in \mathcal{Z}} |\boldsymbol{f}(x) - \boldsymbol{g}(x)| \leqslant \epsilon$$

then $K(\cdot,\cdot)$ has universal approximating property and is called a universal kernel.

With universal kernels, we can immediately see that the optimal function estimated from RKHS is also the optimal among all measureable functions. This is critical in the establishment of CP-STM consistency. Thus, our first result shows that the kernel function in CP-STM can be universal.

Proposition 2.3.1. *For a d-way CP tensor kernel function*

$$K(\mathcal{X}_1, \mathcal{X}_2) = \sum_{l,k=1}^{r} \prod_{j=1}^{d} K^{(j)}(\boldsymbol{x}_{1,l}^{(j)}, \boldsymbol{x}_{2,k}^{(j)})$$

with $\mathcal{X}_1 = \sum_{k=1}^{r} \boldsymbol{x}_{1,k}^{(1)} \circ \boldsymbol{x}_{1,k}^{(2)} ... \circ \boldsymbol{x}_{1,k}^{(d)}$ and $\mathcal{X}_2 = \sum_{k=1}^{r} \boldsymbol{x}_{2,k}^{(1)} \circ \boldsymbol{x}_{2,k}^{(2)} ... \circ \boldsymbol{x}_{2,k}^{(d)}$. If all mode-wise kernel functions $K^{(j)}(\cdot,\cdot)$ are satisfying the universal approximation property in definition 2.3.1, then it also satisfy the universal approximating property in the sense that for all continuous function defined on the compact tensor product space $\boldsymbol{\mathcal{X}} = \otimes_{j=1}^{d} \boldsymbol{\mathcal{V}}^{(j)}$, there exits a function $\boldsymbol{g} \in \boldsymbol{\mathcal{K}}(\boldsymbol{\mathcal{X}})$ in the tensor kernel section such that

$$||\boldsymbol{f} - \boldsymbol{g}||_\infty = \sup_{\mathcal{X} \in \boldsymbol{\mathcal{X}}} |\boldsymbol{f}(\mathcal{X}) - \boldsymbol{g}(\mathcal{X})| \leqslant \epsilon$$

The proof of this proposition is provided in the appendix A.1. Notice that since distance-based kernel functions such as Gaussian RBF and polynomial are universal ([107]), we can use one of it or both for all $K^{(j)}(\cdot,\cdot)$ to create universal tensor kernel functions.

2.3.2 Consistency of CP-STM

With universal tensor kernels, the classification consistency of CP-STM is established with the following theorem. The notations of classification risks are borrowed from section 1.3.

Theorem 2.3.1. *Let $\{\boldsymbol{f_n} : n \in \mathbb{N}\}$ be a sequence of CP-STM classifiers in equation (2.4), which are estimated from different training sets T_n with size n. $\mathcal{R}^*$ is the Bayes risk of tensor binary classification problem for data from the joint distribution $\boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}$. $\boldsymbol{\mathcal{X}}$ is a d-way tensor product space with dimension $I_1 \times I_2 \times ... \times I_d$ defined in definition 1.2.1, and $\boldsymbol{\mathcal{Y}} = \{-1, 1\}$. The CP-STM decision rule $\{\boldsymbol{f_n} : n \in \mathbb{N}\}$ is statistically consistent and the excess classification risk of $\boldsymbol{f_n}$, $\mathcal{R}(\boldsymbol{f_n})$, converges to the optimal Bayes risk*

$$\mathcal{R}(\boldsymbol{f_n}) \rightarrow \mathcal{R}^* \quad (n \rightarrow \infty)$$

If the following conditions are satisfied:

Con.1 *$\boldsymbol{\mathcal{X}}$ is a compact subspace of $\mathbb{R}^{I_1 \times I_2 \times ... \times I_d}$ such that there is a constant $0 < B_x < \infty$. For all $\mathfrak{X} \in \boldsymbol{\mathcal{X}}$ and $\mathfrak{X} = \sum_{k=1}^{r} \boldsymbol{x}_k^{(1)} \circ \boldsymbol{x}_k^{(2)} ... \circ \boldsymbol{x}_k^{(d)}$, $||\boldsymbol{x}_k^{(j)}||^2 \leqslant B_x < \infty$.*

Con.2 *The loss function $\mathcal{L}$ is self-calibrated (see [128]), and is $C(W)$ local Lipschitz continuous in the sense that for $|a| \leqslant W < \infty$ and $|b| \leqslant W < \infty$*

$$|\mathcal{L}(a, y) - \mathcal{L}(b, y)| \leqslant C(W)|a - b|$$

In addition, the loss function is bounded on the second variable, i.e.

$$\sup_{y \in \{1,-1\}} \mathcal{L}(0, y) \leqslant L_0 < \infty$$

.

Con.3 *The kernel functions $K^{(j)}(\cdot, \cdot)$ used to composite the coupled tensor kernel (2.2) are regular vector-based kernels satisfying the universal approximating property in definition 2.3.1. Additionally, they are all bounded so that there is a constant $0 < K_{max} < \infty$, $\sup \sqrt{K^{(j)}(\cdot, \cdot)} \leqslant K_{max}$ for all $j = 1, .., d$.*

Con.4 *The hyper-parameter in the objective function (2.1) $\lambda = \lambda_n$ satisfies:*

$$\lambda_n \rightarrow 0 \quad n\lambda_n \rightarrow \infty \quad as \quad n \rightarrow \infty$$

The proof of this theorem is provided in the appendix A.2. At the end of this section, we can conclude that all the tensor-based classifiers reviewed in this chapter are statistically consistent, meaning that they all have promising prediction accuracy if certain conditions are satisfied. However, their performance in practice can be of various, since their excess risks may converge at different rates depending on the data distribution in specific applications. In next section, we compare the performance of these classification methods with two real data applications.

2.4 Real Data Analysis

In this section, we provide two examples in Neuroimaging and Computer Vision studies, and apply all the reviewed tensor-based classifier in imaging classification problems.

2.4.1 MRI Classification for Alzheimer's Disease

Alzheimer's Disease (AD) is a progressive, irreversible loss of brain function which would impact memory, thinking, language, judgment, and behavior. The disease could destroy patients' memory and thinking ability, and eventually make it difficult for patients to carry out even the simplest tasks of daily living. In the research of AD, there are lots of novel technologies developed to collect information from patients for diagnostic purposes, which include genetic analysis, biological and neurological tests, Magnetic Resonance Imaging (MRI), and Positron Emission Tomography (PET) imaging. Utilizing these information to predict patients' biological status is of great interests in early detection and biomarker development for Alzheimer's Disease. In this study, we consider using patients' voxel MRI data to predict if patients are early AD or Normal Coherent (NC).

Brain MRI technology collects 3D image to show anatomical structure of brains. The data is measured in voxels, which are similar to pixels used in displaying regular images. Voxels are however cuboids in 3D having specific dimensions. Each voxel contains one value standing for the average signal measured at a fixed position. A standard MRI image, also called volume, is arranged in a 3D array to reflect brain structure by placing all voxels in their corresponding positions. Thus, voxel-level MRI images are multi-dimensional arrays loaded from Neuroimaging

	AD	NC
Num Subjects	183	219
Age (Mean ± sd)	75.28 ± 7.55	75.80 ± 4.98
Gender (Female / Male)	88 / 95	110 / 109
MMSE (Mean ± sd)	23.28 ± 2.04	29.11 ± 1.00

Table 2.1: Biological Information for Subjects in ADNI Study; MMSE: baseline Mini-Mental State Examination

Informatics Technology Initiative (Nifti) files (see [85]), and are tensors in nature. We can use our proposed tensor-based classifier to handle voxel-level MRI data.

We collect data from Alzheimer's Disease Neuroimaging Initiative ADNI. It is a huge longitudinal study on Alzheimer's Disease. In ADNI, the structural brain MRI is believed to be highly correlated to the patients' cognition, thus can be utilized to predict patients' status, AD vs. NC. The MRI data is collected from the screening session in ADNI-1. During the session, there are 818 patients selected to enter the study and received 1.5T MRI scan. These images are pre-processed by ADNI with normalization and bias correction, and are provided in the ADNI-1.5T Screening standardized data set. The data set includes both MRI scans and patients' dementia status labeled as Normal Coherent(NC), Mild Cognition Impaired (MCI), and Alzheimer's Disease (AD). In this study, we are especially interested about predicting if a patient has AD or NC. Thus, we only collect image data from NC and AD patients. The biological information about AD group and NC group is provided in the table 2.1. MMSE in the table stands for Mini-Mental State Examination, which is a test of cognition functions for patients with dementia. The MMSE in the table 2.1 shows the scores of MMSE test for subjects.

There are 402 MRI images in the data collection. We further use Matlab Imaging Processing Toolbox to register and align all the images for classification and comparison. We also use image resize function in the toolbox to unify the voxel dimensions in all MRI images to be 6mm by 6mm by 9mm. After resizing, all images are in the shape of 40 by 40 by 21. This step is necessary since ADNI MRI images are acquired from multiple sites. Resizing guarantees all the images can be represented by tensors in the same size. The choice of such image size is referred from other

Models	Accuracy	Precision	Sensitivity	Specificity	AUC
CP-GLM	$0.58_{0.04}$	$0.58_{0.07}$	$1.00_{0.00}$	$0.00_{0.00}$	$0.50_{0.00}$
CMDA	$0.70_{0.03}$	$0.69_{0.05}$	$0.67_{0.09}$	$0.73_{0.10}$	$0.65_{0.17}$
DGTDA	$0.70_{0.02}$	$0.71_{0.02}$	$0.59_{0.06}$	$0.80_{0.01}$	$0.64_{0.18}$
CP-STM1	$0.73_{0.03}$	$1.00_{0.00}$	$0.41_{0.07}$	$1.00_{0.00}$	$0.64_{0.20}$
CP-STM2	$\mathbf{0.74}_{0.04}$	$1.00_{0.00}$	$0.43_{0.08}$	$1.00_{0.00}$	$0.65_{0.20}$

Table 2.2: Real Data: ADNI Classification Comparison I

similar statistical analysis works such as [155, 114, 45].

We conduct our numerical experiment in the same protocol as the simulation study. We randomly sample 80% of images from AD group and 80% from NC group to form the training set with size 321. AD is labeled as positive class, and NC is labeled as negative class. The rest images are used as test set to evaluate model performance. For each classification model, we evaluate its performance by calculating its accuracy, precision, sensitivity, and specificity on the test set. Such step is replicated for multiple times, and the average accuracy, precision, sensitivity, and specificity are reported in the table in percentages. The standard deviation of these performance metrics are also provided (in subscripts) in the parenthesis. In the table 2.2, we use CP-STM1 to denote CP-STM with Hinge loss, and CP-STM2 to denote CP-STM with Square Hinge loss. Figure 2.1 summarizes the comparison in a more illustrative way. The area under the curves (AUC) of ROC curves are reported in the table 2.2 as well.

The results in table 2.2 shows that all the tensor-based classifiers have close accuracy in prediction, while CP-STM is slightly better than the others. Also, these methods all have pretty low sensitivity, indicating that the chances of correctly detecting AD patients are pretty low. The performance of CP-GLM is much worse than the others, which may due to the fact that its parameteric form and data distribution assumption are not appropriate for the real data. Comparing two CP-STMs, we notice that one with Squared Hinge loss is better, which may due to the fact that Squared Hinge loss has a bigger penalty on points which strongly violates the margin during the training procedure.

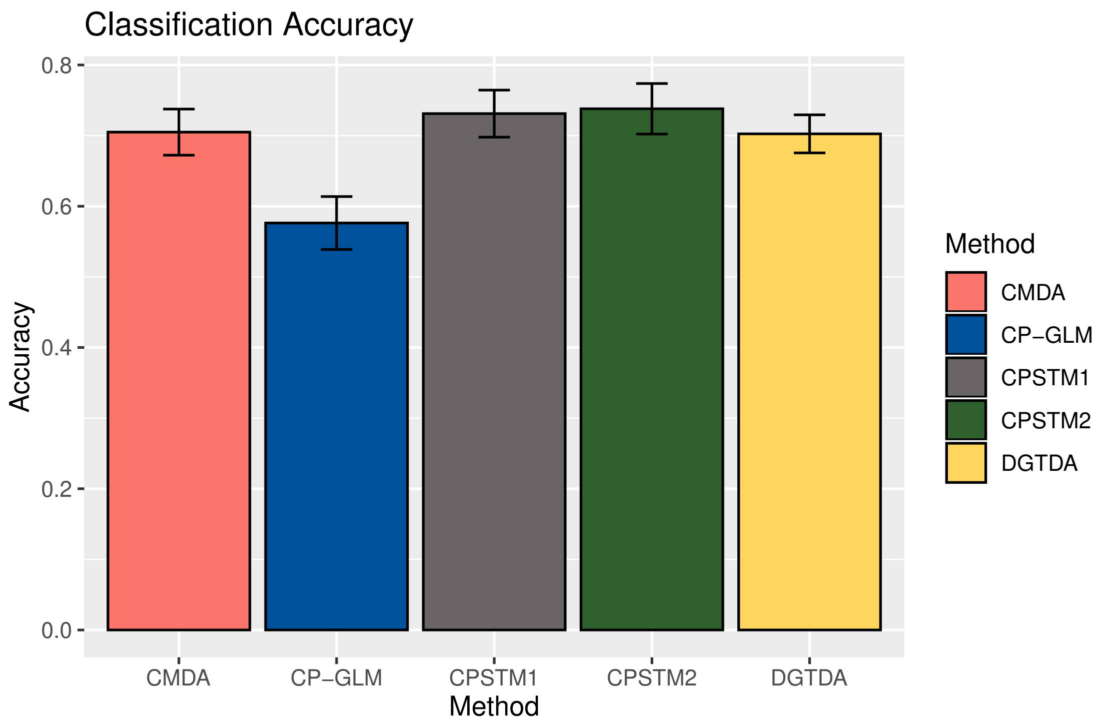


Figure 2.1: Real Data: ADNI Classification Reults I

2.4.2 KITTI Traffic Images

The second application we conduct is traffic image data recognition. Traffic image data recognition is an important computer vision problem. We considered the image data from the KITTI Vision Benchmark Suit. [53], [49], and [52] provided a detailed description and some preliminary studies about the data set. In this application, a 2D object detection task asks us to recognize different objects pointed out by bounding boxes in pictures captured by a camera on streets. There are various types of objects in the pictures, most of which are pedestrians and cars. We selected images containing only pedestrians or cars to test the performance of our classifier.

The first step we did before training our classifier is image pre-processing, which includes cropping the images and dividing them into different categories. We picked patterns indicated by bounding boxes from images and smoothed them into a uniform dimension $224 \times 224 \times 4$. Then we transform all these colored images into grey-scale to drop color information in order to avoid potential problems caused by the extreme dimension imbalance among the three different modes.

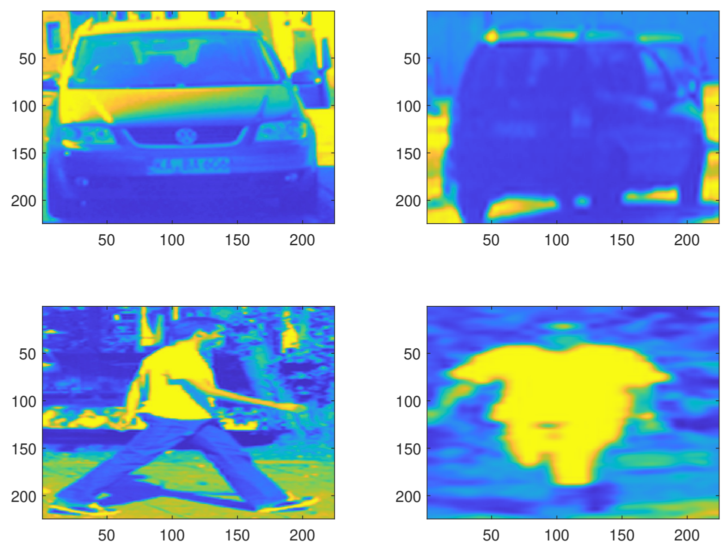


Figure 2.2: Real Data: Examples of Traffic Objects in KITTI Data

The processed images are in size 224 by 224 which can be modeled by two-mode tensors. Figure 2.2 shows few examples of processed images for cars and pedestrians.

The total number of images are 33229, among which 4487 are car images and 28742 are pedestrian images. Next, we divide these images into three groups basing on their qualities and visibility. Images having more than 40 pixels in height and fully visible will go to the easy group. Partly visible images having 25 pixels or more in height are in the moderate group. Those images which are difficult to see with bare eyes are going to the hard group. These three groups of images are utilized to define three different classification tasks with levels of difficulties as easy, moderate, and hard. To overcome the class imbalance in all the three groups of images, We randomly select 200 car images and 200 pedestrian images to form a balanced data set in each group for our numerical experiments. Pedestrian images are considered as the positive class data, and car images are negative class data.

The following procedures are repeated for 50 times in all three tasks with the balanced data sets. We randomly sample 80% of images as training and validation set. Classification models are estimated and tuning parameters (if any) are selected using this part of the data. Then the models with selected tuning parameters are applied on the rest 20% data for testing. The sampling is conducted in a stratified way so that the proportion of pedestrian and car images are approximately same in both training and testing set. For each repetition, we calculate the same performance metrics, accuracy rates, precision (positive predictive rates), sensitivity (true positive rates), and specificity (true negative rates), for each classification method using the testing set. The average value of these rates and their standard deviations (in subscripts) are reported in the table 2.3. The areas under the ROC curves (AUC) are also reported for all the methods. All the classifiers reviewed in section 2.2 are included, and are denoted with the same notations from the previous ADNI application. The comparison of model prediction accuracy rates is also illustrated by the figure 2.3, in which accuracy rates are shown by bar charts and the standard deviations of accuracy rates are shown by error bars.

Task	Methods	Accuracy	Precision	Sensitivity	Specificity	AUC
	CP-STM1	$\mathbf{0.85}_{0.03}$	$0.84_{0.05}$	$0.85_{0.05}$	$0.84_{0.06}$	$\mathbf{0.85}_{0.03}$
	CP-STM2	$0.83_{0.04}$	$0.83_{0.05}$	$0.83_{0.05}$	$0.83_{0.06}$	$0.83_{0.04}$
	CMDA	$0.63_{0.07}$	$0.58_{0.06}$	$0.95_{0.08}$	$0.30_{0.16}$	$0.63_{0.07}$
Easy	DGTDA	$0.84_{0.04}$	$0.77_{0.04}$	$0.96_{0.03}$	$0.72_{0.07}$	$0.84_{0.04}$
	CP-GLM	$0.57_{0.05}$	$0.57_{0.06}$	$0.59_{0.07}$	$0.55_{0.09}$	$0.57_{0.05}$
	CP-STM1	$\mathbf{0.78}_{0.05}$	$0.78_{0.06}$	$0.77_{0.07}$	$0.78_{0.07}$	$\mathbf{0.78}_{0.05}$
	CP-STM2	$0.73_{0.06}$	$0.75_{0.07}$	$0.72_{0.08}$	$0.75_{0.09}$	$0.73_{0.06}$
	CMDA	$0.59_{0.05}$	$0.55_{0.04}$	$0.89_{0.11}$	$0.28_{0.13}$	$0.59_{0.05}$
Moderate	DGTDA	$0.74_{0.06}$	$0.72_{0.06}$	$0.79_{0.08}$	$0.69_{0.08}$	$0.74_{0.05}$
	CP-GLM	$0.53_{0.05}$	$0.53_{0.05}$	$0.54_{0.07}$	$0.52_{0.08}$	$0.53_{0.05}$
	CP-STM1	$\mathbf{0.76}_{0.04}$	$0.84_{0.06}$	$0.64_{0.07}$	$0.87_{0.05}$	$\mathbf{0.76}_{0.04}$
	CP-STM2	$0.74_{0.04}$	$0.80_{0.06}$	$0.63_{0.07}$	$0.84_{0.06}$	$0.74_{0.04}$
	CMDA	$0.53_{0.04}$	$0.52_{0.02}$	$0.91_{0.09}$	$0.16_{0.12}$	$0.53_{0.04}$
Hard	DGTDA	$0.72_{0.05}$	$0.68_{0.04}$	$0.84_{0.06}$	$0.60_{0.08}$	$0.72_{0.05}$
	CP-GLM	$0.51_{0.06}$	$0.51_{0.06}$	$0.54_{0.07}$	$0.49_{0.07}$	$0.51_{0.06}$

Table 2.3: Real Data: Traffic Image Classification I

By comparing the prediction accuracy rates and AUC values, CP-STM models have significant advantages in classification performance than other tensor-based classification models. Different from previous study, CP-STM with Hinge loss outperforms CP-STM with Squared Hinge loss. The

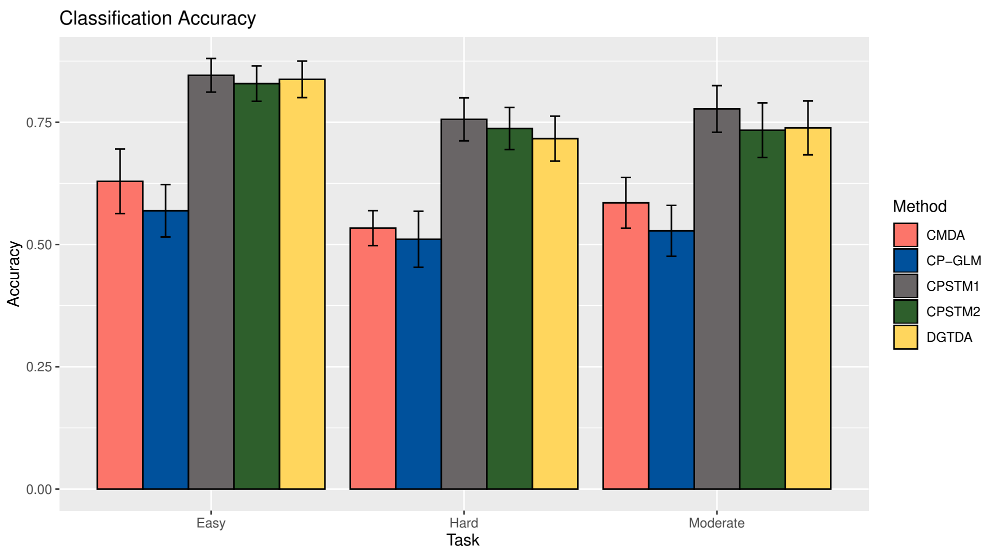


Figure 2.3: Real Data: Traffic Classification Result I

reason for this might be that there are more tensors sitting close to the margin and weakly violate the margin, i.e. $y_i \boldsymbol{f_n}(\mathcal{X}_i) < 1$ and $y_i \boldsymbol{f_n}(\mathcal{X}_i) \approx 1$. As a result, Hinge loss penalizes these points more than Squared Hinge loss in the model estimation procedure, providing a better decision function. Comparing to our previous results in ADNI study, we can conclude that the performance of CP-STM with Hinge ans Squared Hinge loss often time depend on the data distribution. Two tensor discriminant analysis have different performance in this application. DGTDA outperforms CMDA with much higher accuracy rates in all three tasks. Particularly, the accuracy rate of DGTDA is only 1% less than CP-STM1 in the easy task. As for CMDA, it turns out that it fails to identify a discriminantive tensor-to-tensor projection in this application. The performance of CP-GLM is not as good as others, which is similar to the results in our ADNI study.

2.5 Conclusion

In this chapter, we explore the possibility of using tensors to model multi-dimensional and structured data, and reviewed few tensor-based classification models. These models utilize tensor algebraic structures and extend traditional classification methods for tensor data. CP-STM and CP-GLM are extension of Support Vector Machine and Generalized Linear Regression with uses tensor CP decomposition. DGTDA and CMDA are generalization of Fisher Discriminant analysis for tensor using tensor subspace learning. Such subspace learning is indeed a variant of tensor Tucker decomposition. These models can be attractive when handling multidimensional data with various structures.

As a part of our contribution, we develop the statistical consistency result for CP-STM. All these tensor-based methods are then can be considered as consistent decision rules with nice generalization ability. Our data experiments also provide some empirical evidence on the model performance. Through our numerical study, CP-STM show the best prediction accuracy in both applications regardless of data distribution. However, the performance comparison between CP-STM with Hinge and Squared Hinge loss often time depends on the data distribution. In comparison to CP-STM, CP-GLM is a little bit restrictive due to its parametric form, and sometimes fail to approximate the true data distribution. DGTDA and CMDA are both non-parametric and are flexible enough to classify different types of tensors. However, CMDA sometimes will fail to converge and its results then become bad.

Inspired by these existing works, one possible direction for future work can be combining more advanced tensor representation and operations with traditional non-parametric classification models for novel tensor classification models. For example, there are various coupled matrix-tensor decomposition methods established in recent research for multimodal data integration and heterogeneous data analysis. Those decomposition methods can be adopted and help to extend CP-STM for multimodal data classification problems. Besides that, tensor compression via random projection or sketch are also popular in multidimensional big data analysis which aims to provide efficient and scalable ways to process tensors in huge sizes. Coin these tensor compression methods with CP-STM be a great potential for big tensor data classification.

CHAPTER 3

TEC: TENSOR ENSEMBLE CLASSIFIER FOR BIG DATA

In this chapter, we consider classification problems for gigantic size multi-dimensional data. Although tensor-based classification methods mentioned in the previous chapter can analyze multi-dimensional data and preserve data structures, they may face more challenges such as long processing time and insufficient computer memory when dealing with big tensor data. Previously we have demonstrated the distribution-free and statistically consistent properties for the CP-STM model, and highlighted its great potential in successfully handling wide varieties of data applications. However, training a CP-STM can be computationally expensive with high-dimensional tensors. To make it feasible for CP-STM to handle large size tensors, we introduce a tensor-shaped random projection technique, and combine it with CP-STM to reduce the computational time and cost for large tensors. The CP-STM estimated with randomly projected tensors is named as Random Projection-based Support Tensor Machine (RPSTM). We further develop a Tensor Ensemble Classifier (TEC) by aggregating multiple RPSTMs to control the excess classification risk brought by random projections. We demonstrate that TEC can balance between the computational costs and excess classification risk, and provide descent performance in numerical studies.

3.1 Introduction

With the advancement of information and engineering technology, modern-day data science problems often come with data in gigantic size and increased complexities. These complexities are often reflected by huge dimensionality and multi-way features in the observed data such as high-resolution brain imaging and spatio-temporal data. Classification problems with such high-dimensional multi-way data raise more challenges to scientists on how to process the gigantic size data while preserving their structures. Even though there are many established studies in the literature that handle the multi-way data structures with tensor representation and tensor-based models [136, 103, 63, 92, 127, 155, 114], the high dimensionality issue for tensors are rarely explored especially for classification problems. High-dimensional tensors, though are already in multi-dimensional structure, can still have huge dimensionalities in different modes. The existences of high-dimensional tensors could make the current tensor-based classification models fail to provide reliable results due to extremely long processing time and huge computational cost.

Current tensor-based classification approaches for high-dimensional data are mostly adopting regularization or feature extraction steps into the models. For example, [114] proposes a objective function with Lasso penalty to learn the mode-wise precision matrices in the tensor probabilistic discriminant analysis instead of estimating them directly by taking the inverse from empirical covariance matrices, which mitigates the inconsistency in estimate due to high dimensionality. Other methods such as [103, 92] utilize various higher-order principle component analysis to extract features and reduce the data complexity before applying tensor-based K-nearest neighbour classifiers for classification. However, these techniques have several deficiencies. First of all, the regularization-based methods still face the challenge of huge computational cost. Even though they can provide more consistent and robust model estimate by doing a trade-off between variance and bias, the estimation procedures are still the same as that without regularization. For instance, The estimates for ℓ_1 or nuclear norm regularized models are often calculated by doing soft-thresholding on the original estimates. Thus, the computational cost for the procedure remains the same, and could be too huge to be carried out for high-dimensional tensors. Secondly, the feature extraction-based methods integrate unsupervised learning procedures to extract the feature, making it difficult to evaluate the classification consistency for the models. Finally, there is a lack of theoretical results in the current approaches that quantifies the excess risks caused by adding regularization terms and feature extracting procedures, making it difficult to depict the trade-off between the classification accuracy and the computational cost. Novel techniques and statistical frameworks are thus desired to not only optimize the computational procedures, but also integrate the statistical analysis for high-dimensional tensor classification problems.

Random projection, comparing to the aforementioned techniques, turns out to be a perfect candidate for simplifying computational complexity in high-dimensional tensor classification problems, since it is easy to apply and can provide straight forward steps for statistical analysis. It projects data into lower dimensional space with randomly generated transformations to reduce data dimension, and is motivated by the well celebrated Johnson Lindenstrauss Lemma (see e.g. [34]). The lemma says that for arbitrary $k > \frac{8 \log n}{\epsilon^2}$, $\epsilon \in (0, 1)$, there is a linear transformation $\boldsymbol{f} : \mathbb{R}^p \to \mathbb{R}^k$ such that for any two vectors $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^p$, $p > k$:

$$(1 - \epsilon)||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 \leqslant ||\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{f}(\boldsymbol{x}_j)||^2 \leqslant (1 + \epsilon)||\boldsymbol{x}_i - \boldsymbol{x}_j||^2$$

with large probability for all $i, j = 1, ..., n$. The linear transformation $\boldsymbol{f}$ preserves the pairwise Euclidean distances between these points. The random projection has been proven to be a decent dimension reduction technique in machine learning literature [19, 46]. A lot of theoretical results on classification consistency are also established for random projection. [41] presents a Vapnik-Chervonenkis type bounds on the generalization error of a linear classifier trained on a single random projection. [29] provides a convergence rate for the classification error of the support vector machine trained with a single random projection. [24] proves that random projection ensemble classifier can reduce the generalization error further. Their results hold several types of basic classifiers such as nearest neighboring and linear/quadratic discriminant analysis. In addition to the computational efficiency and statistical consistency, [133, 71, 119] demonstrate that random projections for tensors can cost low memory, suggesting that the techniques are memory efficient.

In this work, we propose a computationally efficient and statistically consistent Tensor Ensemble Classifier (TEC) which aggregates multiple CP-Support Tensor Machines (CP-STM). The new STM distinguishes from the existing works [63, 64] by combining the estimation procedure with a newly proposed tensor-shaped random projection to reduce the size of tensors and simplify the computation, making it extremely useful for high-dimensional tensors. The new STM is named as Random Projection-base Support Tensor Machine (RPSTM). Mutiple RPSTMs are then aggregated to form an ensemble classifier, TEC, to mitigate the potential information loss and reduce the extra classification risk brought by random projections. This idea is motivated by [24] and the well known Random Forest model [19]. Similar to these methods, TEC aggregates base classifiers which are estimated from randomly sampled or projected features, and makes predictions for new test points by majority votes. Results from [60, 101] show that such an aggregation of decision can be a very effective tool for improving unstable classifiers like RPSTM.

Our contribution: Our work alleviates the limitations of existing tensor approaches in handling big data classification problems. Specifically, the contribution of this work is threefold.

1. We successfully adopt the well known random-projection technique into high dimensional tensor classification applications and provide an ensemble classifier that can handle extremely big-sized tensor data. The adoption of random projection is shown to be a low-memory cost operation, and makes it feasible to directly classify big tensor data on regular machines efficiently. We further aggregate multiple RPSTM to form our TEC classifier, which can be statistically consistent while remaining computationally efficient. Since the aggregated base classifiers are independent of each other, the model learning procedure can be accelerated in a parallel computing platform.

2. Some theoretical results are established in order to validate the prediction consistency of our classification model. Unlike [29] and [24], we adopt the Johnson-Lindenstrauss lemma further for tensor data and show that the CP-STM can be estimated with randomly projected tensors. The average classification risk of the estimated model converges to the optimal Bayes risk under some specific conditions. Thus, the ensemble of multiple RPSTMs can have robust parameter estimation and provide strongly consistent label predictions. The results also highlight the trade-off between classification risk and dimension reduction created by random projections. As a result, one can take a balance between the computational cost and prediction accuracy in practice.

3. We provide an extensive numerical study with synthetic and real tensor data to reveal our ensemble classifier's decent performance. It performs better than the traditional methods such as linear discriminant analysis and random forest, and other tensor-based methods in applications like brain MRI classification and traffic image recognition. It can also handle large tensors generated from tensor CANDECOMP/PARAFAC (CP) models, which are

widely applied in spatial-temporal data analysis. Besides, the computational cost is much lower for the TEC comparing with the existing methods. All these results indicate a great potential for the proposed TEC in big data and multi-modal data applications.

The contents in this chapter are organized as follow: Section 3.2 reviews the basic concepts about CP-STM classification problem and tensor random projection. Section 3.3 describes our TEC classifier for high-dimensional tensor data, which includes an introduction to our proposed tensor-shaped random projection, the RPSTM, and the ensemble classifier TEC. We provide two different estimation methods for TEC model in section 3.4. In section 3.5, we establish the statistical consistency for the TEC classifier, and provide an explicit upper bound on the excess classification brought by random projection. Simulation studies and real data experiments are in section 3.7. Section 3.8 concludes the work in this chapter.

3.2 Related Works

We briefly review the CP-STM for tensor classification problems and some related works about random projection.

3.2.1 CP-STM for Tensor Classification

Assume there is a collection of data $T_n = \{(\mathcal{X}_1, y_1), (\mathcal{X}_2, y_2), ..., (\mathcal{X}_n, y_n)\}$, where $\mathcal{X}_i \in \boldsymbol{\mathcal{X}} \subset \mathbb{R}^{I_1 \times I_2 \times ... \times I_d}$ are d-way tensors. $\boldsymbol{\mathcal{X}}$ is a compact tensor space, which is a subspace of $\mathbb{R}^{I_1 \times I_2 \times ... \times I_d}$. $y_i \in \{1, -1\}$ are binary labels. CP-STM assumes the tensor predictors are in CP representation, and can be classified by the function which minimizes the objective function

$$\min \quad \lambda ||\boldsymbol{f}||^2 + \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\boldsymbol{f}(\mathcal{X}_i), y_i) \tag{3.1}$$

$\mathcal{L}$ is a loss function for classification, and λ is a tuning parameter. $||\boldsymbol{f}||^2 =< \boldsymbol{f}, \boldsymbol{f} >= \int \boldsymbol{f}(\mathcal{X})^2 d\mathcal{X}$ is the square of functional norm for $\boldsymbol{f}$. By using tensor kernel function

$$K(\mathcal{X}_1, \mathcal{X}_2) = \sum_{l,m=1}^{r} \prod_{j=1}^{d} K^{(j)}(\boldsymbol{x}_{1,l}^{(j)}, \boldsymbol{x}_{2,m}^{(j)}) \tag{3.2}$$

where $\mathcal{X}_1 = \sum_{l=1}^{r} \boldsymbol{x}_{1l}^{(1)} \circ .. \circ \boldsymbol{x}_{1l}^{(d)}$ and $\mathcal{X}_2 = \sum_{l=1}^{r} \boldsymbol{x}_{2l}^{(1)} \circ .. \circ \boldsymbol{x}_{2l}^{(d)}$ are two different tensors. The STM classifier can be written as

$$\boldsymbol{f}(\mathcal{X}) = \sum_{i=1}^{n} \alpha_i y_i K(\mathcal{X}_i, \mathcal{X}) = \boldsymbol{\alpha}^T \boldsymbol{D}_y \boldsymbol{K}(\mathcal{X}) \tag{3.3}$$

where $\mathcal{X}$ is a new d-way rank-r tensor with shape $I_1 \times I_2 \times ... \times I_d$. $\boldsymbol{\alpha} = [\alpha_1, ..., \alpha_n]^T$ are the coefficients. $\boldsymbol{D}_y$ is a diagonal matrix whose diagonal elements are $y_1, .., y_n$. $\boldsymbol{K}(\mathcal{X}) = [K(\mathcal{X}_1, \mathcal{X}), ..., K(\mathcal{X}_n, \mathcal{X})]^T$ is a column vector, whose values are kernel values between training data and the new test data. We denote the collection of functions in the form of (3.3) with $\boldsymbol{\mathcal{H}}$, which is a functional space also known as Reproducing Kernel Hilbert Space (RKHS). The optimal classifier CP-STM $\boldsymbol{f} \in \boldsymbol{\mathcal{H}}$ can be estimated by plugging function (3.3) into objective function (3.1) and minimize it with Hinge loss and Squared Hinge loss. These steps are reviewed in section 2.2.1, algorithm 1 and 2. The coefficient vector of the optimal CP-STM model is denoted by $\boldsymbol{\alpha}^*$. The classification model is statistically consistent if the tensor kernel function satisfying the universal approximating property, as we introduced in the section 2.3.

However, one potential issue of CP-STM for big tensor classification problems is the curse of dimensionality. Even a high-dimensional tensor is decomposed into its CP representation, $\mathcal{X} = \sum_{l=1}^{r} \boldsymbol{x}_{1l}^{(1)} \circ .. \circ \boldsymbol{x}_{1l}^{(d)}, \boldsymbol{x}_l^{(j)}$ can still be in high-dimensional form, making it expensive to calculate the value of kernel functions. For example, Gaussian RBF kernel computes the ℓ_2 norm of the difference between two input tensor CP factors. Such computation can be intensive if the inputs are in high-dimensional. Thus, we propose the RPSTM to simplify the computation with random projection, and avoid the potential issues.

3.2.2 Random Projection

As a dimension reduction technique, the traditional random projection transforms vector data into a lower dimensional space via a linear transformation. The linear transformation is usually defined by a randomly generated projection matrix, $\boldsymbol{A}$, whose element $a_{i,j}$ are either from independently and identically Gaussian distribution $\mathcal{N}(0, 1)$ [34] or a multinomial distribution with three possible outcomes [7]. The two types of random projection matrices are called Gaussian random matrix and sparse random matrix shown below.

$$\boldsymbol{A} = [a_{i,j}] \sim \mathcal{N}(0,1) \qquad \boldsymbol{A} = [a_{i,j}] = \begin{cases} \sqrt{3} & \mathbb{P} = \frac{1}{6} \\ 0 & \mathbb{P} = \frac{2}{3} \\ -\sqrt{3} & \mathbb{P} = \frac{1}{6} \end{cases}$$

Gaussian Random Matrix Sparse Random Matrix

$\mathbb{P}$ stands for probability. With either option of random projection matrices $\boldsymbol{A} \in \mathbb{R}^{k \times p}$, $k > o(\frac{\log n}{\epsilon^2})$, $\epsilon \in (0,1)$, any two vectors $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^p$, $p > k$, the random projection satisfies

$$(1-\epsilon)||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 \leqslant ||\boldsymbol{A}\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_j||^2 \leqslant (1+\epsilon)||\boldsymbol{x}_i - \boldsymbol{x}_j||^2$$

with large probability for all $i, j = 1, ..., n$. This is called Johnson Lindenstrauss (JL) property for the random projection transformation.

For higher-order tensor data, random projection is still defined as a mapping $\boldsymbol{f}_{\text{TRP}} : \mathbb{R}^{I_1 \cdots \times I_d} \rightarrow \mathbb{R}^P$ that transforming a high-dimensional tensor into a vector. In general, the function $\boldsymbol{f}_{\text{TRP}}$ is considered to be

$$\boldsymbol{f}_{\text{TRP}} =< \mathcal{A}, \mathcal{X} > \tag{3.4}$$

$\mathcal{A}$ is a projection tensor in the same size as $\mathcal{X}$. To reduce the memory used for $\mathcal{A}$ and computational cost, [133] proposes a memory efficient random projection for with the assumption that the projection tensor $\mathcal{A}$ is formed by Khatri-rao product product of random matrices. They shows the transformation has the JL property for 2-way tensors (matrices). [71] proposes another random projection satisfying the JL property for rank-one tensors with a projection tensor $\mathcal{A}$ defined by Kroncker product of random matrices.

More related to our work, [119] develops a tensor random projection by assuming the projection tensor $\mathcal{A}$ is in either CP or tensor-train decomposition. The transformations are called CP and tensor-train random projection. Both projections are equipped with the JL property. The CP random projection is a multi-linear map $[\boldsymbol{f}_{\text{TRP-CP}}(\mathcal{X})]_p : \mathbb{R}^{I_1 \times I_2 \cdots \times I_d} \rightarrow \mathbb{R}^P$ that for the p-th element in the output vector

$$[\boldsymbol{f}_{\text{TRP-CP}}(\mathcal{X})]_p = \frac{1}{\sqrt{P}} < \boldsymbol{\mathcal{A}}_p, \mathcal{X} >=< [\![\boldsymbol{A}_p^{(1)}, \boldsymbol{A}_p^{(2)}, ..., \boldsymbol{A}_p^{(d)}]\!], \mathcal{X} > \tag{3.5}$$

$p = 1, 2, ..., P$. $[\![\boldsymbol{A}_p^{(1)}, \boldsymbol{A}_p^{(2)}, ..., \boldsymbol{A}_p^{(d)}]\!]$ is the CP factor of the projection tensor $\boldsymbol{\mathcal{A}}_p$, and $\boldsymbol{A}_p^{(j)} \in \mathbb{R}^{I_i \times r_a}$, $j = 1, .., d$ are Gaussian random matrices. r_a is the CP rank of the random projection tensor $\boldsymbol{\mathcal{A}}_i$, which is independent to the tensor data $\mathcal{X}$. This CP random projection can be applied efficiently when the input tensor $\mathcal{X}$ is also given in CP form. However, since the projection (3.5) transforms tensors into vectors, it may destroy the multi-way features in tensors. It also requires element-wise transformation, which is an extra burden when the dimension of mapping P is large. We propose an alternation to the CP random projection and combine it with our CP-STM for efficient computation.

3.3 Methology

In this section, we present the methodology of our TEC classifier for high-dimensional tensors. We first introduce an alternative tensor-shaped random projection, and combine it with CP-STM to construct RPSTM classifier. The ensemble classifier TEC is then developed by aggregating multiple RPSTMs.

3.3.1 Tensor-Shaped Random Projection

We propose an alternative CP tensor-to-tensor random projection using rank-1 projection tensors $\boldsymbol{\mathcal{A}}$ that can preserve the multi-way structure of tensors after the projection. The proposed tensor-to-tensor random projection is shown to be equivalent to the CP random projection (3.5) with rank-1 projection tensors $\boldsymbol{\mathcal{A}}$ up to a folding-unfolding manner.

Definition 3.3.1. *Suppose a d-mode CP tensor* $\mathcal{X} = [\![\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, ..., \boldsymbol{X}^{(d)}]\!]$ *has size* $I_1 \times I_2 \times ... \times I_d$ *and CP rank r.* $\boldsymbol{X}^{(j)} \in \mathbb{R}^{I_j \times r}$ *are the CP factors of tensor* $\mathcal{X}$ *in matrix form. A rank-1 CP tensor-to-tensor random projection,* $\boldsymbol{f}_{TPR\text{-}CP\text{-}TT} : \mathbb{R}^{I_1 \times I_2 ... \times I_d} \rightarrow \mathbb{R}^{P_1 \times P_2 \times ... P_d}$ *is defined as*

$$\boldsymbol{f}_{TPR\text{-}CP\text{-}TT}(\mathcal{X}) = \frac{1}{\sqrt{P}} [\![\boldsymbol{A}^{(1)}\boldsymbol{X}^{(1)}, \boldsymbol{A}^{(2)}\boldsymbol{X}^{(2)}, ..., \boldsymbol{A}^{(d)}\boldsymbol{X}^{(d)}]\!] \tag{3.6}$$

where $\boldsymbol{A}^{(j)} \in \mathbb{R}^{P_j \times I_j}$ are Gaussian random projection matrices or Sparse random projection matrices. $P = P_1 \times P_2 \times ...P_d$. The projection is uniquely defined by the projection tensor $\boldsymbol{\mathcal{A}} = \{\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, ..., \boldsymbol{A}^{(d)}\}$, the collection of random matrices.

Comparing to the CP random projection (3.5), $\boldsymbol{f}_{\text{TPR-CP-TT}}$ assumes the projection tensor $\boldsymbol{\mathcal{A}}$ to be rank-1 and perform projections directly on the CP tensor factors instead of element-wise. Please notice that $\boldsymbol{A}^{(j)}$ are not CP component matrices for $\boldsymbol{\mathcal{A}}$, and $\boldsymbol{\mathcal{A}} = \{\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, ..., \boldsymbol{A}^{(d)}\}$ is not the notation of Kruskal tensors. We use this notation for convenience since the random projection in definition 3.3.1 with the collection of matrices $\{\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, ..., \boldsymbol{A}^{(d)}\}$ is equivalent to the CP random projection using tensors in equation (3.5). We show this in the following proposition.

Proposition 3.3.1. *Let $\boldsymbol{\pi} : P_1 \times P_2 \times ...P_d \rightarrow P$ be a invertable unfolding rule such that $\mathcal{X}_{p_1,..,p_d} = Vec(\mathcal{X})_{\boldsymbol{\pi}(p_1,..,p_d)}$. $1 \leqslant p_j \leqslant P_j$, and $1 \leqslant p = \boldsymbol{\pi}(p_1, .., p_d) \leqslant P$ are indices. For random projection (3.5) with rank-1 projection tensor $\boldsymbol{\mathcal{A}}$, it is equivalent to the projection 3.6 up to the unfolding rule $\boldsymbol{\pi}$.*

The proof is provided in the appendix B.1. The projection can reduce the computational cost significantly for tensor-based models as it transforms tensor CP components into lower dimensional spaces. Now, we introduce our RPSTM classifiers estimated from the output of tensor random projection (3.6).

3.3.2 Random-Projection-Based Support Tensor Machine (RPSTM)

With tensor-shaped CP random projection, we reformulate the model for the tensor classification problem. Let $T_n^{\boldsymbol{\mathcal{A}}} = \{(\mathcal{X}_1^{\boldsymbol{\mathcal{A}}}, y_1), (\mathcal{X}_2^{\boldsymbol{\mathcal{A}}}, y_2), ..., (\mathcal{X}_n^{\boldsymbol{\mathcal{A}}}, y_n)\}$ be the random projection of the original training data T_n such that

$$\mathcal{X}_i^{\boldsymbol{\mathcal{A}}} = \boldsymbol{f}_{\text{TPR-CP-TT}}(\mathcal{X}_i) \tag{3.7}$$

for all $\mathcal{X}_i$ from T_n. The random projection $\boldsymbol{f}_{\text{TPR-CP-TT}}$ is uniquely defined by the fixed CP random projection tensor $\boldsymbol{\mathcal{A}} = \{\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, ..., \boldsymbol{A}^{(d)}\}$, where each $\boldsymbol{A}^{(j)} \in \mathbb{R}^{P_j \times I_j}$. The original training tensors are transformed into a lower dimensional space with size $P_1 \times P_2 ... \times P_d$, i.e.

$\mathcal{X}_i^{\mathcal{A}} \in \mathbb{R}^{P_1 \times P_2 \cdots \times P_d}$. Similar to CP-STM, RPSTM tries to find an optimal function $\boldsymbol{f}$ such that it optimizes the objective function

$$\min \quad \lambda ||\boldsymbol{f}||^2 + \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\boldsymbol{f}(\mathcal{X}_i^{\mathcal{A}}), y_i) \tag{3.8}$$

Instead of using the original data T_n, the objective function measures the empirical classification loss on the randomly projected training data $T_n^{\mathcal{A}}$. A new kernel function for randomly projected tensors is defined as follow: For any pair of CP tensors $\mathcal{X}_1 = [\![\boldsymbol{X}_1^{(1)}, \boldsymbol{X}_1^{(2)}, ..., \boldsymbol{X}_1^{(d)}]\!], \mathcal{X}_2 = [\![\boldsymbol{X}_2^{(1)}, \boldsymbol{X}_2^{(2)}, ..., \boldsymbol{X}_2^{(d)}]\!]$, the kernel function is

$$\begin{aligned} K(\mathcal{X}_1^{\mathcal{A}}, \mathcal{X}_2^{\mathcal{A}}) &= K\left(\boldsymbol{f}_{\text{TPR-CP-TT}}(\mathcal{X}_1), \boldsymbol{f}_{\text{TPR-CP-TT}}(\mathcal{X}_2)\right) \\ &= K\left(\frac{1}{\sqrt{P}}[\boldsymbol{A}^{(1)}\boldsymbol{X}_1^{(1)}, \boldsymbol{A}^{(2)}\boldsymbol{X}_1^{(2)}, ..., \boldsymbol{A}^{(d)}\boldsymbol{X}_1^{(d)}], \right. \\ &\qquad \left. \frac{1}{\sqrt{P}}[\boldsymbol{A}^{(1)}\boldsymbol{X}_2^{(1)}, \boldsymbol{A}^{(2)}\boldsymbol{X}_2^{(2)}, ..., \boldsymbol{A}^{(d)}\boldsymbol{X}_2^{(d)}]\right) \\ &= \sum_{l,k=1}^{R} \prod_{j=1}^{d} K^{(j)}(\frac{1}{\sqrt{P}}\boldsymbol{A}^{(j)}\boldsymbol{x}_{1,l}^{(j)}, \frac{1}{\sqrt{P}}\boldsymbol{A}^{(j)}\boldsymbol{x}_{2,k}^{(j)}) \end{aligned} \tag{3.9}$$

where $\boldsymbol{A}^{(1)}, ..., \boldsymbol{A}^{(d)}$ are the projection matrices of $\mathcal{A}$ that defines the projection $\boldsymbol{f}_{\text{TPR-CP-TT}}$. $\boldsymbol{x}_{1,l}^{(j)}$ are the columns of $\boldsymbol{X}_1^{(j)}$, and $\boldsymbol{x}_{2,k}^{(j)}$ are the columns of $\boldsymbol{X}_2^{(j)}$. $K^{(j)}$ are still vector-based kernel functions measuring inner products for factors in different tensor modes.

A Random Projection-based Support Tensor Machine (RPSTM), with the kernel function (3.9), will be in the form of

$$\begin{aligned} \boldsymbol{g}(\mathcal{X}) &= \sum_{i=1}^{n} \beta_i y_i K\left(\boldsymbol{f}_{\text{TPR-CP-TT}}(\mathcal{X}_i), \boldsymbol{f}_{\text{TPR-CP-TT}}(\mathcal{X})\right) \\ &= \boldsymbol{\beta}^T \boldsymbol{D}_y \boldsymbol{K}\left(\boldsymbol{f}_{\text{TPR-CP-TT}}(\mathcal{X})\right) \end{aligned} \tag{3.10}$$

for a new given tensor $\mathcal{X}$ due the representer theorem [9]. Notice that we use $\boldsymbol{g}$ to denote functions spanned by tensor kernels to distinguish it from the random projection function $\boldsymbol{f}_{TPR-CP-TT}$ and the original STM classifier $\boldsymbol{f}$. $\boldsymbol{K}\left(\boldsymbol{f}_{TPR-CP-TT}(\mathcal{X})\right)$ is again a column vector whose elements are kernel values between projected training data $\boldsymbol{f}_{\text{TPR-CP-TT}}(\mathcal{X}_i)$ and the projected new observation $\boldsymbol{f}_{\text{TPR-CP-TT}}(\mathcal{X})$. $\boldsymbol{D}_y$ is the diagonal matrix whose diagonal is $\boldsymbol{y} = [y_1, ..., y_n]^T$. $\boldsymbol{\beta}$ is the coefficient vector and is differentiated from the notation of CP-STM. We denote the collection of functions in the form of (3.10) with $\mathcal{H}^{\mathcal{A}}$, which is also a reproducing kernel Hilbert space (RKHS). The optimal classifier can be estimated by plugging the function (3.10) into the objective function (3.8) and minimize it. Let $\boldsymbol{g_n}$ denote the optimal function satisfying

$$\boldsymbol{g_n} = \arg \min_{\boldsymbol{f} \in \mathcal{H}^{\mathcal{A}}} \quad \lambda ||\boldsymbol{f}||^2 + \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\boldsymbol{f}(\mathcal{X}_i^{\mathcal{A}}), y_i) \tag{3.11}$$

then $\boldsymbol{g_n}$ is the RPSTM classifier associated with the fixed random projection $\boldsymbol{f}_{TPR-CP-TT}$ that we estimated from the training data. The label for new observation tensor $\mathcal{X}$ will be predicted by $\text{Sgn}[\boldsymbol{g_n}(\mathcal{X})]$. Thanks to the tensor random projection, the estimation of RPSTM can be computationally efficient and feasible for high-dimensional tensors. The computational benefit will be discussed in details in the model estimation part of this paper.

3.3.3 TEC: Ensemble of RPSTM

While random projection provides extra efficiency by transforming tensor CP components into lower dimension, there is no guarantee that the projected data will preserve the same margin for every single random projection. As a result, the expected excess risk of RPSTM may be larger than the original CP-STM. In order to mitigate the impact of random projection and provide robust class assignments, multiple RPSTMs are aggregated to form a Tensor Ensemble Classifier (TEC).

Let

$$\boldsymbol{\tau_{n,b}}(\mathcal{X}) = \frac{1}{b} \sum_{m=1}^{b} \text{Sgn}[\boldsymbol{g_{m,b}}(\mathcal{X})] \tag{3.12}$$

b is the number of RPSTM classifiers estimated with different random projections. $\boldsymbol{g_{m,b}}$ are RPSTM classifiers learned independently from the training data T_n. The TEC classifier is then defined as

$$\boldsymbol{e_{n,b}}(\mathcal{X}) = \begin{cases} 1 & \text{if} \quad \boldsymbol{\tau_{n,b}}(\mathcal{X}) \geqslant \gamma \\ -1 & \quad \text{Otherwise} \end{cases} \tag{3.13}$$

for a new test tensor $\mathcal{X}$ in CP form. γ is the threshold parameter. For simple majority vote and class-balanced binary classification, $\gamma = 0$. However, it can be different values if any prior information is provided.

3.4 Model Estimation

In this section, We present two estimation procedures for TEC model using two different loss functions. More importantly, we emphasize significant computational efficiency of TEC model by comparing its algorithmic steps and memory costs to the CP-STM, and even to the naive vectorized support vector machine models.

Since TEC is an aggregation of multiple independent RPSTMs, we only have to show the details for a single RPSTM estimation and aggregate them. Similar to the estimation of CP-STM, we can use Hinge loss and Squared Hing loss in objective function (3.8) to measure the empirical classification risk and estimate RPSTM classifiers. With Hinge loss, the objective function becomes

$$\min_{\boldsymbol{f}\in\mathcal{H}^{\mathcal{A}}} \quad \lambda||\boldsymbol{f}||^2 + \frac{1}{n}\sum_{i=1}^{n} \max\left(0, 1 - \boldsymbol{f}(\mathcal{X}_i^{\mathcal{A}}) \cdot y_i\right) \tag{3.14}$$

Like CP-STM, the optimization problem (3.14) is equivalent to a quadratic programming problem, which is shown in [25]. Once we calculate the kernel matrix $\boldsymbol{K}^{\mathcal{A}}$ as

$$\boldsymbol{K}^{\mathcal{A}} = \begin{bmatrix} K(\mathcal{X}_1^{\mathcal{A}}, \mathcal{X}_1^{\mathcal{A}}) & K(\mathcal{X}_1^{\mathcal{A}}, \mathcal{X}_2^{\mathcal{A}}) & \dots & K(\mathcal{X}_1^{\mathcal{A}}, \mathcal{X}_n^{\mathcal{A}}) \\ K(\mathcal{X}_2^{\mathcal{A}}, \mathcal{X}_1^{\mathcal{A}}) & K(\mathcal{X}_2^{\mathcal{A}}, \mathcal{X}_2^{\mathcal{A}}) & \dots & K(\mathcal{X}_2^{\mathcal{A}}, \mathcal{X}_n^{\mathcal{A}}) \\ \dots & \dots & \dots & \dots \\ K(\mathcal{X}_n^{\mathcal{A}}, \mathcal{X}_1^{\mathcal{A}}) & K(\mathcal{X}_n^{\mathcal{A}}, \mathcal{X}_2^{\mathcal{A}}) & \dots & K(\mathcal{X}_n^{\mathcal{A}}, \mathcal{X}_n^{\mathcal{A}}) \end{bmatrix} \tag{3.15}$$

with tensor kernel function (3.9). The quadratic programming problem is defined as

$$\begin{aligned} \min_{\boldsymbol{\beta}\in\mathbb{R}^n} \quad & \frac{1}{2}\boldsymbol{\beta}^T\boldsymbol{D}_y\boldsymbol{K}^{\mathcal{A}}\boldsymbol{D}_y\boldsymbol{\beta} - \mathbf{1}^T\boldsymbol{\beta} \\ \text{S.T.} \quad & \boldsymbol{\beta}^T\boldsymbol{y} = 0 \\ & 0 \le \boldsymbol{\beta} \le \frac{1}{2n\lambda} \end{aligned} \tag{3.16}$$

Same optimization techniques used in CP-STM can be adopted to solve this problem. A TEC classifier can then be estimated by repeating the procedure for multiple times with different random projections. The steps are summarized in the algorithm 7 below. $\boldsymbol{X}_h^{(j)}[:, l]$ is the l-th column of the tensor CP factor matrix $\boldsymbol{X}_h^{(j)}$. The output of the algorithm contains a list of RPSTM coefficients

Algorithm 7 Hinge TEC

```
1:  procedure TEC Train
2:      Input: Training set $T_n = \{\mathcal{X}_i\}$, $\boldsymbol{y}$, kernel function $K$, tensor rank r, $\lambda$, number of ensemble
    $b$
3:      for i = 1, 2,...n do
4:          $\mathcal{X}_i = [\boldsymbol{X}_i^{(1)}, ..., \boldsymbol{X}_i^{(d)}]$                              ▷ CP decomposition
5:      for m = 1, 2, ..., b do
6:          Generate random projection tensor $\boldsymbol{\mathcal{A}}_m = \{\boldsymbol{A}_m^{(1)}, \boldsymbol{A}_m^{(2)}, ..., \boldsymbol{A}_m^{(d)}\}$
7:          Create initial matrix $\boldsymbol{K}^{\boldsymbol{\mathcal{A}}_m} \in \mathbb{R}^{n\times n}$
8:          for i = 1,...,n do
9:              for h = 1,...,i do
10:                 $\boldsymbol{K}^{\boldsymbol{\mathcal{A}}_m}[i,h] = \sum_{k,l=1}^{r} \prod_{j=1}^{d} K(\boldsymbol{A}_m^{(j)}\boldsymbol{X}_i^{(j)}[:,k], \boldsymbol{A}_m^{(j)}\boldsymbol{X}_h^{(j)}[:,l])$      ▷ Kernel values
11:                 $\boldsymbol{K}^{\boldsymbol{\mathcal{A}}_m}[h,i] = \boldsymbol{K}^{\boldsymbol{\mathcal{A}}_m}[i,h]$
12:         Solve the quadratic programming problem (3.16) and find the optimal $\boldsymbol{\beta}^{m*}$.
13:         Output: $\boldsymbol{\beta}^{m*}, \boldsymbol{\mathcal{A}}_m$
        Output: $[\boldsymbol{\beta}^{1*},\boldsymbol{\beta}^{2*},...,\boldsymbol{\beta}^{b*}]$, $[\boldsymbol{\mathcal{A}}_1, ..., \boldsymbol{\mathcal{A}}_b]$
```

$[\boldsymbol{\beta}^{1*},\boldsymbol{\beta}^{2*},...,\boldsymbol{\beta}^{b*}]$ and its corresponding random projection tensors $[\boldsymbol{\mathcal{A}}_1, ..., \boldsymbol{\mathcal{A}}_b]$. The projection is still needed for new test point prediction.

To estimate RPSTM with Squared Hinge loss, we can use Gaussian-Newton method to optimize the objective function

$$\min_{\boldsymbol{f}\in\mathcal{H}^{\boldsymbol{\mathcal{A}}}} \quad \lambda||\boldsymbol{f}||^2 + \frac{1}{n}\sum_{i=1}^{n} \max\left(0, 1 - \boldsymbol{f}(\mathcal{X}_i^{\boldsymbol{\mathcal{A}}}) \cdot y_i\right)^2 \tag{3.17}$$

by letting its derivative to be zero. Since the procedure is identical to the derivation (2.9) in section 2.2, we provide the updating rule for the parameter $\boldsymbol{\beta}$ directly

$$\boldsymbol{\beta}^* = \frac{1}{n}\boldsymbol{D}_y(\lambda\boldsymbol{I} + \frac{1}{n}\boldsymbol{I_s}\boldsymbol{K}^{\boldsymbol{\mathcal{A}}})^{-1}\boldsymbol{I_s}\boldsymbol{y} \tag{3.18}$$

where $\boldsymbol{I_s}$ is the diagonal matrix whose diagonal elements are indicating if the corresponding tensors are support tensors. With a initial value of $\boldsymbol{\beta}$, we can update the parameter iteratively with (3.18) until its value converges. The steps are summarized in the algorithm 8. In the algorithm, $\boldsymbol{k}_i^{\boldsymbol{\mathcal{A}}_m}$ is the i-th column vector of kernel matrix $\boldsymbol{K}^{\boldsymbol{\mathcal{A}}_m}$.

With estimated TEC model, we can make prediction for new test points. The steps for prediction is identical no matter whether the model are estimated with Hinge loss or Squared Hinge loss. The

Algorithm 8 Squared Hinge TEC

```
1: procedure TEC TRAIN
2:     Input: Training set $T_n = \{\mathcal{X}_i\}$, $\boldsymbol{y}$, kernel function $K$, tensor rank r, $\lambda$, $\eta$, maxiter, number of ensemble $b$
3:     for i = 1, 2,...n do
4:         $\mathcal{X}_i = [\boldsymbol{X}_i^{(1)}, ..., \boldsymbol{X}_i^{(d)}]$                          ▷ CP decomposition
5:     for m = 1, ..., b do
6:         Create initial matrix $\boldsymbol{K} \in \mathbb{R}^{n \times n}$
7:         Generate random projection tensor $\boldsymbol{\mathcal{A}}_m = \{\boldsymbol{A}_m^{(1)}, \boldsymbol{A}_m^{(2)}, ..., \boldsymbol{A}_m^{(d)}\}$
8:         for i = 1,...,n do
9:             for h = 1,...,i do
10:                $\boldsymbol{K}^{\boldsymbol{\mathcal{A}}_m}[i,h] = \sum_{k,l=1}^{r} \prod_{j=1}^{d} K(\boldsymbol{A}_m^{(j)}\boldsymbol{X}_i^{(j)}[:,k], \boldsymbol{A}_m^{(j)}\boldsymbol{X}_h^{(j)}[:,l])$   ▷ Kernel values
11:                $\boldsymbol{K}^{\boldsymbol{\mathcal{A}}_m}[h,i] = \boldsymbol{K}^{\boldsymbol{\mathcal{A}}_m}[i,h]$
12:        Create $\boldsymbol{\beta}^{m*} = \boldsymbol{1}_{n\times 1}, \boldsymbol{\beta}^{m} = \boldsymbol{0}_{n\times 1}$                 ▷ Initial Value
13:        Iteration = 0
14:        while $||\boldsymbol{\beta}^{m*} - \boldsymbol{\beta}^{m}||_2 \geqslant \eta$ & Iteration $\leqslant$ maxiter do
15:            $\boldsymbol{\beta}^{m} = \boldsymbol{\beta}^{m*}$
16:            Find $\boldsymbol{s} \in \mathbb{R}^{n\times 1}$. $\boldsymbol{s}_i \in \{0,1\}$ such that $\boldsymbol{s}_i = 1$ if $y_i \boldsymbol{k}_i^{\boldsymbol{\mathcal{A}}_m T}\boldsymbol{\beta}^m < 1$   ▷ Support tensors
17:            $\boldsymbol{I}_s = \text{diag}(\boldsymbol{s})$                 ▷ Create diagonal matrix with $S$ as diagonal
18:            $\boldsymbol{\beta}^{m*} = \frac{1}{n}\boldsymbol{D}_y(\lambda \boldsymbol{I} + \frac{1}{n}\boldsymbol{I_s}\boldsymbol{K}^{\boldsymbol{\mathcal{A}}_m})^{-1}\boldsymbol{I_s y}$                 ▷ Update
19:        Output: $\boldsymbol{\beta}^{m*}$
20:    Output: $[\boldsymbol{\beta}^{1*}, \boldsymbol{\beta}^{2*}, ..., \boldsymbol{\beta}^{b*}]$, $[\boldsymbol{\mathcal{A}}_1, ..., \boldsymbol{\mathcal{A}}_b]$
```

steps for prediction are stated in the algorithm 9. For convenience, we keep using the notation for decomposed training tensors $\mathcal{X}_i = [\boldsymbol{X}_i^{(1)}, ..., \boldsymbol{X}_i^{(d)}]$ from the estimation steps. $\boldsymbol{k}^{\boldsymbol{\mathcal{A}}_m}$ is a new column vector in length n, whose i-th element is kernel value between training tensors $\mathcal{X}_i$ and the new test point $\mathcal{X}$.

Suppose the projected training tensors are in the shape of $P_1 \times P_2 \times ... \times P_d$, the time complexity for kernel matrix computation is $O(n^2 r^2 d \sum_{j=1}^{d} P_j)$. Notice that the choices of P_j are free from the original tensor dimensions I_j, and are only related to the training data size n following the JL lemma. We can choose relatively small P_js so that the total time complexity of algorithm 7 and 8 are $O(n^2 r^2 d \sum_{j=1}^{d} P_j + l_1)$ and $O(n^2 r^2 d \sum_{j=1}^{d} P_j + l_2)$, when the training data are given in their projected CP decomposition forms. l_1 and l_2 are the necessary steps to perform quadratic programming in algorithm 7 and the iterations in algorithm 8. They are bounded by the order of $O(n^2)$ empirically,

Algorithm 9 TEC Prediction

```
1:  procedure TEC PREDICT
2:      Input: TEC coefficients $[\boldsymbol{\beta}^{1*}, \boldsymbol{\beta}^{2*}, ..., \boldsymbol{\beta}^{b*}]$, random projection tensors $[\boldsymbol{\mathcal{A}}_1, ..., \boldsymbol{\mathcal{A}}_b]$, kernel
    function $K$, tensor rank r, new test point $\mathcal{X}$, threshold parameter $\gamma$
3:      $\mathcal{X} = [\boldsymbol{X}^{(1)}, ..., \boldsymbol{X}^{(d)}]$                ▷ CP decomposition for New observation
4:      $\boldsymbol{\tau_{n,b}} = 0$                                        ▷ Initial value in equation (3.12)
5:      for m = 1,...,b do
6:          for i = 1,...,n do
7:              $\boldsymbol{k}^{\boldsymbol{\mathcal{A}}_m}[i] = \sum_{k,l=1}^{r} \prod_{j=1}^{d} K(\boldsymbol{A}_m^{(j)}\boldsymbol{X}_i^{(j)}[:,k], \boldsymbol{A}_m^{(j)}\boldsymbol{X}^{(j)}[:,l])$        ▷ Kernel vector
8:              $\boldsymbol{\tau_{n,b}} = \boldsymbol{\tau_{n,b}} + \mathrm{Sign}[\boldsymbol{k}^{\boldsymbol{\mathcal{A}}_m T}\boldsymbol{D}_y\boldsymbol{\beta}^{m*}]$             ▷ Update equation (3.12)
9:      If $\boldsymbol{\tau_{n,b}} \geqslant \gamma$, the prediction is class 1. Otherwise it is -1.      ▷ Equation (3.13)
10:     Output: Prediction
```

which is shown by [25]. Since $I_j >> P_j$ and $I_j >> n$ for $j = 1, ..., d$ in high-dimensional tensor problems, the time complexities of RPSTM in algorithms 7 and 8 are significantly smaller than CP-STM, which are $O(n^2r^2d \sum_{j=1}^{d} I_j + l_1)$ and $O(n^2r^2d \sum_{j=1}^{d} I_j + l_2)$. Due to the fact that each RPSTM is estimated independently, TEC model can be fitted in a parallel computing manner. As a result, the time complexity of TEC can be roughly the same as RPSTM, which is also much smaller than CP-STM. As for the memory complexity, CP-STM requires $O(nr \sum_{j=1}^{d} I_j + n)$, which is more prohibitive and infeasible than $O(bnr \sum_{j=1}^{d} P_j + bn)$ required by TEC with b aggregated RPSTMs. As the memory complexity are dominated by the dimension of projected CP factors, TEC turns out to be more efficient. If we further consider the naive vectorized SVM model, its time and memory complexities are $O(n^2 \prod_{j=1}^{d} I_j + l_1)$ and $O(n \prod_{j=1}^{d} I_j)$. Through the comparison, it is obvious that TEC model is much more computationally efficient than both CP-STM and the traditional vectorized SVM. This comparison is summarized in the table 3.1.

One may notice that our discussion above does not include the complexity from tensor CP decomposition and random projection. Since both the RPSTM and CP-STM requires CP decomposition, subtracting this part of complexity does not affect the comparison. Moreover, novel CP decomposition methods from [116, 137] reach a time complexity of $O(nr \sum_{j=1}^{d} I_j)$. Neither CP-STM nor RPSTM will have a larger time complexity than the vectorized SVM by adding this part. As

Models	Time Complexity	Memory Complexity
TEC (Parallel)	$O(n^2r^2d\sum_{j=1}^{d} P_j + l_1)$ / $O(n^2r^2d\sum_{j=1}^{d} P_j + l_2)$	$O(bnr\sum_{j=1}^{d} P_j + bn)$
RPSTM	$O(n^2r^2d\sum_{j=1}^{d} P_j + l_1)$ / $O(n^2r^2d\sum_{j=1}^{d} P_j + l_2)$	$O(nr\sum_{j=1}^{d} P_j + n)$
CP-STM	$O(n^2r^2d\sum_{j=1}^{d} I_j + l_1)$ / $O(n^2r^2d\sum_{j=1}^{d} I_j + l_2)$	$O(nr\sum_{j=1}^{d} I_j + n)$
Vectorized SVM	$O(n^2\prod_{j=1}^{d} I_j + l_1)$	$O(n\prod_{j=1}^{d} I_j)$

Table 3.1: TEC: Comparison of Computational Complexity

for the random projection, we define that the projection tensor can be composed by sparse projection matrices in definition 3.3.1. As a result, we can utilize techniques such as low-rank matrix decomposition to reduce the costs of computation and memory. In addition, [90] showed that the projection matrices can be very sparse, making the complexity of random projection to be trivial comparing to other estimation steps.

We want to briefly discuss the tuning parameter selection at the end of this section. The number of ensemble classifiers, b, and the threshold parameter, γ, are chosen by cross-validation. We first let $\gamma = 0$, which is the middle of two labels, -1 and 1. Then we search b in a reasonable range, between 2 to 20. The optimal b is the one that provides the best classification model. In the next step, we fix b and search γ between 1 and -1 with step size to be 0.1, and find optimal value which has the best classification accuracy. For simple majority vote, γ can be set to be zero directly. The choice of random projection matrices is more complicated. Although we can generate random projection matrices, the dimension of matrices is remain unclear. Our guideline, JL-lemma, only provides a lower bound for dimension, and is only for vector situation. As a result, we can only choose the dimension based on our intuition and cross-validation results in practice. Empirically, we suggest to choose the projection dimension $P_j \approx \text{int}(0.7 \times I_j)$ for each mode.

3.5 Statistical Properties

In this section, we develop the statistical consistency for both TEC and RPSTM models. In addition, we establish an explicit upper bound on the excessive risk brought by random projection, highlighting the trade-off between computational efficiency and potential risks.

For the convenience, we introduce a few more notations. $\mathcal{R}_{\mathcal{L}}(\boldsymbol{f})$ is the classification risk of a specific decision function $\boldsymbol{f}$, which is defined as

$$\mathcal{R}_{\mathcal{L}}(\boldsymbol{f}) = \mathbb{E}_{(\mathcal{X}\times\mathcal{Y})}\mathcal{L}(y, \boldsymbol{f}(\mathfrak{X})) = \int \mathcal{L}(y, \boldsymbol{f}(\mathfrak{X}))d\mathbb{P}$$

where $\mathcal{L}$ is a loss function. The empirical risk of the decision function $\boldsymbol{f}$ over the training data T_n is

$$\mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f}) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}\big(y_i, \boldsymbol{f}(\mathfrak{X}_i)\big)$$

$\mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f})$ is the estimate of $\mathcal{R}_{\mathcal{L}}(\boldsymbol{f})$ on finite training data. The subscript $\mathcal{L}$ in the notation of risks emphasizes that the risks are calculated using specific loss functions. We also use risk notations without the subscript $\mathcal{L}$, like $\mathcal{R}(\boldsymbol{f})$, to denote the risk of $\boldsymbol{f}$ calculated with zero-one loss $\mathcal{L}(y, z) = \mathbf{1}\{y \neq z\}$. $\mathbf{1}$ is indicator function. The definition of classification consistency is defined on the zero-one loss initially. We use $\mathcal{R}^*$ to denote the Bayes risk of the tensor classification problem over the joint distribution $\boldsymbol{\mathcal{X}}\times\boldsymbol{\mathcal{Y}}$. It is the optimal risk in the sense that for any measureable function $\boldsymbol{f} : \boldsymbol{\mathcal{X}} \to \mathbb{R}$, $\mathcal{R}^* = \min\limits_{\boldsymbol{f}}\mathcal{R}(\boldsymbol{f})$. A decision rule is said to be consistent if $\mathcal{R}(\boldsymbol{f_n}) \to \mathcal{R}^*$ as $n \to \infty$, see [36]. We have to show this for TEC and RPSTM in order to establish their consistency properties. However, existing results [13, 128] show that the convergence in surrogate loss-based risks indicates the convergence in zero-one loss-based risks, i.e.

$$\mathcal{R}_{\mathcal{L}}(\boldsymbol{f_n}) \to \mathcal{R}^*_{\mathcal{L}} \quad \Rightarrow \quad \mathcal{R}(\boldsymbol{f_n}) \to \mathcal{R}^*$$

$\mathcal{R}^*_{\mathcal{L}} = \min\limits_{\boldsymbol{f}}\mathcal{R}_{\mathcal{L}}(\boldsymbol{f})$ for any decision rule $\{\boldsymbol{f_n}\}$. This conclusion holds as long as the loss function is self-calibrated. Both Hinge loss and Squared Hinge loss used in our models are self-calibrated. As a result, we only have to show $\mathcal{R}_{\mathcal{L}}(\boldsymbol{f_n}) \to \mathcal{R}^*_{\mathcal{L}}$ for TEC and RPSTM.

Recall that in section 3.2 and section 3.3, we use $\boldsymbol{\mathcal{H}}$ and $\boldsymbol{\mathcal{H}}^{\boldsymbol{\mathcal{A}}}$ to denote the reproducing kernel Hilbert space spanned by tensor kernels (3.2) and projected tensor kernels (3.9). Let $\boldsymbol{f_n^\lambda}, \boldsymbol{f^\lambda} \in \boldsymbol{\mathcal{H}}$ such that

$$\boldsymbol{f_n^\lambda} = \arg\min_{\boldsymbol{f}\in\boldsymbol{\mathcal{H}}} \quad \lambda||\boldsymbol{f}||^2 + \mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f}) \qquad \boldsymbol{f^\lambda} = \arg\min_{\boldsymbol{f}\in\boldsymbol{\mathcal{H}}} \quad \lambda||\boldsymbol{f}||^2 + \mathcal{R}_{\mathcal{L}}(\boldsymbol{f})$$

$\boldsymbol{f_n^\lambda}$ is the CP-STM classifier estimated from the training data T_n that minimizes the objective function on T_n. $\boldsymbol{f^\lambda}$ is the optimal CP-STM learned from infinite size training data. The superscript $\boldsymbol{\lambda}$ in both functions denote that they are optimal with the given value of λ in the objective functions. Notice that both $\boldsymbol{f_n^\lambda}$ and $\boldsymbol{f^\lambda}$ do not minimize the empirical risk and expected risk, but the objective functions which have regularization terms. We further denote $\mathcal{R}^*_{\mathcal{L},\boldsymbol{\mathcal{H}}} = \min_{\boldsymbol{f}\in\boldsymbol{\mathcal{H}}} \mathcal{R}_{\mathcal{L}}(\boldsymbol{f})$, the optimal risk functions in $\boldsymbol{\mathcal{H}}$ can be achieved. Similarly, we define $\boldsymbol{g_n^\lambda}, \boldsymbol{g^\lambda} \in \boldsymbol{\mathcal{H}}^{\boldsymbol{\mathcal{A}}}$ as

$$\boldsymbol{g_n^\lambda} = \arg\min_{\boldsymbol{g}\in\boldsymbol{\mathcal{H}}^{\boldsymbol{\mathcal{A}}}} \quad \lambda||\boldsymbol{g}||^2 + \mathcal{R}_{\mathcal{L},T_n^{\boldsymbol{\mathcal{A}}}}(\boldsymbol{g}) \qquad \boldsymbol{g^\lambda} = \arg\min_{\boldsymbol{g}\in\boldsymbol{\mathcal{H}}^{\boldsymbol{\mathcal{A}}}} \quad \lambda||\boldsymbol{g}||^2 + \mathcal{R}_{\mathcal{L}}(\boldsymbol{g})$$

where $\mathcal{R}_{\mathcal{L},T_n^{\boldsymbol{\mathcal{A}}}}(\boldsymbol{g}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\big(y_i, \boldsymbol{g}(\boldsymbol{\mathcal{X}}_i^{\boldsymbol{\mathcal{A}}})\big)$ and $\mathcal{R}_{\mathcal{L}}(\boldsymbol{g}) = \mathbb{E}_{(\mathcal{X}\times\mathcal{Y})}\mathcal{L}(y, \boldsymbol{g}(\boldsymbol{\mathcal{X}}^{\boldsymbol{\mathcal{A}}}))$ for a given tensor random projection defined by $\boldsymbol{\mathcal{A}}$. $\boldsymbol{g_n^\lambda}$ is the optimal RPSTM model estimated from the projected training data $T_n^{\boldsymbol{\mathcal{A}}}$, and $\boldsymbol{g^\lambda}$ is the infinite-sample estimate.

We derive the proof of consistency for TEC $\boldsymbol{e_{n,b}^\lambda}$ and RPSTM $\boldsymbol{g_n^\lambda}$ models with the regularization parameter λ. Notice that $\mathcal{R}_{\mathcal{L}}(\boldsymbol{e_{n,b}^\lambda})$ and $\mathcal{R}_{\mathcal{L}}(\boldsymbol{g_n^\lambda})$ are calculated with a specific random projection tensor $\boldsymbol{\mathcal{A}}$. Thus, we develop the consistency results on the expected risk $\mathbb{E}_{\boldsymbol{\mathcal{A}}}\left[\mathcal{R}_{\mathcal{L}}(\boldsymbol{e_{n,b}^\lambda})\right]$ and $\mathbb{E}_{\boldsymbol{\mathcal{A}}}\left[\mathcal{R}_{\mathcal{L}}(\boldsymbol{g_n^\lambda})\right]$ instead to demonstrate the average performance and integrate the impacts of different possible random projections. The expectation is taken over the distribution of random projection tensors.

3.5.1 Excess Risk of TEC

We first boud the expected risk of our TEC classifier $\boldsymbol{e_{n,b}^\lambda}$ by using the result from [24], theorem 2.

Theorem 3.5.1. *For each $b \in \mathbb{N}$, $\boldsymbol{e_{n,b}^\lambda}$ is the TEC classifier aggregating b independent RPSTMs*

$\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}$. *$\lambda$ is the parameter for the functional norm in the objective function (3.8). Then*

$$\mathbb{E}_{\mathcal{A}}\left[\mathcal{R}(\boldsymbol{e}_{\boldsymbol{n},\boldsymbol{b}}^{\boldsymbol{\lambda}})\right] - \mathcal{R}^* \leqslant \frac{1}{\min(\gamma, 1-\gamma)}\left[\mathbb{E}_{\mathcal{A}}\left[\mathcal{R}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}})\right] - \mathcal{R}^*\right]$$

This result says that the ensemble model TEC is statistically consistent as long as the base classifier is consistent. With the surrogate loss property, we only need to develop the consistency for RPSTM model by showing excessive risk of surrogate loss $\mathbb{E}_{\mathcal{A}}\left[\mathcal{R}_{\mathcal{L}}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}})\right] - \mathcal{R}_{\mathcal{L}}^*$ converges to zero.

3.5.2 Excess Risk of RPSTM

To show the consistency of RPSTM, we first use the following proposition to decompose the excess risk of RPSTM, $\mathbb{E}_{\mathcal{A}}\left[\mathcal{R}_{\mathcal{L}}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}})\right] - \mathcal{R}_{\mathcal{L}}^*$, into several parts and bound them separately.

Proposition 3.5.1. *The excess risk is bounded above:*

$$\begin{aligned}\mathbb{E}_{\mathcal{A}}\left[\mathcal{R}_{\mathcal{L}}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}})\right] - \mathcal{R}_{\mathcal{L}}^* \leqslant & \left[\mathbb{E}_{\mathcal{A}}\left[\mathcal{R}_{\mathcal{L}}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}})\right]\right] + \left[\mathbb{E}_{\mathcal{A}}\left[\mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}})\right]\right] \\ & + \left[\mathcal{R}_{\mathcal{L}}(\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}})\right] + \left[\mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}})\right] \\ & + \left[\mathbb{E}_{\mathcal{A}}\left[\mathcal{R}_{\mathcal{L}}(\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}) + \lambda||\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}||^2\right] - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \lambda||\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}||^2\right] \\ & + D(\lambda) + \mathcal{R}_{\mathcal{L},\boldsymbol{\mathcal{H}}}^* - \mathcal{R}_{\mathcal{L}}^*\end{aligned} \tag{3.19}$$

Where $D(\lambda) = \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}) + \lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}|| - \mathcal{R}_{\mathcal{L},\boldsymbol{\mathcal{H}}}^*$. $\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}} = \boldsymbol{\alpha}^{*T}\boldsymbol{D}_y\boldsymbol{K}\left(\boldsymbol{f}_{TPR\text{-}CP\text{-}TT}(\mathfrak{X})\right)$, which is a function in $\boldsymbol{\mathcal{H}}^{\boldsymbol{\mathcal{A}}}$ with the coefficient vector being the optimal coefficient estimate from CP-STM.

Notice that $\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}$ is different from $\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}$ since its coefficients are estimated from the CP-STM model and original tensor data T_n. Recall that we use $\boldsymbol{\alpha}^*$ and $\boldsymbol{\alpha}$ in section 3.2 to denote optimal CP-STM and CP-STM coefficients. In other words, the coefficients of $\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}$ are the same as the $\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}$. However, their kernel basis functions will have different values, making them as two different functions. The proof of proposition 3.5.1 is provided in the appendix B.2. The proposition unveils the fact that the excess risk can be bounded by four types of risks:

1. Gaps between empirical risk and expected risk:

$$\left[\mathbb{E}_{\mathcal{A}}\left[\mathcal{R}_{\mathcal{L}}(g_n^\lambda) - \mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(g_n^\lambda)\right]\right] \quad \left[\mathcal{R}_{\mathcal{L}}(f_n^\lambda) - \mathcal{R}_{\mathcal{L},T_n}(f_n^\lambda)\right]$$

$$\left[\mathbb{E}_{\mathcal{A}}\left[\mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(f_{\mathcal{A},n}^\lambda) - \mathcal{R}_{\mathcal{L}}(f_{\mathcal{A},n}^\lambda)\right]\right] \quad \left[\mathcal{R}_{\mathcal{L},T_n}(f^\lambda) - \mathcal{R}_{\mathcal{L}}(f^\lambda)\right]$$

2. Extra risk brought by random projection:

$$\left[\mathbb{E}_{\mathcal{A}}\left[\mathcal{R}_{\mathcal{L}}(f_{\mathcal{A},n}^\lambda) + \lambda||f_{\mathcal{A},n}^\lambda||^2\right] - \mathcal{R}_{\mathcal{L}}(f_n^\lambda) - \lambda||f_n^\lambda||^2\right]$$

3. $D(\lambda)$, approximation error between the minimal regularized objective function and the risk of class optimal. This term depicts how regularized objective function approaches to the class optimal risk $\mathcal{R}^*_{\mathcal{L},\mathcal{H}}$ as the parameter λ vanishes.

4. $\mathcal{R}^*_{\mathcal{L},\mathcal{H}} - \mathcal{R}^*_{\mathcal{L}}$ measures the approximation error of the reproducing kernel Hilbert space $\mathcal{H}$. Later we show that with "nice" kernel functions, the functions in the RKHS $\mathcal{H}$ can approximate any measureable function as close as possible (in terms of infinite norm).

Next, we develop explicit bounds all these components.

In the following part, we suppose that all the conditions listed below hold.

AS.1 The loss function $\mathcal{L}$ is $C(W)$ local Lipschitz continuous in the sense that for $|a| \leqslant W < \infty$ and $|b| \leqslant W < \infty$

$$|\mathcal{L}(a, y) - \mathcal{L}(b, y)| \leqslant C(W)|a - b|$$

In addition, we need $\sup_{y \in \{1,-1\}} \mathcal{L}(0, y) \leqslant L_0 < \infty$.

AS.2 The kernel functions $K^{(j)}(\cdot, \cdot)$ used to composite the coupled tensor kernel (3.2) are regular vector-based kernels satisfying the universal approximating property, see [107]. A kernel has this property if it satisfies the following condition. Suppose $\mathcal{X}$ is a compact subset of the Euclidean space $\mathbb{R}^p$, and $C(\mathcal{X}) = \{f : \mathcal{X} \to \mathbb{R}\}$ is the collection of all continuous functions defined on $\mathcal{X}$. The kernel function is also defined on $\mathcal{X} \times \mathcal{X}$, and its reproduction kernel Hilbert space (RKHS) is $\mathcal{H}$. Then $\forall g \in C(\mathcal{X})$, $\exists f \in \mathcal{H}$ such that $\forall \epsilon > 0$, $||g - f||_\infty = \sup_{x \in \mathcal{X}} |g(x) - f(x)| \leqslant \epsilon$.

AS.3 Assume the tensor kernel function (3.2) is bounded, i.e. $\sqrt{\sup K(\cdot,\cdot)} = K_{max} < \infty$. As a result, the projected kernel function (3.9) is also bounded by K_{max} for any arbitrary random projection.

AS.4 For each component $K^{(j)}(\cdot,\cdot)$ in the kernel function (3.2), we assume $K^{(j)}(a,b) = \boldsymbol{h}^{(j)}(||a-b||^2)$ or $\boldsymbol{h}^{(j)}(\langle a,b\rangle)$. $\boldsymbol{h} : \mathbb{R} \to \mathbb{R}$ are functions. We assume that all of them are $L_K^{(j)}$-Lipschitz continuous

$$|\boldsymbol{h}^{(j)}(t_1) - \boldsymbol{h}^{(j)}(t_2)| \leqslant L_K^{(j)}|t_1 - t_2|$$

where $t_1, t_2 \in \mathbb{R}^{I_j}$ are different CP components. Further, let $L_K = \max_{j=1,..,d} L_K^{(j)}$.

AS.5 For random projection tensors $\boldsymbol{\mathcal{A}} = \{\boldsymbol{A}^{(1)}, ..., \boldsymbol{A}^{(d)}\}$, suppose all the $\boldsymbol{A}^{(j)}$ have their elements identically independently distributed as $\mathcal{N}(0,1)$. The dimension of $\boldsymbol{A}^{(j)}$ is $P_j \times I_j$. For a $\delta_1 \in (0,1)$ and $\epsilon > 0$, we assume

$$P_j = O(\frac{[\log \frac{n}{\delta_1}]^{\frac{1}{d}}}{\epsilon^2}), \quad j = 1, 2, ..., d$$

ϵ is considered as the error or distortion caused by random projection.

AS.6 The hyper-parameter in the regularization term $\lambda = \lambda_n$ satisfies:

$$\lambda_n \to 0 \quad n\lambda_n \to \infty \quad n\lambda_n^2 \to \infty \quad \text{as} \quad n \to \infty$$

AS.7 For all the tensor data $\mathcal{X} = \sum_{k=1}^{r} \boldsymbol{x}_k^{(1)} \circ \boldsymbol{x}_k^{(2)} ... \circ \boldsymbol{x}_k^{(d)}$, assume $||\boldsymbol{x}_k^{(j)}||^2 \leqslant B_x < \infty$.

AS.8 Suppose that there is a CP random projection defined by $\boldsymbol{\mathcal{A}}$ such that the Bayes risk in the projected data, $\mathcal{R}^*_{\mathcal{L},\boldsymbol{\mathcal{A}}}$, remains unaltered

$$\mathcal{R}^*_{\mathcal{L},\boldsymbol{\mathcal{A}}} = \mathcal{R}^*_{\mathcal{L}}$$

$\mathcal{R}^*_{\mathcal{L},\boldsymbol{\mathcal{A}}} = \min_{\boldsymbol{f}} \mathcal{R}_{\mathcal{L},\boldsymbol{\mathcal{A}}}(f)$ where $\boldsymbol{f}$ is any measurable function mapping projected data into class assignments.

AS.9 For $D(\lambda)$, we assume there is a relation between $D(\lambda)$ and λ

$$D(\lambda) = c_\eta \lambda^\eta \qquad 0 < \eta \leqslant 1$$

c_η is a constant depending on η.

AS.10 Suppose the projection error ratio for each mode vanishes at rates depending on loss function.

$$\frac{\epsilon_n}{\lambda_n^q} \longrightarrow 0 \qquad n \longrightarrow \infty$$

Where ϵ_n is the ϵ in the assumption **AS.5**. For hinge loss $q = 1$ and square hinge loss $q = \frac{3}{2}$.

AS.11 The probability of projection error is diminishing with increase of sample size,

$$\delta_1 = O\big(n \exp(-n^{\frac{1}{d}})\big)$$

The assumption **AS.1**, **AS.3**, **AS.6**, and **AS.7** are commonly used in supervised learning problems with kernel tricks.(see, e.g. [139, 82, 128]) Assumption **AS.4** and **AS.5** are needed to help to establish the explicit bound on extra errors brought by random projections. Assumption **AS.10** further gives out the condition that the extra errors brought by random projections can converge to zero as n goes to infinity. Condition **AS.8** assumes that it is possible to learn the optimal decision rule from randomly projected tensors. The optimal risk is still achievable after random projection. This helps to align our results to the definition of consistency in [36], and guarantees that RPSTM are consistent if $\mathbb{E}_{\mathcal{A}}\left[\mathcal{R}_{\mathcal{L}}(\boldsymbol{g_n^\lambda})\right] \longrightarrow \mathcal{R}^*_{\mathcal{L}}$. A more detailed discussion about this condition is provided in the appendix B.3. Condition **AS.2** is the sufficient condition that the tensor kernel function (3.2) is universal (see proof in [89]), making $\mathcal{R}^*_{\mathcal{L},\boldsymbol{\mathcal{H}}} - \mathcal{R}^*_{\mathcal{L}}$ to be zero. Finally, condition **AS.9** guarantees that $\mathcal{R}^*_{\mathcal{L},\boldsymbol{\mathcal{H}}}$ has a minimizer, and $\boldsymbol{f^\lambda}$ converges to the minimizer as λ goes to zero. (See definition 5.14 and corollary 5.18 in the section 5.4 of [128])

With the assumption **AS.7**, the gaps between empirical risk and expected risk are easily bounded by the Hoeffding Inequality (see e.g. [36]). Also, result from [89] says $\mathcal{R}^*_{\mathcal{L},\boldsymbol{\mathcal{H}}} = \mathcal{R}^*_{\mathcal{L}}$ due to condition **AS.2**. There are only two terms in the proposition 3.5.1 left to be bounded. The extra risk from random projection and the approximation error $D(\lambda)$. For these two parts, our strategy is proving the convergence or risk under a single random projection first, and then use the dominant convergence theorem to show the convergence of expected risks. Condition **AS.11** entails that probability of projection as well as expected risk difference vanishes with increase in sample size (ℓ_1 convergence).

3.5.3 Price of Random Projection

Applying random projection in the training procedure is indeed doing a trade-off between prediction accuracy and computational cost. We give out an explicit upper bound on the extra risk brought by random projections.

Without taking expectation, the following proposition gives out an upper bound on the extra risk when a random projection $\boldsymbol{\mathcal{A}}$ is given.

Proposition 3.5.2. *Assume a tensor CP random projection is defined by $\boldsymbol{\mathcal{A}}$, whose components are generated independently and identically from a standard Gaussian distribution. With the assumptions **AS.1**, **AS.4**, **AS.5**, **AS.6**, and **AS.7**, for the ϵ^d described in **AS.5**. With probability $(1-2\delta_1)$ and $q=1$ for hinge loss, and $q=\frac{3}{2}$ for square hinge loss function respectively.*

$$|\mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}) + \lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}||^2 - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}) - \lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}||^2| = O(\frac{\epsilon^d}{\lambda^q})$$

where n is the size of training set, d is the number of modes of tensor.

The proof of this proposition is provided in the appendix B.4. The value of q depends on loss function as well as kernel and geometric configuration of data, which is discussed in the appendix. This proposition highlights the trade-off between dimension reduction and prediction risk. As the reduced dimension P_j is related to ϵ negatively, small P_j can make the term converges at a very slow rate.

3.5.4 Convergence of Risk

Now we summarize the previous results and establish the convergence of risk for RPSTM classifier under a single random projection. The following theorem unveils the explicit convergence rate of

RPSTM classifier model.

Theorem 3.5.2 (RPSTM Convergence Rate). *Suppose all the assumptions **AS.1** - **AS.8** hold. For* $\epsilon > 0$*, let the projected dimension* $P_j = \lceil 3r^{\frac{4}{d}}\epsilon^{-2}[log(n/\delta_1)]^2\rceil + 1$ *for each* $j = 1, 2, ..d$*. The excess risk of a RPSTM with a specific random projection is bounded with probability at least* $(1-2\delta_1)(1-\delta_2)$*, i.e.,*

$$\mathcal{R}_{\mathcal{L}}(\boldsymbol{g_n^\lambda}) - \mathcal{R}^*_{\mathcal{L}} \leqslant V(1) + V(2) + V(3)$$

- $V(1) = 12C(K_{max}\sqrt{\frac{L_0}{\lambda}}) \cdot K_{max}\frac{\sqrt{L_0}}{\sqrt{n\lambda}} + 9\tilde{\zeta}_\lambda\sqrt{\frac{\log(2/\delta_2)}{2n}} + 2\zeta_\lambda\sqrt{\frac{2\log(2/\delta_2)}{n}}$
- $V(2) = D(\lambda)$
- $V(3) = C_{d,r}\Psi \cdot [C(K_{max}\sqrt{\frac{L_0}{\lambda}}) + \lambda\Psi]\epsilon^d$

where

- $C(K_{max}\sqrt{\frac{L_0}{\lambda}})$ *is a constant depending on* $K_{max}\sqrt{\frac{L_0}{\lambda}}$.
- $\delta_1 \in (0, \frac{1}{2})$ *and* $\delta_2 \in (0, 1)$
- $\zeta_\lambda = \sup\{\mathcal{L}(\boldsymbol{f^\lambda}(\mathcal{X}), y) : (\mathcal{X}, y) \in \boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}, \}$
- $\tilde{\zeta}_\lambda = \sup\{\mathcal{L}(\boldsymbol{f}(\mathcal{X}), y) :$ *all* $\boldsymbol{f} : \boldsymbol{\mathcal{X}} \to \mathbb{R}, ||\boldsymbol{f}||_\infty \leqslant K_{max}\sqrt{\frac{L_0}{\lambda}}$*, and all* $(\mathcal{X}, y) \in \boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}\}$
- $C_{d,r} = (2L_K B_x^2)^d r^2$
- $\Psi = \sup\{||\boldsymbol{\alpha}||_1 = \sum_{i=1}^n |\alpha_i| : \boldsymbol{f}(\mathcal{X}) = \boldsymbol{\alpha}^T\boldsymbol{D}_y\boldsymbol{K}(\mathcal{X}) \in \boldsymbol{\mathcal{H}}\}$

We prove the theorem, and explain the terms listed above in appendix B.5. The symbol $\lceil\cdot\rceil$ means rounding a value to the nearest integer above its current value. The theorem provides an upper bound controlling the convergence of excessive risk for RPSTM, which holds with probability at least $(1-2\delta_1)(1-\delta_2)$. This probability is defined on the join distribution of three random variables, tensor data $\mathcal{X}$, labels y, and the random projection $\boldsymbol{\mathcal{A}}$. The component $(1-\delta_2)$ is from the randomness in sampling the training data T_n, and $(1-2\delta_1)$ is caused by random projection. It is clear that the projection error ϵ, random projection probability parameter δ_1, and projection dimension P_j are connected by equation $P_j = \lceil 3r^{\frac{4}{d}}\epsilon^{-2}[log(n/\delta_1)]^2\rceil + 1$. As a result, one can express the projection error as $\epsilon^d = r^2 log(\frac{n}{\delta_1})/\prod_{j=1}^{d} P_j$, which is a function of δ_1 when fixing the size of training data n and projected dimension P_j. To obtain a higher probability of getting upper bounds, one can consider decreasing δ_1 and allowing the projection error to increase. Alternative, one can choose a higher projected dimension P_j to have the same level of projection error, but a higher chance of bounding the excessive risk. It worth noting that P_j grows as sample size n goes to infinity. However, huge P_j will make our proposed model infeasible and prohibitive. Thus, the projection error ϵ should be replaced by ϵ_n in assumption **AS.10** to guarantee $P_j << I_j$.

Theorem 3.5.2 provides an upper bound in general to control the excess risk of RPSTM. In the theorem, there are few quantities related to the loss function $\mathcal{L}$. These terms can be further expressed with specific loss functions such as Hinge and Squared Hinge loss. The next two propositions extend the theorem 3.5.2 with Hinge and Squared Hinge loss, and provide explicit upper bound on RPSTM.

Proposition 3.5.3. *For square hinge loss, let* $\epsilon = (\frac{1}{n})^{\frac{\mu}{2d}}$ *for* $0 < \mu < 1$, *and* $\lambda = (\frac{1}{n})^{\frac{\mu}{2\eta+3}}$ *for some* $0 < \eta \leqslant 1$. *Assume* $P_j = \lceil 3r^{\frac{4}{d}} n^{\frac{\mu}{d}}[log(n/\delta_1)]^2\rceil + 1$ *for each mode* $j = 1, 2, ..., d$. *For some* $\delta_1 \in (0, \frac{1}{2})$ *and* $\delta_2 \in (0, 1)$, *with probability* $(1 - \delta_2)(1 - 2\delta_1)$

$$\mathcal{R}_{\mathcal{L}}(\boldsymbol{g_n^\lambda}) - \mathcal{R}^*_{\mathcal{L}} \leqslant C\sqrt{log(\frac{2}{\delta_2})(\frac{1}{n})^{\frac{\mu\eta}{2\eta+3}}} \tag{3.20}$$

Where C is a constant.

The rate of convergence is faster with increase in sample size, when high value of μ is chosen. For $\mu \longrightarrow 1$ the risk difference rate becomes $(\frac{1}{n})^{\frac{d}{5}}$. The proof of this result is in appendix B.6.

Proposition 3.5.4. *For hinge loss,Let* $\epsilon = (\frac{1}{n})^{\frac{\mu}{2d}}$ *for* $0 < \mu < 1$ *and* $\lambda = (\frac{1}{n})^{\frac{\mu}{2\eta+2}}$ *for some* $0 < \eta \leqslant 1$, $P_j = \lceil 3r^{\frac{4}{d}} n^{\frac{\mu}{d}}[log(n/\delta_1)]^2\rceil + 1$, *For some* $\delta_1 \in (0, \frac{1}{2})$ *and* $\delta_2 \in (0, 1)$ *with probability* $(1 - 2\delta_1)(1 - \delta_2)$

$$\mathcal{R}_{\mathcal{L}}(\boldsymbol{g_n^\lambda}) - \mathcal{R}^*_{\mathcal{L}} \leqslant C\sqrt{log(\frac{2}{\delta_2})(\frac{1}{n})^{\frac{\mu\eta}{2\eta+2}}} \tag{3.21}$$

Where C is a constant.

The rate of convergence is faster with increase in sample size, when high value of μ is chosen. For $\mu \to 1$ the risk difference rate becomes $(\frac{1}{n})^{\frac{d}{4}}$. The proof of this result is in appendix B.6.

Finally, we show the convergence of expected risk

Theorem 3.5.3 (Convergence of Expected Risk). *Suppose assumptions **AS.1 - AS.11** hold. The excess risk goes to zero in expectation as sample size increases, the $\mathbb{E}_{\mathcal{A}}$ denote expectation with respect to tensor random projection $\mathcal{A}$, $\mathbb{E}_n$ denote expectation with respect to uniform measure on samples*

$$\mathbb{E}_n|\mathbb{E}_{\mathcal{A}}[\mathcal{R}_{\mathcal{L}}(g_n^{\lambda})] - \mathcal{R}_{\mathcal{L}}^*| \to 0$$

This is the expected risk convergence building on top of our previous results. The proof is provided in the appendix B.7. This theorem concludes that the expected risk of RPSTM converges to the optimal Bayes risk under surrogate loss $\mathcal{L}$. With the aforementioned property about $\mathcal{L}$ and theorem 3.5.1, the RPSTM and our ensemble model TEC are statistically consistent.

3.6 Simulation Study

We provide a simulation study in this section to compare the empirical performance of our TEC model and some other classification methods. Both vector-based and tensor-based methods in the current literature are considered in this comparison. For vector-based methods, we include Gaussian-RBF SVM from [128], BudgetSVM from [38], Linear Discriminant Analysis from [48], and Random Forest from [21]. For tensor-based methods, we select few highly cited models including Direct General Tensor Discriminant Analysis (DGTDA) and Constrained Multilinear Discriminant Analysis (CMDA) from [92].

The synthetic tensor data are generated using the idea from [43], which creates CP tensors by generating random tensor factors and computing the sum of mulit-way outer products. To have a more comprehensive comparison, we also generate Tucker tensors (see [78]) to show how TEC performs when the tensors are not well approximated by CP decomposition. We listed out the data generating models below:

1. **F1 Model**: Low dimensional rank 1 tensor factor model with each components confirming the same distribution. Shape of tensors is $50 \times 50 \times 50$.

$$\mathcal{X}_1 = \boldsymbol{x}^{(1)} \circ \boldsymbol{x}^{(2)} \circ \boldsymbol{x}^{(3)} \quad \boldsymbol{x}^{(j)} \sim \mathcal{N}(\mathbf{0}, I_{50}), j = 1,2,3$$

$$\mathcal{X}_2 = \boldsymbol{x}^{(1)} \circ \boldsymbol{x}^{(2)} \circ \boldsymbol{x}^{(3)} \quad \boldsymbol{x}^{(j)} \sim \mathcal{N}(\mathbf{0.5}, I_{50}), j = 1,2,3$$

2. **F2 Model:** High dimensional rank 1 tensor with normal distribution in each component. Shape of tensors is $50 \times 50 \times 50 \times 50$.

$$\mathcal{X}_1 = \boldsymbol{x}^{(1)} \circ \boldsymbol{x}^{(2)} \circ \boldsymbol{x}^{(3)} \circ \boldsymbol{x}^{(4)} \quad \boldsymbol{x}^{(j)} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}^{(j)}), j = 1,2,3,4$$

$$\mathcal{X}_2 = \boldsymbol{x}^{(1)} \circ \boldsymbol{x}^{(2)} \circ \boldsymbol{x}^{(3)} \circ \boldsymbol{x}^{(4)} \quad \boldsymbol{x}^{(j)} \sim \mathcal{N}(\mathbf{1}, \mathbf{\Sigma}^{(j)}), j = 1,2,3,4$$

$$\mathbf{\Sigma}^{(1)} = \boldsymbol{I}, \quad \mathbf{\Sigma}^{(2)} = AR(0.7), \quad \mathbf{\Sigma}^{(3)} = AR(0.3), \quad \mathbf{\Sigma}^{(4)}_{i,j} = \min(i,j)$$

3. **F3 Model**: High dimensional rank 3 tensor factor model. Components confirm different Gaussian distribution. Shape of tensors is $50 \times 50 \times 50 \times 50$.

$$\mathcal{X}_1 = \sum_{k=1}^{3} \boldsymbol{x}_k^{(1)} \circ \boldsymbol{x}_k^{(2)} \circ \boldsymbol{x}_k^{(3)} \circ \boldsymbol{x}_k^{(4)} \quad \boldsymbol{x}_k^{(j)} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}), j = 1,2,3,4$$

$$\mathcal{X}_2 = \sum_{k=1}^{3} \boldsymbol{x}_k^{(1)} \circ \boldsymbol{x}_k^{(2)} \circ \boldsymbol{x}_k^{(3)} \circ \boldsymbol{x}_k^{(4)} \quad \boldsymbol{x}_k^{(j)} \sim \mathcal{N}(\mathbf{1}, \mathbf{\Sigma}), j = 1,2,3,4$$

$$\mathbf{\Sigma}^{(1)} = \boldsymbol{I}, \quad \mathbf{\Sigma}^{(2)} = AR(0.7), \quad \mathbf{\Sigma}^{(3)} = AR(0.3), \quad \mathbf{\Sigma}^{(4)}_{i,j} = \min(i,j)$$

4. **F4 Model**: Low dimensional rank 1 tensor factor model with components confirming different distributions. Shape of tensor is $50 \times 50 \times 50$.

$$\mathcal{X}_1 = \boldsymbol{x}^{(1)} \circ \boldsymbol{x}^{(2)} \circ \boldsymbol{x}^{(3)} \quad \text{each element of } \boldsymbol{x}^{(1)} \sim \Gamma(4,2),$$

$$\boldsymbol{x}^{(2)} \sim \mathcal{N}(\mathbf{1}, \boldsymbol{I}), \quad \text{each element of } \boldsymbol{x}^{(3)} \sim \boldsymbol{U}(1,2)$$

$$\mathcal{X}_2 = \boldsymbol{x}^{(1)} \circ \boldsymbol{x}^{(2)} \circ \boldsymbol{x}^{(3)} \quad \text{each element of } \boldsymbol{x}^{(1)} \sim \Gamma(6,2),$$

$$\boldsymbol{x}^{(2)} \sim \mathcal{N}(\mathbf{1}, \boldsymbol{I}), \quad \text{each element of } \boldsymbol{x}^{(3)} \sim \boldsymbol{U}(1,2)$$

5. **F5 Model**: A higher dimensional version of F4 model. Tensors are having four modes with dimension $50 \times 50 \times 50 \times 50$

$$\begin{aligned}
\mathcal{X}_1 =& \boldsymbol{x}^{(1)} \circ \boldsymbol{x}^{(2)} \circ \boldsymbol{x}^{(3)} \circ \boldsymbol{x}^{(4)} \quad \text{each element of } \boldsymbol{x}^{(1)} \sim \Gamma(4, 2), \\
& \boldsymbol{x}^{(2)} \sim \mathcal{N}(\mathbf{1}, \boldsymbol{I}), \quad \text{each element of } \boldsymbol{x}^{(3)} \sim \Gamma(2, 1), \\
& \text{each element of } \boldsymbol{x}^{(4)} \sim \boldsymbol{U}(3.5, 4.5) \\
\mathcal{X}_2 =& \boldsymbol{x}^{(1)} \circ \boldsymbol{x}^{(2)} \circ \boldsymbol{x}^{(3)} \circ \boldsymbol{x}^{(4)} \quad \text{each element of } \boldsymbol{x}^{(1)} \sim \Gamma(5, 2), \\
& \boldsymbol{x}^{(2)} \sim \mathcal{N}(\mathbf{1}, \boldsymbol{I}), \quad \text{each element of } \boldsymbol{x}^{(3)} \sim \Gamma(2, 1), \\
& \text{each element of } \boldsymbol{x}^{(4)} \sim \boldsymbol{U}(4.5, 5.5)
\end{aligned}$$

6. **T1 Model**: A Tucker model. $\boldsymbol{Z}^{(1)}, \boldsymbol{Z}^{(2)} \in \mathbb{R}^{50\times50\times50}$ with elements independently and identically distributed. The size of factor matrices are all 50 by 50.

$$\begin{aligned}
&\mathcal{X}_1 = \boldsymbol{Z}^{(1)} \times_1 \boldsymbol{\Sigma}^{(1)} \times_2 \boldsymbol{\Sigma}^{(2)} \times_3 \boldsymbol{\Sigma}^{(3)} \quad \text{each element of } \boldsymbol{Z}^{(1)} \sim \mathcal{N}(0, 1) \\
&\mathcal{X}_2 = \boldsymbol{Z}^{(2)} \times_1 \boldsymbol{\Sigma}^{(1)} \times_2 \boldsymbol{\Sigma}^{(2)} \times_3 \boldsymbol{\Sigma}^{(3)} \quad \text{each element of } \boldsymbol{Z}^{(2)} \sim \mathcal{N}(0.5, 1) \\
&\boldsymbol{\Sigma}^{(1)} = \boldsymbol{I}, \quad \boldsymbol{\Sigma}^{(2)}\text{Random Orthogonal Matrix} \quad \boldsymbol{\Sigma}^{(3)} = AR(0.7)
\end{aligned}$$

The models F1 - F5 generates CP tensors, whose components confirm various probability distributions. F3 is a rank-3 CP tensor model. T1 is a Tucker tensor models constructed using mode-wise product (see [78]). The mode-1 factor is an identity matrix, mode-2 factor is a randomly generated orthogonal matrix, and mode-3 factor is an auto-regression matrix. We call each classification problem using tensors generated from these models as classification tasks. Thus, there are six tasks in this simulation study.

For each synthetic data, we generate 100 samples from class 1 and another 100 samples from class 2. Each time, we first subsample the data to form a training set of size 160, then use the remaining 40 observations to form the test set. We conduct stratified sampling to form training and test sets so that the percentages of each class are the same in both training and test sets. The training set is used to train and validate classifiers. Then the classifiers with the optimal tuning parameters are evaluated on the test set. We record the percentage of true predictions, $\frac{\text{True Predictions}}{\text{Total Predictions}} \times 100\%$, as the accuracy of a classifier on the test set. The experiments are repeated for multiple times, and the mean and the standard deviation of accuracy over all repetitions are reported in the table 3.2 below. For fair comparison, all the computations are done on a desktop with a 12-core CPU and 32GB RAM. We record the average time cost for model estimation over all repetitions, and notate "NA" (Not Available) in the table if a classifier cannot be estimated with the limited resources. We believe this could give an overview about the capabilities of different classifiers when handling big tensor data. More technical details about this simulation study is provided in the appendix B.6.

Notice that in the table 3.2, we use TEC1 and TEC2 to denote TEC models estimated with Hinge and Square Hinge Loss. AAM, LLSM, and BSGD are three variants of SVM for scalable and high-dimensional data analysis from BudgetedSVM package. The first thing we observe from the simulation study is that all the vector-based methods fail to deliver result in F2, F3, and F5 due to memory insufficiency. We later test these models using the same simulation data but on a high performance cluster which has 128GB memory. Their performance and the comparison are included in the appendix B.9. Among the tensor-based methods, such space insufficiency would not be an issue since data storing in tensors can better utilize computer memories. However, CMDA method fail to provide results in F2, F3, and F5 as its optimization procedure takes extremely long time. This failure is not due to memory limitation but high time complexity. On the other hand, our TEC models utilize tensors to handle big data with limited memory, and provide results in all tasks with high efficiency. The CP decomposition and random projection techniques in our TEC model further reduces the number of elements to be stored in memory. Notice that although our original tensors have the same number of elements as the vectorized data, the tensor decomposition can be done independently on each data. As a result, when the data is in huge dimension, such as the data from F2, F3, and F5, we can process the tensor decomposition and random projection one by one, storing only the randomly projected CP factors in memory and recycling memory space by deleting the original tensor. This processing pipeline distinguishes itself from other dimension reduction techniques such as principle component analysis as it can process all observations in the data set independently. It does not require to load all the data at one time and then perform

Model	Methods	TEC1	TEC2	RBF-SVM	AAM	LLSVM
F1	Accuracy (%)	83.96	**85.70**	82.00	73.75	62.81
	STD (%)	3.85	3.29	3.12	6.36	12.94
	Time (s)	1.05	1.26	2.56	1.15	5.06
F2	Accuracy (%)	**98.08**	86.70	NA	NA	NA
	STD (%)	1.06	1.87	NA	NA	NA
	Time (s)	1.44	1.56	NA	NA	NA
F3	Accuracy (%)	96.78	**98.63**	NA	NA	NA
	STD (%)	2.71	1.72	NA	NA	NA
	Time (s)	12	12.6	NA	NA	NA
F4	Accuracy (%)	93.80	94.10	**94.13**	46.33	53.13
	STD (%)	2.20	2.08	3.65	8.12	14.96
	Time (s)	1.75	2.10	1.31	1.94	6.21
F5	Accuracy (%)	89.38	**89.78**	NA	NA	NA
	STD (%)	2.55	2.25	NA	NA	NA
	Time (s)	1.49	1.63	NA	NA	NA
T1	Accuracy (%)	100	100	100	84.13	100
	STD (%)	0.00	0.00	0.00	5.40	0.00
	Time (s)	1.05	1.26	2.55	1.59	6.17

Model	Methods	BSGD	LDA	RF	CMDA	DGTDA
F1	Accuracy (%)	79.84	83.75	68.45	55.25	64.25
	STD (%)	20.16	4.25	6.15	1.25	11.68
	Time (s)	7.62	5.12	1.10	21.50	0.57
F2	Accuracy (%)	NA	NA	NA	NA	81.50
	STD (%)	NA	NA	NA	NA	4.89
	Time (s)	NA	NA	NA	NA	195
F3	Accuracy (%)	NA	NA	NA	NA	93.75
	STD (%)	NA	NA	NA	NA	3.23
	Time (s)	NA	NA	NA	NA	198
F4	Accuracy (%)	57.75	82.88	84.50	80.75	77.50
	STD (%)	7.40	5.20	4.72	5.01	4.61
	Time (s)	18.68	5.21	0.89	22.35	0.56
F5	Accuracy (%)	NA	NA	NA	NA	77.25
	STD (%)	NA	NA	NA	NA	6.56
	Time (s)	NA	NA	NA	NA	1.93
T1	Accuracy (%)	100	100	100	85.71	85.00
	STD (%)	0.00	0.00	0.00	22.59	22.91
	Time (s)	1.42	5.12	0.45	22.85	0.58

Table 3.2: TEC Simulation Results I: Desktop with 32GB RAM

feature extraction and dimension reduction, making it appealing for extremely high-dimensional data analysis. Using only the projected tensor factors also makes the proposed TEC model finishing all the computation in a very short time. Empirical evidence in table 3.2 shows that the processing time is tremendously less than DGTDA and CMDA (no results due to high time complexity) in tasks F2, F3 and F5 where the tensor dimensions are huge.

Apart from the efficiency, the results in table 3.2 highlight the promising performance of our TEC models. In tasks F1, F4 and T1 where the data dimensions are low, our TEC models have the similar performance as the RBF-SVM and its variants in BudgetedSVM. In particular, our TEC with Square Hinge loss outperforms RBF-SVM and all other competitors with significantly higher accuracy rates in task F1. Their performances in F4 are still decent, providing much higher accuracy rates than other classifiers except RBF-SVM. Their accuracy rates in F4 are only 0.5% less than that of RBF-SVM in this task. In Tucker tensor classification task T1, our TEC models continuing providing as solid performance as all other classifiers. Although the classification task is relatively easy, it still can demonstrate the capability of our TEC models in handling tensors which are not well approximated by tensor CP decomposition. The performance advantage of TEC models are even more impressive in higher-dimensional tensor classification tasks F2, F3, and F5. Due to the fact that all vector-based classifiers fail to deliver results on the testing platform, we only compare TEC1 and TEC2 with tensor discriminant analysis DGTDA. In F2 and F5, TEC models have about 10% more average rates than DGTDA. This advantage reduces to 3% in task F3, however, is still significant.

In conclusion, the simulation study demonstrates computational efficiency as well as solid performance for our proposed tensor ensemble classifier TEC.

3.7 Real Data Analysis

In this section, we compare the performance of our proposed TEC models with other existing tensor-based classifiers reviewed in chapter 2. We continue using the two real data sets in chapter 2 for experiments. CP-STM with Hinge and Squared Hinge loss from [63], CMDA and DGTDA

Models	Accuracy	Precision	Sensitivity	Specificity	AUC
TEC1	$0.71_{0.04}$	$0.80_{0.09}$	$0.50_{0.07}$	$0.89_{0.05}$	$0.64_{0.19}$
TEC2	$0.73_{0.03}$	$0.84_{0.04}$	$0.52_{0.09}$	$0.91_{0.03}$	$\mathbf{0.66}_{0.19}$
CP-GLM	$0.58_{0.04}$	$0.58_{0.07}$	$1.00_{0.00}$	$0.00_{0.00}$	$0.50_{0.00}$
CMDA	$0.70_{0.03}$	$0.69_{0.05}$	$0.67_{0.09}$	$0.73_{0.10}$	$0.65_{0.17}$
DGTDA	$0.70_{0.02}$	$0.71_{0.02}$	$0.59_{0.06}$	$0.80_{0.01}$	$0.64_{0.18}$
CP-STM1	$0.73_{0.03}$	$1.00_{0.00}$	$0.41_{0.07}$	$1.00_{0.00}$	$0.64_{0.20}$
CP-STM2	$\mathbf{0.74}_{0.04}$	$1.00_{0.00}$	$0.43_{0.08}$	$1.00_{0.00}$	$0.65_{0.20}$

Table 3.3: Real Data: ADNI Classification Comparison II

from [92], and CP-GLM from [155] are included for comparison.

3.7.1 MRI Classification for Alzheimer's Disease

The first data set is ADNI MRI data from ADNI-1 screening session. The introduction of the data is provided earlier in section 2.4, and thus is omitted here. We randomly sample 80% of images from AD group and 80% from NC group to form the training set with size 321. AD is labeled as positive class, and NC is labeled as negative class. The rest images are used as test set to evaluate model performance. For each classification model, we evaluate its performance by calculating its accuracy, precision, sensitivity, and specificity on the test set. Such step is replicated for multiple times, and the average accuracy, precision, sensitivity, and specificity are reported in the table 3.3. The standard deviation of these performance metrics are also provided in the subscripts. Since the image data are already in tensors, we do not destroy their spatial structure and just compare tensor-base classifiers in this study.

The results in table 3.3 shows that all the tensor-based classifiers have close accuracy in prediction, while CP-STM2 and TEC2 are slightly better than the others. Although CP-STM with Squared Hinge loss has 0.01 more in accuracy rate than TEC models, the AUC of TEC2 is slightly greater. This comparison unveils that tensor compression through random projections may not affect the model classification accuracy negatively, while it can provide a much better computation efficiency.

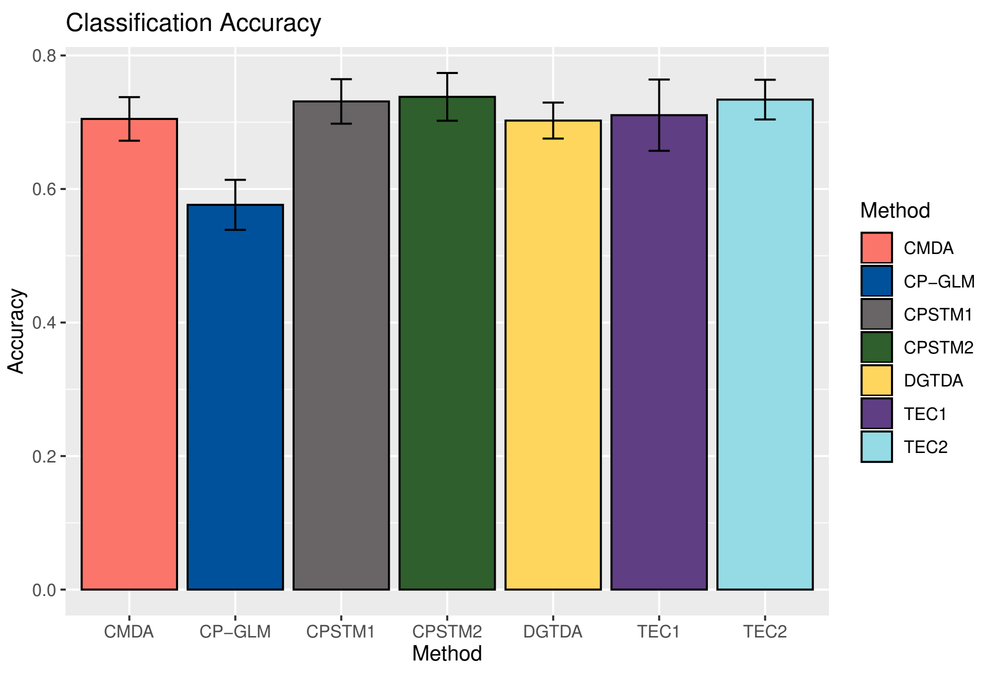


Figure 3.1: Real Data: ADNI Classification Result II

3.7.2 KITTI Traffic Image Classification

Our second real data application use the same traffic image data set from section 2.4. The introduction and data pre-processing is omitted and can be found in section 2.4. After imaging processing, the data set are divided into three groups which defines three classification tasks with levels of difficulties as easy, moderate, and hard. To maintain the balance between car and pedestrian images in the data sets, we randomly select 200 car images and 200 pedestrian images in each group for our numerical experiments. Pedestrian images are considered as the positive class data, and car images are negative class data.

The following procedures are repeated for 50 times in all three tasks with the balanced data sets. We randomly sample 80% of images as training set, and use the rest 20% data as the testing set. The sampling is conducted in a stratified way so that the proportion of pedestrian and car images are approximately same in both training and test set. Classification models are estimated and validated in training set. Then the models with selected tuning parameters are applied on the testing set. For each repetition, we calculate the same performance metrics, accuracy rates, precision (positive predictive rates), sensitivity (true positive rates), and specificity (true negative rates), for each classification method using the testing set. The average value of these rates and their standard deviations (in subscripts) are reported in the table 3.4. The areas under the ROC curves (AUC) are also reported for all the methods. Figure 3.2 shows the comparison of prediction accuracy rates, in which average accuracy rates of each method is shown by the bar charts and their standard deviations are shown by the error bars.

Task	Methods	Accuracy	Precision	Sensitivity	Specificity	AUC
Easy	CP-STM1	$0.85_{0.03}$	$0.84_{0.05}$	$0.85_{0.05}$	$0.84_{0.06}$	$0.85_{0.03}$
	CP-STM2	$0.83_{0.04}$	$0.83_{0.05}$	$0.83_{0.05}$	$0.83_{0.06}$	$0.83_{0.04}$
	TEC1	$\mathbf{0.88}_{0.04}$	$0.88_{0.06}$	$0.89_{0.05}$	$0.87_{0.07}$	$\mathbf{0.88}_{0.04}$
	TEC2	$0.74_{0.05}$	$0.75_{0.06}$	$0.73_{0.07}$	$0.76_{0.08}$	$0.74_{0.05}$
	CMDA	$0.63_{0.07}$	$0.58_{0.06}$	$0.95_{0.08}$	$0.30_{0.16}$	$0.63_{0.07}$
	DGTDA	$0.84_{0.04}$	$0.77_{0.04}$	$0.96_{0.03}$	$0.72_{0.07}$	$0.84_{0.04}$
	CP-GLM	$0.57_{0.05}$	$0.57_{0.06}$	$0.59_{0.07}$	$0.55_{0.09}$	$0.57_{0.05}$
Moderate	CP-STM1	$0.78_{0.05}$	$0.78_{0.06}$	$0.77_{0.07}$	$0.78_{0.07}$	$0.78_{0.05}$
	CP-STM2	$0.73_{0.06}$	$0.75_{0.07}$	$0.72_{0.08}$	$0.75_{0.09}$	$0.73_{0.06}$
	TEC1	$\mathbf{0.85}_{0.04}$	$0.85_{0.05}$	$0.84_{0.06}$	$0.85_{0.06}$	$\mathbf{0.85}_{0.04}$
	TEC2	$0.74_{0.04}$	$0.73_{0.04}$	$0.77_{0.08}$	$0.71_{0.06}$	$0.74_{0.04}$
	CMDA	$0.59_{0.05}$	$0.55_{0.04}$	$0.89_{0.11}$	$0.28_{0.13}$	$0.59_{0.05}$
	DGTDA	$0.74_{0.06}$	$0.72_{0.06}$	$0.79_{0.08}$	$0.69_{0.08}$	$0.74_{0.05}$
	CP-GLM	$0.53_{0.05}$	$0.53_{0.05}$	$0.54_{0.07}$	$0.52_{0.08}$	$0.53_{0.05}$
Hard	CP-STM1	$0.76_{0.04}$	$0.84_{0.06}$	$0.64_{0.07}$	$0.87_{0.05}$	$0.76_{0.04}$
	CP-STM2	$0.74_{0.04}$	$0.80_{0.06}$	$0.63_{0.07}$	$0.84_{0.06}$	$0.74_{0.04}$
	TEC1	$\mathbf{0.77}_{0.05}$	$0.78_{0.05}$	$0.75_{0.07}$	$0.78_{0.06}$	$\mathbf{0.77}_{0.05}$
	TEC2	$0.67_{0.05}$	$0.70_{0.05}$	$0.67_{0.07}$	$0.66_{0.07}$	$0.67_{0.04}$
	CMDA	$0.53_{0.04}$	$0.52_{0.02}$	$0.91_{0.09}$	$0.16_{0.12}$	$0.53_{0.04}$
	DGTDA	$0.72_{0.05}$	$0.68_{0.04}$	$0.84_{0.06}$	$0.60_{0.08}$	$0.72_{0.05}$
	CP-GLM	$0.51_{0.06}$	$0.51_{0.06}$	$0.54_{0.07}$	$0.49_{0.07}$	$0.51_{0.06}$

Table 3.4: Real Data: Traffic Image Classification II

Besides the conclusion we already obtained from section 2.4, the results in table 3.4 show that TEC1 model outperforms CP-STM1, the winner in our previous study, with a significant advantage in all three classification tasks. In particular, TEC model with Hinge loss has 7% more prediction accuracy rates than CP-STM1 in the moderate level classification task. This key observation can be a strong empirical evidence for our proposed TEC models supporting that the models can provide

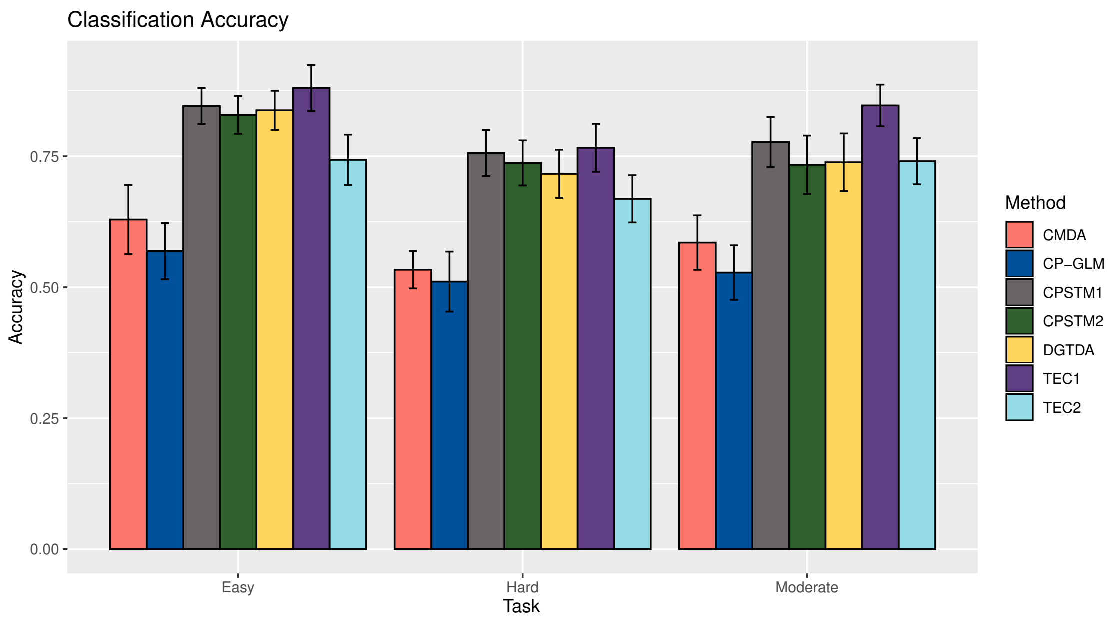


Figure 3.2: Real Data: Traffic Image Classification Result II

significantly better prediction accuracy rates when the data are noisy and the projected data are sufficient for classification.

3.8 Conclusion

We have proposed a tensor ensemble classifier with the CP support tensor machine and random projection in this work. The proposed method can handle high-dimensional tensor classification problems much faster comparing with the existing regularization based methods. Thanks to the Johnson-Lindenstrauss lemma and its variants, we have shown that the proposed ensemble classifier has a converging classification risk and can provide consistent predictions under some specific conditions. Tests with various synthetic tensor models and real data applications show that the proposed TEC can provide optimistic predictions in most classification problems.

Our primary focus in this work is on the classification applications on high-dimensional multi-way data such as images. Support tensor ensemble turns out to be an efficient way of analyzing such data. However, model interpretation has not been considered here. The features in the projected space are not able to provide any information about variable importance. Alternative approaches are possible for constructing explainable tensor classification models, but they are out of this article's scope. Besides that, selection for the dimension (size) of projected tensor P_js cannot be addressed well at this moment. Although our theoretical result points out the connection between the classification risk and $\min P_j$, discussion about how to set P_j for each mode of tensor may have to be developed in the future.

In conclusion, TEC offers a new option in tensor data analysis. The key features highlighted in work are that TEC can efficiently analyze high-dimensional tensor data without compromising the estimation robustness and classification risk. We anticipate that this method will play a role in future application areas such as neural imaging and multi-modal data analysis.

CHAPTER 4

COUPLED SUPPORT TENSOR MACHINE FOR MULTIMODAL NEUROIMAGING DATA

In this chapter, we consider a classification problem with multimodal tensor predictors. Multimodal neuroimaging data arise in various applications where information about the same phenomenon is acquired from multiple sensors and across different imaging modalities. Learning from multimodal data is of great interest in machine learning and statistics research as it offers the possibility of capturing complementary information among modalities. Multimodal learning increases model performance, explains the interdependence between heterogeneous data sources, discovers new insights that may not be available from a single modality, and improves decision-making. Recently, coupled matrix-tensor factorization has been introduced for multimodal data fusion to jointly estimate latent factors and identify complex interdependence among the latent factors. However, prior work on coupled matrix-tensor factorization mostly focuses on unsupervised learning, and very few of them utilize the jointly estimated latent factors for supervised learning. This paper considers the multimodal tensor data classification problem and proposes a Coupled Support Tensor Machine (C-STM), which is built upon the latent factors jointly estimated from Advanced Coupled Matrix Tensor Factorization (ACMTF). C-STM combines individual and shared latent factors with multiple kernels and estimates a maximal-margin classifier for coupled matrix tensor data. The classification risk of C-STM is shown to converge to the optimal Bayes risk, making it a statistically consistent rule. C-STM is validated through simulation studies as well as simultaneous EEG-fMRI analysis. The empirical evidence shows that C-STM can utilize information from multiple sources and provide a better classification performance than traditional unimodal classifiers.

4.1 Introduction

Advances in clinical neuroimaging and computational bioinformatics have dramatically increased our understanding of various brain functions using multiple modalities such as Magnetic

Resonance Imaging (MRI), functional Magnetic Resonance Imaging (fMRI), electroencephalogram (EEG), and Positron Emission Tomography (PET). The strong connection of these modalities to the patients' biological status and disease pathology suggests the great potential of their predictive power in disease diagnostics. Numerous studies using vector- and tensor-based statistical models illustrate how to utilize these imaging data at both the voxel- and Region-of-Interest (ROI) levels to develop efficient biomarkers that predict disease status. For example, [8] propose a classification model using functional connectivity MRI for autism disease with 89% diagnostic accuracy. [123] utilize network models and brain imaging data to develop novel biomarkers for Parkinson's disease. Many works in Alzheimer's disease research such as [109, 74, 100, 37, 94] use EEG, MRI and PET imaging data to predict patient's cognition and detect early-stage Alzheimer's diseases. Although these studies have provided impressive results, utilizing imaging data from single modality such as individual MRI sequences are known to have limited predictive capacity, especially in the early phases of the disease. For instance, [94] use brain MRI volumes from regions of interest to identify patients in early-stage Alzheimer's disease with 77% prediction accuracy. In recent years, it has been common to acquire multiple neuroimaging modalities in clinical studies such as simultaneous EEG-fMRI, MRI and fMRI. Even though each modality measures different physiological phenomena, they are interdependent and mutually informative. Learning from multimodal neuroimaging data may help integrate information from multiple sources and facilitate biomarker development in clinical studies. It also raises the need for novel supervised learning techniques for multimodal data in statistical learning literature.

The existing statistical approaches to multimodal data science are dominated by unsupervised learning methods. These methods analyze multimodal neuroimaging data jointly by performing decomposition, and try to discover how the common information is overlaid across different modalities. During optimization, the decomposed factors bridging two or more modalities are estimated to interpret connections between multimodal data. Examples of these methods include matrix-based joint Independent Component Analysis (ICA) [23, 56, 86, 97, 129, 6] which assume bilinear correlations between factors in different modalities. When tensors are utilized for multi-dimensional imaging modeling, various coupled matrix-tensor decomposition methods are established such as [5, 4, 6, 26, 27, 73, 110] which impose different types of soft or hard multilinear constrains between factors from different modalities. These methods further extend possible correlations between multimodal data, providing more flexibility in data modeling.

Current supervised learning approaches for multimodal data simply concatenate data modalities as extra features without exploring their interdependence. For example, [155, 93] build generalized regression models by appending tensor and vector predictors linearly for image prediction and classification. [114] develop a discriminant analysis by including tensor and vector predictors in a linear fashion. [91] propose an integrative factor regression for multimodal neuroimaging data assuming that data from different modalities can be decomposed into common factors. Another type of integration utilizes kernel tricks and combines information from multimodal data with multiple kernels. [55] provide a survey on various multiple kernel learning techniques for multimodal data fusion and classification with support vector machines. Combining kernels linearly or non-linearly in different modalities, instead of original data, provides more flexibility in information integration. [11] proposed a multiple kernel regression model with group lasso penalty, which integrates information by multiple kernels and selects the most predictive data modalities.

Despite these accomplishments, the current approaches have several shortcomings. First, they mainly focus on exploring the interdependence between multimodal neuroimaging data, ignoring the representative and discriminative power of the learned components. Thus, the methods cannot further bridge the imaging data to the patients' biological status, which is not helpful in biomarker development. Second, the supervised techniques integrate information primarily by data or feature concatenation without explicitly considering the possible correlations between different modalities. This lack of consideration for interdependence may cause issues like overfitting and parameter identifiability. Third, current multimodal approaches are mostly vector-based. Since many neuroimaging data are multi-dimensional, these approaches may fail to utilize the multi-way features as well as the multi-way interdependence between different modalities. Finally, although many empirical studies demonstrate the success of using multimodal data, there is a lack of mathematical and statistical clarity to the extent of generalizability and associated uncertainties. The absence of a sound statistical framework for multimodal data analysis makes it impossible to interpret the generalization ability of a certain statistical model.

In this paper, we propose a two-stage Coupled Support Tensor Machine (C-STM) for multimodal tensor-based neuroimaging classification. The model accommodates current multimodal data science issues and provides a sound statistical framework to interpret the interdependence between modalities and quantify the model consistency and generalization ability. The **major contributions** of this work are:

1. To extract individual and common components from multimodal tensor data in the first stage using Advanced Coupled Matrix Tensor Factorization (ACMTF), and identify interdependence between multimodal data through latent factors.

2. To build a novel CP Support Tensor Machine with both the individual and common factors for classification. This new model is named Coupled Support Tensor Machine (C-STM).

3. To show the proposed model is a consistent classification rule.

A Matlab package is also provided in the supplemental material, including all functions for C-STM classification and detailed data processing pipeline. The rest part of this chapter is organized as follow. Section 4.2 reviews current approaches about coupled matrix tensor factorization and multiple kernel learning, which are the basis of this work. Section 4.3 introduce our classification model. The model estimation is presented in section 4.4 using nonlinear conjugate gradient descent optimization. A simulation study is presented in section 4.6 to compare the performance of multimodal classification with single modal classification, highlighting the benefits of using information from multiple sources. Then we adopt the C-STM model in a simultaneous EEG-fMRI data trial classification problem in section 4.7. The conclusion of this chapter is in section 4.8.

4.2 Related Work

In this section, we review some backgorund and prior work on tensor decomposition and support tensor machine. In this work, we denote numbers and scalars by letters such as x, y, N. Vectors are denoted by boldface lowercase letters, e.g. $\boldsymbol{a}, \boldsymbol{b}$. Matrices are denoted by boldface capital letters like $\boldsymbol{A}, \boldsymbol{B}$. Multi-dimensional tensors are denoted by boldface Euler script letters such as $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}$. The order of a tensor is the number of dimensions of the data hypercube, also known as ways or modes. For example, a scalar can be regarded as a zeroth-order tensor, a vector is a first-order tensor, and a matrix is a second-order tensor.

Let $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ be a tensor of order N, where $x_{i_1, i_2, \ldots, i_N}$ denotes the $(i_1, i_2, \ldots, i_N)$th element of the tensor. Vectors obtained by fixing all indices of the tensor except the one that corresponds to nth mode are called mode-n fibers and denoted as $\boldsymbol{x}_{i_1, \ldots i_{n-1}, i_{n+1}, \ldots i_N} \in \mathbb{R}^{I_n}$. The mode-$n$ unfolding of $\boldsymbol{\mathcal{X}}$ is defined as $\boldsymbol{\mathcal{X}}_{(n)} \in \mathbb{R}^{I_n \times \prod_{n'=1, n' \neq n}^{N} I_{n'}}$ where the mode-n fibers of the tensor $\boldsymbol{\mathcal{X}}$ are the columns of $\boldsymbol{\mathcal{X}}_{(n)}$ and the remaining modes are organized accordingly along the rows.

4.2.1 CP Decomposition

Let $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_d}$ be a tensor with d modes. Rank-r Canonical/Polyadic (CP) decomposition of $\boldsymbol{\mathcal{X}}$ is defined as:

$$\boldsymbol{\mathcal{X}} \approx \sum_{k=1}^{r} \zeta_k \cdot \boldsymbol{x}_k^{(1)} \circ \boldsymbol{x}_k^{(2)} \ldots \circ \boldsymbol{x}_k^{(d)} = [\![\boldsymbol{\zeta}; \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(d)}]\!], \tag{4.1}$$

where $\boldsymbol{X}^{(j)} \in \mathbb{R}^{I_j \times r}, j \in \{1, .., d\}$ are defined as factor matrices whose columns are $\boldsymbol{x}_k^{(j)}$ and "$\circ$" represents the vector outer product. The right side of (4.1) is called Kruskal tensor, which is a convenient representation for CP tensors, see [81]. We denote a Kruskal tensor by $\boldsymbol{\mathcal{U}}_x = [\![\boldsymbol{\zeta}; \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(d)}]\!]$ where $\boldsymbol{\zeta} \in \mathbb{R}^r$ is a vector holding the weights of rank one components. In the special case of matrices, $\boldsymbol{\zeta}$ corresponds to singular values of a matrix. If all the elements in $\boldsymbol{\zeta}$ are 1, then $\boldsymbol{\zeta}$ can be dropped from the notation. In general, it is assumed that the rank r is small so that equation (4.1) is also called low-rank approximation for a tensor $\mathcal{X}$. Such an approximation can be estimated by an Alternating Least Square (ALS) approach, see [78].

Motivated by the fact that joint analysis of data from multiple sources can potentially unveil complex data structures and provide more information, Coupled Matrix Tensor Factorization (CMTF) ([2]) was proposed for multimodal data fusion. CMTF estimates the underlying latent factors for both tensor and matrix data simultaneously by taking the coupling between tensor and matrix data into account. This feature makes CMTF a promising model in analyzing heterogeneous data, which generally have different structures and modalities.

Let $\mathcal{X}_1 \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_d}$ and $\boldsymbol{X}_2 \in \mathbb{R}^{I_1 \times J_2}$. Assuming the factors from the first mode of the tensor $\mathcal{X}_1$ span the column space of the matrix $\boldsymbol{X}_2$, CMTF tries to estimate all factors by minimizing:

$$\boldsymbol{Q}(\mathfrak{U}_1, \mathfrak{U}_2) = \frac{1}{2}\|\mathcal{X}_1 - [\![\boldsymbol{X}_1^{(1)}, \boldsymbol{X}_1^{(2)}, \ldots \boldsymbol{X}_1^{(d)}]\!]\|_{\text{Fro}}^2 + \frac{1}{2}\|\boldsymbol{X}_2 - \boldsymbol{X}_2^{(1)}\boldsymbol{X}_2^{(2)\top}\|_{\text{Fro}}^2, \quad \text{s.t. } \boldsymbol{X}_1^{(1)} = \boldsymbol{X}_2^{(1)}, \tag{4.2}$$

where $\boldsymbol{X}_p^{(m)}$ are the factor matrices for modality p and mode m. The factor matrices $\boldsymbol{X}_1^{(1)} = \boldsymbol{X}_2^{(1)}$ are the coupled factors between tensor and matrix data. These factor matrices can also be represented in Kruskal form, $\mathfrak{U}_1 = [\![\boldsymbol{X}_1^{(1)}, \boldsymbol{X}_1^{(2)}, \ldots \boldsymbol{X}_1^{(d)}]\!]$ and $\mathfrak{U}_2 = [\![\boldsymbol{X}_2^{(1)}, \boldsymbol{X}_2^{(2)}]\!]$. By minimizing the objective function $\boldsymbol{Q}(\mathfrak{U}_1, \mathfrak{U}_2)$, CMTF estimates latent factors for both the tensor and matrix data jointly which allows it to utilize information from both modalities. [2] uses a gradient descent algorithm to optimize the objective function (4.2).

Although CMTF provides a successful framework for joint data analysis, it often fails to obtain a unique estimation when both shared and individual components exist. As a result, any further statistical analysis and learning from CMTF estimation will suffer from the large uncertainty in latent factors. To address this issue, [3] proposed Advanced Coupled Matrix Tensor Factorization (ACMTF) by introducing a sparsity penalty to the weights of latent factors in the objective function (4.2), and restricting the norm of the columns of the factors to be unity to allow unique results up to a permutation. This modification provides a more precise estimation of latent factors compared to CMTF, and makes it possible to develop further stable statistical models upon the estimated factors.

4.2.2 CP Support Tensor Machine (CP-STM)

CP-STM has been previously studied by [136, 63, 64] and use CP model to construct STMs. Given a collection of data $T_n = \{(\mathcal{X}_1, y_1), (\mathcal{X}_2, y_2), ..., (\mathcal{X}_n, y_n)\}$, where $\mathcal{X}_i \in \boldsymbol{\mathcal{X}} \subset \mathbb{R}^{I_1 \times I_2 \times ... \times I_d}$ are d-way tensors, $\boldsymbol{\mathcal{X}}$ is a compact tensor space which is a subspace of $\mathbb{R}^{I_1 \times I_2 \times ... \times I_d}$, and $y_i \in \{1, -1\}$ are binary labels. CP-STM assumes the tensor predictors have a CP structure, and can be classified by the function which minimizes the objective function $\lambda ||\boldsymbol{f}||^2 + \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\boldsymbol{f}(\mathcal{X}_i), y_i)$. By using tensor kernel function

$$K(\mathcal{X}_1, \mathcal{X}_2) = \sum_{l,m=1}^{r} \prod_{j=1}^{d} K^{(j)}(\boldsymbol{x}_{1,l}^{(j)}, \boldsymbol{x}_{2,m}^{(j)}), \tag{4.3}$$

where $\mathcal{X}_1 = \sum_{l=1}^{r} \boldsymbol{x}_{1,l}^{(1)} \circ .. \circ \boldsymbol{x}_{1,l}^{(d)}$ and $\mathcal{X}_2 = \sum_{l=1}^{r} \boldsymbol{x}_{2,l}^{(1)} \circ .. \circ \boldsymbol{x}_{2,l}^{(d)}$, the STM classifier can be written as

$$\boldsymbol{f}(\mathcal{X}) = \sum_{i=1}^{n} \alpha_i y_i K(\mathcal{X}_i, \mathcal{X}) = \boldsymbol{\alpha}^T \boldsymbol{D}_y \boldsymbol{K}(\mathcal{X}), \tag{4.4}$$

where $\mathcal{X}$ is a new d-way rank-r tensor of size $I_1 \times I_2 \times ... \times I_d$. In (4.4), $\boldsymbol{\alpha} = [\alpha_1, ..., \alpha_n]^T$ are the coefficients, $\boldsymbol{D}_y$ is a diagonal matrix whose diagonal elements are $y_1, .., y_n$, and $\boldsymbol{K}(\mathcal{X}) = [K(\mathcal{X}_1, \mathcal{X}), ..., K(\mathcal{X}_n, \mathcal{X})]^T$ is a column vector, whose entries are kernel values computed between training data and the new test data. We denote the collection of functions in the form of (4.4) with $\boldsymbol{\mathcal{H}}$, which is a functional space also known as Reproducing Kernel Hilbert Space (RKHS). The optimal CP-STM classifier, $\boldsymbol{f} \in \boldsymbol{\mathcal{H}}$, can be estimated by plugging function (4.4) into the objective function, and minimize it with Hinge or Squared Hinge loss. The coefficients of the optimal CP-STM model is denoted by $\boldsymbol{\alpha}^*$. The classification model is statistically consistent if the tensor kernel function satisfies the universal approximating property as shown in [89].

4.2.3 Multiple Kernel Learning

Multiple kernel learning (MKL) creates new kernels using a linear or non-linear combination of single kernels to measure inner products between data. Statistical learning algorithms such as support vector machine and kernel regression can then utilize the new combined kernels instead of single kernels to obtain better learning results and avoid the potential bias from kernel selection.

([55]) A more important and more related reason for using MKL is that different kernels can take inputs from various data representations possibly from different sources or modalities. Thus, combining kernels and using MKL is one possible way of integrating multiple information sources.

Given a collection of kernel functions $\{K_1(\cdot,\cdot),...K_m(\cdot,\cdot)\}$, a new kernel function can be constructed by

$$K(\cdot,\cdot) = \boldsymbol{f_\eta}(\{K_1(\cdot,\cdot),...K_m(\cdot,\cdot)\}|\boldsymbol{\eta}) \tag{4.5}$$

where $\boldsymbol{f_\eta}$ is a linear or non-linear function. $\boldsymbol{\eta}$ is a vector whose elements are weight coefficients for the kernel combination. Linear combination methods are the most popular multiple kernel learning, where the kernel function is parameterized as

$$\begin{aligned} K(\cdot,\cdot) &= \boldsymbol{f_\eta}(\{K_1(\cdot,\cdot),...K_m(\cdot,\cdot)\}|\boldsymbol{\eta}) \\ &= \sum_{l=1}^{m} \eta_l K_l(\cdot,\cdot) \end{aligned} \tag{4.6}$$

The weight parameters η_l can be simply assumed to be the same (unweighted) ([115, 14]), or be determined by looking at some performance measures from each kernel or data representation ([134, 118]). There are few more advanced approaches such as optimization-based, Bayesian approaches, and boosting approaches that can also be adopted ([84, 50, 140, 75, 76, 54, 32, 15]).

Motivated by the elegant framework and consistency property of CP-STM, we decide to extend it for multimodal tensor classification problems by combining it with ACMTF decomposition. We further consider linear combination (4.6) of kernels to integrate latent factors from multimodal data, and select the kernel weight parameters in a heuristic data driven way to construct our C-STM model. The preciseness of ACMTF may offer chance to capture the true latent structures from multimodal tensors resulting in better classification performance.

4.3 Methodology

Let $T_n = \{(\mathcal{X}_{1,1}, \boldsymbol{X}_{1,2}, y_1), ..., (\mathcal{X}_{n,1}, \boldsymbol{X}_{n,2}, y_n)\}$ be training data, where each sample $t \in \{1, \ldots, n\}$ has two data modalities $\mathcal{X}_{t,1}, \boldsymbol{X}_{t,2}$, and a corresponding binary label $y_t \in \{1, -1\}$. In this work, following [2], we assume that the first data modality is a third-order tensor, $\mathcal{X}_{t,1} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$,

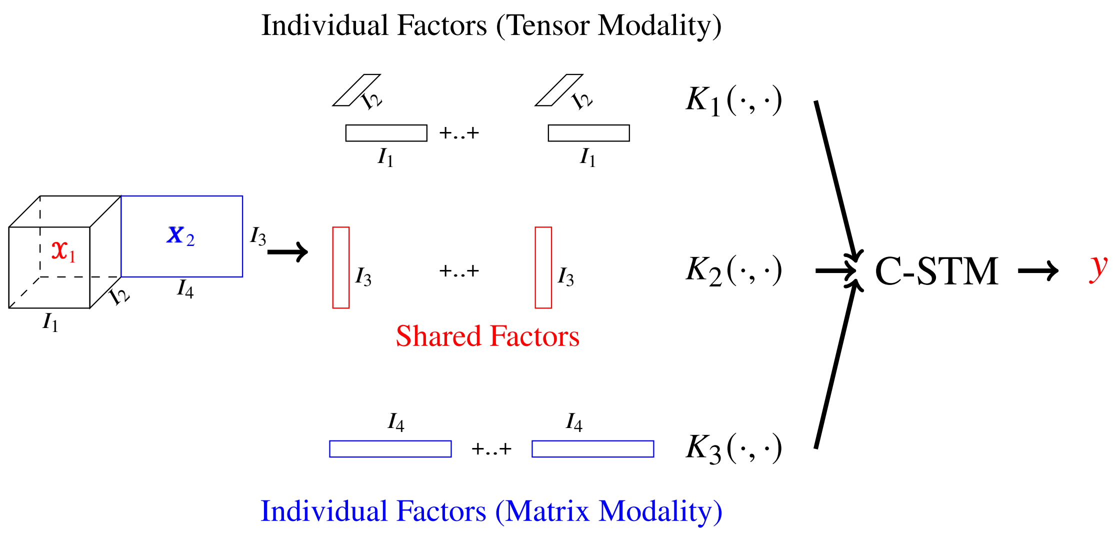


Figure 4.1: C-STM Model Pipeline

and the other is a matrix, $\boldsymbol{X}_{t,2} \in \mathbb{R}^{I_4 \times I_3}$. The third mode of $\mathcal{X}_{t,1}$ and the second mode of $\boldsymbol{X}_{t,2}$ are assumed to be coupled for each t, i.e., the factor matrix is assumed to be fully or partially shared across these modes. Utilizing this coupling, one can extract factors that better represent the underlying structure of the data, and preserve and utilize the discriminative power of the factors from both modalities. Our approach, C-STM (see Figure 4.1), consists of two stages: Multimodal tensor factorization, ACMTF, and coupled support tensor machine. We present both stages in this section.

4.3.1 ACMTF

In this stage, we aim to perform a joint factorization across two modalities for each training sample, t. Let $\mathfrak{U}_{t,1} = [\![\boldsymbol{\zeta}; \boldsymbol{X}_{t,1}^{(1)}, \boldsymbol{X}_{t,1}^{(2)}, \boldsymbol{X}_{t,1}^{(3)}]\!]$ denote the Kruskal tensor of $\mathcal{X}_{t,1}$, and $\mathfrak{U}_{t,2} = [\![\boldsymbol{\sigma}; \boldsymbol{X}_{t,2}^{(1)}, \boldsymbol{X}_{t,2}^{(2)}]\!]$ denote the singular value decomposition of $\boldsymbol{X}_{t,2}$. The weights of the columns of each factor matrix $\boldsymbol{X}_{t,p}^{(m)}$, where p is the modality index and m is the mode index, are denoted by $\boldsymbol{\zeta}$ and $\boldsymbol{\sigma}$. The norms of these columns are constrained to be 1 to avoid redundancy. The objective function of ACMTF decomposition is then given by:

$$\begin{aligned}
&\boldsymbol{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2}) = \gamma_1 \|\mathfrak{X}_{t,1} - [\![\boldsymbol{\zeta}; \boldsymbol{X}_{t,1}^{(1)}, \boldsymbol{X}_{t,1}^{(2)}, \boldsymbol{X}_{t,1}^{(3)}]\!]\|_{\text{Fro}}^2 + \gamma_2 \|\boldsymbol{X}_{t,2} - \boldsymbol{X}_{t,2}^{(1)} \boldsymbol{\Sigma} \boldsymbol{X}_{t,2}^{(2)\top}\|_{\text{Fro}}^2 \\
&\quad + \beta_1 \|\boldsymbol{\zeta}\|_1 + \beta_2 \|\boldsymbol{\sigma}\|_1 \qquad (4.7) \\
&\quad \text{s.t.} \quad \boldsymbol{X}_{t,1}^{(3)} = \boldsymbol{X}_{t,2}^{(2)}, \quad \|\boldsymbol{x}_{t,1,k}^{(1)}\|_2 = \|\boldsymbol{x}_{t,1,k}^{(2)}\|_2 = \|\boldsymbol{x}_{t,1,k}^{(3)}\|_2 = \|\boldsymbol{x}_{t,2,k}^{(1)}\|_2 = \|\boldsymbol{x}_{t,2,k}^{(2)}\|_2 = 1
\end{aligned}$$

$\forall k \in \{1, \dots, r\}$. $\boldsymbol{\Sigma}$ is a diagonal matrix whose elements are the singular values of the matrix $\boldsymbol{X}_{t,2}$ and $\boldsymbol{x}_{t,m,k}^{(j)} \in \mathbb{R}^{I_j}$ denotes the columns of the factor matrices of $\mathfrak{X}_{t,m}$. The objective function in (4.7) includes ℓ_1 penalties for weights in both tensor and matrix decomposition. Thus, the model identifies the shared and individual components. In our experiments, we set $\gamma_1 = \gamma_2 = 1$, and $\beta_1 = \beta_2 = 0.01$. These parameters can also be learned through optimization. These factors are then considered as extracted data representations for multimodal data, and used to predict the labels y_t in C-STM classifier.

4.3.2 Coupled Support Tensor Machine (C-STM)

C-STM uses the idea of multiple kernel learning and considers the coupled and uncoupled factors from ACMTF decomposition as various data representations. As a result, we use three different kernel functions to measure their inner products. One can think of these three kernels inducing three different feature maps transforming multimodal factors into different feature spaces. In each feature space, the corresponding kernel measures the similarity between factors in this specific data modality. The similarities of multimodal factors are then integrated by combining the kernel functions through a linear combination. This procedure is illustrated in Figure 4.1. In particular, the kernel K_1 is a tensor kernel (equation (4.3)) since the first individual factors are tensor CP factors. For two pairs of decomposed factors $(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})$ and $(\mathfrak{U}_{i,1}, \mathfrak{U}_{i,2})$, the kernel function for C-STM is defined as

$$K\Big((\mathfrak{X}_{t,1}, \boldsymbol{X}_{t,2}), (\mathfrak{X}_{i,1}, \boldsymbol{X}_{i,2})\Big) = K\Big((\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2}), (\mathfrak{U}_{i,1}, \mathfrak{U}_{i,2})\Big)$$

$$= \sum_{k,l=1}^{r} w_1 K_1^{(1)}(\boldsymbol{x}_{t,1,k}^{(1)}, \boldsymbol{x}_{i,1,l}^{(1)}) K_1^{(2)}(\boldsymbol{x}_{t,1,k}^{(2)}, \boldsymbol{x}_{i,1,l}^{(2)}) + w_2 K_2(\boldsymbol{x}_{t,1,k}^{(3)*}, \boldsymbol{x}_{i,1,l}^{(3)*}) + w_3 K_3(\boldsymbol{x}_{t,2,k}^{(1)}, \boldsymbol{x}_{i,2,l}^{(1)}). \quad (4.8)$$

$\boldsymbol{x}^{(3)*}_{t,1,k}$ is the average of the estimated shared factors $\frac{1}{2}[\boldsymbol{x}^{(3)}_{t,1,k} + \boldsymbol{x}^{(2)}_{t,2,k}]$ since ACMTF algorithm cannot guarantee $\boldsymbol{x}^{(3)}_{t,1,k} = \boldsymbol{x}^{(2)}_{t,2,k}$ numerically. w_1, w_2, and w_3 are three weight parameters combining the three kernel functions and can be tuned by cross-validation.

With kernel function (4.8), C-STM model tries to estimate a bivariate decision function $\boldsymbol{f}$ from a collection of functions $\boldsymbol{\mathcal{H}}$ such that

$$\boldsymbol{f} = \arg\min_{\boldsymbol{f}\in\boldsymbol{\mathcal{H}}} \quad \lambda \cdot ||\boldsymbol{f}||^2 + \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\boldsymbol{f}(\mathcal{X}_i), y_i) \tag{4.9}$$

where $\mathcal{L}(\mathcal{X}_i, y_i) = \max\left(0, 1 - \boldsymbol{f}(\mathcal{X}_i) \cdot y_i\right)$ is Hinge loss. $\boldsymbol{\mathcal{H}}$ is defined as the collection of all functions in the form of

$$\boldsymbol{f}(\mathcal{X}_1, \boldsymbol{X}_2) = \sum_{t=1}^{n} \alpha_i y_i K((\mathcal{X}_{t,1}, \boldsymbol{X}_{t,2}), (\mathcal{X}_1, \boldsymbol{X}_2)) = \boldsymbol{\alpha}^T \boldsymbol{D}_y \boldsymbol{K}(\mathcal{X}_1, \boldsymbol{X}_2) \tag{4.10}$$

due to the well-known representer theorem ([9]) for any pair of test data $(\mathcal{X}_1, \boldsymbol{X}_2)$ and for $\boldsymbol{\alpha} \in \mathbb{R}^n$. For all possible values of $\boldsymbol{\alpha}$, equation (4.10) defines the data collection $\boldsymbol{\mathcal{H}}$. $\boldsymbol{D}_y$ is a diagonal matrix whose diagonal elements are labels from the training data T_n. $\boldsymbol{K}(\mathcal{X}_1, \boldsymbol{X}_2)$ is a n by 1 column vector whose t-th element is $K\big((\mathcal{X}_{t,1}, \boldsymbol{X}_{t,2}), (\mathcal{X}_1, \boldsymbol{X}_2)\big)$. The optimal C-STM decision function, denoted by $\boldsymbol{f_n} = \boldsymbol{\alpha}^{*T} \boldsymbol{D}_y \boldsymbol{K}(\mathcal{X}_1, \boldsymbol{X}_2)$, can be estimated by solving the quadratic programming problem

$$\begin{aligned} &\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \quad \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{D}_y \boldsymbol{K} \boldsymbol{D}_y \boldsymbol{\alpha} - \boldsymbol{1}^T \boldsymbol{\alpha}, \\ &\text{S.T.} \quad \boldsymbol{\alpha}^T \boldsymbol{y} = 0, \quad 0 \le \boldsymbol{\alpha} \le \frac{1}{2n\lambda}, \end{aligned} \tag{4.11}$$

where $\boldsymbol{K}$ is the kernel matrix constructed by function (4.8). Problem (4.11) is the dual problem of (4.9), and its optimal solution $\boldsymbol{\alpha}^*$ also minimizes the objective function (4.9) when plugging functions in the form of (4.10). For a new pair of test points $(\mathcal{X}_1, \boldsymbol{X}_2)$, the class label is predicted as $\text{Sgn}\left[\boldsymbol{f_n}(\mathcal{X}_1, \boldsymbol{X}_2)\right]$.

4.4 Model Estimation

In this section, we first present the estimation procedure for tensor matrix decomposition (4.7), and then combine it with the classification procedure to summarize the algorithm for C-STM.

To satisfy the constraints in the objective function (4.7), we convert the function $\boldsymbol{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})$ into a differentiable and unconstrained form

$$\begin{aligned}\boldsymbol{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2}) =& \gamma_1 ||\mathfrak{X}_{t,1} - [\![\boldsymbol{\zeta}; \boldsymbol{X}_{t,1}^{(1)}, \boldsymbol{X}_{t,1}^{(2)}, \boldsymbol{X}_{t,1}^{(3)}]\!] ||_{\text{Fro}}^2 + \gamma_2 ||\mathfrak{X}_{t,2} - \boldsymbol{X}_{t,2}^{(1)} \boldsymbol{\Sigma} \boldsymbol{X}_{t,2}^{(2)^\top} ||_{\text{Fro}}^2 \\ &+ \tau ||\boldsymbol{X}_{t,1}^{(3)} - \boldsymbol{X}_{t,2}^{(2)}||_{Fro}^2 \\ &+ \sum_{k=1}^{r} \Big[\beta \sqrt{\zeta_k^2 + \epsilon} + \beta \sqrt{\sigma_k^2 + \epsilon} + \theta \big[(||\mathfrak{X}_{t,1,k}^{(1)}||_2 - 1)^2 + (||\mathfrak{X}_{t,1,k}^{(2)}||_2 - 1)^2 \\ &+ (||\mathfrak{X}_{t,1,k}^{(3)}||_2 - 1)^2 + (||\mathfrak{X}_{t,2,k}^{(1)}||_2 - 1)^2 + (||\mathfrak{X}_{t,2,k}^{(2)}||_2 - 1)^2 \big] \Big]\end{aligned} \tag{4.12}$$

ℓ_1 norm penalties in (4.7) are replaces with differentiable approximations. τ and θ are Lagrange multipliers. $\epsilon > 0$. This unconstrained optimization problem can be solved by nonlinear conjugate gradient descent ([2, 5, 110]). Let $\mathfrak{T}_t$ be the full tensor of $\mathfrak{U}_{t,1}$ (converting Kruskal tensor into multidimensional array form), and $\boldsymbol{M}_t = \boldsymbol{X}_{t,2}^{(1)} \boldsymbol{\Sigma} \boldsymbol{X}_{t,2}^{(2)^\top}$, the partial derivative of each latent factors can be derived as follow:

$$\frac{\delta \boldsymbol{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta \boldsymbol{X}_{t,1}^{(1)}} = \gamma_1 (\mathfrak{T}_t - \mathfrak{X}_{t,1})_{(1)} (\boldsymbol{\zeta}^\top \odot \boldsymbol{X}_{t,1}^{(3)} \odot \boldsymbol{X}_{t,1}^{(2)}) + \theta (\boldsymbol{X}_{t,1}^{(1)} - \bar{\boldsymbol{X}}_{t,1}^{(1)}) \tag{4.13}$$

$$\frac{\delta \boldsymbol{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta \boldsymbol{X}_{t,1}^{(2)}} = \gamma_1 (\mathfrak{T}_t - \mathfrak{X}_{t,1})_{(2)} (\boldsymbol{\zeta}^\top \odot X_{t,1}^{(3)} \odot \boldsymbol{X}_{t,1}^{(1)}) + \theta (\boldsymbol{X}_{t,1}^{(2)} - \bar{\boldsymbol{X}}_{t,1}^{(2)}) \tag{4.14}$$

$$\frac{\delta \boldsymbol{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta \boldsymbol{X}_{t,1}^{(3)}} = \gamma_1 (\mathfrak{T}_t - \mathfrak{X}_{t,1})_{(3)} (\boldsymbol{\zeta}^\top \odot X_{t,1}^{(2)} \odot \boldsymbol{X}_{t,1}^{(1)}) + \tau (\boldsymbol{X}_{t,1}^{(3)} - \boldsymbol{X}_{t,2}^{(2)}) + \theta (\boldsymbol{X}_{t,1}^{(3)} - \bar{\boldsymbol{X}}_{t,1}^{(3)}) \tag{4.15}$$

$$\frac{\delta \boldsymbol{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta \boldsymbol{X}_{t,2}^{(1)}} = \gamma_2 (\boldsymbol{M}_t - \boldsymbol{X}_{t,2}) \boldsymbol{X}_{t,2}^{(2)} \boldsymbol{\Sigma} + \theta (\boldsymbol{X}_{t,2}^{(1)} - \bar{\boldsymbol{X}}_{t,2}^{(1)}) \tag{4.16}$$

$$\frac{\delta \boldsymbol{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta \boldsymbol{X}_{t,2}^{(2)}} = \gamma_2 (\boldsymbol{M}_t - \boldsymbol{X}_{t,2})^\top \boldsymbol{X}_{t,2}^{(1)} \boldsymbol{\Sigma} + \tau (\boldsymbol{X}_{t,2}^{(2)} - \boldsymbol{X}_{t,1}^{(3)}) + \theta (\boldsymbol{X}_{t,2}^{(2)} - \bar{\boldsymbol{X}}_{t,2}^{(2)}) \tag{4.17}$$

$$\frac{\delta \boldsymbol{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta \zeta_k} = \gamma_1 (\mathfrak{T}_t - \mathfrak{X}_{t,1}) \times_1 \boldsymbol{x}_{t,1,k}^{(1)} \times_2 \boldsymbol{x}_{t,1,k}^{(2)} \times_3 \boldsymbol{x}_{t,1,k}^{(3)} + \frac{\beta}{2} \frac{\zeta_k}{\sqrt{\zeta_k^2 + \epsilon}}; k = 1, ..., r \tag{4.18}$$

$$\frac{\delta \boldsymbol{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta \sigma_k} = \gamma_2 \boldsymbol{x}_{t,2,k}^{(1)\top}(\boldsymbol{M}_t - \boldsymbol{X}_{t,2})\boldsymbol{x}_{t,2,k}^{(2)} + \frac{\beta}{2}\frac{\sigma_k}{\sqrt{\sigma_k^2 + \epsilon}}; k = 1, ..., r \tag{4.19}$$

Here $\mathfrak{T}_{(j)}$ denotes the mode-j unfolding of a tensor $\mathfrak{T}$. $\times_j$ denotes mode-wise product, and $\odot$ denotes Khatri-Rao product. (see Section 1.2). The matrix notation with a overline $\bar{\boldsymbol{M}}$ denotes a normalized matrix $\boldsymbol{M}$ whose columns are divided by their respective ℓ_2 norms. If we combine all the derived parts above, the partial derivative of the objective function is

$$\begin{aligned}\nabla \boldsymbol{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2}) = \Bigg[&\frac{\delta \boldsymbol{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta \boldsymbol{X}_{t,1}^{(1)}}, \frac{\delta \boldsymbol{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta \boldsymbol{X}_{t,1}^{(2)}}, \frac{\delta \boldsymbol{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta \boldsymbol{X}_{t,1}^{(3)}}, \\ &\frac{\delta \boldsymbol{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta \boldsymbol{X}_{t,2}^{(2)}}, \frac{\delta \boldsymbol{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta \zeta_1}, ... \frac{\delta \boldsymbol{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta \sigma_1}, ...\Bigg]^\top\end{aligned} \tag{4.20}$$

which is a $5 + 2r$ dimensional vector. If we use

$$(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2}) = [\boldsymbol{X}_{t,1}^{(1)}, \boldsymbol{X}_{t,1}^{(2)}, \boldsymbol{X}_{t,1}^{(3)}, \boldsymbol{X}_{t,2}^{(1)}, \boldsymbol{X}_{t,2}^{(2)}, \zeta_1, ...\sigma_1, ...]^\top$$

to denote the latent factors and the weights we have to estimate, the algorithm uses the negative gradient of $\boldsymbol{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})$ as the direction to update all the components in $(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})$ simultaneously. We first describe this estimation procedure in the algorithm 10. The algorithm keeps updating $(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})$ until convergence. Note that this is a non-convex optimization problem and its convergence properties has been discussed in [112, 117, 143, 144].

Once the factors for all data pairs in the training set T_n are estimated, we can create the kernel matrix using the kernel function in 4.8. By solving the quadratic programming problem (4.11), we can obtain the optimal decision function $\boldsymbol{f_n}$. This two-stage procedure for C-STM estimation is summarized in the algorithm 11 below.

4.5 Theory

We discuss the statistical property of C-STM in this section. Let's assume the risk of a decision function, $\boldsymbol{f}$, is $\mathcal{R}(\boldsymbol{f}) = \mathbb{E}_{\boldsymbol{\mathcal{X}}\times\boldsymbol{\mathcal{Y}}}\left[\mathbf{1}\{\boldsymbol{f}(\mathfrak{X}) \neq y\}\right]$, where $\boldsymbol{\mathcal{X}} \subset \mathbb{R}^{I_1 \times .. \times I_d}$ is a subspace of $\mathbb{R}^{I_1 \times .. \times I_d}$. $\boldsymbol{\mathcal{Y}} = \{1, -1\}$. The function $\mathbf{1}\{\cdot\}$ is an indicator function measuring the loss of classification

Algorithm 10 ACMTF Decomposition

```
1: procedure ACMTF
2:     Input: Multimodal data ($\mathfrak{X}_1, \boldsymbol{X}_2$) tensor rank r, $\eta$, maxiter
3:     $\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2} = \mathfrak{U}^0_{t,1}, \mathfrak{U}^0_{t,2}$                ▷ Initial value
4:     $\Delta_0 = -\nabla \boldsymbol{Q}(\mathfrak{U}^0_{t,1}, \mathfrak{U}^0_{t,2})$
5:     $\varphi_0 = \arg\min_\varphi \boldsymbol{Q}\left[(\mathfrak{U}^0_{t,1}, \mathfrak{U}^0_{t,2}) + \varphi\Delta_0\right]$
6:     $\mathfrak{U}^1_{t,1}, \mathfrak{U}^1_{t,2} = (\mathfrak{U}^0_{t,1}, \mathfrak{U}^0_{t,2}) + \varphi_0\Delta_0$
7:     $\boldsymbol{g}_0 = \Delta_0$
8:     while s < maxiter and $|\boldsymbol{Q}(\mathfrak{U}^s_{t,1}, \mathfrak{U}^s_{t,2}) - \boldsymbol{Q}(\mathfrak{U}^{s-1}_{t,1}, \mathfrak{U}^{s-1}_{t,2})| \geqslant \eta$ do
9:         $\Delta_{s+1} = -\nabla \boldsymbol{Q}(\mathfrak{U}^s_{t,1}, \mathfrak{U}^s_{t,2})$
10:        $\boldsymbol{g}_{s+1} = \Delta_{s+1} + \frac{\Delta^\top_{s+1}(\Delta_{s+1}-\Delta_s)}{-\boldsymbol{g}^\top_s(\Delta_{s+1}-\Delta_s)}\boldsymbol{g}_s$
11:        $\varphi_{s+1} = \arg\min_\varphi \boldsymbol{Q}\left[(\mathfrak{U}^s_{t,1}, \mathfrak{U}^s_{t,2}) + \varphi\boldsymbol{g}_{s+1}\right]$
12:        $\mathfrak{U}^{s+1}_{t,1}, \mathfrak{U}^{s+1}_{t,2} = (\mathfrak{U}^s_{t,1}, \mathfrak{U}^s_{t,2}) + \varphi_{s+1}\boldsymbol{g}_{s+1}$
13:    Output: $\mathfrak{U}^*_{t,1}, \mathfrak{U}^*_{t,2}$
```

Algorithm 11 Coupled Support Tensor Machine

```
1: procedure C-STM
2:     Input: Training set $T_n = \{(\mathfrak{X}_{1,1}, \boldsymbol{X}_{1,2}, y_1), ..., (\mathfrak{X}_{n,1}, \boldsymbol{X}_{n,2}, y_n)\}$, $\boldsymbol{y}$, kernel function $K$,
   tensor rank r, $\lambda$, $\eta$, maxiter
3:     for t = 1, 2,...n do
4:         $\mathfrak{U}^*_{t,1}, \mathfrak{U}^*_{t,2}$ = ACMDF(($\mathfrak{X}_{t,1}, \boldsymbol{X}_{t,2}$), tensor rank r, $\eta$, maxiter)
5:     Create initial matrix $\boldsymbol{K} \in \mathbb{R}^{n\times n}$
6:     for t = 1,...,n do
7:         for i = 1,...,i do
8:             $\boldsymbol{K}[i,t] = K\Big((\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2}), (\mathfrak{U}_{i,1}, \mathfrak{U}_{i,2})\Big)$        ▷ Kernel values
9:             $\boldsymbol{K}[i,t] = \boldsymbol{K}[t,i]$
10:    Solve the quadratic programming problem (4.11) and find the optimal $\boldsymbol{\alpha}^*$.
11:    Output: $\boldsymbol{\alpha}^*$
```

function $\boldsymbol{f}$. If there is a $\boldsymbol{f}^* : \boldsymbol{\mathcal{X}} \to \boldsymbol{\mathcal{Y}}$ from the collection of all measurable functions such that $\boldsymbol{f}^* = \arg\min \mathcal{R}(\boldsymbol{f})$, its risk is called the Bayes risk for the classification problem with data from $\boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}$. We denote the Bayes risk as $\mathcal{R}^* = \mathcal{R}(\boldsymbol{f}^*)$. With different training sets T_n, we can estimate a sequence of decision functions $\boldsymbol{f_n}$ under the same training procedure. This sequence of decision functions $\{\boldsymbol{f_n}\}$ is called a decision rule. A decision rule is statistically consistent if $\mathcal{R}(\boldsymbol{f_n})$ converges to the Bayes risk $\mathcal{R}^*$ as the size of training data n increases, see, e.g., [36]. Our next result shows that C-STM is a statistically consistent decision rule.

Proposition 4.5.1. *Given the tensor and matrix factors for all data in the domain, the classification risk of C-STM, $\mathcal{R}(\boldsymbol{f_n})$, converges to the optimal Bayes risk almost surely, i.e.*

$$\mathcal{R}(\boldsymbol{f_n}) \to \mathcal{R}^* \quad a.s. \quad n \to \infty$$

if the following conditions are satisfied:

AS.1 *The loss function $\mathcal{L}$ is self-calibrated, see [128], and is $C(W)$ local Lipschitz continuous in the sense that for $|a| \leqslant W < \infty$ and $|b| \leqslant W < \infty$, $|\mathcal{L}(a, y) - \mathcal{L}(b, y)| \leqslant C(W)|a - b|$. In addition, we need $\sup_{y \in \{1,-1\}} \mathcal{L}(0, y) \leqslant L_0 < \infty$.*

AS.2 *The kernel functions $K_1^{(1)}(\cdot, \cdot)$, $K_1^{(2)}(\cdot, \cdot)$, $K_2(\cdot, \cdot)$, and $K_3(\cdot, \cdot)$ used to compose the coupled tensor kernel (4.8) are regular vector-based kernels satisfying the universal approximating property. A kernel has this property if it satisfies the following condition. Suppose $\boldsymbol{\mathcal{X}}$ is a compact subset of the Euclidean space $\mathbb{R}^p$, and $C(\boldsymbol{\mathcal{X}}) = \{\boldsymbol{f} : \boldsymbol{\mathcal{X}} \to \mathbb{R}\}$ is the collection of all continuous functions defined on $\boldsymbol{\mathcal{X}}$. The kernel function is also defined on $\boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{X}}$, and its reproduction kernel Hilbert space (RKHS) is $\boldsymbol{\mathcal{H}}$. Then $\forall \boldsymbol{g} \in C(\boldsymbol{\mathcal{X}})$, $\exists \boldsymbol{f} \in \boldsymbol{\mathcal{H}}$ such that $\forall \epsilon > 0$, $||\boldsymbol{g} - \boldsymbol{f}||_\infty = \sup_{\boldsymbol{x} \in \boldsymbol{\mathcal{X}}} |\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x})| \leqslant \epsilon$.*

AS.3 *The kernel functions $K_1^{(1)}(\cdot, \cdot)$, $K_1^{(2)}(\cdot, \cdot)$, $K_2(\cdot, \cdot)$, and $K_3(\cdot, \cdot)$ used to compose the coupled tensor kernel (4.8) are all bounded, satisfying $\sqrt{\sup K(\cdot, \cdot)} \leqslant K_{max} < \infty$.*

AS.4 *The hyper-parameter in the regularization term $\lambda = \lambda_n$ satisfies $\lambda_n \to 0 \quad as \quad n \to \infty$ and $n\lambda_n \to \infty \quad as \quad n \to \infty$.*

This proposition is an extension of our previous result for the statistical consistency of CP-STM. The proof of this proposition is provided in Appendix C.1.

4.6 Simulation Study

We present a simulation study to demonstrate the benefit of utilizing C-STM with multimodal data in classification problems. To show the advantage of using multiple modalities, we compare with CP-STM from [63], Constrained Multilinear Discriminant Analysis (CMDA), and Direct General Tensor Discriminant Analysis (DGTDA) from [92]. These existing approaches can only take a single tensor / matrix as the input for classification. Thus, we apply these approaches on each modality separately and compare their classification performance with C-STM.

We generate synthetic data with two modalities using the idea from [43] as follows:

$$\mathcal{X}_{t,1} = \sum_{k=1}^{3} \boldsymbol{x}_{k,t,1}^{(1)} \circ \boldsymbol{x}_{k,t,1}^{(2)} \circ \boldsymbol{x}_{k,t,1}^{(3)}, \quad \boldsymbol{X}_{t,2} = \sum_{k=1}^{3} \boldsymbol{x}_{k,t,2}^{(1)} \circ \boldsymbol{x}_{k,t,2}^{(2)}, \tag{4.21}$$

where $\mathcal{X}_{t,1} \in \mathbb{R}^{30\times20\times10}$ and $\boldsymbol{X}_{t,2} \in \mathbb{R}^{50\times10}$ with ranks equal to 3. To generate data for the simulation study, we first generate the latent factors (vectors) from various multivariate normal distributions (with the parameters given in Table 4.1), and then use equation (4.21) to construct the tensors $\mathcal{X}_{t,1}$ and matrices $\boldsymbol{X}_{t,2}$. In Table 4.1, we use $c = 1, 2$ to denote data from two different classes. Eight different cases are considered in our simulation study. In cases 1 - 3, the discriminative information about the two classes is capture by one of the tensor factors and one of the matrix factors. This means that tensor and matrix data both contain class information (discriminative power) which may be different in the two modalities. Notice that the discriminative power in the tensor factor remains the same across cases 1 - 3, while the discriminative power in the matrix factor increases. Cases 4 and 5 assume the class information exists only in a single modality. In case 4, the distribution of one of the tensor factors is varied across classes and the discrimination power between the two classes is captured by the tensor factor. The discriminative factor becomes the matrix factor in case 5. In case 6, the difference between the two classes is captured by the shared factors, meaning that both tensor and matrix data modalities contain class information.

For each simulation case, we generate 50 pairs of tensor and matrix data from both classes, collecting 100 pairs of observations in total. We then randomly choose 20 samples as the testing

		Tensor Factors		Shared Factors	Matrix Factors
Simulation	c	$\boldsymbol{x}^{(1)}_{k,t,1}$	$\boldsymbol{x}^{(2)}_{k,t,1}$	$\boldsymbol{x}^{(3)}_{k,t,1} = \boldsymbol{x}^{(2)}_{k,t,2}$	$\boldsymbol{x}^{(1)}_{k,t,2}$
Case 1	1	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$
	2	$\boldsymbol{MVN}(\mathbf{1.5}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1.5}, \boldsymbol{I})$
Case 2	1	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$
	2	$\boldsymbol{MVN}(\mathbf{1.5}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1.75}, \boldsymbol{I})$
Case 3	1	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$
	2	$\boldsymbol{MVN}(\mathbf{1.5}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{2}, \boldsymbol{I})$
Case 4	1	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$
	2	$\boldsymbol{MVN}(\mathbf{2}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$
Case 5	1	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$
	2	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{2}, \boldsymbol{I})$
Case 6	1	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$
	2	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{2}, \boldsymbol{I})$	$\boldsymbol{MVN}(\mathbf{1}, \boldsymbol{I})$

Table 4.1: Distribution Specifications for Simulation; $\boldsymbol{MVN}$: multivariate normal distribution. $\boldsymbol{I}$: identity matrices. Bold numbers are vectors whose elements are all the same.

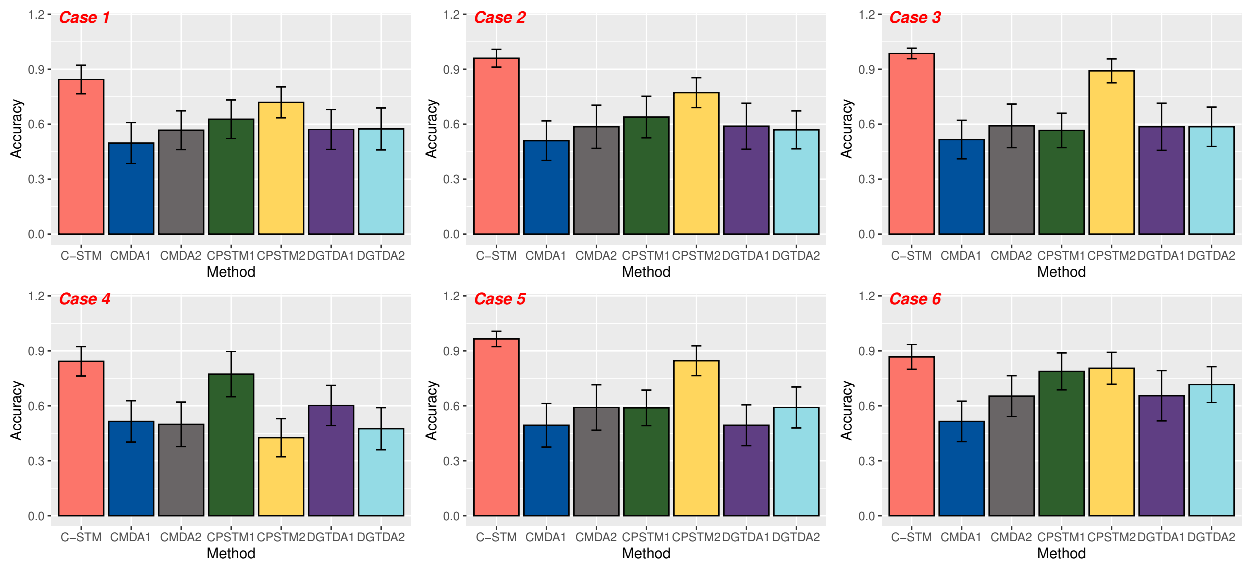


Figure 4.2: Simulation: Average accuracy(bar plot) with standard deviation (error bar)

set, and use the remaining data as the training set. The random selection of testing set is conducted in a stratified sampling manner such that the proportion of samples from each class remains the same in both training and testing sets. For all models, we report the model prediction accuracy, the proportion of correct predictions over total predictions, on the testing set as the performance metric. The random selection of training and testing data is repeated 50 times. The average prediction accuracy and standard deviation of these 50 repetitions for all cases are reported in Figure 4.2. The results of CP-STM, CMDA, and DGTDA with tensor data are denoted by CPSTM1, CMDA1, and DGTDA1, respectively. The results using matrix data are denoted by CPSTM2, CMDA2, and DGTDA2.

From Figure 4.2, we can conclude that our C-STM has a more favorable performance in this multimodal classification problem compared with single modality methods. Its accuracy rates are significantly larger than other methods in most cases. In particular, we can see that the accuracy rates of C-STM (pink) increase from case 1 to case 3, while the accuracy rates of CP-STM using only the tensor data remains the same. This is because the difference between the class mean vectors for the first tensor factor does not change from case 1 to 3. However, the difference between class mean vectors for the matrix factor increases. Due to this fact, both C-STM and CP-STM (yellow) which utilize matrix data have better performance for case 3. More importantly, C-STM always outperforms CP-STM with matrix data as it enjoys the extra class information from multiple modalities. In cases 4 and 5, where class information is in a single modality, the advantage of C-STM is not as significant as the previous cases, though its performance is still better than CP-STM. This indicates that C-STM can provide robust classification results even when the additional modalities do not provide any class information. In case 6, where the class information is from the shared factors, C-STM recovers the shared factors and provides significantly better classification accuracy. Through this simulation, we showed that C-STM has a clear advantage when using multimodal data in classification problems, and is robust to redundant data modalities. The performance of tensor discriminant analysis is not as good as C-STM and CP-STM because they are not designed for CP tensors.

4.7 Trial Classification for Simultaneous EEG-fMRI Data

In this section, we present the application of the proposed method on simultaneous EEG-fMRI data. The data is obtained from [141]. In this study, there is data from seventeen individuals (six females, average age 27.7) participating in three runs each of visual and auditory oddball paradigms. 375 (125 per run) total stimuli per task were presented for 200 ms each with a 2-3 s uniformly distributed variable inter-trial interval. A trial is defined as a time window in which subjects receive the stimuli and give responses. In the visual task, a large red circle on isoluminant gray backgrounds was considered as the target stimuli while a small green circle was the standard stimulus. For the auditory task, the standard and oddball stimuli were, respectively, 390 Hz pure tones and broadband sounds which sound like "laser guns". During the experiment, the stimuli were presented to all subjects, and their EEG and fMRI data are collected simultaneously and continuously. We obtain the data from OpenNeuro website (`https://openneuro.org/datasets/ds000116/versions/00003`). We utilize both EEG and fMRI in this data set with our C-STM model to classify stimulus types across trials.

We pre-process both the EEG and fMRI data with Statistical Parametric Mapping (SPM 12) ([10]) and Matlab. Details of data pre-processing are provided in Appendix C.2. For each trial, we construct a three-mode tensor corresponding to the EEG data for all subjects where the modes represent channel × time × subject denoted as $\mathcal{X}_{t,1} \in \mathbb{R}^{34\times121\times16}$. For fMRI data, there is only one 3D scan of fMRI collected from a single subject during each trial. Time mode does not exist in fMRI data because the trial duration is less than the repetition time of fMRI (time for obtaining a single 3D volume fMRI). We further extract fMRI volumes from voxels in the regions of interest (ROI) only for our study. ROI selection and data extraction are provided in Appendix C.2. We extract fMRI volumes from 178 voxels for auditory oddball tasks, and 112 voxels for auditory tasks. As a result, fMRI data for each trial are modeled by matrices whose rows and columns stand for voxels and subjects: $\boldsymbol{X}_{t,2} \in \mathbb{R}^{16\times178}$ for auditory task data, and $\boldsymbol{X}_{t,2} \in \mathbb{R}^{16\times112}$ for visual task data.

To classify trials with oddball and standard stimulus, we collect 140 multimodal data samples $(\mathcal{X}_{t,1}, \boldsymbol{X}_{t,2})$ from auditory tasks, and 100 samples from visual tasks. For both types of tasks, the

Task	Method	Accuracy	Precision	Sensitivity	Specificity	AUC
Auditory	C-STM	$\mathbf{0.89}_{0.05}$	$0.83_{0.07}$	$1.00_{0.00}$	$0.77_{0.11}$	$\mathbf{0.89}_{0.06}$
	CP-STM1	$0.80_{0.08}$	$0.71_{0.11}$	$1.00_{0.00}$	$0.60_{0.12}$	$0.78_{0.06}$
	CP-STM2	$0.83_{0.06}$	$0.76_{0.07}$	$0.99_{0.05}$	$0.65_{0.11}$	$0.82_{0.05}$
	CDMA1	$0.55_{0.10}$	$0.51_{0.09}$	$0.96_{0.09}$	$0.20_{0.21}$	$0.55_{0.06}$
	CDMA2	$0.67_{0.09}$	$0.61_{0.11}$	$0.92_{0.07}$	$0.46_{0.14}$	$0.70_{0.08}$
	DGTDA1	$0.55_{0.09}$	$0.51_{0.09}$	$0.94_{0.07}$	$0.23_{0.12}$	$0.59_{0.06}$
	DGTDA2	$0.67_{0.09}$	$0.60_{0.10}$	$0.90_{0.09}$	$0.46_{0.13}$	$0.68_{0.08}$
Visual	C-STM	$\mathbf{0.86}_{0.06}$	$0.82_{0.09}$	$0.93_{0.07}$	$0.77_{0.12}$	$\mathbf{0.86}_{0.06}$
	CP-STM1	$0.76_{0.08}$	$0.66_{0.11}$	$1.00_{0.00}$	$0.54_{0.12}$	$0.78_{0.05}$
	CP-STM2	$0.77_{0.08}$	$0.70_{0.11}$	$0.98_{0.08}$	$0.58_{0.17}$	$0.77_{0.07}$
	CDMA1	$0.53_{0.12}$	$0.52_{0.11}$	$0.94_{0.11}$	$0.11_{0.18}$	$0.54_{0.08}$
	CDMA2	$0.65_{0.13}$	$0.61_{0.14}$	$0.91_{0.09}$	$0.43_{0.19}$	$0.66_{0.09}$
	DGTDA1	$0.56_{0.11}$	$0.54_{0.11}$	$0.94_{0.06}$	$0.17_{0.12}$	$0.56_{0.07}$
	DGTDA2	$0.64_{0.10}$	$0.60_{0.13}$	$0.86_{0.10}$	$0.44_{0.18}$	$0.64_{0.07}$

Table 4.2: Real Data Result: Simultaneous EEG-fMRI Data Trial Classification (Mean of Performance Metrics with Standard Deviations in Subscripts)

numbers of oddball and standard trials are equal. We consider the trials with oddball stimulus as the positive class, and the trials with standard stimulus as the negative class. Similar to the simulation study, we select 20% of data as testing set, and use the remaining 80% for model estimation and validation. The classification accuracy, precision (positive predictive rate), sensitivity (true positive rate), specificity (true negative rate), and the area under the curve (AUC) of classifiers are calculated using the test set for each experiment. The experiment is repeated for multiple times, and the average accuracy, precision, sensitivity, and specificity, along with their standard deviations (in subscripts) are reported in Table 4.2. The single mode classifiers CPSTM, CMDA, and DGTDA are also applied on either EEG or fMRI data for comparison. The single mode classifiers applied on EEG data are denoted by appending the method name with the number “1" , and those applied on fMRI data are denoted by appending the method name with the number “2".

It can be seen that the classification accuracy for C-STM using multimodal data is higher than any classifier based on single modality with a significant improvement in terms of average accuracy rates and average AUC values. This improvement is observed for both auditory and visual tasks. Particularly, the accuracy rate of C-STM in visual task is 9% higher than CP-STM using fMRI data, the model with the second best performance. This significant performance improvement demonstrates the clear advantage of our C-STM with multimodal data, which is consistent with the previous conclusions from the simulation study. Similarly, the tensor discriminant analysis does not work as well as CP-STM and C-STM, which also agrees with our observations in the simulation study.

4.8 Conclusion

In this work, we have proposed a novel coupled support tensor machine classifier for multimodal data by combining advanced coupled matrix tensor factorization and support tensor machine. The most distinctive feature of this classifier is its ability to integrate features across different modalities and structures. The approach can simultaneously process matrix and tensor data for classification and can be extended to more than two modalities. Moreover, the coupled matrix tensor decomposition helps unveil the intrinsic correlation structure in different modalities, making it possible to integrate information from multiple sources efficiently. The newly designed kernel functions in C-STM serve as a feature-level information fusion, combining discriminant information from different modalities. In addition, the kernel formulation makes it possible to utilize the most discriminative features from each modality by tuning the weight parameters in the function. Our theoretical results demonstrate that the C-STM decision rule is statistically consistent.

An important theoretical extension of our approach would be the development of excess risk for C-STM. In particular, we look for an explicit expression for the excess risk in terms of data factors from multiple modalities to quantify the contribution of each modality to minimizing the excess risk. By doing so, we are able to interpret the importance of each data modality in classification tasks. In addition, quantifying the uncertainty of tensor and matrix factor estimation and their impact on the excess risk will build the foundation to the next level statistical inference. Another possible future work can be learning the weight parameters in kernel function via optimization problems in the algorithmic aspect. As [55] introduced, the weights in the kernel function can be further estimated by including a group lasso penalty in the objective function. Such a weight estimation procedure can identify the significant data components and reduce the burden of parameter selection.

In conclusion, we believe C-STM offers many encouraging possibilities for multimodal data integration and analysis. Its ability to handle multimodal tensor inputs will make it appropriate in many advanced data applications in neuroscience research.

APPENDICES

APPENDIX A

APPENDIX FOR CHAPTER 2

A.1 Proof of Proposition 2.3.1

Proof. The proof of this theorem is quite straightforward. We need to use the tensor product space defined in section 1.2. In addition, since we are discussing general tensor product, we will use $\otimes$ to denote it. $\otimes$ can be replaced with outer product $\circ$ or Kharti-Rao product $\odot$ when facing specific vector or matrix data. The proof will still holds.

Let $\mathcal{V}^{(j)}, j = 1,,,.d$ be compact subsets of $\mathbb{R}^{I_j}, j = 1, ..., d$. The tensor product of these subsets $\boldsymbol{\mathcal{X}} = \otimes_{j=1}^{d} \mathcal{V}^{(j)}$ will be again a compact subspace of the tensor space $\mathbb{R}^{I_1 \times \ldots \times I_d}$. Let $\boldsymbol{\mathcal{K}}(\boldsymbol{\mathcal{X}})$ be the kernel sections of tensor kernels we defined in equation (2.2), and $C(\boldsymbol{\mathcal{X}}) = \{\boldsymbol{f} : \boldsymbol{\mathcal{X}} \to \mathbb{R}\}$ be the collection of all continuous real-valued functions mapping CP tensors to scalars. We have to prove for any $\boldsymbol{f} \in C(\boldsymbol{\mathcal{X}})$, there exist an approximation in $\boldsymbol{\mathcal{K}}(\boldsymbol{\mathcal{X}})$. We will show such kinds of approximation exists.

If $\boldsymbol{f} \in C(\boldsymbol{\mathcal{X}})$, it has $\sup_{\boldsymbol{\mathcal{X}}} |\boldsymbol{f}| < \infty$ due to continuity. Further, since $\boldsymbol{f}$ is defined on $\boldsymbol{\mathcal{X}}$ and continuous, then $\boldsymbol{f} \in C(\boldsymbol{\mathcal{X}})$ and can be written as

$$\boldsymbol{f} = \sum_{k=1}^{r} \lambda_k \boldsymbol{f}_k^{(1)} \otimes \boldsymbol{f}_k^{(2)} \ldots \otimes \boldsymbol{f}_k^{(d)} + \epsilon \tag{A.1}$$

where $\boldsymbol{f}_k^{(j)}$ are continuous function defined on $\mathcal{V}^{(j)}, j = 1,,,.d$. This decomposition exists due to the fact that $\boldsymbol{f}$ is defined on $\boldsymbol{\mathcal{X}}$. As a result, $\boldsymbol{f}$ belongs to the functional tensor product space $\{\boldsymbol{f} : \boldsymbol{\mathcal{X}} \to \mathbb{R}\}$ in definition 1.2.2. It can also be exlpained by the fact that $\boldsymbol{f}$ is continuous on a compact space, thus it is multilinear and has such a decomposition (Lemma 4.30 from [59]). ϵ is a reminder here, and can be as small as possible since $\boldsymbol{f}$ is uniformly bounded. For simplicity, we shall ignore the ϵ in the later proof for a while and mention it at the end. λ_k are bounded since $\boldsymbol{f}$ is bounded.

If in every mode of the kernel functions are universal, the kernel functions are universal. For each $\boldsymbol{f}_k^{(j)}, k = 1, ..., r; j = 1, ...d$, there is a function $\boldsymbol{g}_k^{(j)} \in \overline{span}\{K_x : \boldsymbol{x} \in \boldsymbol{\mathcal{V}}^{(j)}\}$, which is from the kernel sections of the corresponding mode, such that

$$\sup_{\boldsymbol{\mathcal{V}}^{(j)}} |\boldsymbol{f}_k^{(j)} - \boldsymbol{g}_k^{(j)}| < \epsilon \qquad k = 1, ..., r; j = 1, ...d \tag{A.2}$$

for any arbitrary $\epsilon > 0$. Then for

$$\boldsymbol{g} = \sum_{k=1}^{r} \lambda_k \boldsymbol{g}_k^{(1)} \otimes \boldsymbol{g}_k^{(2)} ... \otimes \boldsymbol{g}_k^{(d)} \tag{A.3}$$

We can have

$$\begin{aligned}\sup_{\boldsymbol{\mathcal{X}} \in \boldsymbol{X}} |\boldsymbol{f} - \boldsymbol{g}| &= \sup_{\boldsymbol{\mathcal{X}} \in \boldsymbol{X}} | \sum_{k=1}^{r} \lambda_k \boldsymbol{f}_k^{(1)} \otimes \boldsymbol{f}_k^{(2)} ... \otimes \boldsymbol{f}_k^{(d)} - \sum_{k=1}^{r} \lambda_k \boldsymbol{g}_k^{(1)} \otimes \boldsymbol{g}_k^{(2)} ... \otimes \boldsymbol{g}_k^{(d)} | \\ &= \sup_{\boldsymbol{\mathcal{X}} \in \boldsymbol{X}} \sum_{k=1}^{r} |\lambda_k| | \prod_{j=1}^{d} \boldsymbol{f}_k^{(j)}(\boldsymbol{x}^{(j)}) - \prod_{j=1}^{d} \boldsymbol{g}_k^{(j)}(\boldsymbol{x}^{(j)}) | \\ &\leqslant rd\epsilon \cdot \max(|\lambda_k|)\end{aligned} \tag{A.4}$$

The last step is because of a simple inequality $|a_1 a_2 - b_1 b_2| \leqslant |a_1||a_2 - b_2| + |b_2||a_1 - b_1|$, and universal property in definition 2.3.1. Since r, d, and λ_k are all bounded, let the ϵ becomes as small as possible, we have

$$\sup_{\boldsymbol{\mathcal{X}} \in \boldsymbol{X}} |\boldsymbol{f} - \boldsymbol{g}| \leqslant \epsilon \quad \forall \boldsymbol{f} \in C(\boldsymbol{X}) \tag{A.5}$$

for any arbitrary $\epsilon > 0$. The proof is completed. □

A.2 Proof of Theorem 2.3.1

Proof. The convergence in the theorem can be showed in two steps. Given the parameter λ, we denote

$$\boldsymbol{f_n^\lambda} = \arg\min_{\boldsymbol{f} \in \boldsymbol{\mathcal{H}}} \quad \lambda ||\boldsymbol{f}||^2 + \mathcal{R}_{\mathcal{L}, T_n}(\boldsymbol{f}) \qquad \boldsymbol{f^\lambda} = \arg\min_{\boldsymbol{f} \in \boldsymbol{\mathcal{H}}} \quad \lambda ||\boldsymbol{f}||^2 + \mathcal{R}_{\mathcal{L}}(\boldsymbol{f})$$

where

$$\mathcal{R}_{\mathcal{L}}(\boldsymbol{f}) = \mathbb{E}_{(\mathcal{X} \times \mathcal{Y})} \mathcal{L}(y, \boldsymbol{f}(\boldsymbol{\mathcal{X}})) = \int \mathcal{L}(y, \boldsymbol{f}(\boldsymbol{\mathcal{X}})) d\mathbb{P}$$

and

$$\mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f}) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}\big(y_i, \boldsymbol{f}(\mathcal{X}_i)\big)$$

$\mathcal{L}$ is a loss function. $\boldsymbol{f_n^{\lambda}}$ is the optimal classifier learned from training data, and $\boldsymbol{f^{\lambda}}$ is the optimal from the RKHS $\boldsymbol{\mathcal{H}}$ generated by tensor kernel function (2.2). Since the Bayes risk under loss function $\mathcal{L}$ is defined as $\mathcal{R}^* = \min_{\boldsymbol{f}:\boldsymbol{\mathcal{X}}\to\boldsymbol{\mathcal{Y}}} \mathcal{R}(\boldsymbol{f})$ over all functions defined on $\boldsymbol{\mathcal{X}}$, we can immediate show that

$$\begin{aligned}|\mathcal{R}(\boldsymbol{f^{\lambda}}) - \mathcal{R}^*| &\leqslant \mathbb{E}_{(\mathcal{X}\times\mathcal{Y})}|\mathcal{L}(y, \boldsymbol{f^{\lambda}}(\mathcal{X})) - \mathcal{L}(y, \boldsymbol{f^*}(\mathcal{X}))| \leqslant C(K_{max}) \sup|\boldsymbol{f^{\lambda}} - \boldsymbol{f^*}| \\ &\leqslant C(K_{max}) \cdot \epsilon\end{aligned} \tag{A.6}$$

This is the result of using condition **Con.1** and **Con.2** in the theorem. $\boldsymbol{f^{\lambda}}$ is in the RKHS and thus bounded by some constant depending on K_{max}. $\boldsymbol{f^*}$ is also continuous on compact subspace $\boldsymbol{\mathcal{X}}$ (because all the tensor components considered are bounded in condition **Con.1**) and thus is bounded. The universal approximating property in condition **Con.3** makes equation (A.6) vanishes as ϵ goes to zero. Thus, the consistency result can be established if we show $|\mathcal{R}(\boldsymbol{f_n^{\lambda}}) - \mathcal{R}(\boldsymbol{f^{\lambda}})|$ converges to zero. This can be done with Hoeffding equality ([36]) and Rademacher complexity (see Theorem B.5.1).

From the objective function (2.4), we have

$$\mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f_n}) + \lambda_n||\boldsymbol{f_n}||^2 \leqslant L_0 \tag{A.7}$$

under condition **Con.2** when we simply let $\boldsymbol{f} = 0$ as a naive classifier. Thus, $||\boldsymbol{f_n}|| \leqslant \sqrt{\frac{L_0}{\lambda_n}}$. Let $M_n = \sqrt{\frac{L_0}{\lambda_n}}$. $\boldsymbol{f_\epsilon} \in \boldsymbol{\mathcal{H}}$ such that $\mathcal{R}_{\mathcal{L}}(\boldsymbol{f_\epsilon}) \leqslant \mathcal{R}_{\mathcal{L}}(\boldsymbol{f^{\lambda}}) + \frac{\epsilon}{2}$. $||\boldsymbol{f_\epsilon}|| \leqslant M_n$ when n is sufficiently large. Due to condition **Con.4**, $\lambda_n \to 0$, making $M_n \to \infty$. Further notice that we introduce $\boldsymbol{f_\epsilon}$ since it is independent of n. As a result, its norm, even though is bounded by M_n, is a constant and is not changing with respect to n. By Rademacher complexity, the following inequality holds with probability at least $1-\delta$, where $0<\delta<1$

$$\begin{aligned}
\mathcal{R}_{\mathcal{L}}(\boldsymbol{f_n^\lambda}) &\leqslant \mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f_n^\lambda}) + \frac{2C(K_{max})M_n}{\sqrt{n}} + (L_0 + C(K_{max})M_n)\sqrt{\frac{\log 2/\delta}{2n}} \\
\text{$\boldsymbol{f_\epsilon}$ is not the optimal in training data} \quad &\leqslant \mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f_\epsilon}) + \lambda_n||\boldsymbol{f_\epsilon}||^2 - \lambda_n||\boldsymbol{f_n^\lambda}||^2 + \frac{2C(K_{max})M_n}{\sqrt{n}} \\
&+ (L_0 + C(K_{max})M_n)\sqrt{\frac{\log 2/\delta}{2n}} \\
\text{Drop } (\lambda_n||\boldsymbol{f_n^\lambda}||^2 > 0) \quad &\leqslant \mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f_\epsilon}) + \lambda_n||\boldsymbol{f_\epsilon}||^2 + \frac{2C(K_{max})M_n}{\sqrt{n}} \\
&+ (L_0 + C(K_{max})M_n)\sqrt{\frac{\log 2/\delta}{2n}} \\
\text{Rademacher Complexity again} \quad &\leqslant \mathcal{R}_{\mathcal{L}}(\boldsymbol{f_\epsilon}) + \lambda_n||\boldsymbol{f_\epsilon}||^2 + \frac{4C(K_{max})M_n}{\sqrt{n}} \\
&+ 2(L_0 + C(K_{max})M_n)\sqrt{\frac{\log 2/\delta}{2n}}
\end{aligned}$$

Let $\delta = \frac{1}{n^2}$, and N large such that for all $n > N$,

$$\lambda_n||\boldsymbol{f_\epsilon}||^2 + \frac{4C(K_{max})M_n}{\sqrt{n}} + 2(L_0 + C(K_{max})M_n)\sqrt{\frac{\log 2/\delta}{2n}} \leqslant \frac{\epsilon}{2}$$

The inequality exists because $||\boldsymbol{f_\epsilon}||$ is a constant with respect to n, and all other terms are converging to zero. Thus

$$\mathcal{R}_{\mathcal{L}}(\boldsymbol{f_n^\lambda}) \leqslant \mathcal{R}_{\mathcal{L}}(\boldsymbol{f_\epsilon}) + \frac{\epsilon}{2} \leqslant \mathcal{R}_{\mathcal{L}}(\boldsymbol{f^\lambda}) + \epsilon$$

with probability $1 - \frac{1}{n^2}$. We conclude that

$$\mathbb{P}(|\mathcal{R}_{\mathcal{L}}(\boldsymbol{f_n^\lambda}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f^\lambda})| \geqslant \epsilon) \to 0 \tag{A.8}$$

for any arbitrary ϵ. This establishes the weak consistency of CP-STM. For strong consistency, we consider for each n

$$\sum_{n=1}^{\infty}\mathbb{P}(|\mathcal{R}_{\mathcal{L}}(\boldsymbol{f_n^\lambda}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f^\lambda})| \geqslant \epsilon) \leqslant N - 1 + \sum_{n=1}^{\infty}\frac{1}{n^2} \leqslant \infty$$

By Borel-Cantelli Lemma ([42]), $\mathcal{R}_{\mathcal{L}}(\boldsymbol{f_n^\lambda}) \to \mathcal{R}_{\mathcal{L}}(\boldsymbol{f^\lambda})$ almost surely. The proof is finished. □

APPENDIX B

APPENDIX FOR CHAPTER 3

B.1 Proof for Proposition 3.3.1

Suppose a CP rank-r tensor $\mathcal{X} = [\![\boldsymbol{X}^1, .., \boldsymbol{X}^d]\!]$ is given with size $I_1 \times I_2 .. \times I_d$. With a rank-1 projection tensor $\mathcal{A}_p$, the CP tensor random projection (3.5) can be written as

$$\begin{aligned}
[\boldsymbol{f}_{\text{TPR-CP}}(\mathcal{X})]_p &=< [\![\boldsymbol{A}_p^{(1)}, ..., \boldsymbol{A}_p^{(d)}]\!], [\![\boldsymbol{X}^{(1)}, .., \boldsymbol{X}^{(d)}]\!] > \\
&=< \boldsymbol{a}_p^{(1)} \circ \boldsymbol{a}_p^{(2)} \circ ... \circ \boldsymbol{a}_p^{(d)}, \sum_{k=1}^{r} \boldsymbol{x}_k^{(1)} \circ \boldsymbol{x}_k^{(2)} \circ ... \circ \boldsymbol{x}_k^{(d)} > \\
&= \sum_{k=1}^{r} < \boldsymbol{a}_p^{(1)T}, \boldsymbol{x}_k^{(1)} > \circ < \boldsymbol{a}_p^{(2)T}, \boldsymbol{x}_k^{(2)} > ... \circ < \boldsymbol{a}_p^{(d)T}, \boldsymbol{x}_k^{(d)} >
\end{aligned} \tag{B.1}$$

$\boldsymbol{a}_p^{(j)}, \boldsymbol{x}_k^{(j)} \in \mathbb{R}^{I_j}, j = 1, ..., d$ are CP factors for the projection tensor $\mathcal{A}_p$ and CP tensor $\mathcal{X}$. $\boldsymbol{a}_p^{(j)} \sim \boldsymbol{MVN}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$ is a multivariate random variable whose elements are identically and independently distributed. Also, $\boldsymbol{a}_p^{(j)}$ are identically and independently distributed with different value of $p = 1, .., P$ and $j = 1, ..., d$. Now we consider the tensor-to-tensor random projection (3.6), and let $\boldsymbol{A}^{(j)} = (\boldsymbol{a}_1^{(j)T}, ..., \boldsymbol{a}_{P_j}^{(j)T})^T \in \mathbb{R}^{P_j \times I_j}$ be the random projection matrices in (3.6). Notice that the rows of matrices $\boldsymbol{A}^{(j)}$ are identically and independently distributed as the $\boldsymbol{a}_p^{(j)}$, since the elements in the matrices $\boldsymbol{A}^{(j)}$ are also identically and independently distributed as $\mathcal{N}(0, \sigma^2)$. The tensor-to-tensor CP random projection (3.6) is

$$\begin{aligned}
\boldsymbol{f}_{\text{TPR-CP-TT}}(\mathcal{X}) &= [\![\boldsymbol{A}^{(1)}\boldsymbol{X}^{(1)}, \boldsymbol{A}^{(2)}\boldsymbol{X}^{(2)}, ..., \boldsymbol{A}^{(d)}\boldsymbol{X}^{(d)}]\!] \\
&= \sum_{k=1}^{r} < \boldsymbol{A}^{(1)}, \boldsymbol{x}_k^{(1)} > \circ < \boldsymbol{A}^{(2)}, \boldsymbol{x}_k^{(2)} > ... \circ < \boldsymbol{A}^{(d)}, \boldsymbol{x}_k^{(d)} > \\
&= \sum_{k=1}^{r} \boldsymbol{v}_k^{(1)} \circ \boldsymbol{v}_k^{(2)} \circ ... \circ \boldsymbol{v}_k^{(d)}
\end{aligned} \tag{B.2}$$

$\boldsymbol{v}_k^{(j)} =< \boldsymbol{A}^{(j)}, \boldsymbol{x}_k^{(j)} > \in \mathbb{R}^{P_j}, j = 1, ..., d$ since it is just matrix vector multiplication. To show the equivalence between (3.5) and (3.6), we have to show that $[\boldsymbol{f}_{\text{TPR-CP-TT}}(\mathcal{X})]_{p_1, p_2, ..., p_d} = [\boldsymbol{f}_{\text{TPR-CP}}(\mathcal{X})]_p$ when a index mapping $\boldsymbol{\pi}$ is given, $\boldsymbol{\pi}(p_1, p_2, ..., p_d) = p$, and $\prod_{j=1}^{d} p_j = p$.

For an arbitrary $p = \prod_{j=1}^{d} p_j$ and $\boldsymbol{\pi}(p_1, p_2, ..., p_d) = p$, we can find the element in projected tensor with index $p_1, p_2, ..., p_d$ is

$$\begin{aligned}
[\boldsymbol{f}_{\text{TPR-CP-TT}}(\mathcal{X})]_{p_1,p_2,...,p_d} &= \sum_{k=1}^{r} \boldsymbol{v}_{p_1}^{(1)} \boldsymbol{v}_{p_2}^{(2)} ... \boldsymbol{v}_{p_d}^{(d)} \\
&= \sum_{k=1}^{r} < \boldsymbol{a}_{p_1}^{(1)T}, \boldsymbol{x}_k^{(1)} > \circ < \boldsymbol{a}_{p_2}^{(2)T}, \boldsymbol{x}_k^{(2)} > ... \circ < \boldsymbol{a}_{p_d}^{(d)T}, \boldsymbol{x}_k^{(d)} >
\end{aligned} \tag{B.3}$$

where $\boldsymbol{a}_{p_j}^{(j)} \in \mathbb{R}^{I_j}$ are rows of matrices $\boldsymbol{A}^{(j)}$. Since $\boldsymbol{a}_{p_j}^{(j)}$ are identically and independently distributed for all p_j, $\boldsymbol{a}_{p_j}^{(j)}$ are equivalent to the $\boldsymbol{a}_p^{(j)}$ in equation (B.1). Thus, the equation (B.3) is equivalent to the equation (B.1). The proof is finished. Indeed, the equivalence can be identified as follow: For each projection tensor $\boldsymbol{\mathcal{A}}_p$ in (3.5), $\boldsymbol{\mathcal{A}}_p = \boldsymbol{a}_{p_1}^{(1)} \circ ... \circ \boldsymbol{a}_{p_d}^{(d)}$, where $\boldsymbol{a}_{p_j}^{(j)} \in \mathbb{R}^{I_j}$ are p_j-th rows of matrices $\boldsymbol{A}^{(j)}$. The order is decided by the index mapping $\boldsymbol{\pi}(p_1, p_2, ..., p_d) = p$.

B.2 Proof of Proposition 3.5.1

We use the adding and subtraction trick to prove the proposition.

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\mathcal{A}}}\left[\mathcal{R}_{\mathcal{L}}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}})\right] - \mathcal{R}_{\mathcal{L}}^* = {} & [\mathbb{E}_{\boldsymbol{\mathcal{A}}}[\mathcal{R}_{\mathcal{L}}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L},T_n^{\boldsymbol{\mathcal{A}}}}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}})] \\
& + \mathbb{E}_{\boldsymbol{\mathcal{A}}}\left[\mathcal{R}_{\mathcal{L},T_n^{\boldsymbol{\mathcal{A}}}}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L},T_n^{\boldsymbol{\mathcal{A}}}}(\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \lambda||\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}||^2\right] \\
& + \mathbb{E}_{\boldsymbol{\mathcal{A}}}[\mathcal{R}_{\mathcal{L},T_n^{\boldsymbol{\mathcal{A}}}}(\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}})] \\
& + [\mathbb{E}_{\boldsymbol{\mathcal{A}}}[\mathcal{R}_{\mathcal{L}}(\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}) + \lambda||\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}||^2] - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \lambda||\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}||^2] \\
& + [\mathcal{R}_{\mathcal{L}}(\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}})] + [\mathcal{R}_{T_n}(\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) + \lambda||\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}||^2 - \mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f}^{\boldsymbol{\lambda}}) - \lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}||^2] \\
& + [\mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}})] + [\mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L},\boldsymbol{\mathcal{H}}}^* + \mathcal{R}_{\mathcal{L},\boldsymbol{\mathcal{H}}}^* - \mathcal{R}_{\mathcal{L}}^* + \lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}||^2] \\
\leqslant {} & \left|\mathbb{E}_{\boldsymbol{\mathcal{A}}}\left[\mathcal{R}_{\mathcal{L}}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L},T_n^{\boldsymbol{\mathcal{A}}}}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}})\right]\right| + \left|\mathbb{E}_{\boldsymbol{\mathcal{A}}}\left[\mathcal{R}_{\mathcal{L},T_n^{\boldsymbol{\mathcal{A}}}}(\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}})\right]\right| \\
& + \left[\mathcal{R}_{\mathcal{L}}(\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}})\right] + \left[\mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}})\right] \\
& + \left|\mathbb{E}_{\boldsymbol{\mathcal{A}}}\left[\mathcal{R}_{\mathcal{L}}(\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}) + \lambda||\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}||^2\right] - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \lambda||\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}||^2\right| \\
& + D(\lambda) + \mathcal{R}_{\mathcal{L},\boldsymbol{\mathcal{H}}}^* - \mathcal{R}_{\mathcal{L}}^*
\end{aligned}$$

Since $D(\lambda) = \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}) + \lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}|| - \mathcal{R}_{\mathcal{L},\boldsymbol{\mathcal{H}}}^*$, there are only two terms are dropped in the derivation.

- The first term dropped is $\mathbb{E}_{\mathcal{A}}[\mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \lambda||\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}||^2]$. As we explained in the paper, $\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}$ is the decision function but with coefficients estimated from CP-STM model. As a result, it is not the optimal of the objective function (3.8). Since $\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}$ minimizes the objective function (3.8) and $\lambda||\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}||^2 \geqslant 0$, we get

$$\begin{aligned} \mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \lambda||\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}||^2 &\leqslant \mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) + \lambda||\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}||^2 - \mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}) \\ &\quad - \lambda||\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}||^2 \\ (\text{Since} \quad \boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}} \quad \text{minimizes (3.8)}) &\leqslant 0 \end{aligned}$$

  The inequality holds for all random projection defined by random tensor $\boldsymbol{\mathcal{A}}$, so

$$\mathbb{E}_{\mathcal{A}}[\mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \lambda||\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}||^2] \leqslant 0$$

- The second term dropped is $[\mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) + \lambda||\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}||^2 - \mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f}^{\boldsymbol{\lambda}}) - \lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}||^2]$. Similar to the previous dropped term, this term is also less or equal to zero. As we defined, $\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}$ minimizes the objective function (3.1) that evaluates loss over the training data T_n. Even though $\boldsymbol{f}^{\boldsymbol{\lambda}}$ is the class optimal with infinite-size training data, its objective function still has a greater value than that of $\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}$. By comparing the values of objective function (3.1) on $\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}$ and $\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}$, we can see that $[\mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) + \lambda||\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}||^2 - \mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f}^{\boldsymbol{\lambda}}) - \lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}||^2] \leqslant 0$

By dropping these two non-positive terms, we prove the proposition.

B.3 Discussion on Assumptions AS.8

Assumption **AS.8** ensures that the Bayes risk remains unchanged after random projection. This assumption has also been made in [24]. This is a necessary condition to align our results with the definition of classification consistency.

For any arbitrary random projection $\boldsymbol{\mathcal{A}}$, it is obvious that $\mathcal{R}^*_{\mathcal{L},\boldsymbol{\mathcal{A}}} \geqslant \mathcal{R}^*_{\mathcal{L}}$. This is because the optimal Bayes risk is achieved by choosing any measureable function. A function compose random projection $\boldsymbol{\mathcal{A}}$ and a decision rule should be measureable, and thus should be considered when searching for Bayes rules. If $\mathcal{R}^*_{\mathcal{L},\boldsymbol{\mathcal{A}}} > \mathcal{R}^*_{\mathcal{L}}$, then smallest achievable risk in projected data will no longer be $\mathcal{R}^*_{\mathcal{L}}$. By definition, a decision rule learned from the projected data just have to reach to the $\mathcal{R}^*_{\mathcal{L},\mathcal{A}}$ to be consistent. This deviates from the result $\mathbb{E}_{\mathcal{A}}\left[\mathcal{R}_{\mathcal{L}}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}})\right] \to \mathcal{R}^*_{\mathcal{L}}$ we show in the paper. Thus, we need the condition to guarantees that $\mathbb{E}_{\mathcal{A}}\left[\mathcal{R}_{\mathcal{L}}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}})\right] \to \mathcal{R}^*_{\mathcal{L}}$ indicating RPSTM's consistency aligns with the definition.

In [33, 24], many examples satisfying this condition are provided. Typically, if there is an random projection $\boldsymbol{\mathcal{A}}$ such that $\mathbb{E}[y|\boldsymbol{f}_{\text{TPR-CP-TT}}(\mathcal{X})]$ and $\mathcal{X}$ are independent, then the condition is satisfied.

B.4 Proof of Proposition 3.5.2

Johnson-Lindenstrauss lemma gives concentration bound on the error introduced by random projection in a single mode. (e.g. see [72] and [34]) We first show how this property is applied at each mode of the tensor CP components in the following lemma.

Lemma B.4.1. *For each fixed mode $j = 1, 2, .., d$ and any two tensor CP factors $\boldsymbol{x}_1^{(j)}, \boldsymbol{x}_2^{(j)} \in \mathbb{R}^{I_j \times 1}$ among n training vectors, with probability at least $(1 - \delta_1)$ and the random projection matrices described in **AS.5**, we have*

$$\left| ||\boldsymbol{A}^{(j)}\boldsymbol{x}_1^{(j)} - \boldsymbol{A}^{(j)}\boldsymbol{x}_2^{(j)}||_2^2 - ||\boldsymbol{x}_1^{(j)} - \boldsymbol{x}_2^{(j)}||_2^2 \right| \leqslant \epsilon ||\boldsymbol{x}_1^{(j)} - \boldsymbol{x}_2^{(j)}||_2^2$$

Proof. The matrix $\boldsymbol{A}^j \in \mathbb{R}^{P_j \times I^j}$, where $P_j = O(\frac{\log \frac{n}{\delta_1}}{\epsilon^2})$. Under condition **AS.5**, the inequality holds due to the JL-property ([72, 34]). □

Next, we apply this lemma to multiple modes of tensor CP factors, and derive a bound for the difference between projected tensor kernel function (3.9) and tensor kernel function (3.2). We need the following lemma and corollary to derive the bound.

Lemma B.4.2. *Consider a $2d$ degree polynomial of independent Centered Gaussian or Rademacher random variables as $Q_{2d}(Y) = Q_{2d}(y_i, .., y_d)$. Then for some $\epsilon > 0$ and $\xi > 0$ constant.*

$$\mathbb{P}(|Q_{2d}(Y) - \mathbb{E}(Q_{2d}(Y))| > \epsilon^d) \leqslant e^2 \exp\left(- \left(\frac{\epsilon^{2d}}{\xi Var[Q_{2d}(Y)]} \right)^{\frac{1}{2d}} \right)$$

Proof. The proof can be found using hypercontractivity, [69] Thm 6.12 and Thm 6.7. This result is also mentioned in [124]. $\square$

From this lemma, we can show a corollary about the difference between projected tensor CP components and original CP components.

Corollary B.4.2.1. *For any two d-mode tensors in rank-r CP form,* $\mathcal{X}_1 = \sum_{k=1}^{r} \boldsymbol{x}_{1,k}^{(1)} \circ \ldots \circ \boldsymbol{x}_{1,k}^{(d)}$ *and* $\mathcal{X}_2 = \sum_{k=1}^{r} \boldsymbol{x}_{2,k}^{(1)} \circ \ldots \circ \boldsymbol{x}_{2,k}^{(d)}$, *concentration bounds of polynomials can be derived below. Given* $\epsilon > 0$, $\xi > 0$ *constant, we have following JL type result.*

$$\mathbb{P}(\sum_{k,l=1}^{r}\prod_{j=1}^{d} |\boldsymbol{A}^{(j)}\boldsymbol{x}_{1,k}^{(j)} - \boldsymbol{A}^{(j)}\boldsymbol{x}_{2,l}^{(j)}||_2^2 - \sum_{k,l=1}^{r}\prod_{j=1}^{d} ||\boldsymbol{x}_{1,k}^{(j)} - \boldsymbol{x}_{2,l}^{(j)}||_2^2 > \epsilon^d \sum_{k,l=1}^{r}\prod_{j=1}^{d} ||\boldsymbol{x}_{1,k}^{(j)} - \boldsymbol{x}_{2,l}^{(j)}||_2^2)$$
$$\leqslant e^2 \exp(-\left(\frac{\epsilon^{2d}\prod_{j=1}^{d} P^{(j)}}{3^d r^4}\right)^{\frac{1}{2d}})$$

Proof. It is known that variable for any vector $\boldsymbol{x}^{(j)} \in \mathbb{R}^{I_j}$ and any matrix $\boldsymbol{A}^{(j)}$ made out of entries of following independent normal with mean 0 and variance $\frac{1}{P_j}$, linear combination $\frac{\boldsymbol{A}^{(j)}\boldsymbol{x}^{(j)}}{\|\boldsymbol{x}^{(j)}\|_2} \sim \boldsymbol{MVN}(\boldsymbol{0}, \frac{1}{P_j}\boldsymbol{I})$. So, $P_j \frac{\|\boldsymbol{A}^{(j)}\boldsymbol{x}^{(j)}\|^2}{\|\boldsymbol{x}^{(j)}\|_2^2}$ follows Chi-square of degree of freedom P_j. Using the fact that expression,

$$\sum_{k,l=1}^{r}\prod_{j=1}^{d} \frac{||\boldsymbol{A}^{(j)}\boldsymbol{x}_{1,k}^{(j)} - \boldsymbol{A}^{(j)}\boldsymbol{x}_{2,l}^{(j)}||_2^2}{||\boldsymbol{x}_{1,k}^{(j)} - \boldsymbol{x}_{2,l}^{(j)}||_2^2}$$

is the sum of r^2 identically distributed random variables with correlation 1. Therefore, the variance of the sum is $(r^2)^2$ times variance of an individual term. Here each element in the summation is a product of d independent scaled Chi-Square variables. Thus, each element in the summation is polynomial of degree equals to 2d of Gaussian random variables. This result follows from lemma B.4.2. In view of similar result can be found in [119], the constant ξ can be assumed to be 1. It is worth noting that for polynomial of degree 2 or $d = 1$, sharper bound can be obtained as illustrated in [72, 34].

$\square$

Now we present the bound for tensor kernels.

Proposition B.4.1. *For any two d-mode tensors in rank-r CP form, $\mathcal{X}_1 = \sum_{k=1}^{r} \boldsymbol{x}_{1,k}^{(1)} \circ \ldots \circ \boldsymbol{x}_{1,k}^{(d)}$ and $\mathcal{X}_2 = \sum_{k=1}^{r} \boldsymbol{x}_{2,k}^{(1)} \circ \ldots \circ \boldsymbol{x}_{2,k}^{(d)}$, concentration bounds of polynomials can be derived below. Suppose the random projection $\boldsymbol{f}_{TPR\text{-}CP\text{-}TT}$ is defined by projection tensor $\boldsymbol{\mathcal{A}}$, which satisfies the assumption **AS.5**. Given $\epsilon > 0$, δ_1 depending on ϵ as given in corollary B.4.2.1, $C_{d,r}$ constant depending on d, we have following JL-type result. For a given tensor kernel function $K(\cdot,\cdot)$,*

$$\mathbb{P}\left(\left|K(\boldsymbol{f}_{TPR\text{-}CP\text{-}TT}(\mathcal{X}_1), \boldsymbol{f}_{TPR\text{-}CP\text{-}TT}(\mathcal{X}_2)) - K(\mathcal{X}_1, \mathcal{X}_2)\right| \geqslant C_{d,r}\epsilon^d\right) \leqslant \delta_1$$

Proof.

$$\begin{aligned}
&|K(\boldsymbol{f}_{\text{TPR-CP-TT}}(\mathcal{X}_1), \boldsymbol{f}_{\text{TPR-CP-TT}}(\mathcal{X}_2)) - K(\mathcal{X}_1, \mathcal{X}_2)| \\
&\qquad = |\sum_{k,l=1}^{r} \prod_{j=1}^{d} K^{(j)}(\boldsymbol{A}^{(j)}\boldsymbol{x}_{1k}^{(j)}, \boldsymbol{A}^{(j)}\boldsymbol{x}_{2l}^{(j)}) - K^{(j)}(\boldsymbol{x}_{1k}^{(j)}, \boldsymbol{x}_{2l}^{(j)})| \\
&\qquad \leqslant \sum_{k,l=1}^{r} \prod_{j=1}^{d} |K^{(j)}(\boldsymbol{A}^{(j)}\boldsymbol{x}_{1k}^{(j)}, \boldsymbol{A}^{(j)}\boldsymbol{x}_{2l}^{(j)}) - K^{(j)}(\boldsymbol{x}_{1k}^{(j)}, \boldsymbol{x}_{2l}^{(j)})| \\
\text{(Lipschitz continuity in } \textbf{AS.4}\text{)} &\qquad \leqslant \sum_{k,l=1}^{r} \prod_{j=1}^{d} L_K^{(j)} |\, ||\boldsymbol{A}^{(j)}\boldsymbol{x}_{1k}^{(j)} - \boldsymbol{A}^{(j)}\boldsymbol{x}_{2l}^{(j)}||_2^2 - ||\boldsymbol{x}_{1k}^{(j)} - \boldsymbol{x}_{2l}^{(j)}||_2^2| \\
\text{(Max } L_k^{(j)} \text{ in } \textbf{AS.4}\text{)} &\qquad \leqslant L_K^d \sum_{k,l=1}^{r} \prod_{j=1}^{d} |\, ||\boldsymbol{A}^{(j)}\boldsymbol{x}_{1k}^{(j)} - \boldsymbol{A}^{(j)}\boldsymbol{x}_{2l}^{(j)}||_2^2 - ||\boldsymbol{x}_{1k}^{(j)} - \boldsymbol{x}_{2l}^{(j)}||_2^2| \\
\text{(Corollary B.4.2.1)} &\qquad \leqslant \epsilon^d L_K^d \sum_{k,l=1}^{r} \prod_{j=1}^{d} ||\boldsymbol{x}_{1k}^{(j)} - \boldsymbol{x}_{2k}^{(j)}||_2^2 \\
\text{(Assumption } \textbf{AS.7}\text{)} &\qquad \leqslant \sum_{k,l=1}^{r} 2^d \epsilon^d L_K^d B_x^{2d} \\
&\qquad \leqslant 2^d r^2 L_K^d B_x^{2d} \epsilon^d \\
\text{(Denote } C_{d,r} = 2^d r^2 L_K^d B_x^{2d}\text{)} &\qquad = C_{d,r}\epsilon^d
\end{aligned}$$

Such part vanishes as ϵ^d becomes as small as possible. □

The proposition shows that the difference between the projected kernel and original kernel function can be bounded with probability at least $1 - \delta_1$, when condition **AS.4**, **AS.5**, and **AS.7** hold.

Now we include conditions **AS.1**, **AS.6** together with the previous results to show proposition 3.5.2. With a single random projection defined by $\mathcal{A}$, the extra risk from random projection

$$|\mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}) + \lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}||^2 - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}) - \lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}||^2|$$

contains two parts. They are bounded in separate ways.

- Difference between risks can be bounded by the following inequality. With probability at least $1 - \delta_1$ (with respect to random projection), $0 < \delta_1 < 1$,

$$\begin{aligned}
\left|\mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}})\right| &= \left|\mathbb{E}_{(\boldsymbol{\mathcal{X}}\times\boldsymbol{\mathcal{Y}})}\left[\mathcal{L}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}(\mathfrak{X}), y) - \mathcal{L}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}(\mathfrak{X}), y)\right]\right| \\
\text{(}\textbf{AS.1}\text{ and Jensen's Inequality)} \quad &\leqslant C(K_{max}\sqrt{\frac{L_0}{\lambda}}) \cdot \mathbb{E}_{(\boldsymbol{\mathcal{X}}\times\boldsymbol{\mathcal{Y}})}\left[|\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}(\mathfrak{X}) - \boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}(\mathfrak{X})|\right] \\
&\leqslant C(K_{max}\sqrt{\frac{L_0}{\lambda}}) \cdot \Big[\sum_{i=1}^{n} |\alpha_i| \mathbb{E}_{(\boldsymbol{\mathcal{X}}\times\boldsymbol{\mathcal{Y}})}\{|y_i| \cdot \\
&\qquad |K(\boldsymbol{f}_{\text{TPR-CP-TT}}(\mathfrak{X}_1), \boldsymbol{f}_{\text{TPR-CP-TT}}(\mathfrak{X}_2)) - K(\mathfrak{X}_1, \mathfrak{X}_2)|\}\Big] \\
\text{(Proposition B.4.1 and } |y_i| = 1\text{)} \quad &\leqslant C(K_{max}\sqrt{\frac{L_0}{\lambda}}) \cdot \Big[\sum_{i=1}^{n} |\alpha_i| \cdot \mathbb{E}_{(\boldsymbol{\mathcal{X}}\times\boldsymbol{\mathcal{Y}})}[C_{d,r}\epsilon^d]\Big] \\
\text{(Expectation over constant)} \quad &\leqslant C(K_{max}\sqrt{\frac{L_0}{\lambda}}) \cdot \Psi C_{d,r}\epsilon^d
\end{aligned}$$

  where $\Psi = \sup\{||\boldsymbol{\alpha}||_1 = \sum_{i=1}^{n} |\alpha_i| : \boldsymbol{f}(\mathfrak{X}) = \boldsymbol{\alpha}^T \boldsymbol{D}_y \boldsymbol{K}(\mathfrak{X}) \in \boldsymbol{\mathcal{H}}\}$.

- Difference between functional norms can be bounded in a similar way. With probability at least $1 - \delta_1$ (with respect to random projection), $0 < \delta_1 < 1$,

$$\begin{aligned}
|\lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}||^2 - \lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}||^2| &\leqslant \lambda \sum_{i=1}^{n}\sum_{l=1}^{n} \alpha_i \alpha_l |y_i||y_l| \cdot \\
\text{(Absolute value)} \quad &\qquad |K(\boldsymbol{f}_{\text{TPR-CP-TT}}(\mathfrak{X}_1), \boldsymbol{f}_{\text{TPR-CP-TT}}(\mathfrak{X}_2)) - K(\mathfrak{X}_1, \mathfrak{X}_2)| \\
\text{(Proposition B.4.1 and } |y_i| = 1\text{)} \quad &\leqslant \lambda(\sum_{i=1}^{n} |\alpha_i|) \cdot (\sum_{l=1}^{n} |\alpha_l|) \cdot C_{d,r}\epsilon^d \\
&\leqslant \lambda \Psi^2 C_{d,r}\epsilon^d
\end{aligned}$$

Each of these two inequalities hold with probability at least $1 - \delta_1$, then two inequalities hold simultaneously with probability at least $1 - 2\delta_1$. This can be showed with simple probability theory, since the probability of at least one inequality does not hold is no more than $2\delta_1$. (Probability of union is no more then the sum of probabilities.)

As a result, we conclude that with probability at least $1 - \delta_1$ with respect to random projection

$$|\mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A},n}}) + \lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A},n}}||^2 - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}) - \lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}||^2| \leqslant C_d\,\Psi\,[C(K_{max}\sqrt{\frac{L_0}{\lambda}}) + \lambda\Psi]\,\epsilon^d = O(\frac{\epsilon^d}{\lambda^q})$$

The proposition 3.5.2 is proved.

B.5 Proof of Theorem 3.5.2

Theorem 3.5.2 establishes an upper bound on the excess risk of RPSTM model under a single random projection. Although it does not give out the statistical consistency of RPSTM we are pursuing, it summarizes the conclusions from proposition 3.5.2 and bound the excess risk in proposition 3.5.1 under a single random projection.

The theorem assumes that if all the conditions **AS.1 - AS.9** hold, then the excess risk under a single random projection can be bounded with probability at least $(1 - 2\delta_1)(1 - \delta_2)$

$$\mathcal{R}_{\mathcal{L}}(\boldsymbol{g}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}) - \mathcal{R}^*_{\mathcal{L}} \leqslant V(1) + V(2) + V(3)$$

- $V(1) = 12C(K_{max}\sqrt{\frac{L_0}{\lambda}}) \cdot K_{max}\frac{\sqrt{L_0}}{\sqrt{n\lambda}} + 9\tilde{\zeta}_\lambda\sqrt{\frac{\log(2/\delta_2)}{2n}} + 2\zeta_\lambda\sqrt{\frac{2\log(2/\delta_2)}{n}}$
- $V(2) = D(\lambda)$
- $V(3) = C_{d,r}\Psi \cdot [C(K_{max}\sqrt{\frac{L_0}{\lambda}}) + \lambda\Psi]\epsilon^d$

$(1 - 2\delta_1)$ is the probability with respect to random projections, and $(1 - \delta_2)$ is with respect to the randomness of choosing training data T_n. As noted in the theorem, $\tilde{\zeta}_\lambda$ can be regarded as the supreme of the infinity norm of a function in the collection $\mathcal{L} \circ \mathcal{F} = \{\boldsymbol{h} : (\mathfrak{X}, y) \to \mathcal{L}(\boldsymbol{f}(\mathfrak{X}), y) : \boldsymbol{f} \in \mathcal{F}\}$, i.e.

$$\tilde{\zeta}_\lambda = \sup_{\boldsymbol{h}\in\mathcal{L}\circ\mathcal{F}} ||\boldsymbol{h}||_\infty = \sup_{\boldsymbol{h}\in\mathcal{L}\circ\mathcal{F}} \sup_{(\mathfrak{X},y)\in(\boldsymbol{\mathcal{X}}\times\boldsymbol{\mathcal{Y}})} |\boldsymbol{h}(\mathfrak{X}, y)|$$

$\mathcal{F} = \{\boldsymbol{f} : ||\boldsymbol{f}||_\infty \leqslant K_{max}\sqrt{\frac{L_0}{\lambda}}\}$. All functions in the collection $\mathcal{L} \circ \mathcal{F}$ are compositing a loss function together with a decision function, and are bi-variate. ζ_λ is a special case of $\tilde{\zeta}_\lambda$ by letting the decision function to be the optimal CP-STM $\boldsymbol{f^{\lambda}}$, i.e.

$$\zeta_{\lambda} = \sup_{(\mathcal{X},y)\in(\boldsymbol{\mathcal{X}}\times\boldsymbol{\mathcal{Y}})} |\mathcal{L}(\boldsymbol{f^{\lambda}}(\mathcal{X}), y)|$$

As for Ψ, it is the supreme of the L-1 norm for CP-STM coefficient vector. In order to show this theorem, we can use the result from proposition 3.5.1, but without taking expectation over random projections.

With a single random projection, the proof of Proposition 3.5.2 in appendix B.4 already shows how the term $V(3)$ is developed. The probability component with respect to random projection depicts the chance of $V(3)$ term being true, and is explained in the proof. $V(2)$ is directly taken from the risk decomposition in Proposition 3.5.1. Thus, we only have to show $V(1)$ to establish the theorem.

Indeed, our discussion in Proposition 3.5.1 unveils that, except the terms already bounded by $V(2)$, $V(3)$, and the term vanishes due to universal tensor kernels, $V(1)$ only bounds the gaps between empirical risk and expected risk, which are listed below.

$$\mathcal{R}_{\mathcal{L}}(\boldsymbol{g_n^{\lambda}}) - \mathcal{R}_{\mathcal{L},T_n^{\boldsymbol{A}}}(\boldsymbol{g_n^{\lambda}}), \quad \mathcal{R}_{\mathcal{L}}(\boldsymbol{f_n^{\lambda}}) - \mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f_n^{\lambda}})$$
$$\mathcal{R}_{\mathcal{L},T_n^{\boldsymbol{A}}}(\boldsymbol{f_{A,n}^{\lambda}}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f_{A,n}^{\lambda}}), \quad \mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f^{\lambda}}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f^{\lambda}})$$

Notice that consider the problem under a single random projection. Thus, the risks are not expectations over all random projections. As we mentioned earlier, one of them can be bounded by Hoeffding equality immediately, and the other three can be bounded by Rademacher Complexity. We list the bound for each term, and explain how the bound is developed below. First, we consider the term $\mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f^{\lambda}}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f^{\lambda}})$ and get the following result.

Proposition B.5.1. *With probability at least* $(1-\delta_2)$ *for* $\delta_2 \in (0,1)$

$$|\mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f^{\lambda}}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f^{\lambda}})| < 2\zeta_{\lambda}\sqrt{\frac{2log\frac{1}{\delta_2}}{n}}$$

Proof. We consider $\mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f^{\lambda}}) = \sum_{i=1}^{n} \mathcal{L}(\boldsymbol{f^{\lambda}}(\mathcal{X}_i), y_i)$ as a sum of independent and identically distributed (i.i.d) random variables since each pair $(\mathcal{X}_i, y_i) \in T_n$ are i.i.d distributed. $\mathcal{R}_{\mathcal{L}}(\boldsymbol{f^{\lambda}})$ is the expectation of $\mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f^\lambda})$. Since loss function is bounded by ζ_λ for every term in $\mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f^\lambda})$, using Hoeffding's inequality ([36]), we obtain $\mathbb{P}[\mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f^\lambda}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f^\lambda})] > \theta) \leq exp(-\frac{n\theta}{8\zeta_\lambda^2})$. Choosing $\delta_2 = exp(-\frac{n\theta}{8\zeta_\lambda^2})$ leads to the above bound. □

However, the other three terms cannot be bounded in the same way. This is because the decision function in the other three terms $\boldsymbol{f^\lambda_{\mathcal{A},n}}$, $\boldsymbol{g^\lambda_n}$, and $\boldsymbol{f^\lambda_n}$ are calculated from the training data and hence conditional on T_n. As a result, the risk of RPSTM model $\mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{g^\lambda_n})$ is not a sum of independent random variables. This violates the assumption of Hoeffding inequality. We need to use Rademacher Complexity, a stronger tool to bound the three terms left. We use this tool to develop a bound between $\mathcal{R}_{\mathcal{L}}(\boldsymbol{g^\lambda_n})$ and $\mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{g^\lambda_n})$.

We use $\mathfrak{R}_n(\boldsymbol{\mathcal{F}})$ to denote the Rademacher complexity of a function class $\boldsymbol{\mathcal{F}}$, and $\hat{\mathfrak{R}}_{D_n}(\boldsymbol{\mathcal{F}})$ to denote the corresponding sample estimate with respect to samples $D_n = \{Z_1, .., Z_n\}$. To make our description more consistent, one may regard each $Z_i = (\boldsymbol{\mathcal{X}}_i, y_i)$, so that D_n is another representation of the training data T_n. The reason for doing this is because we need to use composite function in forms of $\mathcal{L} \circ \boldsymbol{\mathcal{F}}$. We shall present few well established results about the Rademacher complexity without proof. One can find details about the proof from [106].

Theorem B.5.1. *Consider a collection of classifiers* $\boldsymbol{\mathcal{F}} = \{\boldsymbol{f} : ||\boldsymbol{f}||_\infty \leqslant \tilde{\zeta}_\lambda\}$. $\forall \delta_2 > 0$, *with probability at least* $1 - \delta_2$, *we obtain:*

$$\sup_{\boldsymbol{f}\in\boldsymbol{\mathcal{F}}} |\mathbb{E}[\boldsymbol{f}(Z)] - \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{f}(Z_i)| \leqslant 2\mathfrak{R}_n(\boldsymbol{\mathcal{F}}) + \tilde{\zeta}_\lambda \sqrt{\frac{\log \frac{2}{\delta_2}}{2n}}$$

The probability is with respect to the draw of D_n.

This is the general inequality for Rademacher Complexity, and can be applied for all types of data and functions $\boldsymbol{f}$. There is a corollary from the Rademacher Complexity developed especially for controlling classification risks. The necessity of this corollary is due to the fact that $\boldsymbol{f}$ is a uni-variate function in the theorem, but loss functions in classification risk measurement are bi-variate. As a result, it is not appropriate to replace $\boldsymbol{f}$ with loss function $\mathcal{L}$, and substitute D_n with our tensor training data T_n directly in Theorem B.5.1.

Corollary B.5.1.1. *Let $\mathcal{F} = \{f : ||f||_\infty \leqslant K_{max}\sqrt{\frac{L_0}{\lambda}}\}$, and its sample Rademacher Complexity is $\hat{\mathfrak{R}}_{T_n}(\mathcal{F})$. Suppose $\mathcal{L} : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}$ is a loss function satisfying condition **AS.1**. Then for all possible training set T_n,*

$$\hat{\mathfrak{R}}_{T_n}(\mathcal{L} \circ \mathcal{F}) \leqslant 2C(K_{max}\sqrt{\frac{L_0}{\lambda}}) \cdot \hat{\mathfrak{R}}_{T_n}(\mathcal{F})$$

*where $\mathcal{L} \circ \mathcal{F} = \{(\mathcal{X}, y) \to \mathcal{L}(f(\mathcal{X}), y) : f \in \mathcal{F}\}$. $C(K_{max}\sqrt{\frac{L_0}{\lambda}})$ is the Lipschitz continuous constant of loss function $\mathcal{L}$ introduced in assumption **AS.1**.*

The corollary bridges the Rademacher Complexity for general functions to the classification problems. This corollary is also know as a useful extension of Ledoux-Talagrand Contraction Theorem. Lastly, since Corollary B.5.1.1 uses sample Rademacher Complexity, one more result from [106] about kernel classes and RKHS is needed. Here, we continue using the notations about CP-STM and RPSTM from our main content.

Corollary B.5.1.2. *Suppose $\mathcal{H}$ is the RKHS generated by projected tensor kernel functions (3.2). With assumption **AS.3** and training data T_n, for collection of function any function $\mathcal{F}(M) \subset \mathcal{H}$*

$$\hat{\mathfrak{R}}_{T_n}\big(\mathcal{F}(M)\big) \leqslant \frac{MK_{max}}{\sqrt{n}}$$

$\mathcal{F}(M) = \big\{f \in \mathcal{H}, ||f|| \leqslant M, M > 0\big\}$.

To use the inequality in Corollary B.5.1.1, we have to bound the infinite norm of the function g_n^λ due to the fact that we have to composite loss function $\mathcal{L}$ and decision function g_n^λ to measure the risk. Thus, we provide the following proposition.

Proposition B.5.2. *Let g_n^λ be the optimal RPSTM model. Under assumption **AS.1**, we have*

$$||g_n^\lambda||_\infty \leqslant K_{max}\sqrt{\frac{L_0}{\lambda}}$$

Proof. Since $\boldsymbol{g_n^\lambda} = \arg\min_{g \in \mathcal{H}^{\mathcal{A}}} \{\mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{g}) + \lambda ||\boldsymbol{g}||^2\}$, we get

$$\begin{aligned}
||\boldsymbol{g_n^\lambda}||^2 &\leqslant \frac{1}{\lambda}\left[\mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(0) + ||0||^2 - \mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{g_n^\lambda})\right] \\
\left(\mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{g_n^\lambda}) \text{ negative and } ||0||^2 = 0\right) \quad &\leqslant \frac{1}{\lambda}\mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(0) \\
\left(\text{Assumption } \textbf{AS.1}\right) \quad &\leqslant \frac{L_0}{\lambda}
\end{aligned}$$

Since $\boldsymbol{g_n^\lambda} \in \mathcal{H}^{\mathcal{A}}$, and $\mathcal{H}^{\mathcal{A}}$ is a RKHS generated by projected tensor kernels. By RKHS property , for any function $\boldsymbol{g} \in \mathcal{H}^{\mathcal{A}}$

$$\boldsymbol{g}(\mathfrak{X}^{\mathcal{A}}) = \langle \boldsymbol{g}, K(,\mathfrak{X}^{\mathcal{A}})\rangle \leqslant ||\boldsymbol{g}||\sqrt{\sup K(\cdot,\cdot)} \leqslant K_{max}\sqrt{\frac{L_0}{\lambda}}$$

The step use Cauchy–Schwarz inequality. The inequality holds for all $\mathfrak{X}^{\mathcal{A}}$ when the function $\boldsymbol{g}$ is replaced with $\boldsymbol{g_n^\lambda}$. □

Inspired by the proof of Proposition B.5.2, we consider a collection of function $\mathcal{G}^\lambda = \{\boldsymbol{g} : \boldsymbol{g} \in \mathcal{H}^{\mathcal{A}}, ||\boldsymbol{g}|| \leqslant \sqrt{\frac{L_0}{\lambda}}\}$, which obviously includes $\boldsymbol{g_n^\lambda}$. Due to Corollary B.5.1.2 and Proposition B.5.2, we have

$$\hat{\mathfrak{R}}_n(\mathcal{G}^\lambda) \leqslant K_{max}\sqrt{\frac{L_0}{n\lambda}}$$

and $||\boldsymbol{g}||_\infty \leqslant K_{max}\sqrt{\frac{L_0}{\lambda}}$ for all $\boldsymbol{g} \in \mathcal{G}^\lambda$. Thus, we can now utilize Theorem B.5.1 to show the bound between $\mathcal{R}_{\mathcal{L}}(\boldsymbol{g_n^\lambda})$ and $\mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{g_n^\lambda})$.

Proposition B.5.3. *Let* $\mathcal{G}^\lambda = \{\boldsymbol{g} : ||\boldsymbol{g}|| \leqslant \sqrt{\frac{L_0}{\lambda}}\}$. *Assume conditions* ***AS.1**, **AS.3**, **AS.4**, and **AS.7**. Let* $\delta_2 > 0$, *with probability at least* $1 - \delta_2$ *and a given random projection defined by* $\boldsymbol{\mathcal{A}}$

$$|\mathcal{R}_{\mathcal{L}}(\boldsymbol{g_n^\lambda}) - \mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{g_n^\lambda})| \leqslant 4C(K_{max}\sqrt{\frac{L_0}{\lambda}}) \cdot K_{max}\sqrt{\frac{L_0}{n\lambda}} + 3\tilde{\zeta}_\lambda\sqrt{\frac{\log(2/\delta_2)}{2n}} \tag{B.4}$$

The probability is with respect to the join distribution of $\boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}$.

Proof. Since $\boldsymbol{g_n^\lambda} \in \mathcal{G}^\lambda$ and $||\boldsymbol{g}||_\infty \leqslant K_{max}\sqrt{\frac{L_0}{\lambda}}$ for all $\boldsymbol{g} \in \mathcal{G}^\lambda$. Let $\mathcal{H}^\lambda = \mathcal{L} \circ \mathcal{G}^\lambda = \{\boldsymbol{h} : \boldsymbol{h} = \mathcal{L}(\boldsymbol{g}(\mathfrak{X}), y), \boldsymbol{g} \in \mathcal{G}^\lambda\}$. Then $||\boldsymbol{h}||_\infty \leqslant \tilde{\zeta}_\lambda$ as we noted in the description of Theorem 3.5.2. Theorem

B.5.1 then suggests given a training data T_n and its projected counterpart,

$$\begin{aligned}
|\mathcal{R}_{\mathcal{L}}(\boldsymbol{g_n^\lambda}) - \mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{g_n^\lambda})| &\leqslant \sup_{\boldsymbol{h}\in\mathcal{H}^\lambda} \Big|\mathbb{E}_{\mathcal{X}\times\mathcal{Y}}\boldsymbol{h}(Z) - \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{h}(Z_i)\Big| \\
\big(\text{Theorem B.5.1 and } ||\boldsymbol{h}||_\infty \leqslant \tilde{\zeta}_\lambda \text{ by definition}\big) \quad &\leqslant 2\mathfrak{R}_n(\mathcal{H}^\lambda) + \tilde{\zeta}_\lambda\sqrt{\frac{\log(2/\delta_2)}{2n}} \\
\big(\text{McDiarmid's inequality}\big) \quad &\leqslant 2\bigg(\hat{\mathfrak{R}}_n(\mathcal{H}^\lambda) + \tilde{\zeta}_\lambda\sqrt{\frac{\log(2/\delta_2)}{2n}}\bigg) + \tilde{\zeta}_\lambda\sqrt{\frac{\log(2/\delta_2)}{2n}} \\
\big(\text{Corollary B.5.1.1}\big) &\leqslant 2\big(2C(K_{max}\sqrt{\frac{L_0}{\lambda}}) \cdot \hat{\mathfrak{R}}_n(\mathcal{G}^\lambda) + \tilde{\zeta}_\lambda\sqrt{\frac{\log(2/\delta_2)}{2n}}\big) + \\
&\quad \tilde{\zeta}_\lambda\sqrt{\frac{\log(2/\delta_2)}{2n}} \\
\big(\text{Corollary B.5.1.2}\big) \quad &\leqslant 4C(K_{max}\sqrt{\frac{L_0}{\lambda}}) \cdot K_{max}\sqrt{\frac{L_0}{n\lambda}} + 3\tilde{\zeta}_\lambda\sqrt{\frac{\log(2/\delta_2)}{2n}}
\end{aligned}$$

The proposition is proved. □

Finally, we can use the exact same way to control $\mathcal{R}_{\mathcal{L}}(\boldsymbol{f_n^\lambda}) - \mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f_n^\lambda})$ and $\mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{f_{\mathcal{A},n}^\lambda}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f_{\mathcal{A},n}^\lambda})$. Since the probability $1-\delta_2$ is with respect to sampling of training data T_n. Given a random projection, if we get T_n, we get $T_n^{\mathcal{A}}$. Hence, the randomness of all the four terms bounded by $V(1)$ holds simultaneously. We can conclude that with probability at least $1-\delta_2$,

$$V(1) = 12C(K_{max}\sqrt{\frac{L_0}{\lambda}})\frac{K_{max}\sqrt{L_0}}{\sqrt{n\lambda}} + 9\tilde{\zeta}_\lambda\sqrt{\frac{\log(2/\delta_2)}{2n}} + 2\zeta_\lambda\sqrt{\frac{2\log(2/\delta_2)}{n}} \tag{B.5}$$

Theorem 3.5.2 is proved. As one can see, the term $V(1)$ and $V(3)$ converge to zero as we assumed. The $D(\lambda)$ term in $V(2)$ actually motivates us to make assumption **AS.9** to make the excess risk converging.

B.6 Convergence Rate of Squared Hinge and Hinge Loss

We establish the explicit convergence rate for Squared Hinge and Hinge loss in this section. To do that, we have to utilize some properties and facts that hold for these two loss functions. These properties are listed below without proof since they can be easily verified. One can refer [128] for the proof. For similarity, we use $\mathcal{L}_1$ and $\mathcal{L}_2$ to denote Hinge and Squared Hinge loss.

1. For both loss function, $\lambda||\boldsymbol{f^\lambda}||_K^2 < D(\lambda)$ with a given $\lambda > 0$.

2. For both loss function, $||\boldsymbol{f^{\lambda}}||_{\infty} = \sup\limits_{\boldsymbol{\mathcal{X}} \in \boldsymbol{\mathcal{H}}} |\boldsymbol{f^{\lambda}}(\boldsymbol{\mathcal{X}})| \leqslant K_{max}||\boldsymbol{f^{\lambda}}|| \leqslant K_{max}\sqrt{\frac{L_0}{\lambda}}$.

3. For hinge loss, $L_0 = 1$, $\zeta_\lambda \leq 1 + K_{max}\sqrt{\frac{1}{\lambda}}$, and $\tilde{\zeta}_\lambda \leq 1 + K_{max}\sqrt{\frac{D(\lambda)}{\lambda}}, C(K_{max}\sqrt{\frac{L_0}{\lambda}}) = 1$

4. For square hinge loss,$L_0 = 1$, $\zeta_\lambda \leq (1+K_{max}\sqrt{\frac{1}{\lambda}})^2$, $\tilde{\zeta}_\lambda \leq 2(1+K\frac{D(\lambda)}{\lambda})$, and $C(K_{max}\sqrt{\frac{L_0}{\lambda}}) = 2K_{max}\sqrt{\frac{1}{\lambda}}$.

5. For hinge loss, $\Psi_{\mathcal{L}_1} \leqslant \frac{1}{\lambda}$ since the dual problem of CP-STM restrict $\alpha_i \leqslant \frac{1}{2n\lambda}$. (See section 2.2)

6. For square hinge loss, using lemma $\Psi_{\mathcal{L}_2} = O(\frac{1}{\lambda})$

For the property 6 about Squared Hinge loss, we provide some discussion here.

Lemma B.6.1. *Let n^+ and n^- be training samples with label $+1$ and -1 respectively. Define $\Psi = \underset{\mathcal{X}_i \in \mathcal{X}}{sup}\{\sum|\beta_i| : f = \sum \beta_i K(\mathcal{X}_i, .)\}$ Supremum of absolute sum of all coefficients over every possible function in RKHS is given by*

$$\Psi_{\mathcal{L}_2} \leqslant \frac{1}{C_{\boldsymbol{K}} + \frac{n\lambda}{4n^+n^-}}$$

where $C_{\boldsymbol{k}} = \min\limits_{\boldsymbol{\beta}:\boldsymbol{\beta}^T\mathbf{1}=1} \boldsymbol{\beta}^T\boldsymbol{K}\boldsymbol{\beta}$ depends on kernel matrix $\boldsymbol{K}$

The proof of this lemma is available at [39]. Using this lemma, we can obtain a corollary and establish the last property about Squared Hinge loss.

Corollary B.6.1.1. *For Squared Hinge loss, supremum of sum of absolute coefficients is finite as sample size grows.*

$$\Psi_{\mathcal{L}_2} = O(\frac{1}{\lambda})$$

Proof. Sum of eigenvalues of $\boldsymbol{K}$ is of order $O(n)$, since trace of $\boldsymbol{K}$ is of order $O(n)$. The trace of $\boldsymbol{K}$ is indicated by the facts that $K(\mathcal{X}_i, \mathcal{X}_i) = O(1)$ guaranteed by assumption **AS.7**. Considering $\frac{4n^+n^-}{n} \leqslant 1$ from arithmetic mean and geometric mean inequality. Assuming that $\boldsymbol{K}$ is positive definite, then $\Psi_{\mathcal{L}_2} = O(1)$. This bound agrees with the bound of Theorem 3.3 in [29], which is of order $O(\frac{1}{\lambda})$ at constant level. In this case $C_{\boldsymbol{K}}$ is 0 under the situation where $\boldsymbol{K}$ is not positive definite. Combining these two conditions for the kernel matrix $\boldsymbol{K}$, the corollary is proved. $\square$

Note that depending on kernel and geometric configuration of data points. This quantity influences error of projected classifier.The above bounds are consistent with [39]. Assuming the data are from bounded domain, i.e $||\boldsymbol{\mathcal{X}}||_2 \leqslant C_{d,r}$ The gram matrix $\boldsymbol{K}$ can have minimum eigenvalue as positive so $C_{\boldsymbol{K}} > 0$. The key idea is to divide the bounded domain into minimal increasing sequence of discs D_n formed between rings of radius R_{n-1} and R_n such that $\mathcal{X} \subseteq \cup_{n=1}^{N} D_n$. So the diameter of $\boldsymbol{\mathcal{X}} \leq 2R_N$ assuming $\sum R_n^2 < \infty$ and then count the number of points in each D_n. So some regularity conditions on distribution of data for each disc D_n is necessary to evaluate bounds on eigenvalues of Gram matrix. For unknown case, we can estimate the Gram matrix. [126] discusses the regularization error in such estimation

B.6.1 Proof of Proposition 3.5.3

For Squared Hinge loss, consider the projected dimension to be $P_j = \lceil 3r^{\frac{4}{d}} \frac{(log\frac{n}{\delta_1})^2}{\epsilon^2} \rceil + 1$ for each mode $j = 1, 2, ..d$. Adopt theorem 3.5.2 and the properties 1 2, 4 and 6, we have with probability at least $(1 - 2\delta_1))(1 - \delta_2)$ and some $\eta \in (0, 1]$

$$\begin{aligned}\mathcal{R}_{\mathcal{L}}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L}}^* &\leqslant \frac{24K_{max}^2}{\sqrt{n\lambda^2}} + 18(1 + K_{max}\frac{D(\lambda)}{\lambda})\sqrt{\frac{log(2/\delta_2)}{2n}} + 2(1 + \frac{K}{\sqrt{\lambda}})^2\sqrt{\frac{2log(2/\delta_2)}{n}} \\ &\quad + D(\lambda) + C_D\ [2K_{max}\frac{1}{\lambda}\sqrt{\frac{1}{\lambda} + \frac{\lambda}{\lambda}}]\ \frac{\epsilon^d}{\lambda} \\ &\leqslant O(\frac{1}{\sqrt{n\lambda^2}})\sqrt{log\frac{2}{\delta_2}} + O(\frac{1}{\sqrt{n\lambda}}) + O(\frac{1}{\sqrt{n\lambda^{2(1-\eta)}}})\sqrt{log\frac{2}{\delta_2}} + O(\lambda^\eta) + O(\frac{\epsilon^d}{\lambda^{\frac{3}{2}}})\end{aligned}$$

$\delta_1 \in (0, \frac{1}{2})$ and $\delta_2 \in (0, 1)$. Plugging in the assumption **AS.10** and replace the last term in the equation above with a number in term of n, we obtain

$$\mathcal{R}_{\mathcal{L}}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L}}^* \leqslant C\sqrt{log(\frac{2}{\delta_2})} \cdot (\frac{1}{n})^{\frac{\mu\eta}{2\eta+3}}$$

Now $\epsilon = (\frac{1}{n})^{\frac{\mu}{2d}}$ for $0 < \mu < 1$, and $\lambda = (\frac{1}{n})^{\frac{\mu}{2\eta+3}}$ for some $0 < \eta \leqslant 1$. The projected dimension becomes $P_j = \lceil 3r^{\frac{4}{d}} \frac{(log\frac{n}{\delta_1})^2}{\epsilon^2} \rceil + 1$.

B.6.2 Proof of Proposition 3.5.4

The convergence rate for Hinge loss are established using the same way as that of Squared Hinge loss. Adopting theorem 3.5.2 together with the properties 1, 2, 3, and 5, we conclude that

$$\begin{aligned}\mathcal{R}_{\mathcal{L}}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L}}^* &\leqslant \frac{12K}{\sqrt{n\lambda}} + 9\frac{(1+K_{max})}{\sqrt{\lambda}}\sqrt{\frac{log(2/\delta_2)}{n}} + 2(1+K_{max}\frac{D(\lambda)}{\sqrt{\lambda}})\sqrt{\frac{2log(2/\delta_2)}{n}} \\ &+ D(\lambda) + C_D[2+\frac{\lambda}{\lambda}]\frac{\epsilon^d}{\lambda} \\ &\leqslant O(\frac{1}{\sqrt{n\lambda}})\sqrt{log(\frac{2}{\delta_2})} + O(\frac{1}{\sqrt{n\lambda^{2(1-\eta)}}})\sqrt{log(\frac{2}{\delta_2})} + O(\lambda^{\eta}) + O(\frac{\epsilon^d}{\lambda})\end{aligned}$$

with $P_j = O(\frac{[log\frac{n}{\delta_1}]^2}{\epsilon^2})$ for each mode $j = 1, 2, ..d$. The inequality holds for probability at least $(1-2\delta_1)(1-\delta_2)$, for some $\delta_1 \in (0, \frac{1}{2})$ and $\delta_2 \in (0,1)$. Use the assumption **AS.10** again, we get

$$\mathcal{R}_{\mathcal{L}}(\boldsymbol{g}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L}}^* \leqslant C\sqrt{log(\frac{2}{\delta_2})}(\frac{1}{n})^{\frac{\mu\eta}{2\eta+2}}$$

The $\epsilon = (\frac{1}{n})^{\frac{\mu}{2d}}$ for $0 < \mu < 1$, and $\lambda = (\frac{1}{n})^{\frac{\mu}{2\eta+2}}$ for some $0 < \eta \leqslant 1$. The projected dimension becomes $P_j = \lceil 3r^{\frac{4}{d}}n^{\frac{\mu}{d}}[log(n/\delta_1)]^2\rceil + 1$.

B.7 Proof of theorem 3.5.3

Theorem 3.5.3 establishes the rate of convergence for expected risk difference, showing ℓ_1 consistency of error vanishing as sample size increases. Thus theorem 3.5.3 establishes stronger optimality of our algorithm. In this subsection, we show theorem 3.5.3 holds, which is a much stronger result than the risk difference vanishing in probability.

First, we show a corollary about the expected difference between projected tensor CP components and original CP components.

Corollary B.7.0.1. *For any two d-mode tensors in rank-r CP form,* $\mathcal{X}_1 = \sum_{k=1}^{r} \boldsymbol{x}_{1,k}^{(1)} \circ ... \circ \boldsymbol{x}_{1,k}^{(d)}$ *and* $\mathcal{X}_2 = \sum_{k=1}^{r} \boldsymbol{x}_{2,k}^{(1)} \circ ... \circ \boldsymbol{x}_{2,k}^{(d)}$, *expectation of difference of tensor norm and its projection have a upper* *bound as shown below.*

$$\mathbb{E}|\sum_{k,l=1}^{r}\prod_{j=1}^{d}||\boldsymbol{A}^{(j)}\boldsymbol{x}_{1,k}^{(j)} - \boldsymbol{A}^{(j)}\boldsymbol{x}_{2,l}^{(j)}||_2^2 - \sum_{k,l=1}^{r}\prod_{j=1}^{d}||\boldsymbol{x}_{1,k}^{(j)} - \boldsymbol{x}_{2,l}^{(j)}||_2^2|$$
$$\leqslant r^2\sqrt{\frac{3^d}{\prod_{j=1}^{d} P_j}}\sum_{k,l=1}^{r}\prod_{j=1}^{d}||\boldsymbol{x}_{1,k}^{(j)} - \boldsymbol{x}_{2,l}^{(j)}||_2^2$$

Proof. It is known that for any real random variable W, $\mathbb{E}(|W|) \leqslant \sqrt{\mathbb{E}(W^2)}$. Using the aforementioned result along with variance of difference of projection stated in the proof of corollary B.4.2.1, we prove the following result. □

Next we derive the expected difference in tensor kernel due to projection

Proposition B.7.1. *For any two d-mode tensors in rank-r CP form,* $\mathcal{X}_1 = \sum_{k=1}^{r}\boldsymbol{x}_{1,k}^{(1)} \circ \ldots \circ \boldsymbol{x}_{1,k}^{(d)}$ *and* $\mathcal{X}_2 = \sum_{k=1}^{r}\boldsymbol{x}_{2,k}^{(1)} \circ \ldots \circ \boldsymbol{x}_{2,k}^{(d)}$. *Suppose the random projection* $\boldsymbol{f}_{TPR\text{-}CP\text{-}TT}$ *is defined by projection tensor* $\boldsymbol{\mathcal{A}}$*, which satisfies the assumption* ***AS.5***. *For a given tensor kernel function* $K(\cdot,\cdot)$. *We have following bound on expected difference of tensor kernel due to projection, here constant* $C_{d,r}$ *is taken from proposition B.4.1*

$$\mathbb{E}\Big(\Big|K(\boldsymbol{f}_{TPR\text{-}CP\text{-}TT}(\mathcal{X}_1), \boldsymbol{f}_{TPR\text{-}CP\text{-}TT}(\mathcal{X}_2)) - K(\mathcal{X}_1, \mathcal{X}_2)\Big|\Big) \leqslant C_{d,r}r^2\sqrt{\frac{3^d}{\prod_{j=1}^{d} P_j}}$$

Proof. We proceed similarly as in proof of proposition B.4.1. Using result on from proposition B.7.0.1, we derive our result. □

Using the results from above proposition B.7.1 and derivations mentioned in proof of proposition 3.5.2. As a result, we conclude that expectation with respect to random projection

$$\mathbb{E}_{\boldsymbol{\mathcal{A}}}|\mathcal{R}_{\mathcal{L}}(\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}) + \lambda||\boldsymbol{f}_{\boldsymbol{\mathcal{A}},\boldsymbol{n}}^{\boldsymbol{\lambda}}||^2 - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}) - \lambda||\boldsymbol{f}_{\boldsymbol{n}}^{\boldsymbol{\lambda}}||^2| \leqslant C_{d,r}\,\Psi\,[C(K_{max}\sqrt{\frac{L_0}{\lambda}}) + \lambda\Psi]\sqrt{\frac{3^d}{\prod_{j=1}^{d} P_j}}$$
$$= O(\frac{1}{\lambda^q\sqrt{\prod_{j=1}^{d} P_j}}) \tag{B.6}$$

However,it should be noted that for fixed sample size n and thus fixed ϵ, the projected dimension P_j changes as a function of probability δ_1 only. Now we prove the following result on expected risk of projected error, which is shown to be vanishing with increasing sample size.

Proposition B.7.2. *Based on conditions **AS.1** to **AS.8** and conditions **AS.10** to **AS.11**, the expected risk of projection error goes to 0 as n increases.*

$$\mathbb{E}_n\mathbb{E}_{\mathcal{A}}|\mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A},n}}) + \lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A},n}}||^2 - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}) - \lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}||^2| = O(\frac{\epsilon^d}{\lambda^q})$$

Proof. From equation B.6 and condition **AS.5**, we obtain following statement
With probability at least $1 - \delta_1$,

$$\begin{aligned} \mathbb{E}_{\mathcal{A}}|\mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A},n}}) + \lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A},n}}||^2 - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}) - \lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}||^2| &\leqslant \frac{1}{\lambda^q\sqrt{\prod_{j=1}^d P_j}} \\ (\textbf{AS.5}\ ) \quad &\leqslant \frac{\epsilon^d}{\lambda^q}O(\frac{1}{[log(\frac{n}{\delta_1})]^d}) \\ (\delta_1 \text{ as function of } n\ \textbf{AS.11}\ ) \quad &\leqslant \frac{\epsilon^d}{\lambda^q}O(\frac{1}{n}) \end{aligned}$$

Using above equation and substituting value of δ_1 as a function of n from condition **AS.11**, we have established that for sufficient large value of n

$$\mathbb{P}_n\left(\mathbb{E}_{\mathcal{A}}|\mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A},n}}) + \lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A},n}}||^2 - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}) - \lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}||^2| \geqslant O(\frac{\epsilon^d}{\lambda^q})\frac{1}{n}\right) \leqslant n\exp(-n^{\frac{1}{d}}) \tag{B.7}$$

.

We utilize the following lemma which states about the expectation of a sequence of random variables with distribution function specific to equation B.7

Lemma B.7.1. *For some $d \geqslant 1$, let W_n be a sequence of positive random variables with distribution, for sufficiently large n*

$$\mathbb{P}_n(W_n \geqslant \frac{1}{n}) \leqslant n\exp(-n^{\frac{1}{d}})$$

Then, the sequence W_n is uniformly bounded almost surely and in expectation L_1 norm.

Proof. We can show that $\sum_{n}^{\infty} \mathbb{P}_n(W_n \geqslant 1) < \infty$. By Borel-Cantelli lemma, the sequence W_n is uniformly bounded above by 1 almost surely. By Dominated convergence theorem, for any sequence of positive random variables that is uniformly bounded almost surely; then the expectation is bounded by the uniform bound or $\mathbb{E}_n(W_n) \geqslant 1$ for all sufficiently large n. $\square$

Let us denote, $\mathbb{E}_{\mathcal{A}}|\mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A},n}}) + \lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A},n}}||^2 - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}) - \lambda||\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}||^2| = W_n\, O(\frac{\epsilon^d}{\lambda^q})$. Assuming d to be the order of feature tensors, we claim using lemma B.7.1 that

$$\mathbb{E}_n\big(W_n\, O(\frac{\epsilon^d}{\lambda^q})\big) \leqslant O(\frac{\epsilon^d}{\lambda^q})$$

Thus, we conclude the proof for risk difference due to projection as stated in proposition B.7.2. $\square$

In the following statements, we bound the expectation of sampling error of training data.

Proposition B.7.3. *Based on condition **AS.1** to **AS.4** and **AS.7**, the expectation of sampling risk vanishes as sampling size increases.*

$$\begin{aligned}\mathbb{E}_n\Bigg(&|\mathcal{R}_{\mathcal{L}}(\boldsymbol{g}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}) - \mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{g}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}) + \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}) + \mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}) \\ &+ \mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A},n}}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A},n}}) + \mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}})|\Bigg) \\ &= O(\frac{1}{\sqrt{n\lambda^2}}) + O(\tilde{\zeta}_\lambda\sqrt{\frac{1}{2n}}) + O(\zeta_\lambda\sqrt{\frac{1}{n}})\end{aligned}$$

Proof. Lets denote the random variable $|\mathcal{R}_{\mathcal{L}}(\boldsymbol{g}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}) - \mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{g}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}) + \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}}) + \mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{n}})$ $+ \mathcal{R}_{\mathcal{L},T_n^{\mathcal{A}}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A},n}}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}}_{\boldsymbol{\mathcal{A},n}}) + \mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f}^{\boldsymbol{\lambda}}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f}^{\boldsymbol{\lambda}})|$ as H_n. Also, H_n is independent of random projection; thus taking over $\mathbb{E}_{\mathcal{A}}$ does not change it.

Ignoring the constants, we derive the following result from equation B.5,

$$\mathbb{P}_n\Bigg(H_n \geqslant O(\frac{1}{\sqrt{n\lambda^2}}) + O(\frac{\tilde{\zeta}_\lambda}{\sqrt{n}}\sqrt{\log(2/\delta_2)}) + O(\frac{\zeta_\lambda}{\sqrt{n}}\sqrt{\log(2/\delta_2)}\Bigg) \leqslant \delta_2$$

We need the following proposition on sub Gaussian random variable to prove above proposition B.7.3

Proposition B.7.4. *For any real random variable W with sub Gaussian tail, meaning $\mathbb{P}\big(W \geqslant \sqrt{\log(2/\delta_2)}\big) \leqslant \delta_2$; then $\mathbb{E}(W) = O(1)$*

Proof. Using change of variable, $\delta_2 = 2e^{-u^2}$; we obtain $\mathbb{P}(W \geqslant u) \leqslant 2e^{-u^2}$. Now using identity that for any positive random variable W, $\mathbb{E}(W) = \int_0^{\infty} \mathbb{P}(W \geqslant w)\, dw$, we proof our proposition. □

We further split $H_n = O(\frac{1}{\sqrt{n\lambda^2}}) + O(\frac{\tilde{\zeta}_\lambda}{\sqrt{n}})W_1 + O(\frac{\zeta_\lambda}{\sqrt{n}})W_2$ where W_1 and W_2 are random variables with Sub Gaussian tails. We can apply the above proposition B.7.4 about sub Gaussian random variables to prove proposition B.7.3. □

Gathering results from theorem 3.5.2, proposition B.7.2 and proposition B.7.3, we can now show the following conclusion.

$$\mathbb{E}_n|\mathbb{E}_{\mathcal{A}}[\mathcal{R}_{\mathcal{L}}(g_n^\lambda)] - \mathcal{R}^*_{\mathcal{L}}| \leqslant O(\frac{1}{\sqrt{n\lambda^2}}) + O(\tilde{\zeta}_\lambda\sqrt{\frac{1}{2n}}) + O(\zeta_\lambda\sqrt{\frac{1}{n}}) + O(D(\lambda)) + O(\frac{\epsilon^d}{\lambda^q}) \tag{B.8}$$

Referring to value $\tilde{\zeta}_\lambda$ and ζ_λ as mentioned in proof of proposition 3.5.4 and proposition 3.5.3. Under conditions **AS.6** and conditions **AS.9** to **AS.11**, we show validity of our claim as a corollary of equation B.8. Therefore, we complete our proof of theorem 3.5.3

B.8 Technical Details about Numerical Studies

All the code for our numerical studies are available at our Github repository `https://github.com/PeterLiPeide/TEC_Tensor_Ensemble_Classifier`. In the Simulation folder, one can find all the values of tuning parameter, optimal rank of CP decomposition, and the dimension of random projection. We also provide a code to regenerate our synthetic data, so that the whole simulation study is reproducible with our CP-STM module.

B.9 Simulation Study Discussion

Model	Methods	RBF-SVM	AAM	LLSVM	BSGD	LDA	RF
F2	Accuracy (%)	94.50	75.31	95.94	96.67	89.13	98.50
	STD (%)	2.15	6.18	1.86	3.95	3.64	1.50
	Time (s)	92	120	360	790	205	9.5
F3	Accuracy (%)	100	83.33	98.50	77.5	97.63	100
	STD (%)	0.00	3.54	1.37	14.39	1.90	0.00
	Time (s)	80	120	385	935	175	7.5
F5	Accuracy (%)	89.63	52.92	50	50	83.75	76.50
	STD (%)	2.80	8.28	0.00	0.00	4.50	7.65
	Time (s)	117	140	350	1820	231	8.65

Table B.1: TEC Simulation Results II: Cluster with 128GB RAM

For the completeness of our numerical study, we further apply the vector-based methods in simulation study 3.6 to those high dimensional classification tasks on a high performance cluster. The cluster is equipped with a 16-core CPU and 128GB of memory. The classification accuracy of all the vector-based classifier in F2, F3, and F5 tasks are provided in table B.1. If we further consider these results obtained from a more power machine, the advantages of our TEC models are more impressive. With much more memory, the BSGD model provides the best accuracy rate as 96.67% in F2. However, our TEC with Hinge loss has 98% average accuracy rate. Similar situation also happens in F5 where the best vector-based method RBF-SVM provides 89.63% accuracy rate. Our TEC with Square Hinge loss outperforms slightly than RBF-SVM. Only in F3, RBF-SVM and RF have the best performance, and are better than TEC models with 2% accuracy rates. Since these performance requires more computer memory, we believe TEC models are in general have greater potential than all these traditional methods.

APPENDIX C

APPENDIX FOR CHAPTER 4

C.1 Proof of Theorem 4.5.1

Proof. To prove the proposition 4.5.1, we introduce few more notations here. Let $\mathcal{L}$ be the loss function satisfying the condition **AS.2**. We denote the classification risk for an arbitrary decision function, $\boldsymbol{f}$, as

$$\mathcal{R}_{\mathcal{L}}(\boldsymbol{f}) = \mathbb{E}_{\mathcal{X}\times\mathcal{Y}}\mathcal{L}(y, \boldsymbol{f}(\mathfrak{X})) = \int \mathcal{L}(y, \boldsymbol{f}(\mathfrak{X}))d\mathbb{P}$$

The expectation is taken over the joint distribution of $\boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}$. Notice that this risk notation, $\mathcal{R}_{\mathcal{L}}(\boldsymbol{f})$, is different from our notation $\mathcal{R}(\boldsymbol{f})$ in section 4.5 since we use the Lipschitz continuous loss $\mathcal{L}$ instead of the "zero-one" loss to measure the classification error. $\mathcal{L}$ is also called surrogate loss for classification problems. Examples of such surrogate loss functions include Hinge loss and Squared Hinge loss. Comparison of these loss functions and their statistical properties can be found in [151]. If we denote the Bayes risk under the surrogate loss $\mathcal{L}$ as $\mathcal{R}^*_{\mathcal{L}}$, i.e. $\mathcal{R}^*_{\mathcal{L}} = \min \mathcal{R}_{\mathcal{L}}(\boldsymbol{f})$ for all measurable function f, then the result from [151] says $\mathcal{R}_{\mathcal{L}}(\boldsymbol{f_n}) \to \mathcal{R}^*_{\mathcal{L}}$ indicates $\mathcal{R}(\boldsymbol{f_n}) \to \mathcal{R}^*$ for any decision rule $\{\boldsymbol{f_n}\}$. This conclusion holds as long as the surrogate loss is "self-calibrated", see [128]. Since we use Hinge loss in our problem, and Hinge loss is known to be Lipschitz and self-calibrated, our assumption **AS.2** holds in our discussion. Thus, we only need to show $\mathcal{R}_{\mathcal{L}}(\boldsymbol{f_n}) \to \mathcal{R}^*_{\mathcal{L}}$ for the proof of our proposition 4.5.1.

Given the tuning parameter λ satisfying condition **AS.4**, we denote

$$\boldsymbol{f_n^{\lambda}} = \underset{\boldsymbol{f}\in\boldsymbol{\mathcal{H}}}{\arg\min} \quad \lambda \cdot ||\boldsymbol{f}||^2 + \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\boldsymbol{f}(\mathfrak{X}_i), y_i)$$

where $\boldsymbol{\mathcal{H}}$ is the reproducing kernel Hilbert space (RKHS) generated by the kernel function (4.8). As we mentioned in the section 4.3, $\boldsymbol{\mathcal{H}}$ is also know as the collection of functions which are in the form of equation (4.10). Now we further assume

$$\boldsymbol{f^{\lambda}} = \underset{\boldsymbol{f}\in\boldsymbol{\mathcal{H}}}{\arg\min} \quad \lambda \cdot ||\boldsymbol{f}||^2 + \mathcal{R}_{\mathcal{L}}(\boldsymbol{f})$$

Then $\boldsymbol{f^\lambda}$ is the optimal decision function from $\boldsymbol{\mathcal{H}}$ such that it minimizes the expected risk. Comparing $\boldsymbol{f_n^\lambda}$ with $\boldsymbol{f^\lambda}$, we can understand that $\boldsymbol{f^\lambda}$ is the version of $\boldsymbol{f_n^\lambda}$ when the size of training data is as large as possible. If we denote $\mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\boldsymbol{f}(\mathcal{X}_i), y_i)$, then $\mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f})$ is a sample estimate of $\mathcal{R}_{\mathcal{L}}(\boldsymbol{f})$. With $\boldsymbol{f^\lambda}$, we can show that

$$|\mathcal{R}_{\mathcal{L}}(\boldsymbol{f_n^\lambda}) - \mathcal{R}^*_{\mathcal{L}}| \leqslant |\mathcal{R}_{\mathcal{L}}(\boldsymbol{f_n^\lambda}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f^\lambda})| + |\mathcal{R}_{\mathcal{L}}(\boldsymbol{f^\lambda}) - \mathcal{R}^*_{\mathcal{L}}|$$

through triangular inequality. Since the Bayes risk under loss function $\mathcal{L}$ is defined as $\mathcal{R}^* = \min_{\boldsymbol{f}:\boldsymbol{\mathcal{X}}\to\boldsymbol{\mathcal{Y}}} \mathcal{R}(\boldsymbol{f})$ over all functions defined on $\boldsymbol{\mathcal{X}}$, we can immediate show that

$$\begin{aligned} |\mathcal{R}(\boldsymbol{f^\lambda}) - \mathcal{R}^*| &\leqslant \mathbb{E}_{(\mathcal{X}\times\mathcal{Y})}|\mathcal{L}(y, \boldsymbol{f^\lambda}(\mathcal{X})) - \mathcal{L}(y, \boldsymbol{f^*}(\mathcal{X}))| \leqslant C(K_{max}) \sup|\boldsymbol{f^\lambda} - \boldsymbol{f^*}| \\ &\leqslant C(K_{max}) \cdot \epsilon \end{aligned} \tag{C.1}$$

This is the result of using condition **AS.1** and **AS.2** in the proposition 4.5.1. $\boldsymbol{f^\lambda}$ is in the RKHS and thus bounded by some constant depending on K_{max}. $\boldsymbol{f^*}$ is also continuous on compact subspace $\boldsymbol{\mathcal{X}}$ (because all the tensor components considered are bounded in condition **AS.1**) and thus is bounded. The universal approximating property in condition **AS.3** makes equation (C.1) vanishes as ϵ goes to zero. Thus, the consistency result can be established if we show $|\mathcal{R}(\boldsymbol{f_n^\lambda}) - \mathcal{R}(\boldsymbol{f^\lambda})|$ converges to zero. This can be done with Rademacher complexity, see Chapter 26 in [125].

From the objective function (4.9), we have

$$\mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f_n}) + \lambda_n||\boldsymbol{f_n}||^2 \leqslant L_0 \tag{C.2}$$

under condition **AS.2** when we simply let $\boldsymbol{f} = 0$ as a naive classifier. Thus, $||\boldsymbol{f_n}|| \leqslant \sqrt{\frac{L_0}{\lambda_n}}$. Let $M_n = \sqrt{\frac{L_0}{\lambda_n}}$. $\boldsymbol{f_\epsilon} \in \boldsymbol{\mathcal{H}}$ such that $\mathcal{R}_{\mathcal{L}}(\boldsymbol{f_\epsilon}) \leqslant \mathcal{R}_{\mathcal{L}}(\boldsymbol{f^\lambda}) + \frac{\epsilon}{2}$. $||\boldsymbol{f_\epsilon}|| \leqslant M_n$ when n is sufficiently large. Due to condition **AS.4**, $\lambda_n \to 0$, making $M_n \to \infty$. Further notice that we introduce $\boldsymbol{f_\epsilon}$ since it is independent of n. As a result, its norm, even though is bounded by M_n, is a constant and is not changing with respect to n. By Rademacher complexity, the following inequality holds with probability at least $1-\delta$, where $0<\delta<1$

$$\begin{aligned}
\mathcal{R}_{\mathcal{L}}(\boldsymbol{f_n^\lambda}) &\leqslant \mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f_n^\lambda}) + \frac{2C(K_{max})M_n}{\sqrt{n}} + (L_0 + C(K_{max})M_n)\sqrt{\frac{\log 2/\delta}{2n}} \\
\text{$\boldsymbol{f_\epsilon}$ is not the optimal in training data} \quad &\leqslant \mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f_\epsilon}) + \lambda_n||\boldsymbol{f_\epsilon}||^2 - \lambda_n||\boldsymbol{f_n^\lambda}||^2 + \frac{2C(K_{max})M_n}{\sqrt{n}} \\
&+ (L_0 + C(K_{max})M_n)\sqrt{\frac{\log 2/\delta}{2n}} \\
\text{Drop } (\lambda_n||\boldsymbol{f_n^\lambda}||^2 > 0) \quad &\leqslant \mathcal{R}_{\mathcal{L},T_n}(\boldsymbol{f_\epsilon}) + \lambda_n||\boldsymbol{f_\epsilon}||^2 + \frac{2C(K_{max})M_n}{\sqrt{n}} \\
&+ (L_0 + C(K_{max})M_n)\sqrt{\frac{\log 2/\delta}{2n}} \\
\text{Rademacher Complexity again} \quad &\leqslant \mathcal{R}_{\mathcal{L}}(\boldsymbol{f_\epsilon}) + \lambda_n||\boldsymbol{f_\epsilon}||^2 + \frac{4C(K_{max})M_n}{\sqrt{n}} \\
&+ 2(L_0 + C(K_{max})M_n)\sqrt{\frac{\log 2/\delta}{2n}}
\end{aligned}$$

Let $\delta = \frac{1}{n^2}$, and N large such that for all $n > N$,

$$\lambda_n||\boldsymbol{f_\epsilon}||^2 + \frac{4C(K_{max})M_n}{\sqrt{n}} + 2(L_0 + C(K_{max})M_n)\sqrt{\frac{\log 2/\delta}{2n}} \leqslant \frac{\epsilon}{2}$$

The inequality exists because $||\boldsymbol{f_\epsilon}||$ is a constant with respect to n, and all other terms are converging to zero. Thus

$$\mathcal{R}_{\mathcal{L}}(\boldsymbol{f_n^\lambda}) \leqslant \mathcal{R}_{\mathcal{L}}(\boldsymbol{f_\epsilon}) + \frac{\epsilon}{2} \leqslant \mathcal{R}_{\mathcal{L}}(\boldsymbol{f^\lambda}) + \epsilon$$

with probability $1-\frac{1}{n^2}$. We conclude that

$$\mathbb{P}(|\mathcal{R}_{\mathcal{L}}(\boldsymbol{f_n^\lambda}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f^\lambda})| \geqslant \epsilon) \to 0 \tag{C.3}$$

for any arbitrary ϵ. This establishes the weak consistency of CP-STM. For strong consistency, we consider for each n

$$\sum_{n=1}^{\infty} \mathbb{P}(|\mathcal{R}_{\mathcal{L}}(\boldsymbol{f_n^\lambda}) - \mathcal{R}_{\mathcal{L}}(\boldsymbol{f^\lambda})| \geqslant \epsilon) \leqslant N - 1 + \sum_{n=1}^{\infty}\frac{1}{n^2} \leqslant \infty$$

By Borel-Cantelli Lemma, $\mathcal{R}_{\mathcal{L}}(\boldsymbol{f_n^\lambda}) \to \mathcal{R}_{\mathcal{L}}(\boldsymbol{f^\lambda})$ almost surely, see [42]. The proof is finished. □

C.2 Data Pre-processing for Section 4.7

We provide further details about our EEG-fMRI data pre-processing and fMRI data extraction in this section. Most of the processing steps are referred from [65].

C.2.1 fMRI Data

The fMRI data processing includes three major steps, which are pre-processing, regions of interests (ROI) identification, and data extraction. We describe all these steps here. All the steps are performed by SPM 12 in Matlab. There are five steps in the image pre-processing part including realignment, co-registration, segment, normalization, and smoothing.

- **Realignment**: It is a procedure to align all the 3D BOLD volumes recorded along the time to remove artifacts caused by head motions, and also to estimate head position. For each task, there are three sessions of fMRI scans, providing 510 scans in total for each subject. These scans are realigned within subject to the average of these 510 scans. (average across time) In SPM, we create three independent sessions to load all the fMRI runs, and choose not to reslice all the images at this step. The reslicing will be done in normalization step. Avoiding extra reslicing can avoid introducing new artifacts. The mean scan is created in this step for co-registration.

- **Co-registration**: Since all the fMRI scans are aligned to the mean scan, we have to transform the T1 weighted anatomical scan to match their orientation. Reason for doing this is that all the data will finally be transformed to a standardized space. Estimating such a transformation with T1 weighted scan can provide a high accuracy, since anatomical scans have higher resolutions. Matching the orientation of T1 weighted scan with all the fMRI scans makes it possible to apply the transformation estimated from T1 scan directly on fMRI data. In this step, we let the mean fMRI scan to be stationary, and move T1 anatomical scan to match it. A resliced T1 weighted scan is created in this step.

- **Segment**: This step estimate a deformation transformation mapping data into MNI 152 template space [83, 22]. A forward deformation field is created in this step.

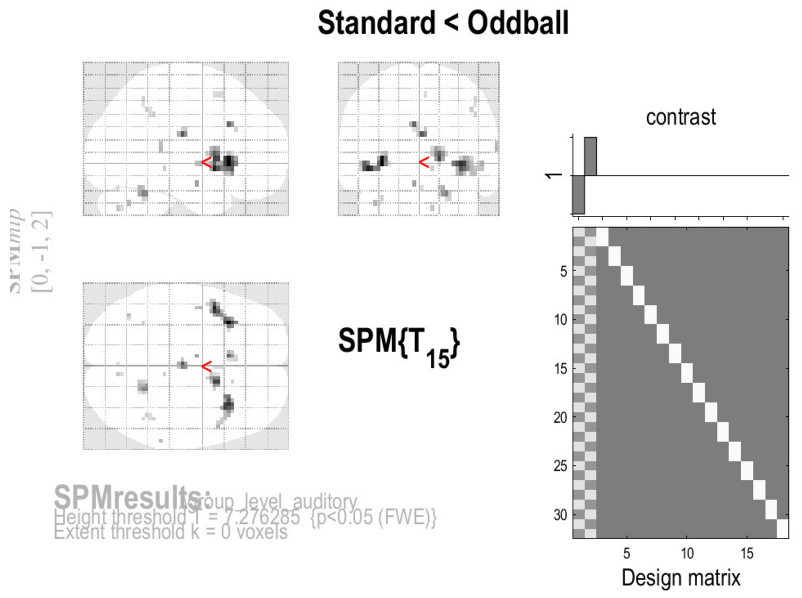
- **Normalization**: In this step, the forward deformation is applied to all realigend fMRI scans, transforming all the data into MNI template space. The voxel size is set to be $3 \times 3 \times 4$ mm, which is the same as the original images.

- **Smoothing**: All normalized fMRI volumes are then smoothed by 3D Gaussian kernels with full width at half maximum (FWHM) parameter being $8 \times 8 \times 8$.

This pre-processing procedure is applied to auditory and visual fMRI scans separately and independently.

For each task, the processed fMRI are used to for statistcal analysis introduced in [96, 145]. These models are basic linear mixed effect model with auto-regression covariance structure. Since these models are standard and are out of the scope of this dissertation, we do not introduce them in this part. For the first level (subject level) analysis, we use the model to estimate two contrast images: standard stimulus over baseline and oddball stimulus over baseline. These two are difference of average BOLD signals during stimulus time and that during no stimulus (baseline) time. They can be understand as the estimate $\hat{\beta}$ in a regression model $y = x\beta + \epsilon$. These contrasts are then pooled together in the group-level analysis. For each voxel, the group-level analysis performs a T-test to compare the BOLD signals in standard contrasts and oddball contrasts. For voxels whose test results is significant, SPM highlighted them as the regions of interest (ROI). The ROI of auditory and visual tasks are presented in the figure C.1 and figure C.2 with P-values. Figure C.3 shows the exact ROIs in the standard brain template in SPM 12.

The coordinates of these activate voxels are also provided in the statistical analysis results. To extract ROI data, we can use "spm_get_data" function in SPM 12. Since we are classifying trials, we only take one fMRI scan for each trial. This is because the trial duration (0.6 sec) is less than the repetition time (2 sec) of fMRI data. For each trial, we take the k-th fMRI scan where "k = round(onset / TR) + 1". This option is also inspired by SPM codes.


Standard < Oddball
SPMmip
[0, -1, 2]
contrast
1
SPM{T_{15}}
SPMresults: group_level_auditory
Height threshold T = 7.276285 {p<0.05 (FWE)}
Extent threshold k = 0 voxels
5
10
15
20
25
30
5
10
15
Design matrix

Statistics: ***p-values adjusted for search volume***

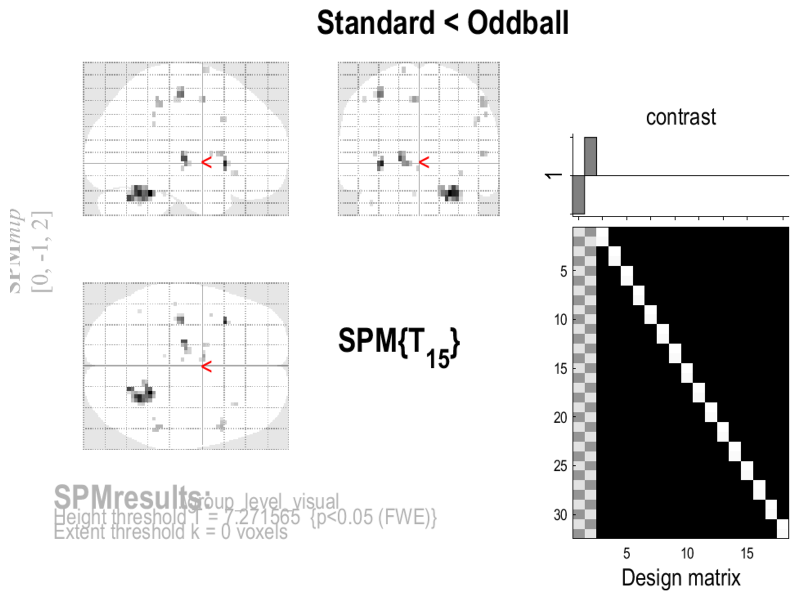
set-level		cluster-level				peak-level					mm mm mm		
p	c	$p_{FWE-corr}$	$q_{FDR-corr}$	k_E	p_{uncorr}	$p_{FWE-corr}$	$q_{FDR-corr}$	T	(Z_E)	p_{uncorr}			
0.00018		0.000	0.000	47	0.000	0.000	0.191	11.77	5.83	0.000	-33	20	-2
						0.000	0.191	11.00	5.67	0.000	-45	8	-6
		0.000	0.000	47	0.000	0.000	0.191	10.62	5.59	0.000	33	20	-2
						0.010	0.765	8.29	5.01	0.000	51	11	-2
						0.017	0.782	7.95	4.91	0.000	45	17	-10
		0.000	0.000	28	0.000	0.001	0.196	10.30	5.52	0.000	15	8	2
		0.000	0.002	9	0.001	0.001	0.230	9.93	5.43	0.000	0	-19	22
		0.000	0.003	8	0.001	0.001	0.249	9.68	5.37	0.000	-6	20	30
		0.000	0.000	17	0.000	0.003	0.421	9.04	5.21	0.000	21	-52	-26
		0.015	0.179	1	0.179	0.008	0.677	8.50	5.07	0.000	-6	-7	-18
		0.005	0.090	2	0.065	0.013	0.767	8.11	4.95	0.000	63	-37	10
		0.005	0.090	2	0.065	0.013	0.767	8.11	4.95	0.000	-45	-4	-2
		0.005	0.090	2	0.065	0.016	0.782	7.98	4.92	0.000	-39	-64	-34
		0.005	0.090	2	0.065	0.022	0.812	7.80	4.86	0.000	9	-73	-38
		0.015	0.179	1	0.179	0.028	0.812	7.65	4.81	0.000	-33	-52	-30
		0.001	0.034	4	0.013	0.028	0.812	7.64	4.81	0.000	45	-43	38
		0.015	0.179	1	0.179	0.028	0.812	7.63	4.81	0.000	9	-34	-6
		0.005	0.090	2	0.065	0.032	0.812	7.56	4.78	0.000	45	-52	50
		0.015	0.179	1	0.179	0.034	0.825	7.51	4.77	0.000	15	23	-2
		0.015	0.179	1	0.179	0.042	0.887	7.39	4.73	0.000	0	-73	-22
		0.002	0.063	3	0.028	0.043	0.887	7.37	4.72	0.000	3	14	54

table shows 3 local maxima more than 8.0mm apart

Height threshold: T = 7.28, p = 0.000 (0.050)
Extent threshold: k = 0 voxels
Expected voxels per cluster, <k> = 0.588
Expected number of clusters, <c> = 0.08
FWEp: 7.276, FDRp: Inf, FWEc: 1, FDRc: 4

Degrees of freedom = [1.0, 15.0]
FWHM = 13.4 13.4 12.6 mm mm mm; 4.5 4.5 3.1 {voxels}
Volume: 1327176 = 36866 voxels = 515.1 resels
Voxel size: 3.0 3.0 4.0 mm mm mm; (resel = 62.78 voxels)

Figure C.1: Auditory fMRI Group Level Analysis


Standard < Oddball
SPMmip
[0, -1, 2]
contrast
1
SPM{T15}
5
10
15
20
25
30
5
10
15
Design matrix
SPMresults: group_level_visual
Height threshold T = 7.271565 {p<0.05 (FWE)}
Extent threshold k = 0 voxels

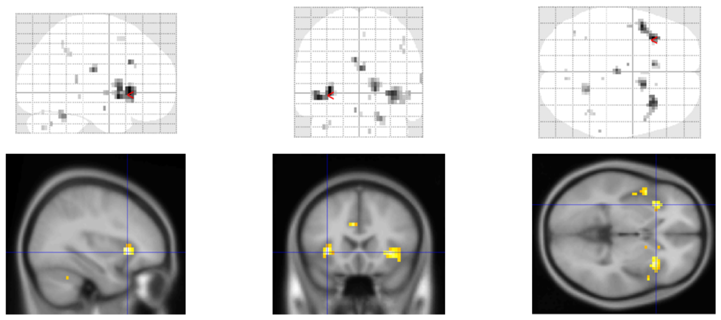
Statistics: ***p-values adjusted for search volume***

set-level		cluster-level				peak-level					mm mm mm		
p	c	$p_{FWE-corr}$	$q_{FDR-corr}$	k_E	p_{uncorr}	$p_{FWE-corr}$	$q_{FDR-corr}$	T	(Z_E)	p_{uncorr}			
0.00017		0.000	0.000	56	0.000	0.001	0.429	10.15	5.49	0.000	24	-46	-26
						0.001	0.429	9.69	5.38	0.000	27	-55	-26
						0.003	0.481	9.20	5.26	0.000	21	-64	-30
		0.000	0.010	6	0.002	0.001	0.429	9.68	5.38	0.000	-33	17	-2
		0.000	0.000	12	0.000	0.003	0.481	9.10	5.23	0.000	-18	-16	2
		0.000	0.000	12	0.000	0.006	0.620	8.70	5.12	0.000	-36	-19	54
		0.001	0.028	4	0.010	0.013	0.910	8.11	4.95	0.000	-6	-1	50
		0.002	0.047	3	0.022	0.018	0.916	7.91	4.89	0.000	51	14	6
		0.000	0.016	5	0.005	0.019	0.916	7.87	4.88	0.000	51	-37	46
		0.002	0.047	3	0.022	0.027	0.916	7.65	4.81	0.000	48	14	34
		0.015	0.161	1	0.161	0.028	0.916	7.63	4.81	0.000	54	-43	38
		0.005	0.093	2	0.055	0.030	0.916	7.60	4.80	0.000	-15	-1	74
		0.005	0.093	2	0.055	0.032	0.916	7.55	4.78	0.000	33	26	-10
		0.015	0.161	1	0.161	0.034	0.916	7.51	4.77	0.000	-3	-19	-6
		0.015	0.161	1	0.161	0.036	0.916	7.48	4.76	0.000	-39	5	6
		0.015	0.161	1	0.161	0.036	0.916	7.48	4.76	0.000	45	-43	30
		0.015	0.161	1	0.161	0.042	0.971	7.38	4.73	0.000	-48	-25	50
		0.015	0.161	1	0.161	0.047	0.993	7.31	4.70	0.000	-6	-34	-26
		0.015	0.161	1	0.161	0.050	0.993	7.28	4.69	0.000	-42	-55	-34

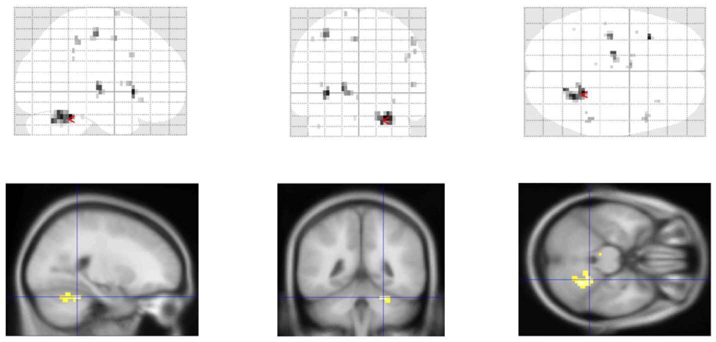
table shows 3 local maxima more than 8.0mm apart

Height threshold: T = 7.27, p = 0.000 (0.050)	Degrees of freedom = [1.0, 15.0]
Extent threshold: k = 0 voxels	FWHM = 13.0 13.0 12.1 mm mm mm; 4.3 4.3 3.0 {voxels}
Expected voxels per cluster, <k> = 0.538	Volume: 1316988 = 36583 voxels = 559.8 resels
Expected number of clusters, <c> = 0.09	Voxel size: 3.0 3.0 4.0 mm mm mm; (resel = 57.29 voxels)
FWEp: 7.272, FDRp: Inf, FWEc: 1, FDRc: 3	

Figure C.2: Visual fMRI Group Level Analysis



(a) Auditory Task



(b) Visual Task

Figure C.3: Region of Interest (ROI)

Tasks	Auditory Oddball	Auditory Standard	Visual Oddball	Visual Standard
Subject 1	75	299	75	299
Subject 2	70	287	70	287
Subject 3	74	296	74	296
Subject 5	74	299	74	299
Subject 6	75	290	75	290
Subject 7	73	295	73	295
Subject 8	72	297	72	297
Subject 9	75	297	75	298
Subject 10	72	298	72	298
Subject 11	70	293	70	293
Subject 12	74	299	74	299
Subject 13	71	297	71	297
Subject 14	75	296	75	296
Subject 15	72	295	72	295
Subject 16	74	293	74	293
Subject 17	73	295	73	295

Table C.1: EEG-fMRI Data: Number of Trials per Subject

C.2.2 EEG Data

The EEG data is collected by a custom built MR-compatible EEG system with 49 channels. [141] provides a version of re-referenced EEG data with 34 channels which are used in our experiment. The original and re-referenced channel positions are provided in the figure C.4. This version of EEG data are sampled at 1,000 Hz, and are downsampled to 200 Hz at the beginning of pre-processing. We use the “resample" function in Matlab Signal Processing toolbox to downsampled EEG data to 200 Hz. Then, we use function "ft_preproc_lowpassfilter" and "ft_preproc_highpassfilter" from SPM 12 toolbox to filter the data. This step intends to remove both low-frequency and high-frequency noise in the data. Finally, we split EEG into epochs for trials which starts 100 ms before the onset and ends 500 ms after the onset. According to [65], such a duration is long enough to capture the event-related potential during each trial for EEG data. We show few examples of latent factors from EEG data estimated by our ACMTF decomposition in figure C.5. For each trial, the topoplot shows the components from channel mode, and the other plot shows the factors from time mode.

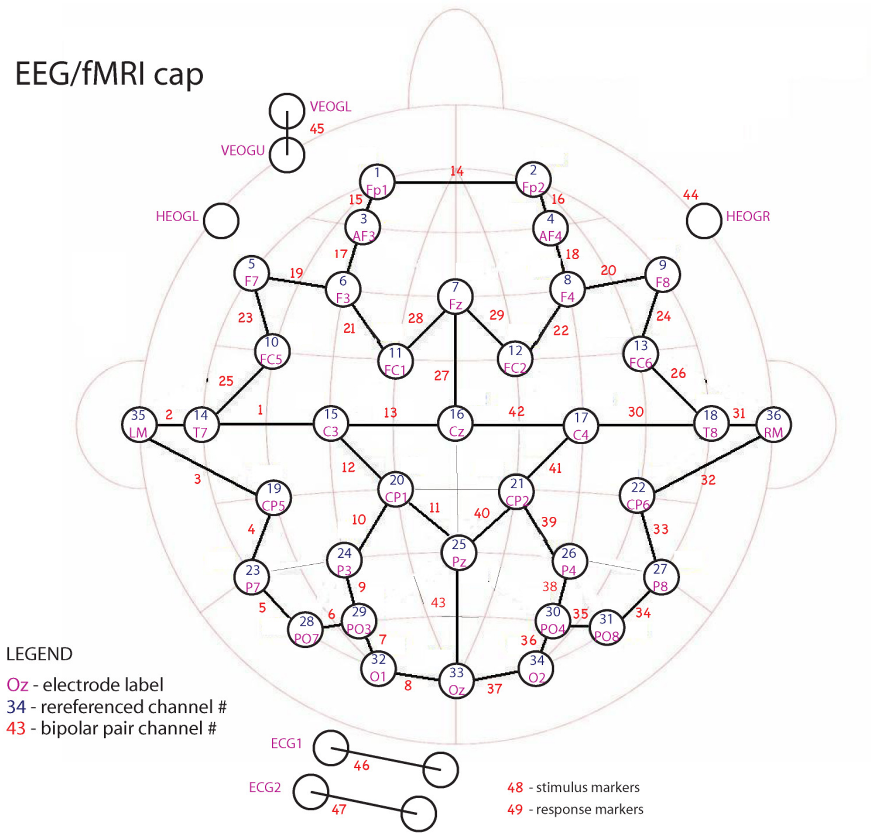


Figure C.4: EEG Channel Position from [141]

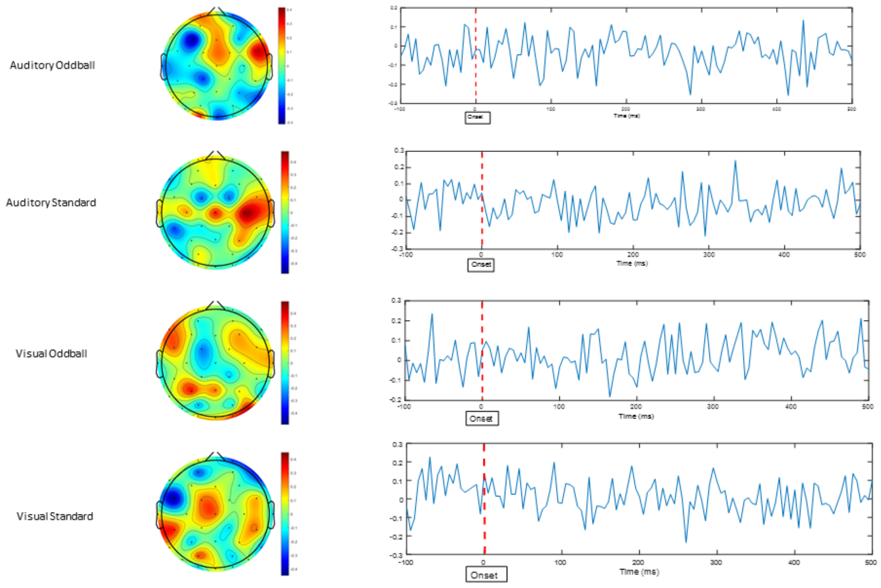


Figure C.5: Examples of EEG Latent Factors (Different Trial and Stimulus Types): Topoplot for Channel Factors (left); Plots for Temporal Factors (right)

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] E. Acar, D. M. Dunlavy, and T. G. Kolda. A scalable optimization approach for fitting canonical tensor decompositions. *Journal of Chemometrics*, 25(2):67–86, 2011.

[2] Evrim Acar, Tamara G Kolda, and Daniel M Dunlavy. All-at-once optimization for coupled matrix and tensor factorizations. *arXiv preprint arXiv:1105.3422*, 2011.

[3] Evrim Acar, Yuri Levin-Schwartz, Vince D Calhoun, and Tülay Adali. Acmtf for fusion of multi-modal neuroimaging data and identification of biomarkers. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 643–647. IEEE, 2017.

[4] Evrim Acar, Yuri Levin-Schwartz, Vince D Calhoun, and Tülay Adali. Tensor-based fusion of eeg and fmri to understand neurological changes in schizophrenia. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4. IEEE, 2017.

[5] Evrim Acar, Evangelos E Papalexakis, Gözde Gürdeniz, Morten A Rasmussen, Anders J Lawaetz, Mathias Nilsson, and Rasmus Bro. Structure-revealing data fusion. *BMC bioinformatics*, 15(1):1–17, 2014.

[6] Evrim Acar, Carla Schenker, Yuri Levin-Schwartz, Vince D Calhoun, and Tülay Adali. Unraveling diagnostic biomarkers of schizophrenia through structure-revealing fusion of multi-modal neuroimaging data. *Frontiers in neuroscience*, 13:416, 2019.

[7] Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of computer and System Sciences*, 66(4):671–687, 2003.

[8] Jeffrey S Anderson, Jared A Nielsen, Alyson L Froehlich, Molly B DuBray, T Jason Druzgal, Annahir N Cariello, Jason R Cooperrider, Brandon A Zielinski, Caitlin Ravichandran, P Thomas Fletcher, et al. Functional connectivity magnetic resonance imaging classification of autism. *Brain*, 134(12):3742–3754, 2011.

[9] Andreas Argyriou, Charles A Micchelli, and Massimiliano Pontil. When is there a representer theorem? vector versus matrix regularizers. *The Journal of Machine Learning Research*, 10:2507–2529, 2009.

[10] John Ashburner, Gareth Barnes, Chun-Chuan Chen, Jean Daunizeau, Guillaume Flandin, Karl Friston, Stefan Kiebel, James Kilner, Vladimir Litvak, Rosalyn Moran, et al. Spm12 manual. *Wellcome Trust Centre for Neuroimaging, London, UK*, 2464, 2014.

[11] Francis R Bach. Consistency of the group lasso and multiple kernel learning. *Journal of Machine Learning Research*, 9(6), 2008.

[12] Ivana Balažević, Carl Allen, and Timothy M Hospedales. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590*, 2019.

[13] Peter L Bartlett, Michael I Jordan, and Jon D McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

[14] Asa Ben-Hur and William Stafford Noble. Kernel methods for predicting protein–protein interactions. *Bioinformatics*, 21(suppl_1):i38–i46, 2005.

[15] Kristin P Bennett, Michinari Momma, and Mark J Embrechts. Mark: A boosting algorithm for heterogeneous kernel models. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 24–31, 2002.

[16] Austin R Benson, David F Gleich, and Jure Leskovec. Tensor spectral clustering for partitioning higher-order network structures. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 118–126. SIAM, 2015.

[17] Xuan Bi, Annie Qu, Xiaotong Shen, et al. Multilayer tensor factorization with applications to recommender systems. *Annals of Statistics*, 46(6B):3308–3333, 2018.

[18] Xuan Bi, Xiwei Tang, Yubai Yuan, Yanqing Zhang, and Annie Qu. Tensors in statistics. *Annual Review of Statistics and Its Application*, 8, 2020.

[19] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250, 2001.

[20] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[21] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[22] Matthew Brett, Kalina Christoff, Rhodri Cusack, Jack Lancaster, et al. Using the talairach atlas with the mni template. *Neuroimage*, 13(6):85–85, 2001.

[23] Vince D Calhoun, Tulay Adali, NR Giuliani, JJ Pekar, KA Kiehl, and GD Pearlson. Method for multimodal analysis of independent source differences in schizophrenia: combining gray matter structural and auditory oddball functional data. *Human brain mapping*, 27(1):47–62, 2006.

[24] Timothy I Cannings and Richard J Samworth. Random-projection ensemble classification. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(4):959–1035, 2017.

[25] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

[26] Christos Chatzichristos, Mike Davies, Javier Escudero, Eleftherios Kofidis, and Sergios Theodoridis. Fusion of eeg and fmri via soft coupled tensor decompositions. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 56–60. IEEE, 2018.

[27] Christos Chatzichristos, Eleftherios Kofidis, Lieven De Lathauwer, Sergios Theodoridis, and Sabine Van Huffel. Early soft and flexible fusion of eeg and fmri via tensor decompositions. *arXiv preprint arXiv:2005.07134*, 2020.

[28] Cong Chen, Kim Batselier, Ching-Yun Ko, and Ngai Wong. A support tensor train machine. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[29] Di-Rong Chen and Han Li. Convergence rates of learning algorithms by random projection. *Applied and Computational Harmonic Analysis*, 37(1):36–51, 2014.

[30] Xinyu Chen, Zhaocheng He, and Jiawei Wang. Spatial-temporal traffic speed patterns discovery and incomplete data recovery via svd-combined tensor decomposition. *Transportation research part C: emerging technologies*, 86:59–77, 2018.

[31] Yanyan Chen, Kuaini Wang, and Ping Zhong. One-class support tensor machine. *Knowledge-Based Systems*, 96:14–28, 2016.

[32] Mario Christoudias, Raquel Urtasun, Trevor Darrell, et al. Bayesian localized multiple kernel learning. *Univ. California Berkeley, Berkeley, CA*, 2009.

[33] R Dennis Cook. *Regression graphics: Ideas for studying regressions through graphics*, volume 482. John Wiley & Sons, 2009.

[34] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003.

[35] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. On the best rank-1 and rank-(r 1, r 2,..., rn) approximation of higher-order tensors. *SIAM journal on Matrix Analysis and Applications*, 21(4):1324–1342, 2000.

[36] Luc Devroye, László Györfi, and Gábor Lugosi. *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media, 2013.

[37] Yiming Ding, Jae Ho Sohn, Michael G Kawczynski, Hari Trivedi, Roy Harnish, Nathaniel W Jenkins, Dmytro Lituiev, Timothy P Copeland, Mariam S Aboian, Carina Mari Aparici, et al. A deep learning model to predict a diagnosis of alzheimer disease by using 18f-fdg pet of the brain. *Radiology*, 290(2):456–464, 2019.

[38] Nemanja Djuric, Liang Lan, Slobodan Vucetic, and Zhuang Wang. Budgetedsvm: A toolbox for scalable svm approximations. *The Journal of Machine Learning Research*, 14(1):3813–3817, 2013.

[39] Leo Doktorski. L2-svm: Dependence on the regularization parameter. *Pattern Recognition and Image Analysis*, 21(2):254–257, 2011.

[40] Olivier Duchenne, Francis Bach, In-So Kweon, and Jean Ponce. A tensor-based algorithm for high-order graph matching. *IEEE transactions on pattern analysis and machine intelligence*, 33(12):2383–2395, 2011.

[41] Robert Durrant and Ata Kabán. Sharp generalization error bounds for randomly-projected classifiers. In *International Conference on Machine Learning*, pages 693–701, 2013.

[42] Rick Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

[43] Hadi Fanaee-T and Joao Gama. Simtensor: A synthetic tensor data generator. *arXiv preprint arXiv:1612.03772*, 2016.

[44] Hadi Fanaee-T and Joao Gama. Tensor-based anomaly detection: An interdisciplinary survey. *Knowledge-Based Systems*, 98:130–147, 2016.

[45] Long Feng, Xuan Bi, and Heping Zhang. Brain regions identified as being associated with verbal reasoning through the use of imaging regression via internal variation. *Journal of the American Statistical Association*, pages 1–15, 2020.

[46] Xiaoli Z Fern and Carla E Brodley. Random projection for high dimensional data clustering: A cluster ensemble approach. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 186–193, 2003.

[47] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[48] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[49] Jannik Fritsch, Tobias Kuehnl, and Andreas Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2013.

[50] Glenn Fung, Murat Dundar, Jinbo Bi, and Bharat Rao. A fast iterative algorithm for fisher discriminant using heterogeneous kernels. In *Proceedings of the twenty-first international conference on Machine learning*, page 40, 2004.

[51] Mostafa Reisi Gahrooei, Hao Yan, Kamran Paynabar, and Jianjun Shi. Multiple tensor-on-tensor regression: An approach for modeling processes with heterogeneous sources of data. *Technometrics*, pages 1–23, 2020.

[52] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[53] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[54] Mark Girolami and Mingjun Zhong. Data integration for classification problems employing gaussian process priors. In *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, volume 19, page 465. MIT Press, 2007.

[55] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *The Journal of Machine Learning Research*, 12:2211–2268, 2011.

[56] Adrian R Groves, Christian F Beckmann, Steve M Smith, and Mark W Woolrich. Linked independent component analysis for multimodal data fusion. *Neuroimage*, 54(3):2198–2217, 2011.

[57] Rajarshi Guhaniyogi, Shaan Qamar, and David B Dunson. Bayesian tensor regression. *The Journal of Machine Learning Research*, 18(1):2733–2763, 2017.

[58] Weiwei Guo, Irene Kotsia, and Ioannis Patras. Tensor learning for regression. *IEEE Transactions on Image Processing*, 21(2):816–827, 2011.

[59] Wolfgang Hackbusch. *Tensor spaces and numerical tensor calculus*, volume 42. Springer, 2012.

[60] Peter Hall and Richard J Samworth. Properties of bagged nearest neighbour classifiers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(3):363–379, 2005.

[61] 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.

[62] Lifang He, Kun Chen, Wanwan Xu, Jiayu Zhou, and Fei Wang. Boosted sparse and low-rank tensor regression. *arXiv preprint arXiv:1811.01158*, 2018.

[63] 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. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 127–135. SIAM, 2014.

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

[65] Richard N Henson, Hunar Abdulrahman, Guillaume Flandin, and Vladimir Litvak. Multimodal integration of m/eeg and f/mri data in spm12. *Frontiers in neuroscience*, 13:300, 2019.

[66] 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.

[67] Heng Huang, Chris Ding, Dijun Luo, and Tao Li. Simultaneous tensor subspace selection and clustering: the equivalence of high order svd and k-means clustering. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge Discovery and Data mining*, pages 327–335, 2008.

[68] Prateek Jain and Sewoong Oh. Provable tensor factorization with missing data. *arXiv preprint arXiv:1406.2784*, 2014.

[69] Svante Janson et al. *Gaussian hilbert spaces*, volume 129. Cambridge university press, 1997.

[70] Yuwang Ji, Qiang Wang, Xuan Li, and Jie Liu. A survey on tensor techniques and applications in machine learning. *IEEE Access*, 7:162950–162990, 2019.

[71] Ruhui Jin, Tamara G Kolda, and Rachel Ward. Faster johnson-lindenstrauss transforms via kronecker products. *arXiv preprint arXiv:1909.04801*, 2019.

[72] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[73] Esin Karahan, Pedro A Rojas-Lopez, Maria L Bringas-Vega, Pedro A Valdés-Hernández, and Pedro A Valdes-Sosa. Tensor analysis and fusion of multimodal brain images. *Proceedings of the IEEE*, 103(9):1531–1559, 2015.

[74] Ali Khazaee, Ata Ebrahimzadeh, and Abbas Babajani-Feremi. Application of advanced machine learning methods on resting-state fmri network for identification of mild cognitive impairment and alzheimer's disease. *Brain imaging and behavior*, 10(3):799–817, 2016.

[75] Fei Yan Krystian Mikolajczyk Josef Kittler and Muhammad Tahir. A comparison of l1 norm and l2 norm multiple kernel svms in image and video classification.

[76] Marius Kloft, Ulf Brefeld, Soeren Sonnenburg, Pavel Laskov, Klaus-Robert Müller, and Alexander Zien. Efficient and accurate lp-norm multiple kernel learning. In *NIPS*, volume 22, pages 997–1005, 2009.

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

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

[79] Tamara Gibson Kolda. Multilinear operators for higher-order decompositions. Technical report, Sandia National Laboratories, 2006.

[80] Jean Kossaifi, Zachary C Lipton, Arinbjörn Kolbeinsson, Aran Khanna, Tommaso Furlanello, and Anima Anandkumar. Tensor regression networks. *Journal of Machine Learning Research*, 21:1–21, 2020.

[81] J. B. Kruskal. Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear Algebra and its Applications*, 18(2):95 – 138, 1977.

[82] Stephen LaConte, Stephen Strother, Vladimir Cherkassky, Jon Anderson, and Xiaoping Hu. Support vector machines for temporal classification of block design fmri data. *NeuroImage*, 26(2):317–329, 2005.

[83] Jack L Lancaster, Diana Tordesillas-Gutiérrez, Michael Martinez, Felipe Salinas, Alan Evans, Karl Zilles, John C Mazziotta, and Peter T Fox. Bias between mni and talairach coordinates analyzed using the icbm-152 brain template. *Human brain mapping*, 28(11):1194–1205, 2007.

[84] Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine learning research*, 5(Jan):27–72, 2004.

[85] Michele Larobina and Loredana Murino. Medical image file formats. *Journal of digital imaging*, 27(2):200–206, 2014.

[86] Xu Lei, Pedro A Valdes-Sosa, and Dezhong Yao. Eeg/fmri fusion based on independent component analysis: integration of data-driven and model-driven methods. *Journal of integrative neuroscience*, 11(03):313–337, 2012.

[87] Jie Li, Guan Han, Jing Wen, and Xinbo Gao. Robust tensor subspace learning for anomaly detection. *International Journal of Machine Learning and Cybernetics*, 2(2):89–98, 2011.

[88] Lexin Li and Xin Zhang. Parsimonious tensor response regression. *Journal of the American Statistical Association*, 112(519):1131–1146, 2017.

[89] Peide Li and Taps Maiti. Universal consistency of support tensor machine. In *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 608–609. IEEE, 2019.

[90] Ping Li, Trevor J Hastie, and Kenneth W Church. Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 287–296. ACM, 2006.

[91] Quefeng Li and Lexin Li. Integrative factor regression and its inference for multimodal data analysis. *arXiv preprint arXiv:1911.04056*, 2019.

[92] Qun Li and Dan Schonfeld. Multilinear discriminant analysis for higher-order tensor data classification. *IEEE transactions on pattern analysis and machine intelligence*, 36(12):2524–2537, 2014.

[93] Xiaoshan Li, Da Xu, Hua Zhou, and Lexin Li. Tucker tensor regression and neuroimaging analysis. *Statistics in Biosciences*, 10(3):520–545, 2018.

[94] Yingjie Li, Liangliang Zhang, Andrea Bozoki, David C Zhu, Jongeun Choi, and Taps Maiti. Early prediction of alzheimer's disease using longitudinal volumetric mri data from adni. *Health Services and Outcomes Research Methodology*, 20(1):13–39, 2020.

[95] Yi Lin. A note on margin-based loss functions in classification. *Statistics & probability letters*, 68(1):73–82, 2004.

[96] Martin A Lindquist et al. The statistical analysis of fmri data. *Statistical science*, 23(4):439–464, 2008.

[97] Jingyu Liu, Godfrey Pearlson, Andreas Windemuth, Gualberto Ruano, Nora I Perrone-Bizzozero, and Vince Calhoun. Combining fmri and snp data to investigate connections between brain function and genetics using parallel ica. *Human brain mapping*, 30(1):241–255, 2009.

[98] Yipeng Liu, Jiani Liu, and Ce Zhu. Low-rank tensor train coefficient array estimation for tensor-on-tensor regression. *IEEE transactions on neural networks and learning systems*, 31(12):5402–5411, 2020.

[99] Eric F Lock. Tensor-on-tensor regression. *Journal of Computational and Graphical Statistics*, 27(3):638–647, 2018.

[100] Xiaojing Long, Lifang Chen, Chunxiang Jiang, Lijuan Zhang, and Alzheimer's Disease Neuroimaging Initiative. Prediction and classification of alzheimer disease based on quantification of mri deformation. *PloS one*, 12(3):e0173372, 2017.

[101] Miles E Lopes. *A sharp bound on the computation-accuracy tradeoff for majority voting ensembles*. eScholarship, University of California, 2013.

[102] Canyi Lu, Jiashi Feng, Yudong Chen, Wei Liu, Zhouchen Lin, and Shuicheng Yan. Tensor robust principal component analysis: Exact recovery of corrupted low-rank tensors via convex optimization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5249–5257, 2016.

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

[104] Wei Lu, Fu-Lai Chung, Wenhao Jiang, Martin Ester, and Wei Liu. A deep bayesian tensor-based system for video recommendation. *ACM Transactions on Information Systems (TOIS)*, 37(1):1–22, 2018.

[105] Wenqi Lu, Zhongyi Zhu, and Heng Lian. High-dimensional quantile tensor regression. *Journal of Machine Learning Research*, 21(250):1–31, 2020.

[106] Ron Meir and Tong Zhang. Generalization error bounds for bayesian mixture algorithms. *Journal of Machine Learning Research*, 4(Oct):839–860, 2003.

[107] Charles A Micchelli, Yuesheng Xu, and Haizhang Zhang. Universal kernels. *Journal of Machine Learning Research*, 7(Dec):2651–2667, 2006.

[108] Sebastian Mika, Gunnar Ratsch, Jason Weston, Bernhard Scholkopf, and Klaus-Robert Mullers. Fisher discriminant analysis with kernels. In *Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop (cat. no. 98th8468)*, pages 41–48. Ieee, 1999.

[109] John C Morris, Catherine M Roe, Elizabeth A Grant, Denise Head, Martha Storandt, Alison M Goate, Anne M Fagan, David M Holtzman, and Mark A Mintun. Pittsburgh compound b imaging and prediction of progression from cognitive normality to symptomatic alzheimer disease. *Archives of neurology*, 66(12):1469–1475, 2009.

[110] Raziyeh Mosayebi and Gholam-Ali Hossein-Zadeh. Correlated coupled matrix tensor factorization method for simultaneous eeg-fmri data fusion. *Biomedical Signal Processing and Control*, 62:102071, 2020.

[111] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Icml*, 2011.

[112] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

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

[114] Yuqing Pan, Qing Mai, and Xin Zhang. Covariate-adjusted tensor classification in high dimensions. *Journal of the American Statistical Association*, pages 1–15, 2018.

[115] Paul Pavlidis, Jason Weston, Jinsong Cai, and William Noble Grundy. Gene functional classification from heterogeneous data. In *Proceedings of the fifth annual international conference on Computational biology*, pages 249–255, 2001.

[116] A. H. Phan, P. Tichavsky, and A. Cichocki. Low complexity damped gauss–newton algorithms for candecomp/parafac. *SIAM Journal on Matrix Analysis and Applications*, 34(1):126–147, 2013.

[117] Michael JD Powell. Nonconvex minimization calculations and the conjugate gradient method. In *Numerical analysis*, pages 122–141. Springer, 1984.

[118] Shibin Qiu and Terran Lane. A framework for multiple kernel support vector regression and its applications to sirna efficacy prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 6(2):190–199, 2008.

[119] Beheshteh T Rakhshan and Guillaume Rabusseau. Tensorized random projections. *arXiv preprint arXiv:2003.05101*, 2020.

[120] Bin Ran, Huachun Tan, Yuankai Wu, and Peter J Jin. Tensor based missing traffic data completion with spatial–temporal correlation. *Physica A: Statistical Mechanics and its Applications*, 446:54–63, 2016.

[121] Garvesh Raskutti, Ming Yuan, Han Chen, et al. Convex regularization for high-dimensional multiresponse tensor regression. *The Annals of Statistics*, 47(3):1554–1584, 2019.

[122] Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. Are loss functions all the same? *Neural computation*, 16(5):1063–1076, 2004.

[123] Katharina A Schindlbeck and David Eidelberg. Network imaging biomarkers: insights and clinical applications in parkinson's disease. *The Lancet Neurology*, 17(7):629–640, 2018.

[124] Warren Schudy and Maxim Sviridenko. Concentration and moment inequalities for polynomials of independent random variables. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 437–446. SIAM, 2012.

[125] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[126] John Shawe-Taylor, Christopher KI Williams, Nello Cristianini, and Jaz Kandola. On the eigenspectrum of the gram matrix and the generalization error of kernel-pca. *IEEE Transactions on Information Theory*, 51(7):2510–2522, 2005.

[127] Marco Signoretto, Quoc Tran Dinh, Lieven De Lathauwer, and Johan AK Suykens. Learning with tensors: a framework based on convex optimization and spectral regularization. *Machine Learning*, 94(3):303–351, 2014.

[128] Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

[129] Jing Sui, Godfrey Pearlson, Arvind Caprihan, Tülay Adali, Kent A Kiehl, Jingyu Liu, Jeremy Yamamoto, and Vince D Calhoun. Discriminating schizophrenia and bipolar disorder by fusing fmri and dti in a multimodal cca+ joint ica model. *Neuroimage*, 57(3):839–855, 2011.

[130] Will Wei Sun and Lexin Li. Store: sparse tensor response regression and neuroimaging analysis. *The Journal of Machine Learning Research*, 18(1):4908–4944, 2017.

[131] Will Wei Sun and Lexin Li. Dynamic tensor clustering. *Journal of the American Statistical Association*, 114(528):1894–1907, 2019.

[132] Yanfeng Sun, Junbin Gao, Xia Hong, Bamdev Mishra, and Baocai Yin. Heterogeneous tensor decomposition for clustering via manifold optimization. *IEEE transactions on pattern analysis and machine intelligence*, 38(3):476–489, 2015.

[133] Yiming Sun, Yang Guo, Joel A Tropp, and Madeleine Udell. Tensor random projection for low memory dimension reduction. In *NeurIPS Workshop on Relational Representation Learning*, 2018.

[134] Hiroaki Tanabe, Tu Bao Ho, Canh Hao Nguyen, and Saori Kawasaki. Simple but effective methods for combining kernels in computational biology. In *2008 IEEE International Conference on Research, Innovation and Vision for the Future in Computing and Communication Technologies*, pages 71–78. IEEE, 2008.

[135] D. Tao, X. Li, X. Wu, and S. J. Maybank. General tensor discriminant analysis and gabor features for gait recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(10), 2007.

[136] Dacheng Tao, Xuelong Li, Weiming Hu, Stephen Maybank, and Xindong Wu. Supervised tensor learning. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 8–pp. IEEE, 2005.

[137] Petr Tichavsky, Anh Huy Phan, and Zbyněk Koldovsky. Cramér-rao-induced bounds for candecomp/parafac tensor decomposition. *IEEE Transactions on Signal Processing*, 61(8):1986–1997, 2013.

[138] Théo Trouillon, Christopher R Dance, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Knowledge graph completion via complex tensor factorization. *arXiv preprint arXiv:1702.06879*, 2017.

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

[140] Manik Varma and Debajyoti Ray. Learning the discriminative power-invariance trade-off. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[141] Jennifer M Walz, Robin I Goldman, Jordan Muraskin, Bryan Conroy, Truman R Brown, and Paul Sajda. "auditory and visual oddball eeg-fmri", 2018.

[142] Huan Wang, Shuicheng Yan, Thomas S Huang, and Xiaoou Tang. A convengent solution to tensor subspace learning. In *IJCAI*, pages 629–634, 2007.

[143] Philip Wolfe. Convergence conditions for ascent methods. *SIAM review*, 11(2):226–235, 1969.

[144] Philip Wolfe. Convergence conditions for ascent methods. ii: Some corrections. *SIAM review*, 13(2):185–188, 1971.

[145] Keith J Worsley, Chien Heng Liao, John Aston, V Petre, GH Duncan, F Morales, and AC Evans. A general statistical analysis for fmri data. *Neuroimage*, 15(1):1–15, 2002.

[146] Kun Xie, Lele Wang, Xin Wang, Gaogang Xie, Jigang Wen, and Guangxing Zhang. Accurate recovery of internet traffic data: A tensor completion approach. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[147] Shuicheng Yan, Dong Xu, Qiang Yang, Lei Zhang, Xiaoou Tang, and Hong-Jiang Zhang. Discriminant analysis with tensor representation. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 526–532. IEEE, 2005.

[148] Rose Yu and Yan Liu. Learning from multiway data: Simple and efficient tensor regression. In *International Conference on Machine Learning*, pages 373–381. PMLR, 2016.

[149] Anru Zhang et al. Cross: Efficient low-rank tensor completion. *Annals of Statistics*, 47(2):936–964, 2019.

[150] Changqing Zhang, Huazhu Fu, Si Liu, Guangcan Liu, and Xiaochun Cao. Low-rank tensor constrained multiview subspace clustering. In *Proceedings of the IEEE international conference on computer vision*, pages 1582–1590, 2015.

[151] Tong Zhang et al. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32(1):56–85, 2004.

[152] Xin Zhang and Lexin Li. Tensor envelope partial least-squares regression. *Technometrics*, 59(4):426–436, 2017.

[153] Xing Zhang, Gongjian Wen, and Wei Dai. A tensor decomposition-based anomaly detection algorithm for hyperspectral image. *IEEE Transactions on Geoscience and Remote Sensing*, 54(10):5801–5820, 2016.

[154] Yanqing Zhang, Xuan Bi, Niansheng Tang, and Annie Qu. Dynamic tensor recommender systems. *Journal of Machine Learning Research*, 22(65):1–35, 2021.

[155] 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.

[156] P. Zhou and J. Feng. Outlier-robust tensor pca. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1–9, 2017.