TENSOR LEARNING WITH STRUCTURE, GEOMETRY AND MULTI-MODALITY

By

Seyyid Emre Sofuoglu

A DISSERTATION

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

Electrical Engineering – Doctor of Philosophy

2022

ABSTRACT

TENSOR LEARNING WITH STRUCTURE, GEOMETRY AND MULTI-MODALITY

By

Seyyid Emre Sofuoglu

With the advances in sensing and data acquisition technology, it is now possible to collect data from different modalities and sources simultaneously. Most of these data are multi-dimensional in nature and can be represented by multiway arrays known as tensors. For instance, a color image is a third-order tensor defined by two indices for spatial variables and one index for color mode. Some other examples include color video, medical imaging such as EEG and fMRI, spatiotemporal data encountered in urban traffic monitoring, etc.

In the past two decades, tensors have become ubiquitous in signal processing, statistics and computer science. Traditional unsupervised and supervised learning methods developed for one-dimensional signals do not translate well to higher order data structures as they get computationally prohibitive with increasing dimensionalities. Vectorizing high dimensional inputs creates problems in nearly all machine learning tasks due to exponentially increasing dimensionality, distortion of data structure and the difficulty of obtaining sufficiently large training sample size.

In this thesis, we develop tensor-based approaches to various machine learning tasks. Existing tensor based unsupervised and supervised learning algorithms extend many well-known algorithms, e.g. 2-D component analysis, support vector machines and linear discriminant analysis, with better performance and lower computational and memory costs. Most of these methods rely on Tucker decomposition which has exponential storage complexity requirements; CANDECOMP-PARAFAC (CP) based methods which might not have a solution; or Tensor Train (TT) based solutions which suffer from exponentially increasing ranks. Many tensor based methods have quadratic (w.r.t the size of data), or higher computational complexity, and similarly, high memory complexity. Moreover, they are not always designed with the particular structure of the data in mind. Many of these methods use purely algebraic measures as the objective which might not capture the local relations within data. Thus, there is a need to develop new models with better computational and memory efficiency, with the particular structure of the data and problem in mind. Finally, as tensors represent the data with more faithfulness to the original structure compared to the vectorization, they also allow coupling of heterogeneous data sources where the underlying physical relationship is known. Still, most of the work on coupled tensor decompositions does not explore supervised problems.

In order to address the issues around computational and storage complexity of tensor based machine learning, in Chapter 2, we propose a new tensor train decomposition structure, which is a hybrid between Tucker and Tensor Train decompositions. The proposed structure is used to implement Tensor Train based supervised and unsupervised learning frameworks: linear discriminant analysis (LDA) and graph regularized subspace learning. The algorithm is designed to solve extremal eigenvalue-eigenvector pair computation problems, which can be generalized to many other methods. The supervised framework, Tensor Train Discriminant Analysis (TTDA), is evaluated in a classification task with varying storage complexities with respect to classification accuracy and training time on four different datasets. The unsupervised approach, Graph Regularized TT, is evaluated on a clustering task with respect to clustering quality and training time on various storage complexities. Both frameworks are compared to discriminant analysis algorithms with similar objectives based on Tucker and TT decompositions.

In Chapter 3, we present an unsupervised anomaly detection algorithm for spatiotemporal tensor data. The algorithm models the anomaly detection problem as a low-rank plus sparse tensor decomposition problem, where the normal activity is assumed to be low-rank and the anomalies are assumed to be sparse and temporally continuous. We present an extension of this algorithm, where we utilize a graph regularization term in our objective function to preserve the underlying geometry of the original data. Finally, we propose a computationally efficient implementation of this framework by approximating the nuclear norm using graph total variation minimization. The proposed approach is evaluated for both simulated data with varying levels of anomaly strength, length and number of missing entries in the tensor as well as urban traffic data.

In Chapter 4, we propose a geometric tensor learning framework using product graph structures for tensor completion problem. Instead of purely algebraic measures such as rank, we use graph smoothness constraints that utilize geometric or topological relations within data. We prove the equivalence of a Cartesian graph structure to TT-based graph structure under some conditions. We show empirically, that introducing such relaxations due to the conditions do not deteriorate the recovery performance. We also outline a fully geometric learning method on product graphs for data completion.

In Chapter 5, we introduce a supervised learning method for heterogeneous data sources such as simultaneous EEG and fMRI. The proposed two-stage method first extracts features taking the coupling across modalities into account and then introduces kernelized support tensor machines for classification. We illustrate the advantages of the proposed method on simulated and real classification tasks with small number of training data with high dimensionality.

ACKNOWLEDGEMENTS

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

All praise be to Allah, the Lord of universe(s). Peace be upon the prophet of Allah, Muhammad (s.a.w.), who is the guide and the best example to all humanity, especially for those who are in search of the truth and wisdom. My thesis is but a miniscule result of the guidance of the prophet of Allah in the way of knowledge.

First and foremost, I would like to thank my advisor, Dr. Aviyente, for her guidance, patience, encouragement, and warm support throughout my research. Without her, the quality of my work would not have been a percent of what it is. She has been a role model, and motivation for me due to her deep dedication, firm discipline, and caring attentiveness to the work of her students.

I am deeply grateful to my wife, who sacrificed so much along this process and has been there for me through thick and thin. No amount of thanks could be enough. I would like to also thank my mother, who deserves more credit than any one person for any achievement or success of mine. Also, to my father, in light of whom, I have considered a PhD is possible for me, who always supported me with his wisdom, and who never lost his belief in me. To my mother-in-law and father-in-law, who have always been there for us, kept us in their prayers, and sincerely interested in my work. Finally, I thank my little son, who was my motivation, happiness, and emotional support.

There are too many people that were a support to me in writing this thesis for me to be able to mention each one of them. I ask for the forgiveness of any of my benefactors, to whom I could not thank in name. May Allah shower all who helped me with success and honour in both their lives.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

With the advance of sensing and data acquisition technology, it is now possible to collect data from different modalities and sources simultaneously. Some examples include medical imaging data such as fMRI, hyperspectral images, computer vision data such as multiview camera recordings, liquid-chromatography/mass spectrometry data in chemometrics, large scale gene expression data in bioinformatics. Most of these data are multi-dimensional in nature and can be represented by multiway arrays known as tensors [48]. For instance, a color image is a third-order tensor defined by two indices for spatial variables and one index for color mode. Similarly, a video comprised of color images is a fourth-order tensor, time being the fourth dimension besides spatial and spectral.

In order to efficiently process large datasets, there is an increasing interest in near real-time processing methods, especially in the case of multimedia, remote-sensing and biological data. Another challenge is to come up with methods that are scalable with the size of data. Although computational power and memory sizes of modern systems keep increasing, traditional methods are exponentially expensive. As the data collected are not always clean and complete, there is also a need for methods that are robust against missing data and outliers, and can account for the underlying structure or the topology of the data. Finally, there is also an emerging interest in extracting information from multiple heterogenous modalities simultaneously. As tensor decompositions are natural, sparse and distributed representations of big data, all the aforementioned problems spurred an interest in efficient tensor algorithms suitable for massive datasets.

Traditional unsupervised and supervised learning methods developed for one-dimensional signals, or vectors, do not translate well to higher order data structures as they get computationally prohibitive with increasing dimensionalities, namely, 'curse of dimensionality'. Vectorizing high dimensional inputs creates problems in nearly all machine learning tasks also due to the difficulty of obtaining sufficiently large training sample size. Moreover, important information about the structure of the high-dimensional space that data lie in is lost through vectorization which reduces the effectiveness of these methods. Therefore, there is a need for extracting compact representations from the original tensor data that result in accurate and interpretable models.

Tensor decompositions can be defined as structures which represent higher order data as a set of low-

order core tensors, thus allowing for better interpretability and computational advantages [45]. Tensor decompositions are natural extensions of matrix decompositions or factorizations such as singular value decomposition (SVD), principal component analysis (PCA) and non-negative matrix factorization (NMF) to higher order data. However, there are various advantages of tensor decompositions compared to matrix factorizations, such as the ability to efficiently parameterize high dimensional spaces and account for intrinsic multidimensional and distributed patterns present in data. Thus, they allow for models to capture multiple interactions and couplings in data, in a more structured and interpretable manner. Tensors first started to gain attention in psychometrics [174, 175] and then chemometrics [27, 160]. After being introduced to many fields such as signal processing [48, 47, 46, 18, 183], statistics and computer science [38, 43, 96], tensors have received great attention. The work on tensors is too diverse and numerous to be covered in this thesis, so we refer the interested reader to survey papers [93, 44, 45, 159, 81].

Recent years have also seen a growth in the development of tensor decomposition methods for machine learning. Various extensions of unsupervised methods such as PCA, robust PCA, SVD, NMF, neighborhood preserving embedding (NPE), locally linear embedding (LLE), Canonical Component Analysis (CCA) [48, 70, 154, 106, 78, 91] and supervised methods such as linear discriminant analysis (LDA), Support Vector Machines (SVM), regression, deep neural networks [195, 196, 168, 112, 104, 74, 38, 97, 137, 96, 95] to tensors have been explored. Some applications include face recognition, object classification, dimensionality reduction, clustering, anomaly detection, learning latent variable models and prediction.

Although most tensor based methods are developed as extensions of existing vector based methods, it is possible to develop new approaches with structures specific to tensors. By utilizing tensors, one can introduce structural models in many different ways such as deep tensor network structures, Kronecker or Cartesian structured product graphs, and multi-modal coupling of data collected from different physical sensors, that are otherwise not relevant or applicable. Tensor based methods also often have milder requirements for theoretical performance analysis such as convergence, sample size, and recovery guarantees [70, 44, 45].

1.1 Background and Notation

In this thesis, we denote numbers and scalars with letters such as x, y, N. Vectors are denoted by boldface lowercase letters, e.g. $\mathbf{x}, \mathbf{y}$. Matrices are denoted by italic capital letters like X, Y. Multi-dimensional tensors are denoted by calligraphic capital letters such as $\mathcal{X}, \mathcal{Y}$. 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 as a first-order tensor, and a matrix as a second-order tensor.

Let $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ be a 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 $\mathbf{a}_{i_1,\ldots i_{n-1},i_{n+1},\ldots i_N} \in \mathbb{R}^{I_n}$.

Definition 1. (Vectorization, Matricization and Reshaping) $\mathbf{V}(.)$ is a vectorization operator such that $\mathbf{V}(\mathcal{A}) \in \mathbb{R}^{I_1 I_2 \ldots I_N \times 1}$. $\mathbf{T}_n(.)$ is a tensor-to-matrix reshaping operator defined as $\mathbf{T}_n(\mathcal{A}) \in \mathbb{R}^{I_1 \ldots I_n \times I_{n+1} \ldots I_N}$ and the inverse operator is denoted as $\mathbf{T}_n^{-1}(.)$.

Definition 2. (Mode-n unfolding and canonical unfolding, [197]) The mode-n unfolding of a tensor $\mathcal{A}$ is defined as $\mathcal{A}_{(n)} \in \mathbb{R}^{I_n \times \prod_{n'=1,n' \neq n}^{N} I_{n'}}$ where the mode-n fibers of the tensor $\mathcal{A}$ are the columns of $\mathcal{A}_{(n)}$ and the remaining modes are organized accordingly along the rows.

Mode-n canonical unfolding, or mode-$1, 2, \ldots, n$ unfolding, of a tensor $\mathcal{Y}$ is defined as $\mathcal{Y}_{[n]} \in \mathbb{R}^{\prod_{i=1}^{n} I_i \times \prod_{i=n+1}^{N} I_i}$.

Definition 3. (Left and right unfolding) The left unfolding operator creates a matrix from a tensor by taking all modes except the last mode as row indices and the last mode as column indices, i.e., $\mathbf{L}(\mathcal{A}) \in \mathbb{R}^{I_1 I_2 \ldots I_{N-1} \times I_N}$ which is equivalent to $\mathbf{T}_{N-1}(\mathcal{A})$. Right unfolding transforms a tensor to a matrix by taking all the first mode fibers as column vectors, i.e. $\mathbf{R}(\mathcal{A}) \in \mathbb{R}^{I_1 \times I_2 I_3 \ldots I_N}$ which is equivalent to $\mathbf{T}_1(\mathcal{A})$. The inverse of these operators are denoted as $\mathbf{L}^{-1}(.)$ and $\mathbf{R}^{-1}(.)$, respectively. A tensor is defined to be left (right)-orthogonal if its left (right) unfolding is orthogonal.

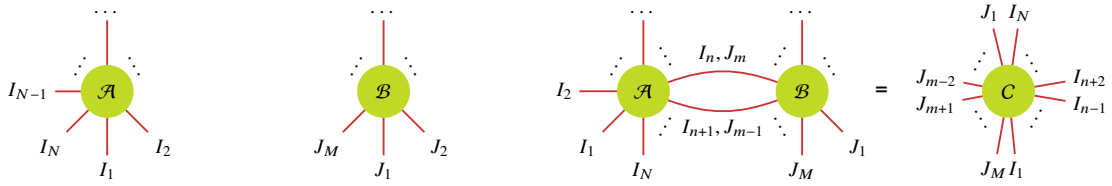
Definition 4. (Tensor trace) Tensor trace is defined on slices of a tensor and contracts them to scalars. Let $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with $I_{k'} = I_k$, then the trace operator on modes k' and k is defined as:

$$\mathcal{D} = tr_{k'}^{k}(\mathcal{A}) = \sum_{i_{k'}=i_k=1}^{I_k} \mathcal{A}(:, \ldots, i_{k'}, :, \ldots, i_k, :, \ldots, :),$$

where $\mathcal{D} \in \mathbb{R}^{I_1 \times \cdots \times I_{k'-1} \times I_{k'+1} \times \cdots \times I_{k-1} \times I_{k+1} \times \cdots \times I_N}$ is a $N-2$-mode tensor.

Definition 5. (Tensor Merging Product) Tensor merging product connects two tensors along some given sets of modes. For two tensors $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\mathcal{B} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_M}$ where $I_n = J_m$ and $I_{n+1} = J_{m-1}$ for some n and m, tensor merging product is given by [45]:

$$\mathcal{C} = \mathcal{A} \times_{n,n+1}^{m,m-1} \mathcal{B}.$$



(a) (b) (c)

Figure 1.1: Illustration of tensors and tensor merging product using tensor network notations. Each node represents a tensor and each edge represents a mode of the tensor. (a) Tensor $\mathcal{A}$, (b) Tensor $\mathcal{B}$, (c) Tensor Merging Product between modes (n, m) and $(n+1, m-1)$.

$C \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times I_{n+2} \times \cdots \times I_N \times J_1 \times \cdots \times J_{m-2} \times J_{m+1} \times \cdots \times J_M}$ is a $(N+M-4)$-mode tensor that is calculated as:

$$C(i_1, \ldots, i_{n-1}, i_{n+2}, \ldots, i_N, j_1, \ldots, j_{m-2}, j_{m+1}, \ldots, j_M) =$$
$$\sum_{t_1=1}^{I_n} \sum_{t_2=1}^{J_{m-1}} \big[\mathcal{A}(i_1, \ldots, i_{n-1}, i_n = t_1, i_{n+1} = t_2, i_{n+1}, \ldots, i_N)$$
$$\mathcal{B}(j_1, \ldots, j_{m-2}, j_{m-1} = t_2, j_m = t_1, j_{m+1}, \ldots, j_M)\big].$$

A graphical representation of tensors $\mathcal{A}$ and $\mathcal{B}$ and the tensor merging product defined above is given in Figure 1.1.

A special case of the tensor merging product can be considered for the case where $I_n = J_m$ for all $n, m \in \{1, \ldots, N-1\}, M \geq N$. In this case, the tensor merging product across the first $N-1$ modes is defined as:

$$C' = \mathcal{A} \times_{1,\ldots,N-1}^{1,\ldots,N-1} \mathcal{B}, \tag{1.1}$$

where $C' \in \mathbb{R}^{I_N \times J_N \times \cdots \times J_M}$. This can equivalently be written as:

$$\mathbf{R}(C') = \mathbf{L}(\mathcal{A})^\top \mathbf{T}_{N-1}(\mathcal{B}), \tag{1.2}$$

where $\mathbf{R}(C') \in \mathbb{R}^{I_N \times \prod_{m=N}^{M} J_m}$.

Definition 6. (Mode-n product) The mode-n product of tensor $\mathcal{A}$ and a matrix $\mathbf{U} \in \mathbb{R}^{J \times I_n}$ is denoted as $\mathcal{B} = \mathcal{A} \times_n \mathbf{U}$ and defined as $b_{i_1,\ldots,i_{n-1},j,i_{n+1}\ldots,i_N} = \sum_{i_n=1}^{I_n} a_{i_1\ldots,i_n,\ldots,i_N} u_{j,i_n}$, where $\mathcal{B} \in \mathbb{R}^{I_1 \times \ldots \times I_{n-1} \times J \times I_{n+1} \times \ldots \times I_N}$. Notice that this is equivalent to a tensor merging product $\mathcal{B} = \mathcal{A} \times_n^2 \mathbf{U}$.

Definition 7. (Mode-n Graph Laplacian) Mode-n adjacency matrix W^n corresponding to mode-n similarity graph of a tensor $\mathcal{Y}$ is defined as:

$$W^n_{ss'} = \begin{cases} e^{-\frac{\|\mathbf{y}_{(n),s} - \mathbf{y}_{(n),s'}\|_F^2}{2\sigma^2}}, & \text{if } \mathbf{y}_{(n),s} \in \mathcal{N}_k(\mathbf{y}_{(n),s'}) \text{ or } \mathbf{y}_{(n),s'} \in \mathcal{N}_k(\mathbf{y}_{(n),s}) \\ 0, & \text{otherwise} \end{cases},$$

where $\mathcal{N}_k(\mathbf{y}_{(n),s})$ is the Euclidean k-nearest neighborhood of the sth row of $\mathcal{Y}_{(n)}$, $\mathbf{y}_{(n),s} \in \mathbb{R}^{\prod_{i=1,i\neq n}^{N} I_i}$. The mode-$n$ graph Laplacian Φ_n is then defined as $\Phi_n = D^n - W^n$, where D^n is a diagonal degree matrix with $D^n_{i,i} = \sum_{i'=1}^{I_n} W^n_{i,i'}$. The eigendecomposition of Φ^n can be written as $\Phi^n = P_n \Lambda_n P_n^\top$, where P_n is the matrix of eigenvectors and Λ_n is a diagonal matrix with the eigenvalues on the diagonal, in a non-descending order.

Definition 8. (Product Graphs) Let $\{\mathcal{G}_n\}$ be a set of N graphs, where $\mathcal{G}_n = \{V_n, E_n\}$ where V_n are the vertices and E_n are the edges of $\mathcal{G}_n$. Let $\Phi_n \in \mathbb{R} I_n \times I_n$ be the graph Laplacian for G_n. A product graph $\mathcal{G}$ is created by unifying $\{\mathcal{G}_n\}$, in a special manner. A Kronecker product graph, or Kronecker graph is defined as [151]:

$$\mathcal{G} = \mathcal{G}_N \otimes \mathcal{G}_{N-1} \otimes \cdots \otimes \mathcal{G}_1, \tag{1.3}$$

and has a Laplacian defined as:

$$\Phi = \Phi_N \otimes \Phi_{N-1} \otimes \cdots \otimes \Phi_1. \tag{1.4}$$

A Cartesian product graph, on the other hand is defined as [151]:

$$\mathcal{G} = \mathcal{G}_N \otimes \mathbf{I}_{\prod_{n=1}^{N-1} I_n} + \mathbf{I}_{I_N} \otimes \mathcal{G}_{N-1} \otimes \mathbf{I}_{\prod_{n=1}^{N-2} I_n} + \cdots + \mathbf{I}_{\prod_{n=2}^{N} I_n} \otimes \mathcal{G}_1, \tag{1.5}$$

with Laplacian:

$$\Phi = \Phi_N \otimes \mathbf{I}_{\prod_{n=1}^{N-1} I_n} + \mathbf{I}_{I_N} \otimes \Phi_{N-1} \otimes \mathbf{I}_{\prod_{n=1}^{N-2} I_n} + \cdots + \mathbf{I}_{\prod_{n=2}^{N} I_n} \otimes \Phi_1. \tag{1.6}$$

The graph smoothness objective defined over all modes of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$:

$$\sum_{n=1}^{N} tr(\mathcal{X}_{(n)}^\top \Phi_n \mathcal{X}_{(n)}), \tag{1.7}$$

is equivalent to:

$$\mathbf{V}(\mathcal{X})^\top \Phi \mathbf{V}(\mathcal{X}), \tag{1.8}$$

where Φ is the graph Laplacian of the Cartesian product graph.

Definition 9. (Mode-n Concatenation) Mode-n concatenation unfolds the input tensors along mode-n and stacks the unfolded matrices across rows:

$$\text{cat}_n(\mathcal{A}, \mathcal{A}) = \begin{bmatrix} \mathcal{A}_{(n)}^\top & \mathcal{A}_{(n)}^\top \end{bmatrix}^\top. \tag{1.9}$$

If the input is a set of tensors, *e.g.* $\{\mathcal{A}\} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_M\}$, where $\mathcal{A}_m \in \mathbb{R}^{I_1 \times \cdots \times I_N}, \forall m \in \{1, \ldots, M\}$, then $\text{cat}_n(\{\mathcal{A}\})$ stacks all mode n unfoldings of tensors $\{\mathcal{A}\}$ across rows into a matrix of size $MI_n \times \prod_{n'=1, n' \neq n}^{N} I_{n'}$.

Definition 10. (Tensor norms) In this thesis, we employ three different tensor norms. Frobenius norm of a tensor is defined as $\|\mathcal{A}\|_F = \sqrt{\sum_{i_1, i_2, \ldots, i_N} y_{i_1, i_2, \ldots, i_N}^2}$. ℓ_1 norm of a tensor is defined as $\|\mathcal{A}\|_1 = \sum_{i_1, i_2, \ldots, i_N} |y_{i_1, i_2, \ldots, i_N}|$. In this thesis, the nuclear norm of a tensor is defined as the weighted sum of the nuclear norms of all mode-n unfoldings of a tensor, namely $\|\mathcal{A}\|_* = \sum_{n=1}^{N} \psi_n \|\mathcal{A}_{(n)}\|_*$, where ψ_ns are the weights corresponding to each mode [173, 70].

Definition 11. (Support Set) Let Ω be a support set defined for tensor $\mathcal{A}$, *i.e.* $\Omega \in \{1, \ldots, I_1\} \times \{1, \ldots, I_2\} \times \cdots \times \{1, \ldots, I_N\}$. The projection operator on this support set, $\mathcal{P}_\Omega$, is defined as:

$$\mathcal{P}_\Omega[\mathcal{A}]_{i_1, i_2, \ldots, i_N} = \begin{cases} \mathcal{A}_{i_1, i_2, \ldots, i_N}, & (i_1, i_2, \ldots, i_N) \in \Omega, \\ 0, & \text{otherwise.} \end{cases} \tag{1.10}$$

The orthogonal complement of the operator $\mathcal{P}_\Omega$ is defined in a similar manner as:

$$\mathcal{P}_{\Omega^\perp}[\mathcal{A}]_{i_1, i_2, \ldots, i_N} = \begin{cases} \mathcal{A}_{i_1, i_2, \ldots, i_N}, & (i_1, i_2, \ldots, i_N) \notin \Omega, \\ 0, & \text{otherwise.} \end{cases} \tag{1.11}$$

1.2 Robust Principal Component Analysis

Principal component analysis (PCA) is one of the most common algorithms for dimensionality reduction [30]. The aim of PCA is to find a subspace with smaller dimension, in which the projected data has the highest variance among all possible subspaces with the same dimension. This model is based on the assumption that the data has an i.i.d. Gaussian distribution. Thus, with grossly corrupted, or partially observed data, PCA fails to estimate the best subspace, or the best low-rank approximation to data. To alleviate the issues with these outliers, robust PCA methods [30, 35] have been proposed where the aim is to separate the data into a low-rank part, corresponding to the clean part, and a sparse part, corresponding to gross corruptions, or outliers.

In [30, 35], this problem is formulated as:

$$\text{minimize}_{L,S} \|L\|_* + \lambda \|S\|_1, \quad L + S = Y \tag{1.12}$$

where the observed data is Y and λ is a regularization parameter. It was proven that under some conditions on the true L and S, an alternating minimization based approach can recover the true L. This algorithm can also be used for matrix completion [30].

Although very successful in recovering low-rank data from gross corruptions, RPCA has some shortcomings. First, the algorithm requires an SVD at every step of the optimization which gets intractable with increasing data size. Second, the underlying assumption of incoherence is not always easy to satisfy as in many real data, the underlying, clean data is structured and can be sparse. Finally, the outliers themselves could be structured, which violates the randomness of the support of the sparse part.

1.3 Geometric Learning

In this thesis, we explore a variety of approaches to supervised and unsupervised learning where underlying geometric or topological relations between each modes rows. Such relations are common in many types of real data, e.g. recommendation systems, drug-target interaction, genomics. Usually, such relations are represented by a graph, or a set of graphs with relationships across these graphs. We unite these approaches under the term geometric learning.

Such geometric structures within data are often not taken into account in many methods as purely algebraic terms, such as rank, cannot account for them [22]. In many modern signal processing applications, graph-based priors have been used to extract low-dimensional structure from high dimensional data [57, 83, 170]. Representation of a signal on a graph is also motivated by the emerging field of signal processing on graphs, based on notions of spectral graph theory [155, 154, 146]. The underlying assumption is that high-dimensional data samples lie on or close to a smooth low-dimensional manifold, represented by a graph G. In this section, we explore several extensions of wide-known algorithms for dimensionality reduction using geometric learning concepts.

In the specific case of manifold learning, geometric learning is utilized as follows. Tensor samples for a data set with no labels are denoted as $\mathcal{Y}_s \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ where $s \in \{1, \ldots, S\}$. For two tensor samples, $\mathcal{Y}_s$ and $\mathcal{Y}_{s'}$ and their respective low-dimensional projections, X_s and $X_{s'}$, regularization that is employed to preserve the underlying geometry can be formulated as:

$$\sum_{s=1}^{S} \sum_{\substack{s'=1 \\ s' \neq s}}^{S} \|X_s - X_{s'}\|_F^2 w_{ss'}, \tag{1.13}$$

where X_s is the projection of $\mathcal{Y}_s$ to a lower dimensional manifold and $w_{s,s'}$ is the mode-$N+1$ adjacency or similarity graph. One can also equivalently rewrite (1.13) as $\text{tr}\left(X_s \Phi^{N+1} X_s^\top\right)$.

1.3.1 Robust PCA on Graphs

In [155], it was shown that a low-rank approximation, U, to a data matrix, X, can be obtained by solving the following optimization problem:

$$\min_U \|X - U\|_1 + \gamma_1 tr(U\Phi^1 U^T) + \gamma_2 tr(U^T \Phi^2 U), \tag{1.14}$$

where Φ^1 and Φ^2 are the graph Laplacians corresponding to graphs connecting the samples (rows) of X and the features (columns) of X, respectively. The above formulation assumes that the data is low-rank on graphs, *i.e.* lies on a smooth low-dimensional manifold. This can be quantified by a graph stationarity measure, $s_r(\Gamma_n) = \frac{\|\text{diag}(\Gamma_n)\|_2}{\|\Gamma_n\|_F}$, where $\Gamma_n = P_n^\top C_n P_n$ with C_n being the covariance matrix of each mode-n unfolding, i.e., $n = 1$ or 2 in the case of data matrices [146, 154].

1.3.2 Spectral Geometric Matrix Completion

In [84, 22], it was shown that using two graphs that encode the relationships between the rows, $\mathcal{G}_1$ and the columns $\mathcal{G}_2$ of a matrix, it is possible to extract a signal X that is band-limited on the Cartesian product graph $\mathcal{G} = \mathcal{G}_2 \otimes \mathbf{I} + \mathbf{I} \otimes \mathcal{G}_1$ from data Y with missing entries and corruption. The problem is formulated as:

$$\text{minimize}_X \|\mathcal{P}_\Omega[X - Y]\|_F^2 + tr(X^\top \Phi^1 X) + tr(X\Phi^2 X^\top), \tag{1.15}$$

where the last two terms are called the graph smoothness terms, and Φ^1 and Φ^2 are graph Laplacians corresponding to the rows and columns of the matrix, respectively. $\Phi = \Phi^2 \times \mathbf{I} + \mathbf{I} \times \Phi^1$ is the Laplacian matrix associated with $\mathcal{G}$.

1.4 Tensor Decompositions

In many machine learning problems, low-rank decompositions or matrix factorizations are the main computational bottleneck as they are often $O(n^3)$, *i.e.* cubic, in computational complexity. Tensor decompositions provide alternative factorization models preserving the original multiway structure of the data which decrease both the computational complexity and the storage cost of the factors. In this section, we review

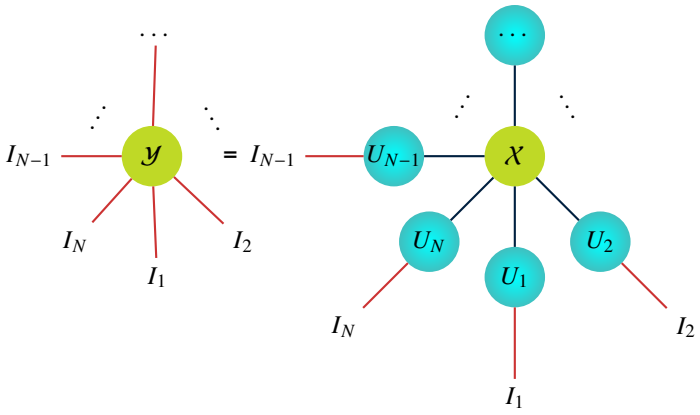


Figure 1.2: Tensor network notation for Tucker decomposition.

three well-known tensor decomposition approaches: Tucker Decomposition (TD), Canonical Decomposition (CANDECOMP/PARAFAC, CP) and Tensor Train Decomposition (TT).

1.4.1 Tucker Decomposition (TD)

Tucker decomposition represents a tensor $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ as:

$$\mathcal{Y} = \mathcal{X} \times_1 U_1 \times_2 U_2 \cdots \times_N U_N,$$

where $U_n \in \mathbb{R}^{I_n \times R_n}$ with $R_n < I_n$ are called the low-rank factor matrices and $\mathcal{X} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ is the core tensor. This decomposition is illustrated in Figure 1.2. Instead of mode-n product, Tucker Decomposition can also be represented by tensor merging product as:

$$\mathcal{Y} = \mathcal{X} \times_1^2 U_1 \times_2^2 U_2 \cdots \times_N^2 U_N,$$

1.4.2 Canonical/Polyadic (CP) Decomposition

CP is a special case of Tucker Decomposition where the core tensor $\mathcal{X}$ is superdiagonal:

$$x_{i_1,i_2,\ldots,i_N} = \begin{cases} r_{i_1,i_2,\ldots,i_N}, & \text{if } i_1 = i_2 = \cdots = i_N, \\ 0, & \text{otherwise.} \end{cases} \tag{1.16}$$

where $r_{i_1,i_2,\ldots,i_N} \in R$ is a constant. CP is often defined equivalently as the sum of rank one tensors computed by the outer products of the columns of the factor matrices U_n:

$$\mathcal{Y} = \sum_{i=1}^{R} \mathbf{u}_{1,i} \circ \mathbf{u}_{2,i} \circ \cdots \circ \mathbf{u}_{N,i}, \tag{1.17}$$

where R is the CP rank of $\mathcal{Y}$. The above equation is also denoted as:

$$\mathcal{Y} = [[U_1, U_2, \ldots, U_N]] \tag{1.18}$$

The above representation is called Kruskal tensor (see [99]), which is a convenient representation for CP tensors. We denote a Kruskal tensor by $\mathfrak{Y} = [[\mathbf{U}_1, ..., \mathbf{U}_N]]$ or $\mathfrak{Y} = [[\zeta; \mathbf{U}_1, ..., \mathbf{U}_N]]$ where $\zeta \in \mathbb{R}^r$ is a vector whose entries are the weights of rank one tensor components. In the special case of matrices, ζ corresponds to singular values of a matrix. In general, it is assumed that the rank R is small so that equation (1.18) is also called low-rank approximation of a tensor $\mathcal{Y}$.

1.4.3 Tensor Train Decomposition (TT)

Most of the existing work in both unsupervised and supervised learning have utilized Tucker or CP decompositions. The matrix product state (MPS) or Tensor Train is one of the best understood tensor networks, or tensor decompositions, for which efficient algorithms have been developed [140, 139]. TT is a special case of a tensor decomposition where a tensor with N indices is factorized into a chain-like product of low-rank, three-mode tensors. TT model has been employed in various applications such as PCA [18], manifold learning [183] and deep learning [137].

Tensor Train decomposition represents each element of $\mathcal{Y}$ using a series of matrix products as:

$$\mathcal{Y}(i_1, i_2, \ldots, i_N) = \mathcal{U}_1(1, i_1, :)\mathcal{U}_2(:, i_2, :) \ldots \mathcal{U}_N(:, i_N, :)\mathbf{x}, \tag{1.19}$$

where $\mathcal{U}_n \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$ are the three mode low-rank tensor factors, $R_n < I_n$ are the TT-ranks of the corresponding modes and $\mathbf{x} \in \mathbb{R}^{R_N \times 1}$ is the projected sample vector. Using tensor merging product form, (1.19) can be rewritten as

$$\mathcal{Y} = \mathcal{U}_1 \times_3^1 \mathcal{U}_2 \times_3^1 \cdots \times_3^1 \mathcal{U}_N \times_3^1 \mathbf{x}. \tag{1.20}$$

A graphical representation of (1.20) can be seen in Figure 1.3. If $\mathcal{Y}$ is vectorized, another equivalent expression for (1.19) in terms of matrix projection is obtained as:

$$\mathcal{V}(\mathcal{Y}) = \mathbf{L}(\mathcal{U}_1 \times_3^1 \mathcal{U}_2 \times_3^1 \cdots \times_3^1 \mathcal{U}_N)\mathbf{x}.$$

Another widely used form of Tensor Train, called Matrix Product States (MPS) [18] is defined as:

$$\mathcal{Y} = \mathcal{U}_1 \times_3^1 \mathcal{U}_2 \times_3^1 \cdots \times_3^1 \mathcal{U}_k \times_3^1 X \times_3^1 \mathcal{U}_{k+1} \times_3^1 \cdots \times_3^1 \mathcal{U}_N. \tag{1.21}$$

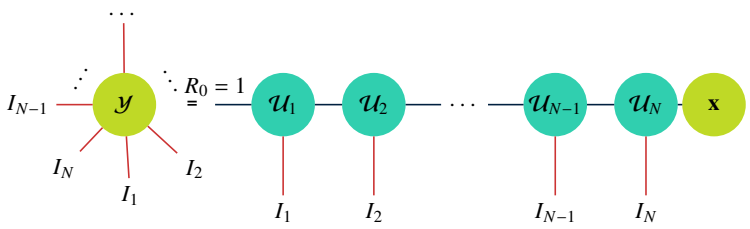


Figure 1.3: Tensor Train Decomposition of $\mathcal{Y}$ using tensor merging products.

Note that the low dimensional projected samples in MPS are matrices while for TT, they are vectors.

Let $U_{\leq k} = \mathbf{L}(\mathcal{U}_1 \times_3^1 \mathcal{U}_2 \times_3^1 \cdots \times_3^1 \mathcal{U}_k) \in \mathbb{R}^{I_1 I_2 \ldots I_k \times r_k}$ and $U_{>k} = \mathbb{R}(\mathcal{U}_{k+1} \times_3^1 \cdots \times_3^1 \mathcal{U}_N) \in \mathbb{R}^{r_{k+1} \times I_{k+1} \ldots I_N}$. When $\mathbf{L}(\mathcal{U}_n)$s for $n \leq k$ are left orthogonal, $U_{\leq k}$ is also left orthogonal [80], i.e. $\mathbf{L}(\mathcal{U}_n)^\top \mathbf{L}(\mathcal{U}_n) = \mathbb{I}_{R_{n-1} I_n}, \forall n$ implies $U^\top U = \mathbb{I}_I, I = \prod_{n=1}^N I_n$ where $\mathbb{I}_I \in \mathbb{R}^{I \times I}$ is the identity matrix. Similarly, when $\mathbb{R}(\mathcal{U}_n)$s for $n > k$ are right orthogonal, $U_{>k}$ is right orthogonal. When $\mathcal{Y}$ is reshaped into a matrix, (1.21) can be equivalently expressed as a matrix projection $\mathbf{T}_k(\mathcal{Y}) = \left[\mathbb{I}_{I_n} \otimes U_{\leq k}\right] X U_{>k}$.

1.5 Robust PCA for Tensors

In order to separate the low-rank part from sparse outliers in tensors, many extensions of RPCA to tensors have been proposed [70, 120, 208, 197]. Although utilizing tensor structures have proven to be useful in many tasks, quantifying the low rankness of tensors is an open ended problem as there is no clear definition of rank for a tensor. Similarly, the nuclear norm of a tensor, which is a convex measure of the rank, is not clearly defined. Depending on the definition of the rank, different nuclear norms are used such as Tucker rank (sum of nuclear norms (SNN) of mode-n unfoldings), tubal rank (tensor nuclear norm (TNN)), TT rank (Schatten norm (TTNN)), etc.

In [120, 70], Tucker rank was used to quantify the rank of a tensor. This interpretation resulted in a SNN based objective as:

$$\text{minimize}_{\mathcal{L},\mathcal{S}} \sum_{n=1}^{N} \|\mathcal{L}_{(n)}\|_* + \|\mathcal{S}\|_1, \tag{1.22}$$

where $\|.\|_1$ is the ℓ_1 norm of a tensor, defined as the sum of the absolute values of all entries of a tensor. There are many extensions to this formulation such as reformulating the objective function for tensor completion, using other definitions of nuclear norm, and weighting the nuclear norms of different modes.

1.6 Linear Discriminant Analysis (LDA) for Tensors

Let $\mathcal{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times K \times C}$ be the collection of samples of training tensors. For a given $\mathcal{Y}$ with C classes and K samples per class, define $\mathcal{Y}_c^k \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ as the sample tensors where $k \in \{1, \ldots, K\}$ is the sample index and $c \in \{1, \ldots, C\}$ is the class index. LDA for tensor data initially applies a vectorization to tensor samples and then finds an orthogonal projection U that maximizes the discriminability of projections by solving[1]:

$$U = \underset{\hat{U}}{\operatorname{argmin}} \left[tr(\hat{U}^\top S_W \hat{U}) - \lambda tr(\hat{U}^\top S_B \hat{U}) \right] =$$
$$\underset{\hat{U}}{\operatorname{argmin}}\, tr(\hat{U}^\top (S_W - \lambda S_B)\hat{U}) = \underset{\hat{U}}{\operatorname{argmin}}\, tr(\hat{U}^\top S \hat{U}), \tag{1.23}$$

where $S = S_W - \lambda S_B$, λ is the regularization parameter that controls the trade-off between S_W and S_B which are within-class and between-class scatter matrices, respectively, defined as:

$$S_W = \sum_{c=1}^{C} \sum_{k=1}^{K} \mathcal{V}(\mathcal{Y}_c^k - \mathcal{M}_c)\mathcal{V}(\mathcal{Y}_c^k - \mathcal{M}_c)^\top, \tag{1.24}$$

$$S_B = \sum_{c=1}^{C} \sum_{k=1}^{K} \mathcal{V}(\mathcal{M}_c - \mathcal{M})\mathcal{V}(\mathcal{M}_c - \mathcal{M})^\top, \tag{1.25}$$

where $\mathcal{M}_c = \frac{1}{K} \sum_{k=1}^{K} \mathcal{Y}_c^k$ is the mean for each class c and $\mathcal{M} = \frac{1}{CK} \sum_{c=1}^{C} \sum_{k=1}^{K} \mathcal{Y}_c^k$ is the total mean of all samples. Since U is an orthogonal projection, (1.23) is equivalent to minimizing the within-class scatter and maximizing the between class scatter of projections. This can be solved by the matrix $U \in \mathbb{R}^{\prod_{n=1}^{N} I_n \times R_N}$ whose columns are the eigenvectors of $S \in \mathbb{R}^{\prod_{n=1}^{N} I_n \times \prod_{n=1}^{N} I_n}$ corresponding to the lowest R_N eigenvalues.

Multilinear Discriminant Analysis (MDA):

MDA extends LDA to tensors using TD by finding a subspace $U_n \in \mathbb{R}^{I_n \times R_n}$ for each mode $n \in \{1, \ldots, N\}$ that maximizes the discriminability along that mode [112, 168, 195]. When the number of modes N is equal to 1, MDA is equivalent to LDA. In the case of MDA, within-class scatter along each mode $n \in \{1, \ldots, N\}$ is defined as:

$$S_W^{(n)} = \sum_{c=1}^{C} \sum_{k=1}^{K} \left[(\mathcal{Y}_c^k - \mathcal{M}_c) \prod_{\substack{m \in \{1,\ldots,N\} \\ m \neq n}} \times_m^1 U_m \right]_{(n)} \left[(\mathcal{Y}_c^k - \mathcal{M}_c) \prod_{\substack{m \in \{1,\ldots,N\} \\ m \neq n}} \times_m^1 U_m \right]_{(n)}^\top. \tag{1.26}$$

[1] The original formulation optimizes the trace ratio. Prior work showed the equivalence of trace ratio to trace difference used in this thesis [63].

Between-class scatter $S_B^{(n)}$ is found in a similar manner. Using these definitions, each U_n is found by optimizing [168]:

$$U_n = \underset{\hat{U}_n}{\operatorname{argmin}}\, tr(\hat{U}_n^\top (S_W^{(n)} - \lambda S_B^{(n)})\hat{U}_n). \tag{1.27}$$

Different implementations of the multilinear discriminant analysis have been introduced including Discriminant Analysis with Tensor Representation (DATER), Direct Generalized Tensor Discriminant Analysis (DGTDA) and Constrained MDA (CMDA). DATER minimizes the ratio $tr(U_n^\top S_W^{(n)} U_n)/tr(U_n^\top S_B^{(n)} U_n)$ [195] instead of (1.27). Direct Generalized Tensor Discriminant Analysis (DGTDA), on the other hand, computes scatter matrices without projecting inputs on U_m, where $m \neq n$ and finds an optimal U_n[112]. Constrained MDA (CMDA) finds the solution in an iterative fashion [112], where each subspace is found by fixing all other subspaces.

1.7 Organization and Contributions of the Thesis

In this thesis, we present novel algorithms for learning from tensor data for different applications including discriminative and multi-modal supervised learning, data recovery, and anomaly detection from spatiotemporal data. In all of the proposed methods, important design challenges such as keeping the storage and computational complexity low while maintaining high performance accuracy are taken into account. The resulting cost functions are formulated as optimization problems and efficient algorithms are presented to solve them. We have also provided analysis of the proposed algorithms in terms of convergence, computational and storage complexity. The algorithms were applied to classification, clustering and anomaly detection tasks.

In Chapter 2, we introduce a tensor decomposition structure that provides computational and storage efficiency while providing high accuracy for both supervised and unsupervised applications. The structure is akin to a hybrid between Tensor Train and Tucker decompositions using the flexibility of tensor networks, and named as multi-branch tensor train decomposition. This new structure is useful for extending existing machine learning methods to tensor type data. This structure is employed in a supervised learning setting for extending LDA to tensor data. We also present an unsupervised learning method that relies on the proposed structure, namely graph regularized tensor train decomposition. The proposed supervised and unsupervised algorithms are compared to state-of-the-art tensor decompositions with similar objectives in, respectively, classification and clustering settings on various data sets.

In Chapter 3, we address the problem of unsupervised anomaly detection in spatiotemporal data. In this chapter, we develop an anomaly detection method by taking the structure of anomalies into account. In particular, we model the spatiotemporal data as a tensor where the underlying data is low-rank and the anomalies are sparse and temporally continuous. In other words, anomalies appear as spatially contiguous groups of locations that show anomalous values consistently for a short duration of time. We present different methods based on these assumptions: LOw-rank plus temporally Smooth Sparse decomposition (LOSS), Graph regularized LOSS (GLOSS) and LOw-rank on Graphs plus temporally Smooth Sparse decomposition (LOGSS). LOSS and GLOSS use the same objective functions except for the graph regularization term included in GLOSS. By including a graph regularization term in the objective, we preserve the local geometry of the data. In LOGSS, different from the previous two, graph regularization is employed to approximate the nuclear norm minimization problem to reduce the computational complexity. As the proposed methods use a tensor completion framework, they are robust against missing entries in the observed spatiotemporal data.

In Chapter 4, we study geometric tensor learning, a framework that explores the effectiveness of product graph structure on tensors. This framework is utilized for a tensor recovery task where the aim is to reconstruct the underlying data from grossly corrupted and partially observed data. We present a new method with Tensor Train structure to model the data. Using TT changes the definition of the product graph as the tensor factorization is not Kronecker structured. We propose two new algorithms to model product graphs for TT decomposition, where in the first approach, we propose using a graph for each canonical mode unfolding, and in the second model, we show the equivalence of canonical unfolding to mode-n unfolding when each mode-n graph has a Kronecker structure.

In Chapter 5, we present a coupled support tensor machine framework for classification. The framework extends a supervised learning task to multiple, possibly heterogenous, data sources such as simultaneous EEG and fMRI. Starting from a kernelized support tensor machine (STM) framework formulated for single modality, we propose a multiple kernel approach. In particular, we estimate latent factors from Advanced Coupled Matrix Tensor Factorization (ACMTF) [3] jointly across modalities. This algorithm decomposes multiple tensors with CP decomposition, where some sets of factors from different modes are coupled together, i.e. has similarity conditions, to account for the same underlying physical phenomenon that affects these factors. A Coupled Support Tensor Machine (C-STM) combines individual and shared latent factors with multiple kernels and estimates a maximal-margin classifier for coupled matrix tensor data.

Finally, Chapter 6 summarizes the main contributions of this thesis and discuss potential future research

directions.

CHAPTER 2

MULTI-BRANCH TENSOR TRAIN STRUCTURE FOR SUPERVISED AND UNSUPERVISED LEARNING

2.1 Introduction

Tensor decompositions have been proposed for various tasks including dimensionality reduction, classification and clustering. In the area of unsupervised learning, many vector or matrix based methods such as PCA, SVD and NMF have been extended to tensors using various tensor decomposition structures including PARAFAC/CP, Tucker decomposition and TT [159, 42, 93, 26, 48, 49, 43, 106, 18, 181]. Tensors have also gained attention in supervised learning literature. Vectorizing high dimensional inputs may result in poor classification performance due to overfitting when the training sample size is relatively small compared to the feature vector dimension [159, 112, 169, 168, 196]. Another issue brought forth by vectorization is small sample size (SSS) problem. This problem is considered to be the main drawback of linear discriminant analysis (LDA) as the objective function becomes ill-defined due to the singularity of the estimated scatter matrices [63]. For this and many other reasons, a variety of supervised tensor learning methods for feature extraction, selection, regression and classification have been proposed [126, 168, 97, 75, 112, 74]. These include extensions of Linear Discriminant Analysis (LDA) to Multilinear Discriminant Analysis (MDA) for face and gait recognition [196, 168, 112]; Discriminant Non-negative Tensor Factorization (DNTF) [202]; Supervised Tensor Learning (STL) where one projection vector along each mode of a tensor is learned [169, 76]. More recently, the linear regression model has been extended to tensors to learn multilinear mappings from a tensorial input space to a continuous output space [73, 201]. Finally, a framework for tensor-based linear large margin classification was formulated as Support Tensor Machines (STMs), in which the parameters defining the separating hyperplane form a tensor [97, 75, 74]. However, most of these methods are based on CP or Tucker decomposition which have various problems.

CP decomposition is very efficient in terms of storage complexity and has useful properties such as uniqueness and interpretability, which is not true for TD or TT. However, finding the correct CP rank is challenging and often is done by iterating over different values. For noisy tensor data, CP decomposition may find false rank. Moreover, there may not be a low-rank approximation, which is referred to as degeneracy [93] and the problem is often ill-posed [44].

Tucker Decomposition is an extension of SVD and a widely used tensor decomposition. Also, it is more numerically stable compared to CP and most of the time, a low-rank approximation exists in Tucker form. However, Tucker representation can be exponential in storage requirements since even when each mode is low-rank with rank r, the dimensionality of a sample core tensor $\mathcal{X}_c^k$ will be r^N [182, 32]. It was also shown in [17] that learning a low-rank approximation using Tucker Decomposition requires a truncated SVD on a fat matrix, which hinders the quality of the low-rank approximation as fat or tall matrices are usually full rank.

TT format exhibits both stability and low storage complexity compared to Tucker format [42]. It was also suggested as a possible solution to the imbalance of unfolding problem encountered in Tucker decomposition in [17]. However, the ranks of the core tensors in the Tensor Train decomposition are frequently not low for real data [45] and increase exponentially with the number of modes. In [204], a solution was proposed by limiting the maximum ranks of tensor factors and optimizing accordingly. This limitation requires the original data to be low TT rank, which is not true for most real data [45, 204], and results in lower approximation accuracy. A widely used solution to these problem with TT or MPS [18] is to start the Tensor Train decomposition from the left, i.e. first mode, and the right, *i.e.* last mode, simultaneously which decreases the computational complexity significantly.

Even though early applications of TT decomposition focused on compression and dimensionality reduction [139, 140], more recently TT has been used in machine learning applications. In [18], MPS is implemented in an unsupervised manner to first compress the tensor of training samples and then the resulting lower dimensional core tensors are used as features for subsequent classification. In [182], TT decomposition is associated with a structured subspace model, namely the Tensor Train subspace. Learning this structured subspace from training data is posed as a non-convex problem referred to as TT-PCA. Once the subspaces are learned from the training data, the resulting low-dimensional subspaces are used to project and classify the test data. [183] extends TT-PCA to manifold learning by proposing a Tensor Train neighborhood preserving embedding (TTNPE). The classification is conducted by first learning a set of tensor subspaces from the training data and then projecting the training and testing data onto the learned subspaces. Apart from employing TT for subspace learning, recent work has also considered the use of TT in classifier design. In [38], a support tensor train machine (STTM) is introduced to replace the rank-1 weight tensor in Support Tensor Machine (STM) [169] by a Tensor Train that can approximate any tensor with a scalable number of parameters.

In order to address the issues of exponential storage requirements and high computational complexity, in this chapter, we introduce supervised and unsupervised subspace learning approaches based on the Tensor Train (TT) structure. Building on the idea of starting the decomposition from the first and last modes simultaneously, we show that using a number of branches in the decomposition, a hybrid between TT and Tucker Decomposition structure can be achieved which in turn can provide a balanced structure and low ranks. When the number of branches is equal to one, the decomposition is equivalent to a standard TT decomposition. When the number of branches is equal to the number of modes, the decomposition is equivalent to Tucker Decomposition. As a special case, when the number of branches is two, the structure is still denoted as TT in previous work[98, 204, 18, 45]. However, the formulation is different from one-way TT as the orthogonality requirements of some factors become right-orthogonality instead of left. By generalizing the proposed idea to different number of branches, we show that our structure is a hybrid tensor decomposition that includes TT and Tucker decomposition as special cases. The significance of the number of branches becomes more apparent for high dimensional data with large number of modes as for these data, the TT ranks tend to get large. The proposed structure is closely related to recent work in the computation of extremal eigenvalue-eigenvector pairs of large Hermitian matrices [45]. Since the main objective is to compute extremal eigenpairs, *i.e.* Ritz pairs, of a large-scale structured matrix, the proposed multi-branch approach is similar to the previously introduced block TT (BTT) format [56, 98, 204].

In particular, we present a discriminant subspace learning approach using the TT model, namely the Tensor Train Discriminant Analysis (TTDA). The proposed approach is based on linear discriminant analysis (LDA) and learns a Tensor Train subspace (TT-subspace) [183, 182] that maximizes the linear discriminant function. Although this approach provides an efficient structure for storing the learned subspaces, it is computationally prohibitive. For this reason, we propose the multi-branch structure for efficient implementations of TTDA. In the second part of this chapter, we apply the multi-branch structure to an unsupervised learning problem that can be posed as an extremal eigenvalue-eigenvector problem. This problem is a combination of manifold learning and linear subspace learning which is posed as a graph regularized dimensionality reduction algorithm.

The contributions of the chapter can be summarized as follows:

- A new tensor network structure is introduced to provide a better trade-off between computational complexity and storage cost. The proposed multi-branch structure is akin to a hybrid between tensor-

train and Tucker decompositions using the flexibility of tensor networks. This structure can also be utilized within other subspace learning tasks, as was done in both supervised and unsupervised settings in this chapter.

- This chapter is the first that uses tensor-train decomposition to formulate LDA for supervised subspace learning. Unlike recent work on TT-subspace learning [17, 18, 182, 183] which focuses on dimensionality reduction, the proposed work learns discriminative TT-subspaces to extract features that optimize the linear discriminant function.

- A theoretical analysis of storage and computational complexity of this new framework is presented. A method to find the optimal implementation of the multi-branch TT model given the dimensions of the input tensor is provided.

The rest of the chapter is organized as follows. In Section 2.2, we introduce an optimization problem to learn the TT-subspace structure that maximizes the linear discriminant function, named as Tensor Train Discriminant Analysis (TTDA). In Section 2.3, we introduce multi-branch implementations of TTDA to address the issue of high computational complexity. In Section 2.4, we provide an analysis of storage cost, computational complexity and convergence of the proposed algorithms. We also provide a procedure to determine the optimal TT structure for minimizing storage complexity. In Section 2.5, we compare the proposed methods with state-of-the-art tensor based discriminant analysis and subspace learning methods for classification applications. In Section 2.6, we discuss graph regularized unsupervised learning on tensors, then, propose an implementation using the two-way structure using an ADMM based optimization, and discuss the convergence properties. In Section 2.7, we compare the proposed method with unsupervised tensor learning methods for clustering applications on two data sets.

2.2 Tensor Train Discriminant Analysis

When the data samples are tensors, traditional LDA first vectorizes them and then finds an optimal projection as shown in (1.23). This creates problems as the intrinsic structure of the data is destroyed. Even though MDA addresses this problem, it is inefficient in terms of storage complexity as it relies on TD [45, 32]. Thus, we propose to solve (1.23) by constraining $U = \mathbf{L}(\mathcal{U}_1 \times_3^1 \mathcal{U}_2 \times_3^1 \cdots \times_3^1 \mathcal{U}_N)$ to be a TT-subspace to reduce the computational and storage complexity and obtain a solution that will preserve the inherent data structure.

The goal of TTDA is to learn left orthogonal tensor factors $\mathcal{U}_n \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}, n \in \{1, \ldots, N\}$ using TT-model such that the discriminability of projections $\mathbf{x}_c^k, \forall c, k$ is maximized. First, $\mathcal{U}_n$s can be initialized using TT decomposition proposed in [140]. To optimize $\mathcal{U}_n$s for discriminability, we need to solve (1.23) for each $\mathcal{U}_n$, which can be rewritten using the definition of U as:

$$\mathcal{U}_n = \underset{\hat{\mathcal{U}}_n}{\operatorname{argmin}}\, tr\left[\mathbf{L}(\mathcal{U}_1 \times_3^1 \cdots \times_3^1 \hat{\mathcal{U}}_n \times_3^1 \cdots \times_3^1 \mathcal{U}_N)^\top S \mathbf{L}(\mathcal{U}_1 \times_3^1 \cdots \times_3^1 \hat{\mathcal{U}}_n \times_3^1 \cdots \times_3^1 \mathcal{U}_N)\right]. \tag{2.1}$$

Using the definitions presented in (1.1) and (1.2), we can express (2.1) in terms of tensor merging product:

$$\mathcal{U}_n = \underset{\hat{\mathcal{U}}_n}{\operatorname{argmin}}\, tr\left[(\mathcal{U}_1 \times_3^1 \cdots \times_3^1 \hat{\mathcal{U}}_n \times_3^1 \cdots \times_3^1 \mathcal{U}_N) \times_{1,\ldots,N}^{1,\ldots,N} \mathcal{S} \times_{N+1,\ldots,2N}^{1,\ldots,N} (\mathcal{U}_1 \times_3^1 \cdots \times_3^1 \hat{\mathcal{U}}_n \times_3^1 \cdots \times_3^1 \mathcal{U}_N)\right], \tag{2.2}$$

where $\mathcal{S} = \mathbf{T}_N^{-1}(S) \in \mathbb{R}^{I_1 \times \cdots \times I_N \times I_1 \times \cdots \times I_N}$. Let $\mathcal{U}_{\leq n-1} = \mathcal{U}_1 \times_3^1 \mathcal{U}_2 \times_3^1 \cdots \times_3^1 \mathcal{U}_{n-1}$ and $\mathcal{U}_{>n} = \mathcal{U}_{n+1} \times_3^1 \cdots \times_3^1 \mathcal{U}_N$. By rearranging the terms in (2.2), we can first compute all merging products and trace operations that do not involve $\mathcal{U}_n$ as:

$$\mathcal{A}_n = tr_4^8\left[\mathcal{U}_{\leq n-1} \times_{1,\ldots,n-1}^{1,\ldots,n-1} \left(\mathcal{U}_{>n} \times_{1,\ldots,N-n}^{n+1,\ldots,N} \left(\mathcal{U}_{\leq n-1} \times_{1,\ldots,n-1}^{N+1,\ldots,N+n-1} (\mathcal{U}_{>n} \times_{1,\ldots,N-n}^{N+n+2,\ldots,2N} \mathcal{S})\right)\right)\right], \tag{2.3}$$

where $\mathcal{A}_n \in \mathbb{R}^{R_{n-1} \times I_n \times R_n \times R_{n-1} \times I_n \times R_n}$ (refer to Figure 2.1 for a graphical representation of (2.3)). Then, we can rewrite the optimization in terms of $\mathcal{U}_n$:

$$\mathcal{U}_n = \underset{\hat{\mathcal{U}}_n}{\operatorname{argmin}} \left(\hat{\mathcal{U}}_n \times_{1,2,3}^{1,2,3} \left(\mathcal{A}_n \times_{4,5,6}^{1,2,3} \hat{\mathcal{U}}_n\right)\right). \tag{2.4}$$

Let $A_n = \mathbf{T}_3(\mathcal{A}_n) \in \mathbb{R}^{R_{n-1} I_n R_n \times R_{n-1} I_n R_n}$, then (2.4) can be rewritten as:

$$\mathcal{U}_n = \underset{\hat{\mathcal{U}}_n}{\operatorname{argmin}}\, \mathbf{V}(\hat{\mathcal{U}}_n)^\top A_n \mathbf{V}(\hat{\mathcal{U}}_n), \qquad \mathbf{L}(\hat{\mathcal{U}}_n)^\top \mathbf{L}(\hat{\mathcal{U}}_n) = \mathbb{I}_{R_n}. \tag{2.5}$$

This is a non-convex function due to unitary constraints and can be solved by the algorithm proposed in [189]. The algorithm employs a curvilinear line search method to solve minimizations with orthogonality constraints such as eigenvalue decompositions or matrix rank minimization. The procedure described above to find the subspaces is computationally expensive as the complexity of finding each $\mathcal{A}_n$ [183] is at least quadratic in the number of elements, *i.e.* $O(I^{2N})$.

When $n = N$, (2.5) does not apply as $\mathcal{U}_{>N}$ is not defined and the trace operation is defined on the third mode of $\mathcal{U}_N$. To update $\mathcal{U}_N$, the following can be used:

$$\mathcal{U}_N = \underset{\hat{\mathcal{U}}_N}{\operatorname{argmin}}\, tr\left(\hat{\mathcal{U}}_N \times_{1,2}^{1,2} \left(\mathcal{A}_N \times_{3,4}^{1,2} \hat{\mathcal{U}}_N\right)\right),$$

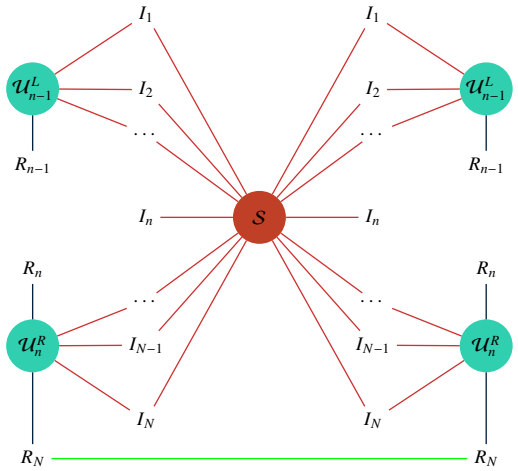


Figure 2.1: Tensor $\mathcal{A}_n$ is formed by first merging $\mathcal{U}^R_n$, $\mathcal{U}^L_{n-1}$ and $\mathcal{S}$ and then applying trace operation across 4^{th} and 8^{th} modes of the resulting tensor. The green line at the bottom of the diagram refers to the trace operator.

where $\mathcal{A}_N = \mathcal{U}_{\leq N-1} \times_{1,\ldots,N-1}^{1,\ldots,N-1} \left(\mathcal{U}_{\leq N-1} \times_{1,\ldots,N-1}^{N+1,\ldots,2N-1} \mathcal{S} \right)$. Once all of the $\mathcal{U}_n$s are obtained, they can be used to extract low-dimensional, discriminative features by the projection operation $U^\top \mathbf{V}(\mathcal{Y}_c^k)$. The pseudocode for TTDA is given in Algorithm 1.

TTDA algorithm described above is computationally expensive as it requires the computation of tensor $\mathcal{A}_n$ through tensor merging products. Also, the size of $\mathcal{A}_n$ grows significantly with increasing R_n, *e.g.* $O(I^{4N})$, which has exponentially growing memory cost. Moreover, since the ranks of TT are bounded by $R_n \leq \prod_{i=1}^n I_i$, the size of A_n may increase up to $\prod_{i=1}^n I_i^2 \times \prod_{i=1}^n I_i^2$. Thus, A_n might end up being larger than the original scatter matrix S, making TT approximation worse than LDA in terms of computational complexity.

2.3 Multi-Branch Tensor Train Discriminant Analysis

In this section, we introduce tensor decomposition structures to reduce computational complexity of TTDA. In particular, instead of using the whole scatter tensor $\mathcal{S}$ for the update of each $\mathcal{U}_n$, we aim to partition tensor factors into several sets where each set will approximate a lower dimensional TT subspace. For the update of each factor set, the original data will be projected to the remaining sets, and then the corresponding scatter matrix will be formed from these low dimensional projections. To further reduce the complexity, tensor merging products between sets will be removed. These collection of sets form a Kronecker structure, approximating U in (1.23).

Algorithm 2.1: Tensor Train Discriminant Analysis (TTDA)

Input: Input tensors $\mathcal{Y}_c^k \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ where $c \in \{1, \ldots, C\}$ and $k \in \{1, \ldots, K\}$, initial tensor factors $\mathcal{U}_n, n \in \{1, \ldots, N\}$, λ, $R_1, \ldots, R_N$, $MaxIter$

Output: $\mathcal{U}_n, n \in \{1, \ldots, N\}$, and $\mathbf{x}_c^k, \quad \forall c, k$

$\mathcal{S} \leftarrow \mathbf{T}_N^{-1}(S_W - \lambda S_B)$, see eqns.(1.24), (1.25).
while $iter < MaxIter$ **do**
  for $n = 1 : N-1$ **do**
    Compute $\mathcal{A}_n$ using (2.3).
    $\mathbf{V}(\mathcal{U}_n) \leftarrow \underset{\hat{\mathcal{U}}_n, \mathbf{L}(\hat{\mathcal{U}}_n)^\top \mathbf{L}(\hat{\mathcal{U}}_n) = \mathbb{I}_{R_n}}{\text{argmin}} \mathbf{V}(\hat{\mathcal{U}}_n)^\top \mathbf{T}_3(\mathcal{A}_n) \mathbf{V}(\hat{\mathcal{U}}_n)$.
  end for
  $\mathcal{A}_N \leftarrow \mathcal{U}_{N-1}^L \times_{1,\ldots,N-1}^{1,\ldots,N-1} \left(\mathcal{U}_{N-1}^L \times_{1,\ldots,N-1}^{N+1,\ldots,2N-1} \mathcal{S} \right)$
  $\mathbf{L}(\mathcal{U}_N) \leftarrow \underset{\hat{\mathcal{U}}_N, \mathbf{L}(\hat{\mathcal{U}}_n)^\top \mathbf{L}(\hat{\mathcal{U}}_N) = \mathbb{I}_{R_N}}{\text{argmin}} tr\Big(\mathbf{L}(\hat{\mathcal{U}}_N)^\top \mathbf{T}_2(\mathcal{A}_N) \mathbf{L}(\hat{\mathcal{U}}_N) \Big)$.
  $iter \leftarrow iter + 1$.
end while
$U = \mathbf{L}(\mathcal{U}_1 \times_3^1 \mathcal{U}_2 \times_3^1 \cdots \times_3^1 \mathcal{U}_N)$
$\mathbf{x}_c^k \leftarrow U^\top \mathbf{V}(\mathcal{Y}_c^k), \quad \forall c, k.$

2.3.1 Two-way Tensor Train Discriminant Analysis (2WTTDA)

As LDA tries to find a subspace U which maximizes discriminability for vector-type data, 2D-LDA tries to find two subspaces V_1, V_2 such that these subspaces maximize discriminability for matrix-type data [198]. If one considers the matricized version of $\mathcal{Y}_c^k$ along mode d, i.e. $\mathbf{T}_d(\mathcal{Y}_c^k) \in \mathbb{R}^{\prod_{i=1}^d I_i \times \prod_{i=d+1}^N I_i}$, where $1 < d < N$, the equivalent orthogonal projection can be written as:

$$\mathbf{T}_d(\mathcal{Y}_c^k) = V_1 X_c^k V_2^\top, \tag{2.6}$$

where $V_1 \in \mathbb{R}^{\prod_{i=1}^d I_i \times R_d}$, $V_2 \in \mathbb{R}^{\prod_{i=d+1}^N I_i \times \hat{R}_d}$, $X_c^k \in \mathbb{R}^{R_d \times \hat{R}_d}$.

In TTDA, since the projections $\mathbf{x}_c^k$ are considered to be vectors, the subspace $U = \mathbf{L}(\mathcal{U}_1 \times_3^1 \mathcal{U}_2 \times_3^1 \cdots \times_3^1 \mathcal{U}_N)$ is analogous to the solution of LDA with the constraint that the subspace admits a TT model. If we consider the projections and the input samples as matrices, now we can impose a TT structure to the left and right subspaces analogous to solution of 2D-LDA. In other words, one can find two sets of TT representations corresponding to V_1 and V_2 in (2.6). Using this structural approximation, (2.6) can be rewritten as:

$$\mathbf{T}_d(\mathcal{Y}_c^k) = \mathbf{L}(\mathcal{U}_1 \times_3^1 \cdots \times_3^1 \mathcal{U}_d) X_c^k \mathbf{R}(\mathcal{U}_{d+1} \times_3^1 \cdots \times_3^1 \mathcal{U}_N), \tag{2.7}$$

which is equivalent to the following representation:

$$\mathcal{Y}_c^k = \mathcal{U}_1 \times_3^1 \cdots \times_3^1 \mathcal{U}_d \times_3^1 X_c^k \times_2^1 \mathcal{U}_{d+1} \times_3^1 \cdots \times_3^1 \mathcal{U}_N.$$

This formulation is graphically represented in Figure 2.2a where the decomposition has two branches, thus we refer to it as Two-way Tensor Train Decomposition (2WTT).

In prior work, decompositions with this structure was also called Tensor Train [204] or Matrix Product States [18] but the formulation is different and some properties of Tensor Train such as ranks, computational and storage complexities change when this structure is employed. Thus, this is a different decomposition than the conventional Tensor Train and the differences will be more higlighted when the number of branches increase. The differences between the two-way structure and TT are:

- In the two-way structure, factors after mode-d are not linearly dependent to the ones before mode-d. This can be viewed easily in 2D-LDA problem, which is a two-way structure with only two modes, or from the representation in Figure 2.2a.

- In 2WTT, modes before mode-d are left-orthogonal, similar to TT, but the modes after mode-d are right orthogonal. This fact with the previous one allow a reduced computational cost in the ALS optimization scheme by a simple linear independence assumption.

- In TT, ranks grow from the first mode to the last mode as the upper bound of each rank, $R_n \leq I_n R_{n-1} \leq \prod_{n'=1}^{n} I_{n'}$, keeps increasing. On the other hand, for 2WTT ranks with $n > d$ are bounded by $R_n \leq I_n R_{n+1} \leq \prod_{n'=n+1}^{N} I_{n'}$. As a quantitative measure, the upper bound of the maximum rank is $\prod_{n=1}^{N} I_n$ for TT, and $\sqrt{\prod_{n=1}^{N} I_n}$ for 2WTT.

To maximize discriminability using 2WTT, an optimization scheme that alternates between the two sets of TT-subspaces can be utilized. When forming the scatter matrices for a set, projections of the data to the other set can be used instead of the full data which is similar to (1.26). This will reduce computational complexity as the cost of computing scatter matrices and the number of matrix multiplications to find $\mathcal{A}_n$ in (2.3) will decrease. We propose the procedure given in Algorithm 2.2 to implement this approach and refer to it as Two-way Tensor Train Discriminant Analysis (2WTTDA) as illustrated in Figure 2.2c. To determine the value of d in (2.7), we use a center of mass approach and find the d that minimizes $|\prod_{i=1}^{d} I_i - \prod_{j=d+1}^{N} I_j|$. In this manner, the problem can be separated into two parts which have similar computational complexities.

Algorithm 2.2: Two-Way Tensor Train Discriminant Analysis (2WTTDA)

Input: Input tensors $\mathcal{Y}_c^k \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ where $c \in \{1, \ldots, C\}$ and $k \in \{1, \ldots, K\}$, initial tensor factors $\mathcal{U}_n, n \in \{1, \ldots, N\}$, d, λ, $R_1, \ldots, R_N$, $MaxIter$, $LoopIter$

Output: $\mathcal{U}_n, n \in \{1, \ldots, N\}$, and $X_c^k, \quad \forall c, k$

while $iter < LoopIter$ **do**
  $\mathcal{Y}_L \leftarrow \mathcal{Y} \times_{d+1,\ldots,N}^{2,\ldots,N-d+1} (\mathcal{U}_{d+1} \times_3^1 \cdots \times_3^1 \mathcal{U}_N)$.
  $[\mathcal{U}_i] \leftarrow TTDA(\mathcal{Y}_L, \lambda, R_i, MaxIter) \forall i \in \{1, \ldots, d\}$.
  $\mathcal{Y}_R \leftarrow \mathcal{Y} \times_{1,\ldots,d}^{2,\ldots,d+1} (\mathcal{U}_1 \times_3^1 \cdots \times_3^1 \mathcal{U}_d)$.
  $[\mathcal{U}_i] \leftarrow TTDA(\mathcal{Y}_R, \lambda, R_i, MaxIter) \forall i \in \{d+1, \ldots, N\}$.
  $iter = iter + 1$.
end while
$\mathcal{X}_c^k \leftarrow \mathbf{L}(\mathcal{U}_1 \times_3^1 \cdots \times_3^1 \mathcal{U}_d)^\top \mathbf{T}_d(\mathcal{Y}_c^k)\mathbf{R}(\mathcal{U}_{d+1} \times_3^1 \cdots \times_3^1 \mathcal{U}_N)^\top$.

2.3.2 Three-way Tensor Train Discriminant Analysis (3WTTDA)

Elaborating on the idea of 2WTTDA, one can increase the number of modes of the projected samples which will increase the number of tensor factor sets, or the number of subspaces to be approximated using TT structure. For example, one may choose the number of modes of the projections as three, i.e. $\mathcal{X}_c^k \in \mathbb{R}^{R_{d_1} \times R_{d_2} \times \hat{R}_{d_2}}$, where $1 < d_1 < d_2 < N$. This model, named as Three-way Tensor Train Decomposition (3WTT), is given in (2.8) and represented graphically in Figure 2.2b.

$$\mathcal{Y}_c^k = \Bigg(\Big(\mathcal{X}_c^k \times_3^{N-d_2+2} (\mathcal{U}_{d_2+1} \times_3^1 \cdots \times_3^1 \mathcal{U}_N) \Big) \times_2^{d_2-d_1+2} (\mathcal{U}_{d_1+1} \times_3^1 \cdots \times_3^1 \mathcal{U}_{d_2}) \Bigg) \times_1^{d_1+2} (\mathcal{U}_1 \times_3^1 \cdots \times_3^1 \mathcal{U}_{d_1}). \tag{2.8}$$

To maximize discriminability using 3WTT, one can utilize an iterative approach as in Algorithm 2.2, where inputs are projected on all tensor factor sets except the set to be optimized, then TTDA is applied to the projections. The flowchart for the corresponding algorithm is illustrated in Figure 2.2d. This procedure can be repeated until a convergence criterion is met or a number of iterations is reached. The values of d_1 and d_2 are calculated such that the product of dimensions corresponding to each set is as close to $(\prod_{i=1}^N I_i)^{1/3}$ as possible. It is important to note that 3WTT will only be meaningful for tensors of order three or higher. For three-mode tensors, 3WTT is equivalent to Tucker Decomposition. When there are more than four modes, the number of branches can be increased accordingly which makes 4W, 5W and NWTT possible.

2.4 Analysis of Storage, Training Complexity and Convergence

In this section, we derive the storage and computational complexities of the aforementioned algorithms as well as providing a convergence analysis for TTDA.

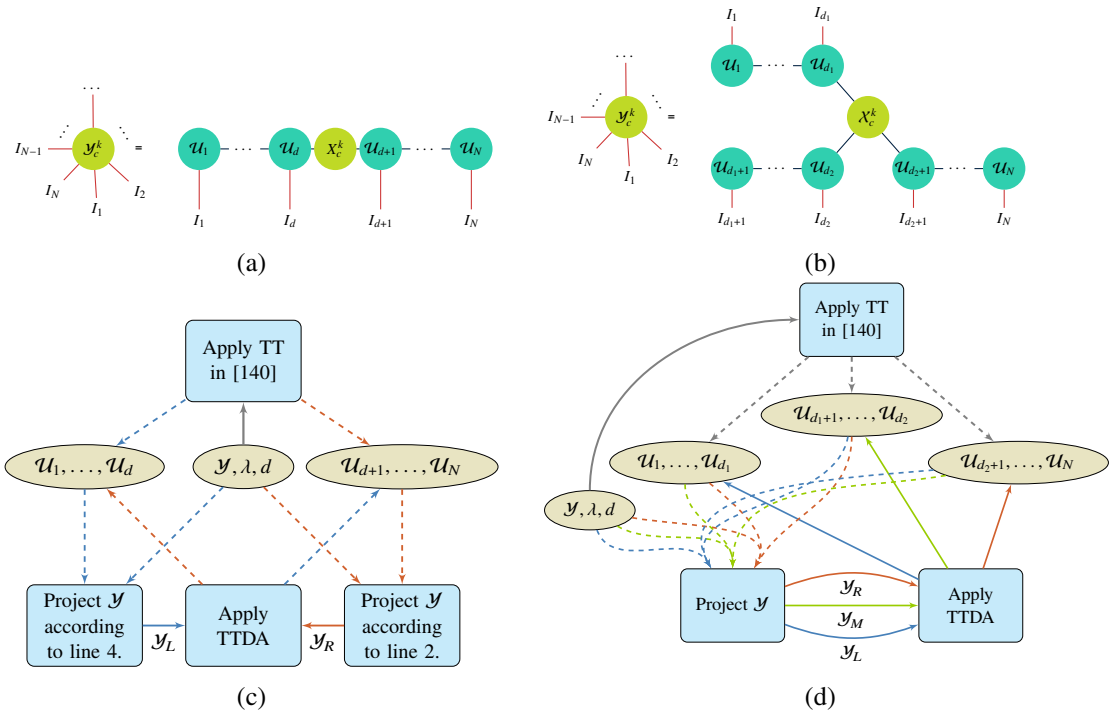


Figure 2.2: Illustration of the proposed methods (*Compare (a) and (b) with Figures 1.2 and 1.3*): (a) The proposed tensor network structure for 2WTT; (b) The proposed tensor network structure for 3WTT; (c) The flow diagram for 2WTTDA (Algorithm 2.2); (d) The flow diagram for 3WTTDA

Table 2.1: Storage Complexities of Different Tensor Decomposition Structures

Methods	Subspaces ($\mathcal{U}_n$s) ($O(.)$)	Projections $\mathcal{X}_c^k$ ($O(.)$)
TT	$(N-1)r^2I + rI$	rCK
2WTT	$(N-2)r^2I + 2rI$	r^2CK
3WTT	$(N-3)r^2I + 3rI$	r^3CK
TD	NrI	r^NCK

Storage Complexity Let $I_n = I, n \in \{1, 2, \ldots, N\}$ and $R_l = r, l \in \{2, \ldots, N-1\}$. Assuming N is a multiple of both 2 and 3, total storage complexities are:

- $O((N-1)r^2I + rI + rCK)$ for TT Decomposition, where $R_1 = 1, R_N = r$;
- $O((N-2)r^2I + 2rI + r^2CK)$ for Two-Way TT Decomposition, where $R_1 = R_N = 1$;
- $O((N-3)r^2I + 3rI + r^3CK)$ for Three-Way TT Decomposition, where $R_1 = R_{d_1} = R_N = 1$;
- $O(NrI + r^NCK)$ for Tucker Decomposition, where $R_1 = R_N = r$.

These results show that when the number of modes for the projected samples is increased, the storage cost increases exponentially for $\mathcal{X}_c^k$ while the cost of storing $\mathcal{U}_n$s decreases quadratically. Using the above, one

Table 2.2: Computational complexities of various algorithms. The number of iterations to find the subspaces are denoted as t_c for CMDA and t_t for TT-based methods. $C_s = 2CK$. ($r << I$, $t_t r(r + N/f - 1) << C_s$, and $I^{N/f} >> r^6$)

Methods	Order of Complexity ($O(.)$)
LDA	$C_s I^{2N} + I^{3N}$
DGTDA	$3I^3 + NC_s I^{N+1}$
CMDA	$2t_c I^3 + t_c N^2 C_s I^N$
TTDA	$C_s I^{2N}$
2WTTDA	$(r/2 + 2)C_s I^N$
3WTTDA	$(rI^{N/3}/2 + 3)C_s I^{2N/3}$

can easily find the optimal number of modes for the projected samples that minimizes storage complexity. Let the number of modes of $\mathcal{X}_c^k$ be denoted by f. The storage complexity of the decomposition is then $O((N - f)r^2 I + frI + r^f CK)$. The optimal storage complexity is achieved by taking the derivative of the complexity in terms of f and equating it to zero. In this case, the optimal f is given by

$$\hat{f} = round\left(\log_r\left(\frac{r^2 I - rI}{CK\ln(r)}\right)\right),$$

where $round(.)$ is an operator that rounds to the closest positive integer.

2.4.1 Computational Complexity

For all of the decompositions mentioned except for DGTDA and LDA, the $\mathcal{U}_n$s and $\mathcal{X}_c^k$ depend on each other which makes these decompositions iterative. The number of iterations will be denoted as t_c and t_t for CMDA and TT-based methods, respectively. For the sake of simplicity, we also define $C_s = 2CK$. The total cost of finding $\mathcal{U}_n$s and $\mathcal{X}_c^k$ $\forall n, c, k$, where $r << I$ is in the order of:

- $O\Big(I^N\big[(C_s + t_t r(r + N - 1))I^N + t_t r^4(I + r^2 I^{-1})\big]\Big)$ for TTDA;
- $O\Big(rI^N\frac{C_s}{2} + 2I^{N/2}\big[(C_s + t_t r(r + N/2 - 1))I^{N/2} + t_t r^4 I + t_t r^6 I^{-1}\big]\Big)$ for 2WTTDA;
- $O\Big(rI^N\frac{C_s}{2} + 3I^{N/3}\big[(C_s + t_t r(r + N/3 - 1))I^{N/3} + t_t r^4 I + t_t r^6 I^{-1}\big]\Big)$ for 3WTTDA.

If convergence criterion is met with a small number of iterations, i.e. $t_t r(r+N/f-1) << C_s$, and $I^{N/f} >> r^6$ for all f, the reduced complexities are as given in Table 2.2.

We can see from Table 2.2 that with increasing number of branches, TT-based methods become more efficient if the algorithm converges in a few number of iterations. This is especially the case if the ranks of tensor factors are low as this reduces the dimensionalities of the search space and the search algorithm finds

a solution to (2.5) faster. When this assumption holds true, the complexity is dominated by the formation of scatter matrices. Note that the ranks are assumed to be much lower than dimensionalities and number of modes is assumed to be sufficiently high. When these assumptions do not hold, the complexity of computing $\mathcal{A}_n$ might be dominated by terms with higher powers of r. This indicates that TT-based methods are more effective when the tensors have higher number of modes and when the TT-ranks of the tensor factors are low. DGTDA has an advantage over all other methods as it is not iterative and the solution for each mode is not dependent on other modes. On the other hand, the solution of DGTDA is not optimal and there are no convergence guarantees except when the ranks and initial dimensions are equal to each other, i.e. when there is no compression.

2.4.2 Convergence

To analyze the convergence of TTDA, we must first establish a lower bound for the objective function of LDA, as (2.1) is lower bounded by the objective value of (1.23).

Lemma 1. *Given that $\lambda \in \mathbf{R}_+$, i.e. a nonnegative real number, the lower bound of $tr(U^\top S_W U) - \lambda tr(U^\top S_B U)$ is achieved when $U \in \mathbb{R}^{\prod_{n=1}^{N} I_n \times r}$ satisfies the following two conditions, simultaneously:*

1. *The columns of U are in the null space of S_W: $\mathbf{u}_j \in null(S_W), \forall j \in \{1, \ldots, r\}$.*
2. *$\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r\}$ are the top-r eigenvectors of S_B.*

In this case, the minimum of $tr(U^\top S_W U) - \lambda tr(U^\top S_B U) = -\lambda \sum_{i=1}^{r} \sigma_i$, where σ_is are the eigenvalues of S_B.

Proof. Since S_W is positive semi-definite,

$$0 \leq \min_U tr(U^\top S_W U),$$

which implies that when the columns of U are in the null space of S_W, i.e. $\mathbf{u}_j \in null(S_W), \forall j \in \{1, \ldots, r\}$, the minimum value will be achieved for the first part of the objective function.

To minimize the trace difference, we need to maximize $tr(U^\top S_B U)$ which is bounded from above as:

$$\max_U tr(U^\top S_B U) \leq \sum_{i=1}^{r} \sigma_i.$$

$tr(U^\top S_B U)$ is maximized when the columns of U are the top-r eigenvectors of S_B. Therefore, the trace difference achieves the lower-bound when $\mathbf{u}_j \in null(S_W), \forall j \in \{1, \ldots, r\}$ and $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r\}$ are the top-r eigenvectors of S_B and this lower-bound is equal to $-\lambda \sum_{i=1}^{r} \sigma_i$. □

As shown above, the objective function of LDA is lower bounded. Thus, the solution to (2.1) is also lower-bounded.

Let $f(\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_N) = tr\,(U^\top S U)$, where $U = \mathbf{L}(\mathcal{U}_1 \times_3^1 \cdots \times_3^1 \mathcal{U}_N)$ and S is defined as in (1.23). If the function f is non-increasing with each update of $\mathcal{U}_n$s, i.e.

$$f(\mathcal{U}_1^t, \mathcal{U}_2^t, \ldots, \mathcal{U}_n^{t-1}, \ldots, \mathcal{U}_N^{t-1}) \geq$$
$$f(\mathcal{U}_1^t, \mathcal{U}_2^t, \ldots, \mathcal{U}_n^t, \ldots, \mathcal{U}_N^{t-1}), \qquad \forall t, n \in \{1, 2, \ldots, N\},$$

then we can claim that Algorithm 2.1 converges to a fixed point as $t \to \infty$ since $f(.)$ is lower-bounded. In [189], an approach to regulate the step sizes in the search algorithm was introduced to guarantee global convergence. In this chapter, this approach is used to update $\mathcal{U}_n$s. Thus, (2.5) can be optimized globally, and the objective value is non-increasing. As Multi-Branch extensions utilize TTDA on the update of each branch, proving the convergence of TTDA is sufficient to prove the convergence of 2WTTDA or 3WTTDA.

2.5 Experiments

The proposed TT based discriminant analysis methods are evaluated in terms of classification accuracy, storage complexity, training complexity and sample size. We compared our methods[1] with both linear supervised tensor learning methods including LDA, DGTDA and CMDA[112][2] as well as other Tensor Train based learning methods such as MPS [18], TTNPE [183][3] and STTM [38][4]. The experiments were conducted on four different data sets: COIL-100, Weizmann Face, Cambridge and UCF-101. For all data sets and all methods, we evaluate the classification accuracy and training complexity with respect to storage complexity.

In this chapter, classification accuracy is evaluated using a 1-NN classifier and quantified as N_{true}/N_{test}, where N_{true} is the number of test samples which were assigned the correct label and N_{test} is the total number of test samples. Normalized storage complexity is quantified as the ratio of the total number of elements in

[1] Our code is in https://github.com/mrsfgl/mbttda
[2] https://github.com/laurafroelich/tensor_classification
[3] https://github.com/wangwenqi1990/TTNPE.
[4] https://github.com/git2cchen/KSTTM

the learnt tensor factors ($\mathcal{U}_n, \forall n$) and projections ($\mathcal{X}_c^k, \forall c, k$) of training data, O_s, to the size of the original training data ($\mathcal{Y}_c^k, \forall c, k$):

$$\frac{O_s}{CK \prod_{n=1}^{N} I_n}.$$

Training complexity is the total runtime in seconds for learning the subspaces. All experiments were repeated 10 times with random selection of the training and test sets and average classification accuracies are reported.

The regularization parameter, λ, for each experiment was selected using a validation set composed of all of the samples in the training set and a small subset of each class from the test set (10 samples for COIL-100, 5 samples for Weizmann, 1 sample for Cambridge, and 10 samples for UCF-101). Utilizing a leave-s-out approach, where s is the aforementioned subset size, 5 random experiments were conducted. The optimal λ was selected as the value that gave the best average classification accuracy among a range of values from 0.1 to 1000 increasing in a logarithmic scale. CMDA, TTNPE and MPS do not utilize the λ parameter while DGTDA utilizes eigendecomposition to find λ [112]. STTM has an outlier fraction parameter which was set to 0.02 according to the original paper [38].

2.5.1 Data Sets

COIL-100

The dataset consists of 7,200 RGB images of 100 objects of size 128×128. Each object has 72 images, where each image corresponds to a different pose angle ranging from 0 to 360 degrees with increments of 5 degrees [133]. For our experiments, we downsampled the grayscale images of all objects to 64×64. Each sample image was reshaped to create a tensor of size $8 \times 8 \times 8 \times 8$. Reshaping the inputs into higher order tensors is common practice and was studied in prior work [90, 140, 209, 45, 17]. 20 samples from each class were selected randomly as training data, i.e. $\mathcal{Y} \in \mathbb{R}^{8\times8\times8\times8\times20\times100}$, and the remaining 52 samples were used for testing.

Weizmann Face Database

The dataset includes RGB face images of size 512×352 belonging to 28 subjects taken from 5 viewpoints, under 3 illumination conditions, with 3 expressions [188]. For our experiments, each image was grayscaled, and downsampled to 64×44. The images were then reshaped into 5-mode tensors of size $4 \times 4 \times 4 \times 4 \times 11$ as in [183]. For each experiment, 20 samples were randomly selected to be the training data, i.e. $\mathcal{Y} \in \mathbb{R}^{4\times4\times4\times4\times11\times20\times28}$, and the remaining 25 samples were used in testing.

Cambridge Hand-Gesture Database

The dataset consists of 900 image sequences of 9 gesture classes, which are combinations of 3 hand shapes and 3 motions. For each class, there are 100 image sequences generated by the combinations of 5 illuminations, 10 motions and 2 subjects [91]. Sequences consist of images of size 240×320 and sequence length varies. In our experiments, we used grayscaled versions of the sequences and we downsampled all sequences to length 30. We also included 2 subjects and 5 illuminations as the fourth mode. Thus, we have 10 samples for each of the 9 classes from which we randomly select 4 samples as the training set, i.e. $\mathcal{Y} \in \mathbb{R}^{30\times40\times30\times10\times4\times9}$, and the remaining 6 as test set.

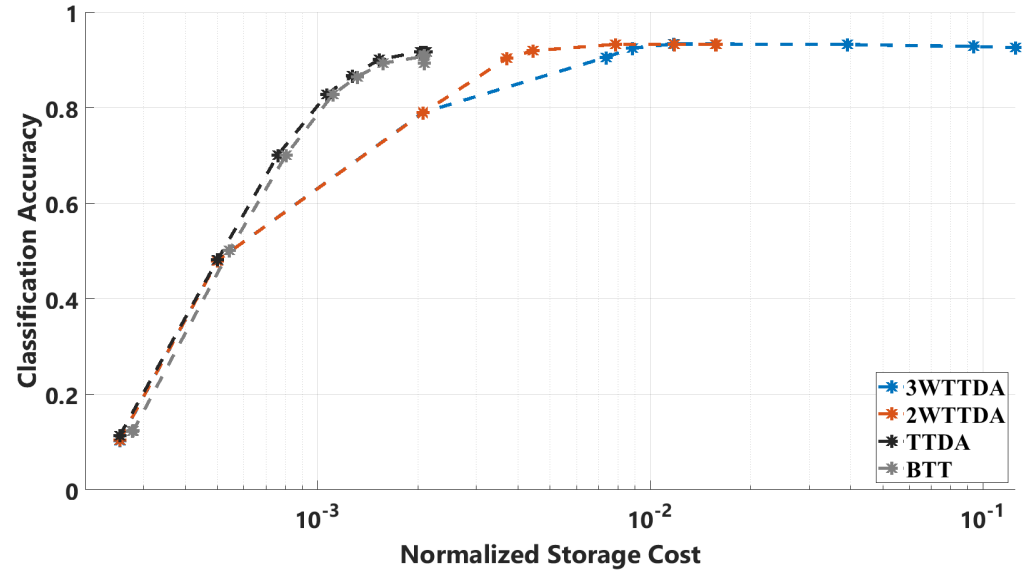
UCF-101 Human Action Dataset

UCF-101 is an action recognition dataset [162]. There are 13320 videos of 101 actions, where each action category might have different number of samples. Each sample is an RGB image sequence with frame size $240 \times 320 \times 3$. The number of frames differs for each sample. In our experiments, we used grayscaled, downsampled frames of size 30×40. From each class, we extracted 100 samples to balance the class sizes where each sample consists of 50 frames obtained by uniformly sampling each video sequence. 60 randomly selected samples from each class were used for training, i.e. $\mathcal{Y} \in \mathbb{R}^{30\times40\times50\times60\times101}$, and the remaining 40 samples were used for testing.

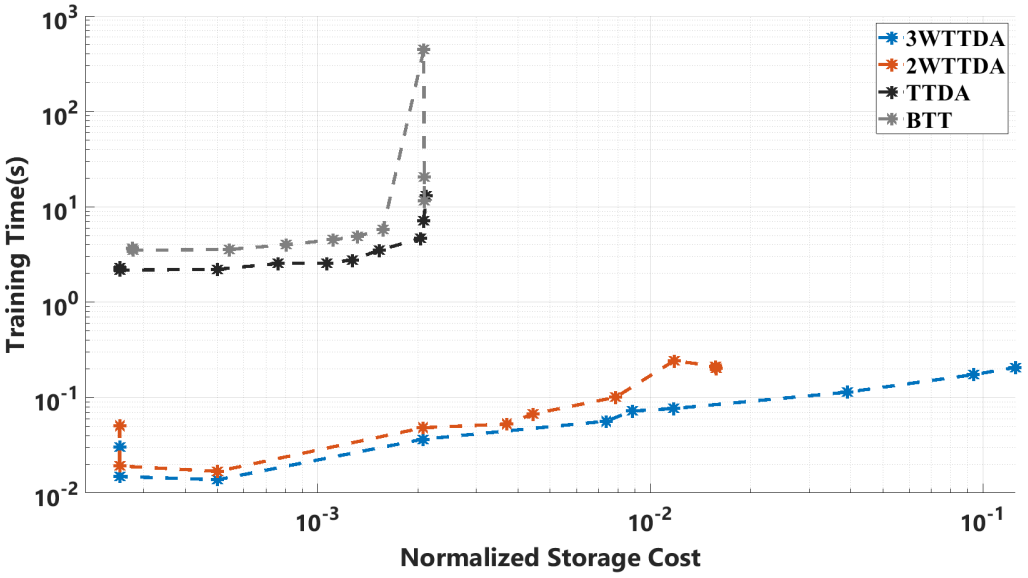
2.5.2 Classification Accuracy

We first evaluate the classification accuracy of the different methods with respect to normalized storage complexity. The varying levels of storage cost are obtained by varying the ranks, R_is, in the implementation of the tensor decomposition methods. Varying the truncation parameter $\tau \in (0, 1]$, the singular values smaller than τ times the largest singular value are eliminated. The remaining singular values are used to determine the ranks R_is for both TT-based and TD-based methods. For TT-based methods, the ranks are selected using TT-decomposition proposed in [140], while for TD-based methods truncated HOSVD was used. We also limit the upper bounds of the ranks to $R_n \leq I_n$ to reduce the memory cost associated with storing A_n. With this limitation, the maximum dimension of projected samples $\mathbf{x}_c^k$ becomes I_N which is a fraction of the original sample size $\prod_{i=1}^{N} I_i$.

In Fig. 2.3, we present comparisons of BTT based eigenpair computation algorithm [56] with the proposed method. The experiments were conducted on COIL-100 data set. BTT performs similarly to



(a)



(b)

Figure 2.3: Comparisons with a BTT based Ritz pair computation algorithm. a) Classification accuracy and b) Training time with respect to normalized storage cost.

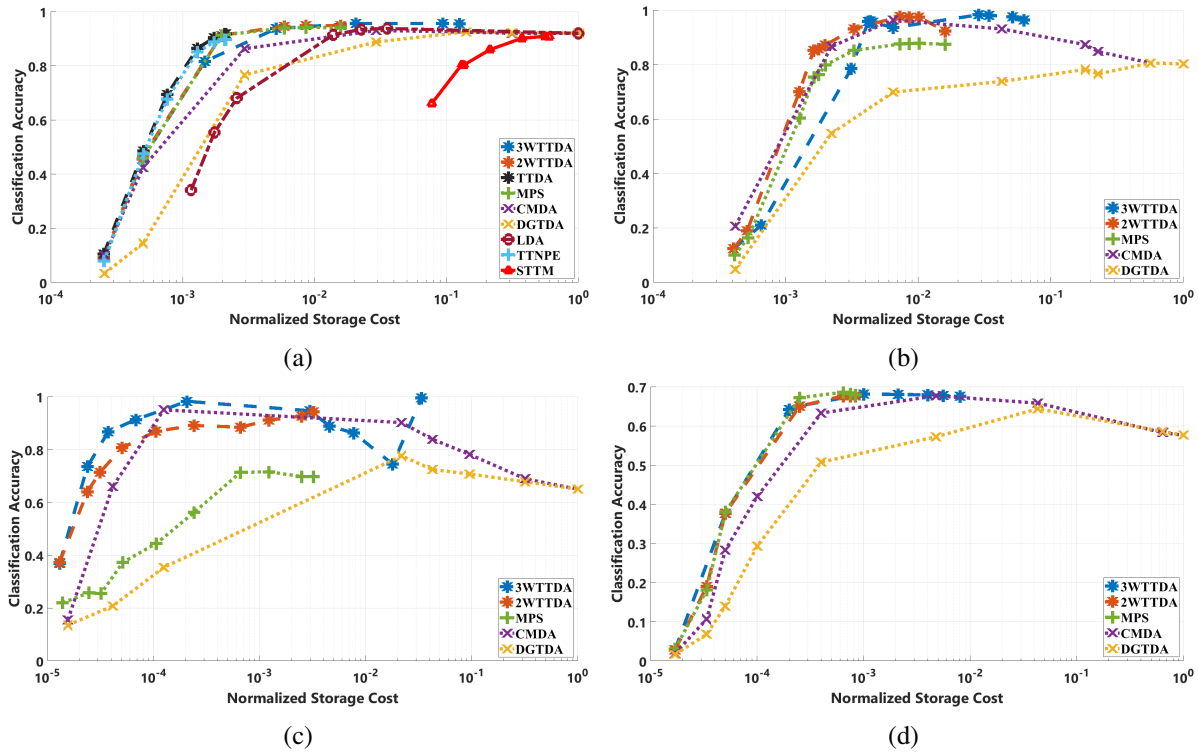


Figure 2.4: Classification accuracy vs. Normalized storage cost of the different methods for: a) COIL-100, b) Weizmann Face, c) Cambridge Hand Gesture and d) UCF-101. All TD based methods are denoted using 'x', TT based methods are denoted using '+' and proposed methods are denoted using '*'. STTM and LDA are denoted using '△' and 'o', respectively.

TTDA with slightly lower accuracy and higher training time for increasing storage complexity. As BTT has adaptive ranks, the factor sizes increase more compared to TTDA which results in a much higher computational cost. Although the scatter tensor is approximated using a TT structure, BTT does not provide better computational complexity than TTDA as the approximation itself has $O(I^{2N})$ complexity. Multi-branch extensions of TTDA outperform BTT both in terms of classification accuracy and computational complexity.

Figure 2.4a illustrates the classification accuracy of the different methods with respect to normalized storage complexity for COIL-100 data set. For this particular dataset, we implemented all of the methods mentioned above. It can be seen that the proposed discriminant analysis framework in its original form, TTDA, gives the highest accuracy results followed by TTNPE. However, these two methods only operate at very low storage complexities since the TT-ranks of tensor factors are constrained to be smaller than the corresponding mode's input dimensions. We also implemented STTM, which does not provide compression rates similar to other TT-based methods. This is due to the fact that STTM needs to learn $\frac{C(C-1)}{2}$ classifiers

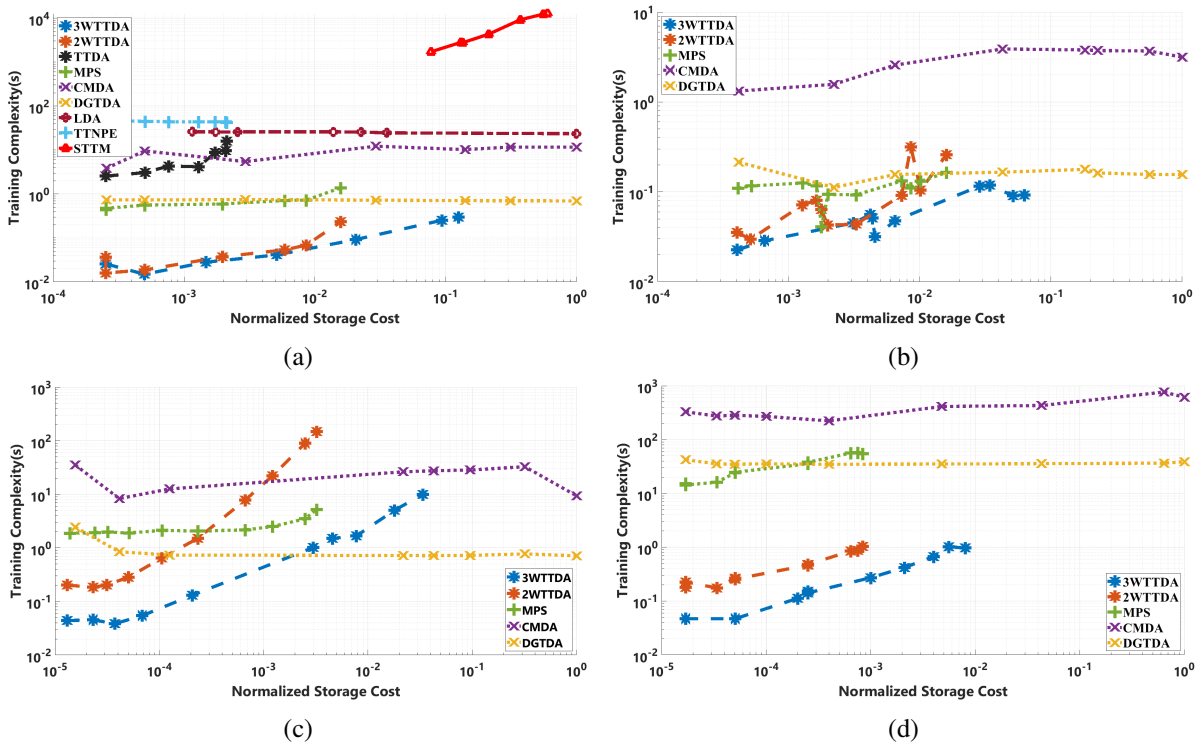


Figure 2.5: Training complexity vs. Normalized storage cost of the different methods for: a) COIL-100, b) Weizmann Face, c) Cambridge Hand Gesture, and d) UCF-101.

with TT structure. Moreover, these methods have very high computational complexity as will be shown in Section 2.5.3. For this reason, they will not be included in the comparisons for the other datasets. For a wide range of storage complexities, MPS and 2WTTDA perform the best and have similar accuracy. It can also be seen that the storage costs of MPS and 2WTTDA stop increasing after some point due to rank constraints. This is in line with the theoretical storage complexity analysis presented in Section 2.4. Tucker based methods, such as CMDA and DGTDA, along with the original vector based LDA have lower classification accuracy.

Figure 2.4b similarly illustrates the classification accuracy of the different methods on the Weizmann Face Database. For all storage complexities, the proposed 2WTTDA and 3WTTDA perform better than the other methods, including TT based methods such as MPS.

Figure 2.4c illustrates the classification accuracy for the Cambridge hand gesture database. In this case, 3WTTDA performs the best for most storage costs. As the number of samples for training, validation and testing is very low for Cambridge dataset, the classification accuracy fluctuates with respect to the dimensionality of the features at normalized storage cost of 0.02. Similar fluctuations can also be seen in the

results of [183].

Finally, we tested the proposed methods on a more realistic, large sized dataset, UCF-101. For this dataset, TT-based methods perform better than the Tucker based methods. In particular, 2WTTDA performs very close to MPS at low storage costs, whereas 3WTTDA performs well for a range of normalized storage costs and provides the highest accuracy overall.

Even though our methods outperform MPS for most datasets, the classification accuracies get close for UCF-101 and COIL-100. This is due to the high number of classes in these datasets. As the number of classes increases, the number of scatter matrices that needs to be estimated also increases which results in a larger bias given limited number of training samples. This improved performance of MPS for datasets with large number of classes is also observed when MPS is compared to CMDA. Therefore, the reason that MPS and the proposed methods perform similarly is a limitation of discriminant analysis rather than the proposed tensor network structure.

2.5.3 Training Complexity

In order to compute the training complexity, for TT-based methods, each set of tensor factors is optimized until the change in the normalized difference between consecutive tensor factors is less than 0.1 or 200 iterations is completed. After updating the factors in a branch, no further optimizations are done on that branch in each iteration. CMDA iteratively optimizes the subspaces for a given number of iterations (which is set to 20 to increase the speed in our experiments) or until the change in the normalized difference between consecutive subspaces is less than 0.1.

Figs. 2.5a, 2.5b, 2.5c, 2.5d illustrate the training complexity of the different methods with respect to normalized storage cost for the four different datasets. In particular, Figure 2.5a illustrates the training complexity of all the methods including TTNPE, TTDA and STTM for COIL-100. It can be seen that STTM has the highest computational complexity among all of the tested methods. This is due to the fact that for a 100-class classification problem, STTM implements $(100)(99)/2$ one vs. one binary classifiers, increasing the computational complexity. Similarly, TTNPE has high computational complexity as it tries to learn the manifold projections which involves eigendecomposition of the embedded graph. Among the remaining methods, LDA has the highest computational complexity as it is based on learning from vectorized samples which increases the dimensionality of the covariance matrices. For the tensor based methods, the proposed

2WTTDA and 3WTTDA have the lowest computational complexity followed by MPS and DGTDA. In particular, for large datasets like UCF-101 the difference in computational complexity between our methods and existing TT-based methods such as MPS is more than a factor of 10^2.

Table 2.3: Classification accuracy (top) and training time (bottom) with standard deviation for various methods and datasets.

Accuracy (%)	3WTTDA	2WTTDA	MPS[18]	CMDA[112]	DGTDA[112]
COIL-100	**95.6 ± 0.4**	94.8 ± 0.5	94.2 ± 0.2	86.3 ± 0.7	76.6 ± 0.9
Weizmann	93.6 ± 2	**97.6 ± 1.2**	87.5 ± 2.3	96.4 ± 1.03	69.9 ± 1.8
Cambridge	**98.2 ± 1.7**	89.1 ± 16.7	56.2 ± 9.8	95 ± 2.8	35.4 ± 8.7
UCF-101	**68.6 ± 0.8**	67.7 ± 0.9	67.9 ± 0.6	67.7 ± 0.8	57.3 ± 2.7
(s)					
COIL-100	**0.09 ± 0.005**	0.24 ± 0.06	1.4 ± 0.13	12.2 ± 6.6	0.7 ± 0.06
Weizmann	**0.05 ± 0.02**	0.09 ± 0.02	0.13 ± 0.01	2.6 ± 0.3	0.16 ± 0.02
Cambridge	**0.11 ± 0.01**	1.7 ± 1.5	2.07 ± 0.25	12.6 ± 0.3	0.7 ± 0.04
UCF-101	**0.67 ± 0.02**	0.853 ± 0.13	56.4 ± 1.9	413.5 ± 24.1	35.3 ± 2.9

2.5.4 Convergence

In this section, we present an empirical study of convergence for TTDA in Figure 2.6 where we report the objective value of TTDA, i.e. the expression inside argmin operator in (2.5), with random initialization of projection tensors. This figure illustrates the convergence of the TTDA algorithm, which is at the core of both 2WTTDA and 3WTTDA, on COIL-100 dataset. It can be seen that even for random initializations of the tensor factors, the algorithm converges in a small number of steps. The convergence rates for 2WTTDA and 3WTTDA are faster than that of TTDA as they update smaller sized projection tensors as shown in Section 2.4.

2.5.5 Effect of Sample Size on Accuracy

We also evaluated the effect of training sample size on classification accuracy for Weizmann Dataset. In Figure 2.7, we illustrate the classification accuracy with respect to training sample size for different methods. It can be seen that 3WTTDA provides a high classification accuracy even for small training datasets, i.e., for 15 training samples it provides an accuracy of 96%. This is followed by CMDA and 2WTTDA. It should also be noted that DGTDA is the most sensitive to sample size as it cannot even achieve the local optima and more data allows it to learn better classifiers.

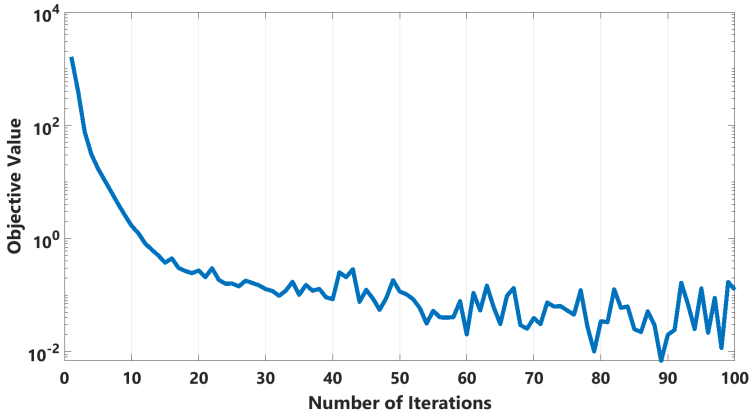


Figure 2.6: Convergence curve for TTDA on COIL-100. Objective value vs. the number of iterations is shown.

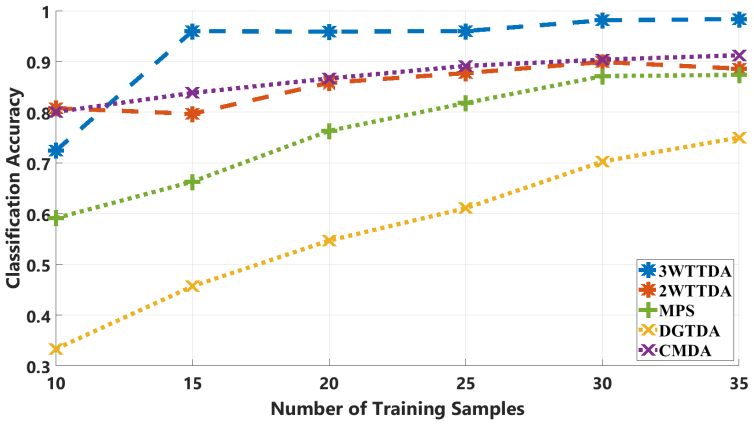


Figure 2.7: Comparison of classification accuracy vs. training sample size for Weizmann Face Dataset for different methods.

2.5.6 Summary of Experimental Results

In Table 2.3, we summarize the performance of the different algorithms for the four different datasets considered in this chapter. In the left half of this table, we report the classification accuracy (mean ± std) of the different methods for a fixed normalized storage cost of about 2.10^{-2} for COIL-100, 6.10^{-3} for Weizmann Face, 2.10^{-4} for Cambridge Hand Gesture and 10^{-3} for UCF-101 datasets. At the given compression rates, for all datasets the proposed 3WTTDA and 2WTTDA perform better than the other tensor based methods. In some cases, the improvement in classification accuracy is significant, e.g. for Weizmann and Cambridge data sets. These results show that the proposed method achieves the best trade-off, i.e. between normalized storage complexity and classification accuracy.

Similarly, the right half of Table 2.3 summarizes the average training complexity for the different

methods for the same normalized storage cost. From this Table, it can be seen that 3WTTDA is the most computationally efficient method for all datasets. This is followed by 2WTTDA. The difference in computational time becomes more significant as the size of the dataset increases, e.g. for UCF-101. Therefore, even if the other methods perform well for some of the datasets, the proposed methods provide higher accuracy at a computational complexity reduced by a factor of 10^2.

2.6 Graph Regularized Tensor Train Decomposition

The geometric relationship between data samples has been shown to be important for learning low-dimensional structures from high-dimensional data [171, 183]. Recently, motivated by manifold learning, dimensionality reduction of tensor objects has been formulated to incorporate the geometric structure [115]. The goal is to learn a low dimensional representation for tensor objects that incorporates the geometric structure while maintaining a low reconstruction error in the tensor decomposition. This idea of manifold learning for tensors has been mostly implemented for the Tucker method, including Graph Laplacian Tucker Decomposition (GLTD) [82] and nonnegative Tucker factorization (NTF). However, this line of work suffers from the limitations of Tucker decomposition mentioned early in the chapter [45].

Earlier in this chapter, we proposed a TT model, called multi-branch Tensor Train, such that the features can be matrices and higher-order tensors. In this way, the computational efficiency could be improved. Utilizing this structure, specifically the two-way approach, we propose a graph-regularized TT decomposition for unsupervised dimensionality reduction.

2.6.1 Problem Statement

Our goal is to find a TT projection such that the geometric structure of the samples $\mathcal{Y}_s \in \mathbf{R}^{I_1 \times \cdots \times I_N}$ is preserved, *i.e.* the distance between the samples, $\mathcal{Y}_s$, should be similar to that between the projections X_s, while the reconstruction error of the low-rank TT decomposition is minimized. This goal can be formulated

through the following cost function as:

$$f_O(\{\mathcal{U}\}, \mathcal{X}) =$$
$$\sum_{s=1}^{S} \|\mathcal{Y}_s - \mathcal{U}_1 \times_3^1 \cdots \times_3^1 \mathcal{U}_k \times_3^1 X_s \times_2^1 \mathcal{U}_{k+1} \times_3^1 \cdots \times_3^1 \mathcal{U}_N\|_F^2$$
$$+\frac{\lambda}{2} \sum_{s=1}^{S} \sum_{\substack{s'=1 \\ s' \neq s}}^{S} \|X_s - X_{s'}\|_F^2 w_{ss'}, \quad \mathbf{L}(\mathcal{U}_n)^\top \mathbf{L}(\mathcal{U}_n) = \mathbb{I}_{r_n}, \forall n$$

where $\{\mathcal{U}\}$ denotes the set of tensor factors $\mathcal{U}_n, \forall n \in \{1, \ldots, N\}$, $\mathcal{X} \in \mathbb{R}^{r_k \times S \times r_{k+1}}$ is the tensor whose slices are X_s and $w_{ss'}$ is the similarity between tensor samples defined by:

$$w_{ss'} = \begin{cases} 1, & \text{if} \quad \mathcal{Y}_s \in \mathcal{N}_k(\mathcal{Y}_{s'}) \quad \text{or} \quad \mathcal{Y}_{s'} \in \mathcal{N}_k(\mathcal{Y}_s) \\ 0, & \text{otherwise} \end{cases}, \tag{2.9}$$

where $\mathcal{N}_k(\mathcal{Y}_s)$ is the k-nearest neighborhood of $\mathcal{Y}_s$.

The objective function can equivalently be expressed as:

$$f_O(\{\mathcal{U}\}, \mathcal{X}) = \|\boldsymbol{\pi}_{k+1}(\mathcal{Y}) - \mathcal{U}_1 \times_3^1 \cdots \times_3^1 \mathcal{U}_k \times_3^1 \mathcal{X} \times_3^1 \mathcal{U}_{k+1} \times_3^1 \cdots \times_3^1 \mathcal{U}_N\|_F^2 + \lambda \text{tr}\left((\mathcal{X} \times_2^1 \Phi) \times_{1,3}^{1,2} \mathcal{X}\right),$$
$$\mathbf{L}(\mathcal{U}_n)^\top \mathbf{L}(\mathcal{U}_n) = \mathbb{I}_{r_n}, \text{ for } n \leq k \text{ and } \mathbf{R}(\mathcal{U}_n)\mathbf{R}(\mathcal{U}_n)^\top = \mathbb{I}_{r_n}, \text{ for } n > k,$$

where $\boldsymbol{\pi}_{k+1}(\mathcal{Y}) \in \mathbb{R}^{I_1 \times \cdots \times I_k \times S \times I_{k+1} \times \cdots \times I_N}$ is the permuted version of $\mathcal{Y}$ such that the last mode is moved to the $(k+1)$th mode and all modes larger than k are shifted by one mode, $W \in \mathbb{R}^{S \times S}$ is the adjacency matrix and $\Phi = D - W \in \mathbb{R}^{S \times S}$ is the graph Laplacian where D is a diagonal degree matrix with, $d_{ss} = \sum_{s'=1}^{S} w_{ss'}$.

2.6.2 Optimization

The goal of obtaining low-rank tensor train projections that preserve the data geometry can be achieved by minimizing the objective function as follows:

$$\underset{\{\mathcal{U}\}, \mathcal{X}}{\text{argmin}} f_O(\{\mathcal{U}\}, \mathcal{X}), \qquad \text{s.t. } \mathbf{L}(\mathcal{U}_n)^\top \mathbf{L}(\mathcal{U}_n) = \mathbb{I}_{r_n}, \text{ for } n \leq k, \tag{2.10}$$
$$\text{and } \mathbf{R}(\mathcal{U}_n)\mathbf{R}(\mathcal{U}_n)^\top = \mathbb{I}_{r_n}, \text{ for } n > k.$$

As we want our tensor factors to be orthogonal, the solutions lie in the Stiefel manifold $\mathcal{S}_n$, i.e. $\mathbf{L}(\mathcal{U}_n) \in \mathcal{S}_n$ for $n \leq k$ and $\mathbf{R}(\mathcal{U}_n)^\top \in \mathcal{S}_n$ for $n > k$. Although the function $f_O(.)$ is convex, the optimization problem is nonconvex due to the manifold constraints on $\mathcal{U}_n$s.

The solution to the optimization problem can be obtained by Alternating Direction Method of Multipliers (ADMM). In order to solve the optimization problem we define $\{\mathcal{V}\}$, as the set of auxiliary variables $\mathcal{V}_n, \forall n \in \{1, \ldots, N\}$ and rewrite the objective function as:

$$\underset{\{\mathcal{U}\},\{\mathcal{V}\},\mathcal{X}}{\operatorname{argmin}} \; f_O(\{\mathcal{V}\}, \mathcal{X}) \qquad \text{subject to} \quad \mathcal{U}_n = \mathcal{V}_n \quad , \forall n$$

$$\mathbf{L}(\mathcal{U}_n) \in \mathcal{S}_n, \forall n \leq k \text{ and } \mathbf{R}(\mathcal{U}_n)^\top \in \mathcal{S}_n, \forall n > k.$$

The partial augmented Lagrangian is given by:

$$\mathcal{L}\left(\{\mathcal{U}\}, \{\mathcal{V}\}, \mathcal{X}, \{\mathcal{Z}\}\right) = f_O(\{\mathcal{V}\}, \mathcal{X}) - \sum_{n=1}^{N} \mathcal{Z}_n \times_{1,2,3}^{1,2,3} (\mathcal{V}_n - \mathcal{U}_n) + \frac{\gamma}{2} \sum_{n=1}^{N} \|\mathcal{V}_n - \mathcal{U}_n\|_F^2, \tag{2.11}$$

where $\mathcal{Z}_n$s are the Lagrange multipliers and γ is the penalty parameter.

As each tensor factor is independent from the others, we update the variables for each mode n using the corresponding part of the augmented Lagrangian:

$$\mathcal{L}_n\left(\mathcal{U}_n, \mathcal{V}_n, \mathcal{Z}_n\right) = f_O(\mathcal{V}_n) - \mathcal{Z}_n \times_{1,2,3}^{1,2,3} (\mathcal{V}_n - \mathcal{U}_n) + \frac{\gamma}{2} \|\mathcal{V}_n - \mathcal{U}_n\|_F^2, \tag{2.12}$$

where $f_O(\mathcal{V}_n)$ denotes the objective function where all variables other than $\mathcal{V}_n$ are fixed. The solution for each variable at iteration $t+1$ can then be found using a step-by-step approach as:

$$\mathcal{V}_n^{t+1} = \underset{\mathcal{V}_n}{\operatorname{argmin}} \, \mathcal{L}_n\left(\mathcal{U}_n^t, \mathcal{V}_n, \mathcal{Z}_n^t\right), \tag{2.13}$$

$$\mathcal{U}_n^{t+1} = \underset{\mathcal{U}_n}{\operatorname{argmin}} \begin{cases} \mathcal{L}_n\left(\mathcal{U}_n, \mathcal{V}_n^{t+1}, \mathcal{Z}_n^t\right), \mathbf{L}(U_n) \in \mathcal{S}_n & \text{for } n \leq k, \\ \mathcal{L}_n\left(\mathcal{U}_n, \mathcal{V}_n^{t+1}, \mathcal{Z}_n^t\right), \mathbf{R}(U_n)^\top \in \mathcal{S}_n & \text{for } n > k, \end{cases}$$

$$\mathcal{Z}_n^{t+1} = \mathcal{Z}_n^t - \gamma(\mathcal{V}_n^{t+1} - \mathcal{U}_n^{t+1}). \tag{2.14}$$

Once $\mathcal{V}_n, \mathcal{U}_n, \mathcal{Z}_n$ are updated for all n, samples $\mathcal{X}$ are computed using:

$$\mathcal{X}^{t+1} = \underset{\mathcal{X}}{\operatorname{argmin}} \, \mathcal{L}\left(\{\mathcal{U}^{t+1}\}, \{\mathcal{V}^{t+1}\}, \mathcal{X}, \{\mathcal{Z}^{t+1}\}\right). \tag{2.15}$$

Solution for $\mathcal{V}_n$: For $n \leq k$, the solution for $\mathcal{V}_n^{t+1}$ can be written explicitly as:

$$\mathcal{V}_n^{t+1} = \underset{\mathcal{V}_n}{\operatorname{argmin}} \, \|\boldsymbol{\pi}_{k+1}(\mathcal{Y}) - \mathcal{V}_1^{t+1} \times_3^1 \cdots \times_3^1 \mathcal{V}_n \times_3^1 \cdots \times_3^1 \mathcal{V}_k^t \times_3^1 \mathcal{X}^t \times_3^1 \mathcal{V}_{k+1}^t \times_3^1 \cdots \times_3^1 \mathcal{V}_N^t\|_F^2$$

$$-\mathcal{Z}_n^t \times_{1,2,3}^{1,2,3} (\mathcal{V}_n - \mathcal{U}_n^t) + \frac{\gamma}{2} \|\mathcal{V}_n - \mathcal{U}_n^t\|_F^2. \tag{2.16}$$

We can equivalently convert this equation into matrix form as:

$$\mathcal{V}_n^{t+1} = \underset{\mathcal{V}_n}{\operatorname{argmin}} \|H\mathbf{L}(\mathcal{V}_n)P - \mathbf{T}_n(\boldsymbol{\pi}_{k+1}(\mathcal{Y}))\|_F^2 - \operatorname{tr}\Big(\mathbf{L}(\mathcal{Z}_n^t)^\top \mathbf{L}(\mathcal{V}_n - \mathcal{U}_n^t)\Big) + \frac{\gamma}{2}\|\mathbf{L}(\mathcal{V}_n) - \mathbf{L}(\mathcal{U}_n^t)\|_F^2, \tag{2.17}$$

where $H = \left[\mathbb{I}_{I_n} \otimes V_{\leq n-1}^{t+1}\right]$, $P = \mathbf{R}(\mathcal{V}_{n+1}^t \times_3^1 \cdots \times_3^1 \mathcal{V}_k^t \times_3^1 \mathcal{X}^t \times_3^1 \cdots \times_3^1 \mathcal{V}_N^t)$. The analytical solution is found by taking the derivative with respect to $\mathbf{L}(\mathcal{V}_n)$ and setting it to zero:

$$2H^\top\Big(H\mathbf{L}(\mathcal{V}_n^{t+1})P - G\Big)P^\top - \mathbf{L}(\mathcal{Z}_n^t) + \gamma\mathbf{L}(\mathcal{V}_n^{t+1}) - \gamma\mathbf{L}(\mathcal{U}_n^t) = 0,$$

$$\mathbf{T}_3(\mathcal{V}_n^{t+1}) = \left(2(PP^\top \otimes H^\top H) + \gamma\mathbb{I}_{r_{n-1}I_n r_n}\right)^{-1}\left(\mathbf{T}_3(\gamma\mathcal{U}_n^t + \mathcal{Z}_n^t) + 2\mathbf{T}_2(H^\top G P^\top)\right), \tag{2.18}$$

where $G = \mathbf{T}_n(\boldsymbol{\pi}_{k+1}(\mathcal{Y}))$. Note that the inverse in the solution will always exist given $\gamma > 0$ as the inverse of a sum of a Hermitian matrix and an identity matrix always exists.

When $n > k$, following (2.17) the solution for $\mathcal{V}_n$ can be written in the same manner but with different H, G and P, where $H = V_{\leq n-1}^t$, $P = \left[V_{>n}^{t+1} \otimes \mathbb{I}_{I_n}\right]$ and $G = \mathbf{T}_{n+1}(\boldsymbol{\pi}_{k+1}(\mathcal{Y}))$.

Solution for $\mathcal{U}_n$: For $n \leq k$, we can solve (2.12) for $\mathcal{U}_n$ using:

$$\begin{aligned}\mathcal{U}_n^{t+1} &= \underset{\mathcal{U}_n:\mathbf{L}(\mathcal{U}_n)\in\mathcal{S}_n}{\operatorname{argmin}} -\operatorname{tr}\Big(\mathbf{L}(\mathcal{Z}_n^t)^\top \mathbf{L}(\mathcal{V}_n^{t+1} - \mathcal{U}_n)\Big) + \frac{\gamma}{2}\|\mathbf{L}(\mathcal{V}_n^{t+1}) - \mathbf{L}(\mathcal{U}_n)\|_F^2 \\ &= \underset{\mathcal{U}_n:\mathbf{L}(\mathcal{U}_n)\in\mathcal{S}_n}{\operatorname{argmin}} \|\mathbf{L}(\mathcal{V}_n^{t+1}) - \frac{1}{\gamma}\mathbf{L}(\mathcal{Z}_n^t) - \mathbf{L}(\mathcal{U}_n)\|_F^2,\end{aligned} \tag{2.19}$$

which is found by applying a singular value decomposition to $\mathbf{L}(\mathcal{V}_n^{t+1}) - \frac{1}{\gamma}\mathbf{L}(\mathcal{Z}_n^t)$. When $n > k$, the optimal solution is similarly found by applying SVD to $\mathbf{R}(\mathcal{V}_n^{t+1}) - \frac{1}{\gamma}\mathbf{R}(\mathcal{Z}_n^t)$.

Solution for $\mathcal{X}$: Let $\boldsymbol{\pi}_2(\mathcal{X}) \in \mathbb{R}^{r_k \times r_{k+1} \times S}$ be the permutation of $\mathcal{X}$, (2.15) can equivalently be rewritten in matrix form as:

$$\underset{\mathcal{X}}{\operatorname{argmin}} \left\|\left[V_{>k}^{t+1\top} \otimes V_{\leq k}^{t+1}\right]\mathbf{L}(\boldsymbol{\pi}_2(\mathcal{X})) - \mathbf{T}_N(\mathcal{Y})\right\|_F^2 + \lambda\operatorname{tr}\left(\mathbf{L}(\boldsymbol{\pi}_2(\mathcal{X}))\Phi\mathbf{L}(\boldsymbol{\pi}_2(\mathcal{X}))^\top\right). \tag{2.20}$$

The solution for $\mathcal{X}^{t+1}$ does not have any constraints, thus it is solved analytically by setting the derivative of (2.20) to zero:

$$\begin{aligned}2H^\top(H\mathbf{L}(\boldsymbol{\pi}_2(\mathcal{X}^{t+1})) - G) + 2\lambda\mathbf{L}(\boldsymbol{\pi}_2(\mathcal{X}^{t+1}))\Phi = 0, \\ H^\top H\mathbf{L}(\boldsymbol{\pi}_2(\mathcal{X}^{t+1})) + \lambda\mathbf{L}(\boldsymbol{\pi}_2(\mathcal{X}^{t+1}))\Phi = H^\top G,\end{aligned} \tag{2.21}$$

where $H = {V_{>k}^t}^\top \otimes V_{\leq k}^{t+1}$ and $G = \mathbf{T}_N(\mathcal{Y})$. (2.21) is a Sylvester equation which can be solved efficiently [14]. Similar to the case for $\mathcal{V}_n$, the solution to this problem always exists.

Solution for $\mathcal{Z}_n$: Finally, we update the Lagrange multipliers $\mathcal{Z}_n^t$ using (2.14).

Algorithm 2.3: Graph Regularized Tensor Train-ADMM(GRTT-ADMM)

Input: Input tensors $\mathcal{Y}_s \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ where $s \in \{1, \ldots, S\}$, initial tensor factors $\{\mathcal{U}^1\}, n \in \{1, \ldots, N\}$, $k, \lambda, r_1, \ldots, r_N, LoopIter, ConvThresh$

Output: $\mathcal{U}_n, n \in \{1, \ldots, N\}$, and $X_s, \quad \forall s$

$\{\mathcal{V}^1\} \leftarrow \{\mathcal{U}^1\}$.
$\{\mathcal{Z}^1\} \leftarrow 0$.
while $t < LoopIter$ **or** $c > ConvThres$ **do**
  for $n = 1 : N$ **do**
    Find $\mathcal{V}_n^{t+1}$ using (2.18).
    Find $\mathcal{U}_n^{t+1}$ using SVD to solve (2.19).
    Find $\mathcal{Z}_n^{t+1}$ using (2.14).
  end for
  Find $\mathcal{X}^{t+1}$ using (2.21).
  $c \leftarrow \frac{1}{N} \sum_{n=1}^{N} \frac{\|\mathcal{V}_n^{t+1} - \mathcal{V}_n^t\|_F^2}{\|\mathcal{V}_n^t\|_F^2}$
  $t = t + 1$.
end while

2.6.3 Convergence

Convergence of ADMM is guaranteed for convex functions but there is no theoretical proof of the convergence of ADMM for nonconvex functions. Recent research has provided some theoretical guarantees for the convergence of ADMM for a class of nonconvex problems under some conditions [187].

Our objective function is nonconvex due to unitary constraints. In [187], it has been shown that this type of nonconvex optimization problems, i.e. convex optimization on a Stiefel manifold, converge under some conditions. We show that these conditions hold for each optimization problem corresponding to mode n. The gradient of f_O with respect to $\mathcal{V}_n$ is Lipschitz continuous with Lipschitz constant $L \geq \|PP^\top \otimes H^\top H\|_2$, which fulfills the conditions given in [187]. Thus, $\mathcal{L}_n$ convergences to a set of solutions $\mathcal{V}_n^t, \mathcal{U}_n^t, \mathcal{Z}_n^t$, given that $\gamma \geq 2L + 1$. The solution for $\mathcal{X}$ is found analytically. As the iterative solutions for each variable converge and the optimization function is nonnegative, *i.e.* bounded from below, the algorithm converges to a local minimum.

2.7 Experiments

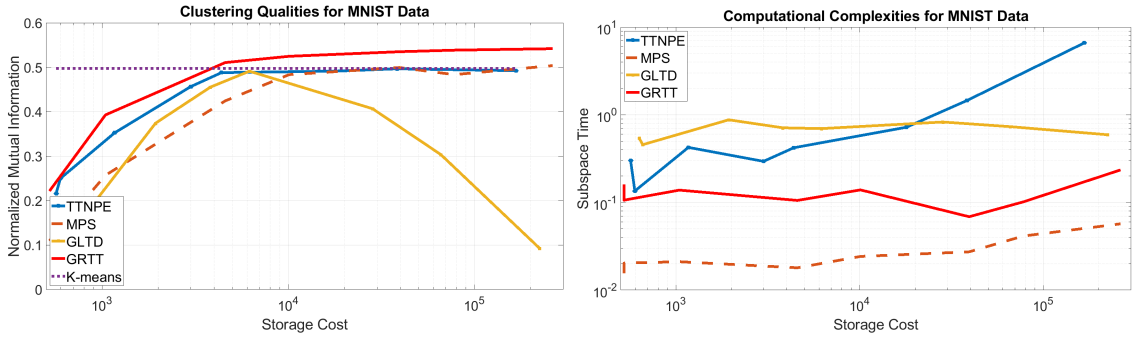
The proposed method is evaluated for clustering and compared to existing tensor clustering methods including k-means, MPS [18], TTNPE [183] and GLTD [82] for Weizmann Face Database and MNIST Dataset. Clustering quality is quantified by Normalized Mutual Information (NMI). Average accuracy with respect to both storage complexity and computation time over 20 experiments are reported for all methods.

In the following experiments, the storage complexity is quantified as the size of the tensor factors ($\mathcal{U}_n, \forall n$) and projections ($\mathcal{X}_s, \forall s$). The varying levels of storage cost are obtained by varying r_ns in the implementation of the tensor decomposition methods. Using varying levels of a truncation parameter $\tau \in (0, 1]$, the singular values smaller than τ times the largest singular value are discarded. The rest are used to determine ranks r_n for both TT based and TD based methods. For GRTT and TTNPE, the ranks are selected using TT decomposition proposed in [140], while for GLTD truncated HOSVD was used. Computational complexity is quantified as the time it takes to learn the tensor factors. In order to compute the run time, for TT based methods, each set of tensor factors is optimized until the change in the normalized difference between consecutive tensor factors is less than 0.01 or 50 iterations are completed.

The regularization parameter, λ, for each experiment was selected using a validation set composed of a small batch of samples not included in the experiments. 5 random experiments were conducted and optimal λ was selected as the value that gave the best average NMI for a range of λ values from 0.001 to 1000 increasing in a logarithmic scale. The similarity graphs were constructed using k-nearest neighbor method with $k = log(S)$ following [177].

2.7.1 MNIST

MNIST is a database of grayscale handwritten digit images where each image is of size 28×28. We transformed each of the images to a $4 \times 7 \times 4 \times 7$ tensor. Reshaping the inputs into higher order tensors is common practice and was employed in prior work [90, 140, 209, 45, 17]. In our experiments, we used a subset of 500 images with 50 images from each class. 50 samples with 5 samples from each class are used as validation set to determine λ.



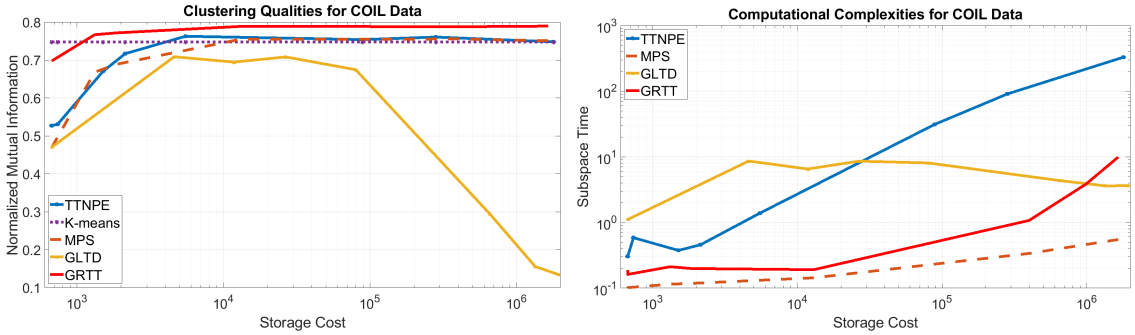
(a) (b)

Figure 2.8: (a) Normalized Mutual Information vs. Storage Complexity of different methods for MNIST dataset. (b) Computation Time vs. Storage Complexity of different methods for MNIST dataset.

In Figure 2.8a, we can see that at all storage complexity levels, our approach gives the best clustering result in terms of NMI. The dotted purple line represents the accuracy of k-means clustering on original tensor data. Even though the performance of TTNPE is the closest to our method, it is computationally inefficient. In Figure 2.8b, we can see that our approach is faster than GLTD and TTNPE at all storage complexities. MPS is the most efficient in terms of speed but it provides poor clustering quality.

2.7.2 COIL

The dataset consists of 7,200 RGB images of 100 objects of size 128×128. Each object has 72 images, where each image corresponds to a different pose angle ranging from 0 to 360 degrees with increments of 5 degrees [133]. We used a subset of 20 classes and 32 randomly selected, downsampled, grayscale samples from each class. Each image was converted to an $8 \times 8 \times 8 \times 8$ tensor. 8 samples from each class are used as validation set.



(a) (b)

Figure 2.9: (a) Normalized Mutual Information vs Storage Complexity of different methods for COIL dataset. (b) Computation Time vs Storage Complexity of different methods for COIL dataset.

From Figure 2.9a, we can see that the proposed method provides the best clustering results compared to all other methods. The results for GLTD seem to deteriorate with increasing ranks, which is a result of using orthonormal tensor factors. TTNPE gives results closest to the proposed method but it is computationally inefficient and gets very slow with increasing r_ns. From Figure 2.9b, we can see that MPS provides best results in terms of run time but the proposed method has a similar computational complexity while providing better clustering accuracy.

2.8 Conclusions

In this chapter, we propose a new Tensor Train implementation structure and two associated learning tasks. In the first part of the chapter, *i.e.* Tensor Train Disciminant Analysis, we introduce a novel approach for Tensor Train based discriminant analysis for tensor object classification. The proposed approach first formulated linear discriminant analysis such that the learnt subspaces have a TT structure. The resulting framework, TTDA, reduces storage complexity at the expense of high computational complexity. This increase in computational complexity is then addressed by reshaping the projection vector into matrices and third-order tensors, resulting in 2WTTDA and 3WTTDA, respectively. A theoretical analysis of storage and computational complexity illustrated the tradeoff between these two quantities and suggest a way to select the optimal number of modes in the reshaping of TT structure. The proposed methods were compared with the state-of-the-art TT based subspace learning methods as well as tensor based discriminant analysis for four datasets. While providing reduced storage and computational costs, the proposed methods also yield higher or similar classification accuracy compared to state-of-the art tensor based learning methods such as CMDA, STTM, TTNPE and MPS.

In the second part of the chapter, *i.e.* Graph Regularized Tensor Train Decomposition we proposed a unsupervised graph regularized tensor train decomposition for dimensionality reduction. To the best of our knowledge, this is the first tensor train based dimensionality reduction method that incorporates manifold information through graph regularization. The proposed method also utilizes a multi-branch structure to implement tensor train decomposition which increases the computational efficiency. An ADMM based algorithm is proposed to solve the resulting optimization problem. The proposed method was compared to GLTD, TTNPE and MPS for unsupervised learning in two different datasets.

The proposed multi-branch structure can also be extended to other unsupervised methods such as dictionary learning, subspace learning, denoising, data recovery, and compression applications. The structure can also be optimized by permuting the modes in a way that the dimensions are better balanced than the original order.

CHAPTER 3

TENSOR METHODS FOR ANOMALY DETECTION ON SPATIOTEMPORAL DATA

3.1 Introduction

Large volumes of spatiotemporal data are ubiquitous in a diverse range of applications including climate science, social sciences, neuroscience, epidemiology [20], transportation systems [55], mobile health, and Earth sciences [207]. One emerging application of interest in spatiotemporal data is anomaly detection. Detecting anomalies from these large data volumes is important for identifying interesting but rare phenomena, e.g. traffic congestion or irregular crowd movement in urban areas, or hot-spots for monitoring outbreaks in infectious diseases. Traditionally, the problem of anomaly detection has been approached mainly through statistical and machine learning techniques [34, 59, 79]. However, these techniques are not as effective on spatiotemporal data as anomalies are highly correlated across time and space, they can no longer be modeled as i.i.d.

The definition of anomaly and the suitability of a particular method is determined by the application. In this chapter, we focus on urban anomaly detection [206, 205, 117, 118, 203], where anomalies correspond to incidental events that occur rarely, such as irregularity in traffic volume, unexpected crowds, etc. Urban data are spatiotemporal data collected by mobile devices or distributed sensors in cities and are usually associated with timestamps and location tags. Detecting and predicting urban anomalies are of great importance to policymakers and governments for understanding city-scale human mobility and activity patterns, inferring land usage and region functions and discovering traffic problems [206, 31].

Urban anomalies are usually modeled as group anomalies within spatiotemporal data. This type of anomalies exhibit themselves as spatially contiguous groups of locations that show anomalous values consistently for a short duration of time stamps. Early approaches to detecting these anomalies decompose the anomaly detection problem by first treating the spatial and temporal properties of the outliers independently using univariate vector modeling frameworks, and then merging together in a post-processing step [58, 191]. Some examples are dynamic linear models (DLMs) [108] including time-varying autoregressive (TVAR) [24, 25] and switching Kalman filtering/smoothing (SKF/SKS) [131, 134]. However, as the number of sensors increases, scalability becomes a critical issue and these methods become inefficient and unreliable [59]. For

these reasons, one natural approach to address ST anomaly detection has been to use tensor decomposition to capture the multiway structure of the data.

While there has been a growing number of papers in low-rank tensor models for anomaly detection [60, 186, 193, 185, 205, 117], most of the existing methods are not particularly suited to the complex structure of anomalies in urban traffic data and the unique challenges associated with them. These challenges include: 1) scarcity of anomalies; 2) the contextual dependency of what constitutes an anomaly, i.e., the criteria for anomaly may vary for different regions and time windows due to external influences such as weather patterns; 3) long-term periodicity across time, e.g., one day or one week and 4) strong short-term temporal correlations.

In this chapter, we address these challenges by proposing temporally regularized, locally consistent, robust low-rank plus sparse tensor models to decompose the urban traffic data into normal and anomalous components. The key contributions of the proposed methods are:

- **Modeling Temporal Persistence**: The traditional low-rank plus sparse tensor decomposition model is modified to account for the characteristics of urban traffic data. As anomalies tend to last for periods of time, i.e., have strong short-term dependencies, we impose temporal smoothness on the sparse part of the tensor through total variation regularization. This regularization ensures that instantaneous changes in the data, which may be due to errors in sensing, are not mistaken for actual anomalies. This formulation leads to our first algorithm; low-rank plus temporally smooth sparse (LOSS) tensor decomposition.

- **Incorporating Local Structure**: We introduce additional structure to the solution of LOSS by enforcing the low-rank tensor corresponding to normal activity to be smooth on manifolds across each mode. These smoothness terms are added to LOSS as graph regularization across each mode yielding GLOSS. While the low-rank structure in LOSS captures the global structure, GLOSS further reduces redundancy in the representation and incorporates the local geometric structure. This new framework exploits the joint structures and correlations across the modes to more accurately model and process ST signals.

- **Using Local Structure for Efficiency**: We reduce the computational cost incurred by the nuclear norm minimization in GLOSS and LOSS by utilizing the graph smoothness term to estimate normal

activity. This allows an efficient algorithm that is still effective in extracting anomalies with some mild constraints on data.

- **Robustness to Missing and Heterogenous Data**: Our optimization framework is formulated in a flexible manner such that it solves a tensor completion problem in conjunction with anomaly detection. This framework takes into account the missing data and fits a structure to the observed part when estimating the underlying structure. Thus, it provides robustness against missing data. The heterogeneity of traffic data is taken into account through the use of weighted nuclear norm minimization to emphasize the difference in low-rank structure across modes.

The rest of the chapter is organized as follows. In Section, 3.2, we review some of the related work in spatiotemporal anomaly detection, in particular tensor based anomaly detection methods. In Section 3.3, we formulate the optimization problems of LOSS and GLOSS, propose ADMM based solutions and analyze the computational complexity of these solutions. In Section 3.4, we formulate the optimization problem of LOGSS, propose an ADMM based solution and analyze the computational complexity. In Section 3.5 we provide an analysis of convergence of all proposed algorithms. In Section 3.6, we explain how the anomaly scores for each element in the data is generated. In Section 3.7, we describe the experimental settings both with synthetic and real data and compare the proposed methods with baseline anomaly detection methods as well as other tensor based methods. Finally, we conclude the chapter in Section 3.8, by discussing the proposed algorithms, experimental results and future work.

3.2 Related Work

Anomaly detection in spatiotemporal data is usually studied under three categories: point anomalies; trajectory anomalies and group anomalies. Point anomalies are defined as spatiotemporal outliers that break the natural ST autocorrelation structure of the normal points. Most ST point anomaly detection algorithms such as ST-DBSCAN [100] assume homogeneity in neighborhood properties across space and time, which can be violated in the presence of ST heterogeneity. Trajectory anomalies are usually detected by computing pairwise similarities among trajectories and identifying trajectories that are spatially distant from the others [105]. Finally, group anomalies appear in ST data as spatially contiguous groups of locations that show anomalous values consistently for a short duration of time stamps. Most approaches for detecting group anomalies decompose the anomaly detection problem by first treating the spatial and temporal properties

of the outliers independently, and then merging together in a post-processing step [58, 191]. However, as the number of sensors increases, scalability becomes a critical issue and these methods become unreliable. For these reasons, one natural approach to address ST group anomaly detection has been to use tensor decomposition.

Low-rank tensor decomposition and completion have been proposed as suitable approaches to anomaly detection in spatiotemporal data as these methods are a natural extension of spectral anomaly detection techniques to multi-way data [59, 113, 68, 207, 39, 118, 117, 193]. These models project the original spatiotemporal data into a low-dimensional latent space, in which the normal activity is represented with better spatial and temporal resolution. The learned features, i.e., factor matrices or core tensors, are then used to detect anomalies by monitoring the reconstruction error at each time point [144, 143, 166, 193] or by applying well-known statistical tests to the extracted multivariate features [60, 207]. For example, Zhang *et al.* [207] proposed a tensor-based method to detect targets in hyperspectral imaging data with both spectral and spatial anomaly characteristics. Shi *et al.* [158] proposed an incremental tensor decomposition algorithm for online anomaly detection. In [60], a hybrid model is constructed from a topology tensor and a flow tensor and Tucker decomposition with an adjustable core size is used to detect anomalies. Xu *et al.* [193] proposed a sliding-window tensor factorization to detect anomalies. Wang *et al.* [185] expanded the traditional Tucker decomposition in a probabilistic manner to detect abnormal activity behaviors.

Although these low-rank tensor models are powerful in identifying abnormal traffic activity, they have multiple shortcomings. First, they rely on well-known factorization models such as Tucker [60, 207, 193] and CP [117]. These methods obtain a low-dimensional projection of the data without taking the particular structure of anomalies, i.e., sparsity, into account. The proposed method incorporates the sparsity of anomalies into the model by assuming that the anomalies lie in the sparse part of the tensor rather than in the low-rank part. Second, prior work on higher order robust principal component analysis (HoRPCA) within the framework of anomaly detection [113, 68] does not ensure temporal smoothness of the detected anomalies. However, in urban data, anomalies typically last for some time and are not instantaneous. Recently, temporally regularized matrix factorization models [149, 50, 200] and their extensions to tensors [199, 179, 184, 88] have been implemented to capture the temporal dynamics. While the smoothness term included in these papers is very similar to ours, in these papers the smoothness is enforced on the factor matrices corresponding to the temporal mode while in our approach temporal regularization is applied directly to the sparse tensor.

Finally, the existing tensor based anomaly detection methods are limited to modeling anomalies that lie in linear subspaces. The proposed method utilizes a simultaneously structured model [84] for robust tensor decomposition. This model structures the low-rank tensor by the proximities within each mode and forces the solution to be smooth on manifolds constructed from each mode. This goal is achieved by adding graph regularization across each mode of the tensor. Although graph regularized tensor decomposition has been used before [115, 135, 82, 148, 13, 180, 192, 51], it is usually with respect to one mode of the tensor and it was not utilized for anomaly detection in urban traffic data. Yet, for many tensors, correlations exist across all modalities. Several recent papers [67, 84, 156, 135, 155, 128, 127] exploit this coupled relationship to co-organize matrices and infer underlying row and column embeddings. The methods GLOSS and LOGSS in this chapter are direct extensions of this work to tensors where we consider graph regularization across all modes to capture the coupled correlation structure in spatiotemporal data.

3.3 Robust Low-Rank Tensor Decomposition for Anomaly Detection

In the following discussions, we model spatiotemporal data as a four mode tensor $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4}$. The first mode corresponds to hours in a day. The second mode corresponds to the days of a week as urban traffic activity shows highly similar patterns on the same days of different weeks. The third mode corresponds to the different weeks and the last mode corresponds to the spatial locations, such as stations for metro data, sensors for traffic data or zones for other urban data.

3.3.1 Problem Statement

Assuming that anomalies are rare events, our goal is to decompose $\mathcal{Y}$ into a low-rank part, $\mathcal{L}$, that corresponds to normal activity and a sparse part, $\mathcal{S}$, that corresponds to the anomalies. This model relies on the assumption that normal activity can be embedded into a lower dimensional subspace while anomalies are outliers. We also take into account the existence of missing elements in the data, *i.e.*, the observed data is $\mathcal{P}_\Omega[\mathcal{Y}]$. This goal can be formulated as:

$$\min_{\mathcal{L},\mathcal{S}} \|\mathcal{L}\|_* + \lambda\|\mathcal{S}\|_1, \quad \text{s.t. } \mathcal{P}_\Omega[\mathcal{L}+\mathcal{S}] = \mathcal{P}_\Omega[\mathcal{Y}], \tag{3.1}$$

where λ is the regularization parameter for sparsity.

Since urban anomalies tend to be temporally continuous, *i.e.*, smooth in the first mode, this assumption

can be incorporated into the above formulation as:

$$\min_{\mathcal{L},\mathcal{S}} \|\mathcal{L}\|_* + \lambda\|\mathcal{S}\|_1 + \gamma\|\mathcal{S} \times_1 \Delta\|_1, \text{ s.t. } \mathcal{P}_\Omega[\mathcal{L} + \mathcal{S}] = \mathcal{P}_\Omega[\mathcal{Y}], \tag{3.2}$$

where γ is the regularization parameter for temporal smoothness and $\|\mathcal{S} \times_1 \Delta\|_1$ quantifies the sparsity of the projection of the tensor $\mathcal{S}$ onto the discrete-time differentiation operator along the first mode where Δ is defined as:

$$\Delta = \begin{bmatrix} 1 & -1 & 0 & \dots & 0 \\ 0 & 1 & -1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & -1 \\ -1 & 0 & \dots & 0 & 1 \end{bmatrix}. \tag{3.3}$$

It is common to incorporate the relationships among data points as auxiliary information in addition to the low-rank assumption to improve the quality of tensor decomposition and to capture the local data structure [132, 164]. This approach, also known as manifold learning, is an effective dimensionality reduction technique leveraging geometric information. The intuitive idea behind manifold learning is that if two objects are close in the intrinsic geometry of data manifold, they should be close to each other after dimensionality reduction. For tensors, this usually reduces to forcing two similar objects to behave similarly in the projected low-dimensional space through a graph Laplacian term. In this section, since we are trying to learn anomalies from a single tensor, we do not have tensor samples and their projections. Instead, we preserve the relationships between each mode unfolding of the tensor data as each mode corresponds to a different attribute of the data.This goal can be achieved through graph regularization across each mode as follows:

$$\min_{\mathcal{L},\mathcal{S}} \|\mathcal{L}\|_* + \frac{\theta}{2}\sum_{n=1}^{4}\sum_{i=1}^{I_n}\sum_{\substack{i'=1 \\ i'\neq i}}^{I_n} \|\mathcal{L}_{(n),i} - \mathcal{L}_{(n),i'}\|_F^2 w_{ii'}^n + \lambda\|\mathcal{S}\|_1 +$$

$$\gamma\|\mathcal{S} \times_1 \Delta\|_1, \qquad s.t. \quad \mathcal{P}_\Omega[\mathcal{L} + \mathcal{S}] = \mathcal{P}_\Omega[\mathcal{Y}],$$

where θ is the weight parameter for graph regularization.

The above optimization problem can equivalently be rewritten using trace norm to represent the graph

regularization and total variation (TV) norm across the temporal mode to describe temporal smoothness as:

$$\min_{\mathcal{L},\mathcal{S}} \|\mathcal{L}\|_* + \theta \sum_{n=1}^{4} \text{tr}\left(\mathcal{L}_{(n)}^{\top} \Phi^n \mathcal{L}_{(n)}\right) + \lambda \|\mathcal{S}\|_1 +$$
$$\gamma \sum_{i_2,i_3,i_4} \|\mathbf{s}_{i_2,i_3,i_4}\|_{\text{TV}}, \qquad \mathcal{P}_\Omega[\mathcal{L}+\mathcal{S}] = \mathcal{P}_\Omega[\mathcal{Y}], \tag{3.4}$$

where $\|.\|_{\text{TV}}$ denotes the total variation norm. The solution to the optimization problems (3.2) and (3.4) are referred to as LOw-rank plus temporally Smooth Sparse (LOSS) decomposition, and Graph regularized LOw-rank plus temporally Smooth Sparse (GLOSS) decomposition, respectively.

3.3.2 Optimization

The proposed objective function is convex. In prior work, ADMM has been shown to be effective at solving similar optimization problems in an iterative fashion [70, 7]. Thus, we follow a similar approach for solving the optimization problem given in (3.4). To separate the minimization of TV, ℓ_1 norm and graph regularization from each other, we introduce auxiliary variables $\mathcal{Z}, \mathcal{W}, \{\mathfrak{L}\} := \{\mathfrak{L}^1, \mathfrak{L}^2, \mathfrak{L}^3, \mathfrak{L}^4\}, \{\mathfrak{G}\} := \{\mathfrak{G}^1, \mathfrak{G}^2, \mathfrak{G}^3, \mathfrak{G}^4\}$ such that the optimization problem becomes:

$$\min_{\mathcal{L},\{\mathfrak{L}\},\{\mathfrak{G}\},\mathcal{S},\mathcal{Z},\mathcal{W}} \sum_{n=1}^{4} \left(\psi_n \|\mathfrak{L}^n_{(n)}\|_* + \theta g(\mathfrak{G}^n, \Phi^n)\right) + \lambda \|\mathcal{S}\|_1 +$$
$$\gamma \|\mathcal{Z}\|_1, \qquad \text{s.t. } \mathcal{W} = \mathcal{S}, \quad \mathcal{Z} = \mathcal{W} \times_1 \Delta,$$
$$\mathcal{P}_\Omega[\mathcal{L}+\mathcal{S}] = \mathcal{P}_\Omega[\mathcal{Y}], \mathfrak{L}^n = \mathcal{L}, \mathfrak{G}^n = \mathcal{L}, n \in \{1,2,3,4\}, \tag{3.5}$$

where $g(\mathfrak{G}^n, \Phi^n) = \text{tr}\left({\mathfrak{G}^n_{(n)}}^{\top} \Phi^n \mathfrak{G}^n_{(n)}\right)$ is the graph regularization term for each auxiliary variable $\mathfrak{G}^n$. To solve the above optimization problem, we propose using ADMM with partial augmented Lagrangian:

$$\sum_{n=1}^{4} \left(\psi_n \|\mathfrak{L}^n_{(n)}\|_* + \theta g(\mathfrak{G}^n, \Phi^n)\right) + \lambda \|\mathcal{S}\|_1 + \gamma \|\mathcal{Z}\|_1 +$$
$$\frac{\beta_1}{2} \|\mathcal{P}_\Omega[\mathcal{L}+\mathcal{S}-\mathcal{Y}-\Gamma_1]\|_F^2 + \frac{\beta_2}{2} \sum_{n=1}^{4} \|\mathfrak{L}^n - \mathcal{L} - \Gamma_2^n\|_F^2 +$$
$$\frac{\beta_3}{2} \sum_{n=1}^{4} \|\mathcal{L} - \mathfrak{G}^n - \Gamma_3^n\|_F^2 + \frac{\beta_4}{2} \|\mathcal{W} \times_1 \Delta - \mathcal{Z} - \Gamma_4\|_F^2 +$$
$$\frac{\beta_5}{2} \|\mathcal{S} - \mathcal{W} - \Gamma_5\|_F^2, \tag{3.6}$$

where $\Gamma_1, \Gamma_2^n, \Gamma_3^n, \Gamma_4, \Gamma_5 \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4}$ are the Lagrange multipliers.

1. $\mathcal{L}$ update: The low-rank variable $\mathcal{L}$ can be updated using:

$$\mathcal{L}^{t+1} = \underset{\mathcal{L}}{\operatorname{argmin}} \frac{\beta_1}{2}\|\mathcal{P}_\Omega[\mathcal{L} + \mathcal{S}^t - \mathcal{Y} - \Gamma_1^t]\|_F^2 + \sum_{n=1}^{4}\left(\frac{\beta_2}{2}\|\mathfrak{L}^{n,t} - \mathcal{L} - \Gamma_2^{n,t}\|_F^2 + \frac{\beta_3}{2}\|\mathcal{L} - \mathfrak{G}^{n,t} - \Gamma_3^{n,t}\|_F^2\right), \tag{3.7}$$

which has the analytical solution:

$$\mathcal{P}_\Omega[\mathcal{L}^{t+1}] = \frac{\mathcal{P}_\Omega\left[\beta_1\mathcal{T}_1 + \beta_2\mathcal{T}_2 + \beta_3\mathcal{T}_3\right]}{\beta_1 + 4(\beta_2 + \beta_3)}, \tag{3.8}$$

$$\mathcal{P}_{\Omega^\perp}[\mathcal{L}^{t+1}] = \frac{\mathcal{P}_{\Omega^\perp}\left[\beta_2\mathcal{T}_2 + \beta_3\mathcal{T}_3\right]}{4(\beta_2 + \beta_3}, \tag{3.9}$$

where $\mathcal{T}_1 = \mathcal{Y} - \mathcal{S}^t + \Gamma_1^t$, $\mathcal{T}_2 = \sum_{n=1}^4 \mathfrak{L}^{n,t} - \Gamma_2^{n,t}$ and $\mathcal{T}_3 = \sum_{n=1}^4 \mathfrak{G}^{n,t} + \Gamma_3^{n,t}$.

2. $\mathfrak{L}^n$ update: The variables $\mathfrak{L}^n$ can be updated using:

$$\mathfrak{L}^{n,t+1} = \underset{\mathfrak{L}^n}{\operatorname{argmin}} \psi_n\|\mathfrak{L}^n_{(n)}\|_* + \frac{\beta_2}{2}\|\mathfrak{L}^n - \mathcal{L}^{t+1} - \Gamma_2^{n,t}\|_F^2, \tag{3.10}$$

which is solved by a soft thresholding operator on singular values of $\left(\mathcal{L}^{t+1} + \Gamma_2^{n,t}\right)_{(n)}$ with a threshold of ψ_n/β_2.

3. $\mathfrak{G}^n$ update: The variables $\mathfrak{G}^n$ can be updated using:

$$\mathfrak{G}^{n,t+1} = \underset{\mathfrak{G}^n}{\operatorname{argmin}} \theta\operatorname{tr}\left(\mathfrak{G}^n_{(n)}{}^\top \Phi^n \mathfrak{G}^n_{(n)}\right) + \frac{\beta_3}{2}\|\mathcal{L}^{t+1} - \mathfrak{G}^n - \Gamma_3^{n,t}\|_F^2, \tag{3.11}$$

which is solved by:

$$\mathfrak{G}^{n,t+1}_{(n)} = \beta_3 G_{inv}\left(\mathcal{L}^{t+1} - \Gamma_3^{n,t}\right)_{(n)}, \tag{3.12}$$

where $G_{inv} = (2\theta\Phi^n + \beta_3\mathbf{I})^{-1}$ always exists and can be computed outside the loop for faster update.

4. $\mathcal{S}$ update: The variable $\mathcal{S}$ can be updated using:

$$\mathcal{S}^{t+1} = \underset{\mathcal{S}}{\operatorname{argmin}} \lambda\|\mathcal{S}\|_1 + \frac{\beta_1}{2}\|\mathcal{P}_\Omega[\mathcal{S} + \mathcal{L}^{t+1} - \mathcal{Y} - \Gamma_1^t]\|_F^2 + \frac{\beta_5}{2}\|\mathcal{S} - \mathcal{W}^t - \Gamma_5^t\|_F^2, \tag{3.13}$$

where the Frobenius norm terms can be combined and the expression can be simplified into:

$$\mathcal{P}_\Omega[\mathcal{S}^{t+1}] = \underset{\mathcal{P}_\Omega[\mathcal{S}]}{\operatorname{argmin}} \|\mathcal{P}_\Omega[\mathcal{S}]\|_1 + \frac{\beta_1 + \beta_5}{2\lambda}\|\mathcal{P}_\Omega[\mathcal{S} - \mathcal{T}_s]\|_F^2, \tag{3.14}$$

$$\mathcal{P}_{\Omega^\perp}[\mathcal{S}^{t+1}] = \underset{\mathcal{P}_{\Omega^\perp}[\mathcal{S}]}{\operatorname{argmin}} \|\mathcal{P}_{\Omega^\perp}[\mathcal{S}]\|_1 + \frac{\beta_5}{2\lambda}\|\mathcal{P}_{\Omega^\perp}[\mathcal{S} - \mathcal{T}_s]\|_F^2, \tag{3.15}$$

where

$$\mathcal{P}_{\Omega}[\mathcal{T}_s] = \mathcal{P}_{\Omega}\left[\frac{\beta_1(\mathcal{Y} - \mathcal{L}^{t+1} + \Gamma_1^t) + \beta_5(\mathcal{W}^t + \Gamma_5^t)}{\beta_1 + \beta_5}\right]$$

$$\mathcal{P}_{\Omega^\perp}[\mathcal{T}_s] = \mathcal{P}_{\Omega^\perp}[\mathcal{W}^t + \Gamma_5^t]$$

The above is solved by setting $\mathcal{P}_{\Omega}[\mathcal{S}^{t+1}] = soft\_thresh(\mathcal{P}_{\Omega}[\mathcal{T}_s], \frac{\lambda}{\beta_1+\beta_5})$ and $\mathcal{P}_{\Omega^\perp}[\mathcal{S}^{t+1}] = soft\_thresh(\mathcal{P}_{\Omega^\perp}[\mathcal{T}_s], \frac{\lambda}{\beta_5})$, where $soft\_thresh(\mathbf{a}, \phi) = sign(\mathbf{a}) \odot max(|\mathbf{a}| - \phi, 0)$ and $\odot$ is elementwise or Hadamard product.

5. $\mathcal{W}$ update: The auxiliary variable $\mathcal{W}$ can be updated using:

$$\mathcal{W}^{t+1} = \underset{\mathcal{W}}{\operatorname{argmin}} \frac{\beta_4}{2}\|\mathcal{W} \times_1 \Delta - \mathcal{Z}^t - \Gamma_4^t\|_F^2 + \frac{\beta_5}{2}\|\mathcal{S}^{t+1} - \mathcal{W} - \Gamma_5^t\|_F^2, \tag{3.16}$$

which is solved analytically by taking the derivative of the expression given above and setting it to zero which results in:

$$\mathcal{W}_{(1)}^{t+1} = W_{inv}\left(\beta_5(\mathcal{S}^{t+1} - \Gamma_5^t)_{(1)} + \beta_4\Delta^\top(\Gamma_4^t + \mathcal{Z}^t)_{(1)}\right), \tag{3.17}$$

where $W_{inv} = (\beta_5\mathbf{I} + \beta_4\Delta^\top\Delta)^{-1}$ always exists and can be computed outside the loop for faster update.

6. $\mathcal{Z}$ update: The auxiliary variable $\mathcal{Z}$, can be updated using:

$$\mathcal{Z}^{t+1} = \underset{\mathcal{Z}}{\operatorname{argmin}}\, \gamma\|\mathcal{Z}\|_1 + \frac{\beta_4}{2}\|\mathcal{W}^{t+1} \times_1 \Delta - \mathcal{Z} - \Gamma_4^t\|_F^2 \tag{3.18}$$

which is solved by $soft\_thresh(\mathcal{W}^{t+1} \times_1 \Delta - \Gamma_4^t, \gamma/\beta_4)$.

7. Dual updates: Finally, dual variables $\Gamma_1, \Gamma_2^n, \Gamma_3^n, \Gamma_4, \Gamma_5$ are updated using:

$$\Gamma_1^{t+1} = \Gamma_1^t - \mathcal{P}_{\Omega}[\mathcal{L}^{t+1} + \mathcal{S}^{t+1} - \mathcal{Y}], \tag{3.19}$$

$$\Gamma_2^{n,t+1} = \Gamma_2^{n,t} - (\mathfrak{L}^{n,t+1} - \mathcal{L}^{t+1}), \tag{3.20}$$

$$\Gamma_3^{n,t+1} = \Gamma_3^{n,t} - (\mathcal{L}^{t+1} - \mathfrak{G}^{n,t+1}), \tag{3.21}$$

$$\Gamma_4^{t+1} = \Gamma_4^t - (\mathcal{W}^{t+1} \times_1 \Delta - \mathcal{Z}^{t+1}), \tag{3.22}$$

$$\Gamma_5^{t+1} = \Gamma_5^t - (\mathcal{S}^{t+1} - \mathcal{W}^{t+1}). \tag{3.23}$$

The pseudocode for the proposed algorithm, GLOSS, is given in Algorithm 3.1. The optimization for LOSS can be similarly computed without the updates on graph regularization and related variables $\{\mathfrak{G}\}, \{\Gamma_3\}$.

Algorithm 3.1: GLOSS

Input: $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4}$, Ω, Φ, parameters λ, γ, θ, $\{\psi\}$, β_1, β_2, β_3, β_4, β_5, max_iter.

Output: $\mathcal{L}$: Low-rank tensor; $\mathcal{S}$: Sparse tensor.

Initialize $\mathcal{S}^0 = 0$, $\mathcal{Z}^0 = 0$, $\mathfrak{L}^{n,0} = 0$, $\mathfrak{G}^{n,0} = 0$, $\Gamma_1^0 = 0$, $\Gamma_2^{n,0} = 0$, $\Gamma_3^{n,0} = 0$, $\Gamma_4^0 = 0$, $\Gamma_5^0 = 0$, $\forall i \in \{1, \ldots, 4\}$.

$W_{inv} \leftarrow (\beta_5 \mathbf{I} + \beta_4 \Delta^\top \Delta)^{-1}$

$G_{inv} \leftarrow (2\theta \Phi^n + \beta_3 \mathbf{I})^{-1}$

for $t = 0$ **to** max_iter **do**

$\mathcal{T}_1 \leftarrow \mathcal{Y} - \mathcal{S}^t + \Gamma_1^t$

$\mathcal{T}_2 \leftarrow \sum_{n=1}^4 \mathfrak{L}^{n,t} - \Gamma_2^{n,t}$

$\mathcal{T}_3 \leftarrow \sum_{n=1}^4 \mathfrak{G}^{n,t} + \Gamma_3^{n,t}$

$\mathcal{P}_\Omega[\mathcal{L}^{t+1}] \leftarrow \mathcal{P}_\Omega \left[\beta_1 \mathcal{T}_1 + \beta_2 \mathcal{T}_2 + \beta_3 \mathcal{T}_3\right] / (\beta_1 + 4(\beta_2 + \beta_3))$

$\mathcal{P}_{\Omega^\perp}[\mathcal{L}^{t+1}] \leftarrow \mathcal{P}_{\Omega^\perp} \left[\beta_2 \mathcal{T}_2 + \beta_3 \mathcal{T}_3\right] / 4(\beta_2 + \beta_3)$

for $n = 1$ **to** 4 **do**

$[U, \Sigma, V] \leftarrow \text{SVD}\left(\mathcal{L}^{t+1} + \Gamma_2^{n,t}\right)_{(n)}$

$\hat{\sigma}_{i,i} \leftarrow \max\left(\sigma_{i,i} - \frac{\psi_n}{\beta_2}, 0\right) \quad \forall i \in \{1, \ldots, I_n\}$

$\mathfrak{L}_{(n)}^{n,t+1} \leftarrow U \hat{\Sigma} V^\top$

$\mathfrak{G}_{(n)}^{n,t+1} \leftarrow \beta_3 G_{inv} \left(\mathcal{L}^{t+1} - \Gamma_3^{n,t}\right)_{(n)}$

end for

$\mathcal{P}_\Omega[\mathcal{T}_s] \leftarrow \mathcal{P}_\Omega \left[\frac{\beta_1(\mathcal{Y} - \mathcal{L}^{t+1} + \Gamma_1^t) + \beta_5(\mathcal{W}^t + \Gamma_5^t)}{\beta_1 + \beta_5}\right]$

$\mathcal{P}_{\Omega^\perp}[\mathcal{T}_s] \leftarrow \mathcal{P}_{\Omega^\perp}[\mathcal{W}^t + \Gamma_5^t]$

$\mathcal{P}_\Omega[\mathcal{S}^{t+1}] \leftarrow soft\_thresh(\mathcal{P}_\Omega[\mathcal{T}_s], \frac{\lambda}{\beta_1 + \beta_5})$

$\mathcal{P}_{\Omega^\perp}[\mathcal{S}^{t+1}] \leftarrow soft\_thresh(\mathcal{P}_{\Omega^\perp}[\mathcal{T}_s], \frac{\lambda}{\beta_5})$

$\mathcal{W}_{(1)}^{t+1} \leftarrow W_{inv}\left(\beta_5(\mathcal{S}^{t+1} - \Gamma_5^t)_{(1)} + \beta_4 \Delta^\top (\Gamma_4^t + \mathcal{Z}^t)_{(1)}\right)$

$\mathcal{Z}^{t+1} \leftarrow soft\_thresh(\mathcal{W}^{t+1} \times_1 \Delta - \Gamma_4^t, \frac{\gamma}{\beta_4})$

$\Gamma_1^{t+1} \leftarrow \Gamma_1^t - \mathcal{P}_\Omega[\mathcal{L}^{t+1} + \mathcal{S}^{t+1} - \mathcal{Y}]$

$\Gamma_4^{t+1} \leftarrow \Gamma_4^t - (\mathcal{W}^{t+1} \times_1 \Delta - \mathcal{Z}^{t+1})$

$\Gamma_5^{t+1} \leftarrow \Gamma_5^t - (\mathcal{S}^{t+1} - \mathcal{W}^{t+1})$

for $n = 1$ **to** 4 **do**

$\Gamma_2^{n,t+1} \leftarrow \Gamma_2^{n,t} - (\mathfrak{L}^{n,t+1} - \mathcal{L}^{t+1})$

$\Gamma_3^{n,t+1} \leftarrow \Gamma_3^{n,t} - (\mathcal{L}^{t+1} - \mathfrak{G}^{n,t+1})$

end for

end for

3.3.3 Computational Complexity

Let $\mathcal{Y}$ be a mode N tensor with dimensions $I_1 = I_2 = \cdots = I_N = I$. The computational complexity of each iteration of ADMM is computed as follows:

1. The update of $\mathcal{L}$ only involves element-wise operations, thus the contribution to computational complexity is not significant.

2. The computational complexity of updating each $\mathfrak{Q}^n$ is $O\left(I^{2N-1}\right)$. Thus, the total computational complexity is $O(NI^{2N-1})$. However, it is possible to reduce the effect of this cost by parallelizing the updates across modes, hence the complexity becomes quadratic in the number of elements, i.e. $O\left(I^{2N-1}\right)$.

3. The update of each $\mathfrak{G}^n$ requires the computation of a matrix inverse with complexity $O(I^3)$ and a matrix multiplication with complexity $O(I^{N+1})$. As mentioned before, the inverse always exists and can be computed outside the loop. Similar to the updates of $\mathfrak{Q}^n$, the total complexity is $O(NI^{N+1})$. This can be reduced by parallelizing across modes.

4. The second update requires a soft thresholding, which has linear complexity, *i.e.*, $O(I^N)$.

5. The computational complexity of updating $\mathcal{W}$ is governed by matrix multiplication resulting in $O(I^{N+1})$ complexity.

6. The update of $\mathcal{Z}$ consists of a matrix product followed by soft thresholding, which results in a total complexity of $O((I+1)I^N)$.

7. The updates of the dual variables do not require any additional multiplication operations. Thus, the computational complexity is negligible.

It can be concluded that the complexity of each loop is governed by the updates of $\mathfrak{Q}^n$. Therefore, the total computational complexity of the algorithm is $O(\text{max\_iter}NI^{2N-1})$. Since the complexity is driven by nuclear norm minimization, in the following section, we propose another method which utilizes graph total variation regularization for low-rank approximation instead of nuclear norm.

3.4 Low-rank On Graphs Plus Temporally Smooth Sparse Decomposition

As mentioned in Section 1.3.1, we will approximate the low-rank tensor, $\mathcal{L}$, through N graph total variation terms corresponding to each mode similar to FRPCAG in (1.14). To this end, the first J_n eigenvectors of Φ^n, $\hat{P}_n$ corresponding to the J_n lowest eigenvalues, are used to quantify the total variation of the low-rank tensor across mode-n with respect to its corresponding similarity graph. As these first J_n eigenvectors capture the low-frequency information of the signal, they can capture the normal activity in the data. Thus, the optimization problem can be written as:

$$\begin{aligned} \min_{\mathcal{L},\mathcal{S}} \theta \sum_{n=1}^{N} \operatorname{tr}\left(\mathcal{L}_{(n)}^{\top} \hat{\Phi}^n \mathcal{L}_{(n)}\right) + \lambda\|\mathcal{S}\|_1 + \gamma\|\mathcal{S} \times_1 \Delta\|_1, \\ s.t. \quad \mathcal{P}_\Omega[\mathcal{Y}] = \mathcal{P}_\Omega[\mathcal{L} + \mathcal{S}], \end{aligned} \tag{3.24}$$

where $\hat{\Phi}^n = \hat{P}_n \hat{\Lambda}_n \hat{P}_n^{\top}$ and $\hat{\Lambda}_n \in \mathbb{R}^{J_n \times J_n}$ is the leading principal submatrix of Λ_n. If we define the projections of each mode-n unfolding of $\mathcal{L}$ to the graph eigenvectors (low frequency graph Fourier basis) as $\mathcal{G}_{(n)}^n = \hat{P}_n^{\top} \mathcal{L}_{(n)}$, then (3.24) can be rewritten as:

$$\begin{aligned} \min_{\mathcal{L},\{\mathcal{G}\}\mathcal{S}} \theta \sum_{n=1}^{N} \operatorname{tr}\left({\mathcal{G}_{(n)}^n}^{\top} \Lambda^n \mathcal{G}_{(n)}^n\right) + \lambda\|\mathcal{S}\|_1 + \gamma\|\mathcal{S} \times_1 \Delta\|_1, \\ s.t. \quad \mathcal{P}_\Omega[\mathcal{Y}] = \mathcal{P}_\Omega[\mathcal{L} + \mathcal{S}], \qquad \mathcal{G}_{(n)}^n = \hat{P}_n^{\top} \mathcal{L}_{(n)}, \end{aligned} \tag{3.25}$$

where $\{\mathcal{G}\} := \{\mathcal{G}^1, \ldots, \mathcal{G}^N\}$. The solution to (3.25) will be called LOw-rank on Graphs plus temporally Smooth Sparse Decomposition (LOGSS).

3.4.1 Optimization

The optimization problem was solved using ADMM. We introduce auxiliary variables $\mathcal{W}$ and $\mathcal{Z}$ similar to LOSS to separate sparsity and temporal smoothness regularization. The problem is then rewritten as:

$$\begin{aligned} \min_{\mathcal{L},\{\mathcal{G}\},\mathcal{S},\mathcal{W},\mathcal{Z}} \theta \sum_{n=1}^{N} \operatorname{tr}\left({\mathcal{G}_{(n)}^n}^{\top} \Lambda^n \mathcal{G}_{(n)}^n\right) + \lambda\|\mathcal{S}\|_1 + \gamma\|\mathcal{S} \times_1 \Delta\|_1, \\ \text{s.t.} \quad \mathcal{P}_\Omega[\mathcal{Y}] = \mathcal{P}_\Omega[\mathcal{L} + \mathcal{S}], \quad \mathcal{G}_{(n)}^n = \hat{P}_n^{\top} \mathcal{L}_{(n)}, \mathcal{S} = \mathcal{W}, \quad \mathcal{Z} = \mathcal{W} \times_1 \Delta. \end{aligned} \tag{3.26}$$

The corresponding augmented Lagrangian is given by:

$$\theta \sum_{n=1}^{N} \text{tr}\left(\mathcal{G}_{(n)}^{n\top} \Lambda_n \mathcal{G}_{(n)}^{n}\right) + \lambda\|\mathcal{S}\|_1 + \gamma\|\mathcal{Z}\|_1 + \frac{\beta_1}{2}\|\mathcal{P}_\Omega[\mathcal{L}+\mathcal{S}-\mathcal{Y}] - \Gamma_1\|_F^2 +$$
$$\frac{\beta_2}{2}\|\mathcal{W} \times_1 \Delta - \mathcal{Z} - \Gamma_2\|_F^2 + \frac{\beta_3}{2}\|\mathcal{S} - \mathcal{W} - \Gamma_3\|_F^2 + \frac{\beta_4}{2}\sum_{n=1}^{N}\|\mathcal{L} - \mathcal{G}^n \times_n \hat{P}_n - \Gamma_4^n\|_F^2, \tag{3.27}$$

where $\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4^n \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4}$ are the Lagrange multipliers. Using (3.27) each variable can be updated alternately.

1. $\mathcal{L}$ update: The update of low-rank variable $\mathcal{L}$ is given by:

$$\mathcal{P}_\Omega[\mathcal{L}^{t+1}] = \mathcal{P}_\Omega\left[\beta_1\mathcal{T}_1 + \beta_4\mathcal{T}_2\right]/(\beta_1 + 4\beta_4),$$
$$\mathcal{P}_{\Omega^\perp}[\mathcal{L}^{t+1}] = \mathcal{P}_{\Omega^\perp}[\mathcal{T}_2]/4, \tag{3.28}$$

where $\mathcal{T}_1 = \mathcal{Y} - \mathcal{S}^t + \Gamma_1^t$, $\mathcal{T}_2 = \sum_{n=1}^{N} \mathcal{G}^{n,t} \times_n \hat{P}_n + \Gamma_4^{n,t}$.

2. $\mathcal{G}^n$ update: The variables $\mathcal{G}^n$ can be updated using:

$$\mathcal{G}^{n,t+1} = (2\frac{\theta}{\beta_4}\hat{\Lambda}_n + \mathbf{I})^{-1}(\mathcal{L}^{t+1} \times_n \hat{P}_n^\top - \Gamma_4^{n,t}), \tag{3.29}$$

where $\mathbf{I} \in \mathbb{R}^{J_n \times J_n}$ is an identity matrix.

3. $\mathcal{S}$ update: The variable $\mathcal{S}$ can be updated using:

$$\mathcal{P}_\Omega[\mathcal{S}^{t+1}] = \mathbf{T}_\lambda(\mathcal{P}_\Omega[\beta_1\mathcal{T}_3 + \beta_3\mathcal{T}_4])/(\beta_1 + \beta_3)$$
$$\mathcal{P}_{\Omega^\perp}[\mathcal{S}^{t+1}] = \mathbf{T}_{\frac{\lambda}{\beta_3}}(\mathcal{P}_{\Omega^\perp}[\mathcal{T}_4]), \tag{3.30}$$

where $\mathcal{T}_3 = \mathcal{Y} - \mathcal{L}^{t+1} + \Gamma_1^t$, $\mathcal{T}_4 = \mathcal{W}^t + \Gamma_3^t$, $\mathbf{T}_\phi(\mathbf{a}) = sign(\mathbf{a}) \odot max(|\mathbf{a}| - \phi, 0)$ and $\odot$ is Hadamard product.

4. $\mathcal{W}$ update: The auxiliary variable $\mathcal{W}$ can be updated using:

$$\mathcal{W}_{(1)}^{t+1} = W_{inv}\left(\beta_3(\mathcal{S}^{t+1} - \Gamma_3^t)_{(1)} + \beta_2\Delta^\top(\Gamma_2^t + \mathcal{Z}^t)_{(1)}\right), \tag{3.31}$$

where $W_{inv} = (\beta_3\mathbf{I} + \beta_2\Delta^\top\Delta)^{-1}$ always exists and can be computed outside the loop for faster update.

5. $\mathcal{Z}$ update: The auxiliary variable $\mathcal{Z}$, can be updated using:

$$\mathcal{Z}^{t+1} = \underset{\mathcal{Z}}{\text{argmin}}\, \gamma\|\mathcal{Z}\|_1 + \frac{\beta_2}{2}\|\mathcal{W}^{t+1} \times_1 \Delta - \mathcal{Z} - \Gamma_2^t\|_F^2, \tag{3.32}$$

which is solved by $\mathbf{T}_{\frac{\gamma}{\beta_2}}(\mathcal{W}^{t+1} \times_1 \Delta - \Gamma_2^t)$.

6. Dual updates: Finally, dual variables $\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4^n$ are updated using:

$$\Gamma_1^{t+1} = \Gamma_1^t - \mathcal{P}_\Omega[\mathcal{L}^{t+1} + \mathcal{S}^{t+1} - \mathcal{Y}], \tag{3.33}$$

$$\Gamma_2^{t+1} = \Gamma_2^t - (\mathcal{W}^{t+1} \times_1 \Delta - \mathcal{Z}^{t+1}), \tag{3.34}$$

$$\Gamma_3^{t+1} = \Gamma_3^t - (\mathcal{S}^{t+1} - \mathcal{W}^{t+1}), \tag{3.35}$$

$$\Gamma_4^{n,t+1} = \Gamma_4^{n,t} - (\mathcal{L}^{t+1} - \mathcal{G}^{n,t+1} \times_n \hat{P}_n). \tag{3.36}$$

The pseudocode for the optimization is given in Algorithm 3.2.

Algorithm 3.2: LOGSS

Input: $\mathcal{Y}$, Ω, Φ_n, parameters θ, λ, γ, β_1, β_2, β_3, β_4, max_iter.
Output: $\mathcal{L}$: Low-rank tensor; $\mathcal{S}$: Sparse tensor.
Initialize $\mathcal{S}^0 = 0$, $\mathcal{W}^0 = 0$, $\mathcal{Z}^0 = 0$, $\mathcal{G}^{n,0} = 0$, $\Gamma_1^0 = 0$, $\Gamma_2^0 = 0$, $\Gamma_3^0 = 0$, $\Gamma_4^{n,0} = 0$, $\forall i \in \{1, \ldots, 4\}$, $W_{inv} = (\beta_3 \mathbf{I} + \beta_2 \Delta^\top \Delta)^{-1}$.
for $t = 1$ **to** max_iter **do**
Update $\mathcal{L}$ using (3.28).
Update $\mathcal{G}^n$s using (3.29).
Update $\mathcal{S}$ using (3.30).
Update $\mathcal{W}$ using (3.31).
Update $\mathcal{Z}$ using (3.32).
Update Lagrange multipliers using (3.33), (3.34), (3.35) and (3.36).
end for

3.4.2 Computational Complexity of LOGSS

Assume $I_1 = I_2 = \cdots = I_N = I$. The complexity of the proposed algorithm is dominated by matrix multiplications which are the updates of $\mathcal{L}$, $\mathcal{W}$, $\mathcal{Z}$ and Γ_4^n. The updates of $\mathcal{G}^n$ require are multiplications since $\hat{\Lambda}_n$s are diagonal. The computational complexity of the matrix multiplications are: $O(NI^N)$ for the update of $\mathcal{L}$, $O(I^N)$ for the updates of $\mathcal{W}, \mathcal{Z}, \Gamma_4^n$. Since the updates of $\mathcal{L}$ and Γ_4^n can be parallelized, the complexity of the algorithm is $O(\text{max\_iter}I^N)$, hence, linear in the number of elements. Thus, using graph total variation regularization instead of nuclear norm we reduce the complexity of LOSS from quadratic to linear.

3.5 Convergence

In this subsection, we analyze the convergence of the proposed algorithms. First, we show that the proposed optimization problem (3.4) can be written as a two-block ADMM. In previous work, linear and global

convergence of ADMM is proven for two-block systems [52] with no dependence on the hyperparameters. We use this proof to derive the convergence of GLOSS. Convergence of LOSS and LOGSS follow as they have similar objective functions and similar operations can be applied to rewrite their objective as two-block ADMM.

In the following discussion, we will assume that there is no missing data, *i.e.* $\mathcal{P}_\Omega[\mathcal{Y}] = \mathcal{Y}$, to simplify the notation. Let $h(\mathcal{S}) = \lambda\|\mathcal{S}\|_1$, $j(\mathcal{Z}) = \gamma\|\mathcal{Z}\|_1$, $f(\{\mathfrak{L}\}) = \sum_{n=1}^4 \psi_n \|\mathfrak{L}^n_{(n)}\|_*$, $g(\{\mathfrak{G}\}) = \sum_{n=1}^4 \text{tr}\left(\mathfrak{G}^n_{(n)}{}^\top \Phi^n \mathfrak{G}^n_{(n)}\right)$, (3.5) can be rewritten as:

$$\begin{aligned}
&\min_{\{\mathfrak{L}\},\{\mathfrak{G}\},\mathcal{S},\mathcal{Z}} h(\mathcal{S}) + j(\mathcal{Z}) + f(\{\mathfrak{L}\}) + \theta g(\{\mathfrak{G}\}),\\
&A_1\mathcal{L}_{(1)} + A_2\mathcal{S}_{(1)} + A_3\text{cat}_1(\{\mathfrak{L}\}) + A_4\text{cat}_1(\{\mathfrak{G}\}) +\\
&A_5\mathcal{Z}_{(1)} + A_6\mathcal{W}_{(1)} = \text{cat}_1(\{\mathcal{Y}, 0, \ldots, 0\}),
\end{aligned} \tag{3.37}$$

where

$$A_1 = \begin{bmatrix} \mathbf{I} \\ \mathbf{I} \\ \mathbf{I} \\ \mathbf{I} \\ \mathbf{I} \\ \mathbf{I} \\ \mathbf{I} \\ \mathbf{I} \\ \mathbf{I} \\ 0 \\ 0 \end{bmatrix}, A_2 = \begin{bmatrix} \mathbf{I} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \mathbf{I} \\ 0 \end{bmatrix}, A_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -\mathbf{I} & 0 & 0 & 0 \\ 0 & -\mathbf{I} & 0 & 0 \\ 0 & 0 & -\mathbf{I} & 0 \\ 0 & 0 & 0 & -\mathbf{I} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, A_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\mathbf{I} & 0 & 0 & 0 \\ 0 & -\mathbf{I} & 0 & 0 \\ 0 & 0 & -\mathbf{I} & 0 \\ 0 & 0 & 0 & -\mathbf{I} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, A_5 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -\mathbf{I} \\ \Delta \end{bmatrix}, A_6 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -\mathbf{I} \end{bmatrix}.$$

From this reformulation, it is easy to see that $A_1^\top A_5 = 0$ and A_2, A_3, A_4, A_6 are all orthogonal to each other, *i.e.* $A_i^\top A_j = 0$ where $i, j \in \{2, 3, 4, 6\}$ and $i \neq j$.

In this manner, the optimization problem reduces to a special case of two-block ADMM as follows. Define variables $V_1 = [\mathcal{L}_{(1)}, \mathcal{W}_{(1)}]$ and $V_2 = [\mathcal{X}_{(1)}, \text{cat}_1(\{\mathfrak{L}\}), \text{cat}_1(\{\mathfrak{G}\}), \mathcal{Z}_{(1)}]$, and matrices $B_1 = [A_1, A_5]$ and $B_2 = [A_2, A_3, A_4, A_6]$. When we create the variable V_1, the order of updating $\mathcal{S}$ and $\mathcal{W}$ needs to change in the new formulation as A_2 and A_5 are not orthogonal. Updates of $\{\mathfrak{L}\}$ and $\{\mathfrak{G}\}$ have no effect on the update

of $\mathcal{W}$ but this is not true for $\mathcal{S}$, so updating $\mathcal{W}$ before $\mathcal{S}$ might affect the solution. However, it was proven in [194] that the change in order gives the equivalent solution if either one of the functions of the variables $\mathcal{S}$ and $\mathcal{W}$ is affine. In our formulation, this is true as the function corresponding to $\mathcal{W}$ is a constant. Thus, the problem reduces to the two-block form:

$$\min_{V_1,V_2} f_1(V_1) + f_2(V_2), \quad \text{s.t. } B_1V_1 + B_2V_2 = C, \tag{3.38}$$

where $f_1(V_1) = 0$ and $f_2(V_2) = h(\mathcal{S})+j(\mathcal{Z})+f(\{\mathfrak{L}\})+\theta g(\{\mathfrak{G}\})$ are both convex and $C = \text{cat}_1(\{\mathcal{Y}, 0, \ldots, 0\})$.

It can easily be shown using Kronecker products and vectorizations that the objective functions (3.2), (3.25) can also be converted into a two-block form. Thus, LOSS and LOGSS also converge using the above results.

3.6 Anomaly Scoring

The methods proposed in this chapter focus on extracting spatiotemporal features for anomaly detection. After extracting the features, *i.e.* the sparse part, a baseline anomaly detector can be applied to obtain an anomaly score. In this chapter, we evaluated three anomaly detection methods: Elliptic Envelope (EE) [150], Local Outlier Factor (LOF) [23] and One Class SVM (OCSVM) [153]. These three methods are used to assign an anomaly score to each element of the sparse tensor. Each method was applied to all third mode fibers which correspond to different weeks' traffic activity. This is equivalent to fitting a univariate distribution to each of the third mode fibers of the tensor. The anomaly scores were used to create an anomaly score tensor. Finally, the elements with the highest anomaly scores were selected as anomalous while the rest were determined to be normal.

3.7 Experiments

In this chapter, we evaluated the proposed method on both real and synthetic datasets. We compared our method to regular HoRPCA and weighted HoRPCA (WHoRPCA) where the nuclear norm of each unfolding is weighted. In addition to Tucker based algorithms we compare with two CP based anomaly detection methods: low rank plus sparse CP (LRSCP) [88] and Bayesian augmented tensor factorization (BATF) [40]. In the case of LRSCP, a low rank plus sparse CP model is used for anomaly detection. While the original algorithm is implemented in an online manner, in this chapter, it is modified to be applicable to the whole ST data for comparability against other methods. BATF is a CP based tensor completion/imputation model with smoothness constraints designed for urban traffic data. BATF utilizes bias vectors for all mode-n slices

which enforces smoothness across each mode. To evaluate the effect of graph regularization term in (3.4), we also compared with LOSS corresponding to the objective function in (3.2). As our method is focused on feature extraction for anomaly detection, we also compared our method to baseline anomaly detection methods such as EE, LOF and OCSVM applied to the original tensor. After the feature extraction stage, unless noted otherwise, such as "GLOSS-LOF", EE was used as the default anomaly scoring method for all tensor feature extraction methods, e.g., HoRPCA, WHoRPCA, BATF, etc. LRSCP scores the anomalies by ordering the magnitude of the elements of the sparse tensor [88]. The number of neighbors for LOF is selected as 10 as this is the suggested lower-bound in [207]. The outlier fraction of OCSVM is set to 0.1 as only the anomaly scores, not labels, generated by OCSVM are used in the experiments. The methods used for comparison and their properties are summarized in Table 3.1.

Table 3.1: Properties of anomaly detection methods used in the experiments. The acronyms refer to the different attributes of the cost function: (LR) low-rank, (SP) sparse, (WLR) weighted low-rank, (SR) smoothness regularization.

		LR	SP	WLR	SR
Tucker Based	GLOSS	+	+	+	+
	LOGSS	-	+	-	+
	LOSS	+	+	+	+
	WHoRPCA	+	+	+	-
	HoRPCA	+	+	-	-
CP Based	BATF	+	+	-	+
	LRSCP	+	+	-	-
	EE	N/A	N/A	N/A	N/A
	LOF	N/A	N/A	N/A	N/A
	OCSVM	N/A	N/A	N/A	N/A

For each data set, a varying K percent of the elements with highest anomaly scores are determined to be anomalous. With varying K, ROC curves were generated for synthetic data and the mean area under the curve (AUC) was computed for 10 random experiments. For real data, number of detected events were reported for varying K.

The number of neighbors for LOF is selected as 10 as this is the suggested lower-bound in [207]. Finally, the outlier fraction of OCSVM is set to 0.1 as only the anomaly scores, not labels, generated by OCSVM is used in the experiments.

3.7.1 Data Description

To evaluate the proposed framework, we use two publicly available datasets as well as synthetic data.

Real Data:

The first dataset is NYC yellow taxi trip records[1] for 2018. This dataset consists of trip information such as the departure zone and time, arrival zone and time, number of passengers, tips for each yellow taxi trip in NYC. In the following experiments, we only use the arrival zone and time to collect the number of arrivals for each zone aggregated over one hour time intervals. We selected 81 central zones to avoid zones with very low traffic [205]. Thus, we created a tensor $\mathcal{Y}$ of size $24 \times 7 \times 52 \times 81$ where the first mode corresponds to hours within a day, the second mode corresponds to days of a week, the third mode corresponds to weeks of a year and the last mode corresponds to the zones. The data is suitable for low-rank on graphs model as graph stationarity measure $s_r(\Gamma)$, given in Section 1.3.1, for each mode is $0.83, 0.98, 0.99, 0.56$, respectively. This implies that the data is mostly low-rank on the temporal modes as there is strong correlation among the different days, hours and weeks, while it is less low-rank across space.

The second dataset is Citi Bike NYC bike trips[2] for 2018. This dataset contains the departure and arrival station numbers, arrival and departure times and user id. In our experiments, we aggregated bike arrival data for taxi zones imported from the NYC yellow taxi trip records dataset, instead of using the original stations, to reduce the dimensionality and to avoid data sparsity. The resulting data tensor is of size $24 \times 7 \times 52 \times 81$.

[1] https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page
[2] https://www.citibikenyc.com/system-data

Synthetic Data Generation:

Evaluating urban events in a real-world setting is an open challenge, since it is difficult to obtain urban traffic data sets with ground truth information, *i.e.* anomalies are not known *a priori* and what constitutes an anomaly depends on the nature of the data and the specific problem. To be able to evaluate our method quantitatively, we generate synthetic data and inject anomalies. Following [205], we generated a synthetic data set by taking the average of the NYC taxi trip tensor, $\mathcal{Y}$, along the third mode, *i.e.* across weeks of a year. We then repeat the resulting three-mode tensor such that for each zone, average data for a week is repeated 52 times. We multiply each element of the tensor by a Gaussian random variable with mean 1 and variance 0.5 to create variation across weeks.

We generate anomalies on randomly selected $m\%$ of the first mode fibers. For each fiber, we set a random time interval of length l, which corresponds to l hours in a day, as anomalous. We multiply the average value of each randomly selected anomalous interval by a parameter c and the modify the entries by adding or subtracting this value from the interval. When c is low, the anomalies will be harder to detect and may be perceived as noise.

3.7.2 Parameter Selection:

In this section, we will discuss how the different parameters in (3.4) are selected. Following [70], we set $\lambda = 1/\sqrt{\max(I_1, \ldots, I_N)}$ for HoRPCA and $\beta_1 = \frac{1}{5\text{std}(vec(\mathcal{Y}))}$, where $vec(\mathcal{Y}) \in \mathbb{R}^{I_1 I_2 \ldots I_N}$ is the vectorization of $\mathcal{Y}$ and std(.) is the standard deviation. The other β parameters for all methods are set to be the same as β_1. The selection of β parameters does not affect the algorithm performance but changes the convergence rate as mentioned in Section 3.5. In GLOSS, the neighboorhood size for each of the k-NN graphs is selected to be $k = log(\sum_{i=1}^{N} I_n)$ following [177]. The σ value in k-NN graphs is selected to be proportional to the Frobenius norm of each mode to ensure similar density levels for all graphs. The ranks for LRSCP and BATF are selected from $\{1, 2, \ldots, 11\}$ as the rank with the best result. Increasing the rank to higher values does not improve the results while increasing complexity.

An important observation about the selection of the parameters is that depending on the data, the optimal values of hyperparameters might be different. This is due to the properties of the data such as size, variance and sparsity level. Moreover, the different hyperparameters are dependent on each other. Hence, a search over the whole parameter space may be costly. Thus, we perform a sensitivity analysis for the different

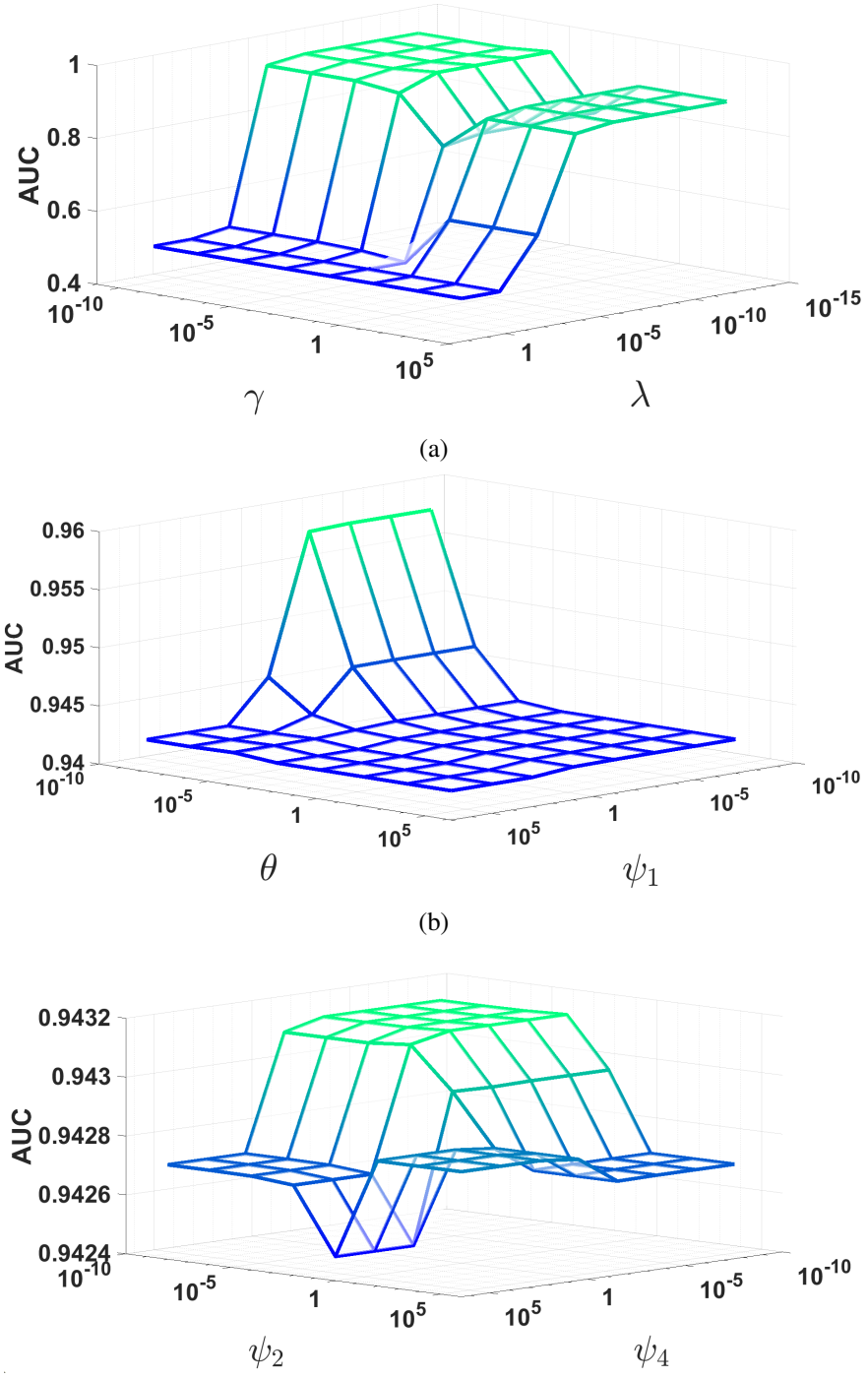


Figure 3.1: Mean AUC values for various choices of: (a) λ and γ, (b) θ and ψ_1, (c) ψ_2 and ψ_4. Mean AUC values across 10 random experiments are reported for each hyperparameter pair. For each set of experiments, the remaining hyperparameters are fixed.

hyperparameters. In Figure 3.1, we present the average AUC for various ranges of the hyperparameters for GLOSS applied on the synthetic data generated with $c = 2.5$. It can be seen from Figure 3.1a that while low values of λ always provide the best results, γ values are optimized around 10^{-5}. Increasing the sparsity penalty λ above 10^{-3} results in a sparse tensor that is mostly zero. At this λ value, AUC becomes equal to 0.5 which is equivalent to randomly guessing the anomalous points. On the other hand, when γ is too large, it smooths out all mode-1 fibers and generates the same anomaly score for each fiber. This is akin to identifying anomalous days, rather than time intervals. From these observations, it can be seen that there is not a strong dependence between γ and λ. For the weight parameters, ψ_n, when one of them is set to 1 and the others are varied across a wide range of values as shown in Figures 3.1c and 3.1b, the accuracy does not change significantly. Similarly, changing the value of θ does not seem to affect the optimal value of ψ_1.

We repeated this analysis for different c values and observed similar results indicating that the proposed method is not sensitive to the selection of hyperparameters as long as they are selected following the guidelines given below. In particular, the choice of λ affects the accuracy more than any of the other hyperparameters. This realization reduces the computational complexity of finding the best set of parameters as the search can be implemented in parallel.

Based on these empirical observations, we select $\lambda = \gamma = 1/\|\mathcal{P}_\Omega(\mathcal{Y})\|_0$, where $\|\mathcal{P}_\Omega(\mathcal{Y})\|_0$ is the number of nonzero elements of $\mathcal{Y}$ for GLOSS and $\lambda = \gamma = 1/max(I_1, \ldots, I_N)$ for LOSS and WHoRPCA similar to [70]. Since the ranks across each mode are closely related to the variance of the data within that mode, weights for each mode in the definition of nuclear norm, ψ_ns, are selected to be inversely proportional to the trace of the square root of the covariance of mode-n, i.e. $\psi_n = \frac{p}{Tr(\sqrt{\Sigma_{Y_{(n)}}})}$, where p is selected such that $\min_n(\psi_n) = 1$. The parameter θ is set to be the geometric mean of ψ_ns, i.e. $\theta = \prod_{n=1}^{4} \psi_n^{1/4}$.

3.7.3 Experiments on Synthetic Data

Effect of anomaly length and percentage:

First, we evaluated the effect of the length l and the percentage m, *i.e.* denseness, of anomalies in synthesized data. For these experiments, we set $c = 2.5$ and $P = 0\%$. From Table 3.2 and Fig. 3.2, it can be seen that as l increases, the performance of LOGSS and LOSS improves while HoRPCA's performance does not show a significant change. This is due to the fact that the temporal total variation regularization will become more suited to the observed data as the anomalies become more temporally persistent, *i.e.* when l increases.

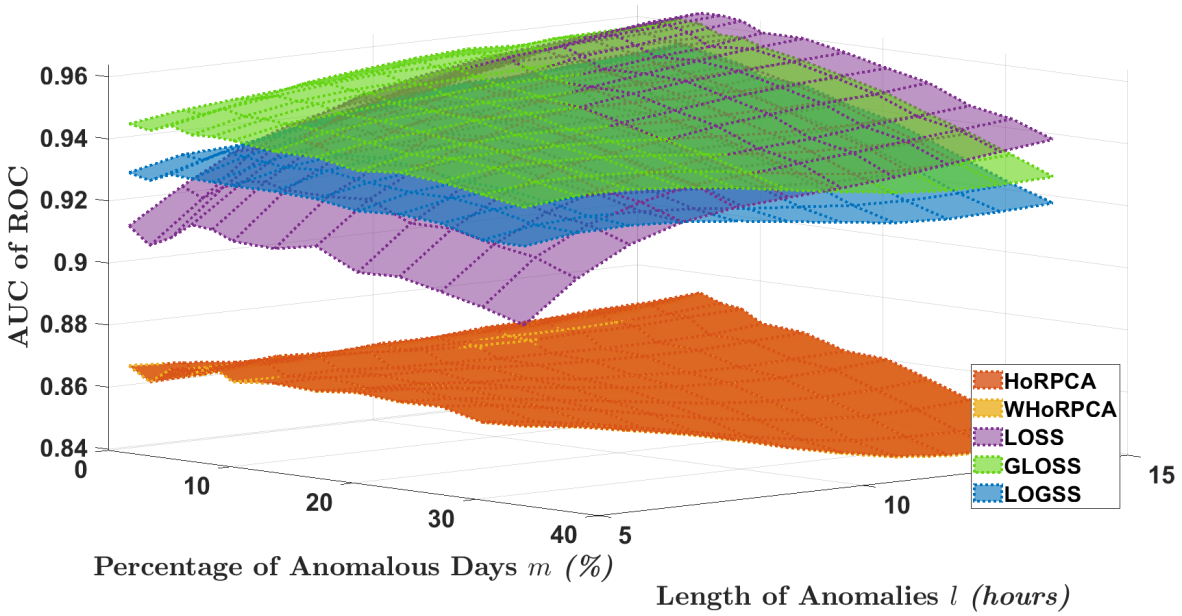


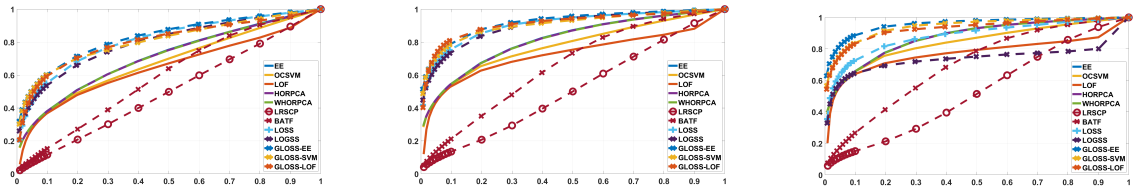
Figure 3.2: AUC of ROC w.r.t. l and m with $c = 2.5$, $P = 0\%$.

Although LOSS performs slightly better than LOGSS when l is large, for low l, it drastically underperforms which is not the case for LOGSS. With increasing m, all methods perform worse due to the assumption of sparsity for the anomalies.

Robustness against noise:

We first evaluated the effect of c, *i.e.* the strength of the anomaly, on the accuracy of the proposed method. Low c values imply that the amplitudes of anomalies are low and they may be indistinguishable from noise. From Fig. 3.3 and Table 3.2, it can be seen that for varying c values, our method (GLOSS-EE) achieves the highest AUC values compared to both baseline methods and HoRPCA, WHoRPCA, LRSCP, BATF, LOSS and LOGSS. Among the remaining methods, LOSS performs better than WHoRPCA, especially when the anomaly strength is small. CP based methods generally underperform compared to all other methods. The proposed methods have higher anomaly detection accuracy compared to EE and HoRPCA which illustrates the benefit of tailoring the optimization problem to anomaly structure. In fact, HoRPCA does not perform better than EE in most cases which means that extracting anomalies using HoRPCA does not have a significant improvement compared to using the original data. It is also important to note that the choice of the anomaly scoring method does not change the performance of GLOSS significantly.

In terms of time complexity, it can be seen from Table 3.3 that LOGSS is up to 10 times faster than both GLOSS and LOSS. BATF, on the other hand, has higher run time compared to the proposed methods



(a) (b) (c)

Figure 3.3: ROC curves for various amplitudes of anomalies. Higher amplitude means more separability. c = (a) 1.5, (b) 2, (c) 2.5. ($P = 0\%$, $l = 7$, $m = 2.3\%$)

(LOSS and GLOSS) with lower accuracy. Although other methods such as EE, LOF, LRSCP and HoRPCA outperform LOSS and GLOSS, the proposed methods outperform them in anomaly detection accuracy as mentioned earlier.

Table 3.2: Mean and standard deviation of AUC values for for various c and P. On experiments of each variable, the rest of the variables are fixed at $c = 2.5$, $P = 0\%$, $l = 7$ and $m = 2.3\%$. The proposed methods, outperform the other algorithms in all cases significantly with $p < 0.001$.

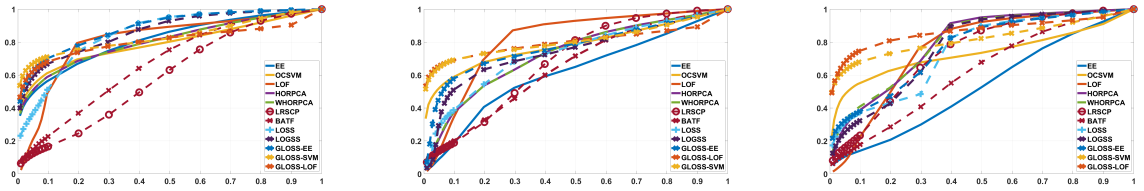
	$c = 1.5$	$c = 2$	$c = 2.5$	$P = 20\%$	$P = 40\%$	$P = 60\%$
EE	0.70±0.004	0.81±0.004	0.87±0.003	0.81±0.004	0.61±0.008	0.53±0.01
LOF	0.68±0.003	0.78±0.004	0.84±0.004	0.79±0.005	0.78±0.005	0.73±0.007
OCSVM	0.65±0.005	0.73±0.004	0.77±0.005	0.81±0.004	0.81±0.005	0.76±0.006
HoRPCA	0.7±0.004	0.81±0.004	0.87±0.003	0.8±0.004	0.73±0.006	0.79±0.005
WHoRPCA	0.7±0.004	0.81±0.004	0.87±0.003	0.8±0.003	0.73±0.006	0.8±0.005
LRSCP	0.51±0.02	0.53±0.03	0.54±0.05	0.61±0.007	0.7±0.01	0.8±0.006
BATF	0.59±0.005	0.65±0.005	0.69±0.005	0.67±0.005	0.64±0.006	0.61±0.006
LOSS	0.81±0.005	0.9±0.004	0.94±0.0035	0.85±0.003	0.73±0.009	0.74±0.009
LOGSS	0.8 ± 0.005	0.9±0.003	0.94±0.002	0.86±0.004	0.74±0.008	0.76±0.005
GLOSS-EE	**0.83±0.005**	**0.92±0.003**	**0.95±0.002**	**0.88±0.004**	0.77±0.008	0.76±0.006
GLOSS-SVM	0.81±0.006	0.9±0.004	0.95±0.002	0.83±0.006	**0.81±0.007**	0.82±0.007
GLOSS-LOF	0.8±0.006	0.91±0.004	**0.95±0.002**	0.81±0.01	0.80±0.007	**0.86±0.007**

Table 3.3: Mean and standard deviation of run times (seconds) for various methods.

EE	LOF	OCSVM	HoRPCA	WHoRPCA	LRSCP	BATF	LOSS	LOGSS	GLOSS
12.1 ± 0.3	17.0 ± 0.3	1016.2 ± 56.1	7.5 ± 0.4	13.5 ± 0.5	0.66 ± 0.17	65.8 ± 0.16	44.7 ± 2.7	**5.0±0.27**	46.7 ± 1.5

Robustness against missing data:

In addition to injecting synthetic anomalies, we also remove varying number of days at random from the tensor to evaluate the robustness of the proposed method to missing data. After generating the synthetic data, a percentage P of the mode-1 fibers is set to zero to simulate missing data, where the number of mode-1 fibers is equal to the total number of days, *i.e.* $7 \times 52 \times 81$. The accuracy of anomaly detection for varying



(a) (b) (c)

Figure 3.4: ROC curves for varying percentage of missing data, (a) $P = 20\%$, (b) $P = 40\%$, (c) $P = 60\%$. ($c = 2.5$, $l = 7$, $m = 2.3\%$)

levels of missing data is illustrated in Fig. 3.4 and the corresponding AUC values (mean ± std) are given in Table 3.2.

While the performance of all the methods degrades with increasing levels of missing data, GLOSS provides the best anomaly detection performance and is robust against missing data compared to the rest of the methods. It is interesting to see that with increasing percentage of missing data, the performance of OCSVM does not degrade too much, and some of the methods such as HoRPCA, WHoRPCA, LRSCP, GLOSS-SVM and GLOSS-LOF have a better performance. This phenomenon occurs partially due to increasing percentage of anomalous points. As some of the false positives are replaced by missing data, and anomalous points are higher in percentage, newly identified data might have more true positives with increasing percentage of missing data. Incorporating temporal smoothness for the anomalies in the objective function lowers the false detection rate by penalizing instantaneous changes in traffic that do not constitute an actual anomaly. We note that while LOSS performs comparable to GLOSS for varying anomaly strength, the performance of LOSS degrades quickly with increasing missing data. Therefore, even though both WHoRPCA and LOSS are equipped to handle missing data, GLOSS is more robust as it uses side information in the form of similarity graphs. Similar to the previous experiments, LOGSS provide the best results in terms of computational efficiency with the expense of a small amount of AUC compared to GLOSS.

3.7.4 Experiments on Real Data

To evaluate the performance of the proposed methods on real data, we compiled a list of 20 urban events, which are listed in Table 3.4, that took place in the important urban activity centers such as city squares, parks, museums, stadiums and concert halls, during 2018. We used the same set of urban events for both taxi and bike data. To detect the activities, top-K percent, with varying K, of the highest anomaly scores of the

Table 3.4: Events of Interest for NYC in 2018.

Event Name	Location	Date and Time
New Year's	Times Square	1/1 12-2AM
Blackhawks vs. Rangers	Madison Square Garden	1/3 4-8PM
Armory Show	Piers 92/94	1/14 9AM-5PM
Woman's March	Central Park West	1/20 8AM-12PM
Big Ten Basketball Final	Madison Square Garden	3/4 3-10PM
Big East Quarter Finals	Madison Square Garden	3/8 3-10PM
St. Patricks Day Parade	5th Avenue, btw 44th and 79th	3/17 11AM-5PM
Nor'easter Storm	Citywide	3/20 11AM-5PM
NIT Quarterfinal Utah vs. St.Mary's	Madison Square Garden	3/21 5 PM- 10PM
U2 Concert	Madison Square Garden	7/1 5-10PM
July 4th Celebrations	Citywide	7/4 5-11PM
UN General Assembly	United Nation Headquarters	9/25 12-5PM
Comic Con	Javits Center	11/4 8 AM-3PM
NYC Marathon	Colombus Circle	11/04 12-5PM
Elton John Concert	Madison Square Garden	11/9 7-11PM
Macy's Thanksgiving Parade	Herald Square	11/22 9PM-12AM
Christmas Tree Lighting	Bryant Park	12/4 7PM-12AM
Golden Knights vs. Rangers	Madison Square Garden	12/16 12-3PM
Phish Concert	Madison Square Garden	12/28 4-8PM
New Year's Eve	Times Square	12/31 8PM-12AM

extracted sparse tensors are selected as anomalies and compared against the compiled list of urban events. In previous work, similar case studies were presented for experiments on real data [206, 39, 203, 205].

Detection performance for all methods is given in Tables 3.5 and 3.6 for the taxi and bike data, respectively. From Table 3.5, it can be seen that anomaly scoring methods applied to the spatiotemporal features extracted by GLOSS perform the best for the NYC Taxi data. The performance of GLOSS is followed by LOSS as temporal smoothness allows for detection of events at lower K by removing anomalies resulting from noise. LOSS performs the best initially but as more points are considered GLOSS finds more cases. Although LOGSS performs better than baseline methods most of the time it does not perform as good as GLOSS or LOSS. In the case of CP based methods, although the performance of LRSCP is not good, BATF shows results competitive with GLOSS especially for Taxi data. Although BATF extracts the anomalies well, it has a high computational complexity. Among the baseline methods, EE performs the best while LOF performs the worst. However, when the features extracted from GLOSS are input to LOF and EE, their performances

become very similar. This shows that GLOSS is effective at separating anomalous entries from noise and normal traffic activity and thus, improves the performance of both LOF and EE. It is important to note that most of the anomalies cannot be detected at low K values by most methods because events such as New Year's Eve or July 4th celebrations change the activity pattern in the whole city and constitute the majority of the anomalies detected at low K values for Taxi Data.

The performance of all methods is significantly reduced in Bike Data as can be seen from Table 3.6. This is because Bike Data is very noisy with a large number of days, or points that would be considered anomalous. Also, some of the selected events do not produce significant changes in Bike Data such as New Year's Eve as usage of bikes at midnight is low even though it's a significant event for taxi traffic. Changes in the weather also affect the performance by increasing the variance of the data, especially across the third mode, which corresponds to the weeks of the year. We see some fluctuations in the performance comparisons as well, such as LOGSS performing better than all methods at higher K, *i.e.* percentages. Still, the proposed methods are better than HoRPCA, WHoRPCA, and baseline methods overall, which shows the improvements brought by the extracted features.

It is important to note that event selection is done manually and the selected events may not correspond to the most significant anomalies. Thus, although it is a widely utilized tool in analyzing the performance on real data, the case study approach might not reflect the true performance of the anomaly detection method as effectively as synthetic data.

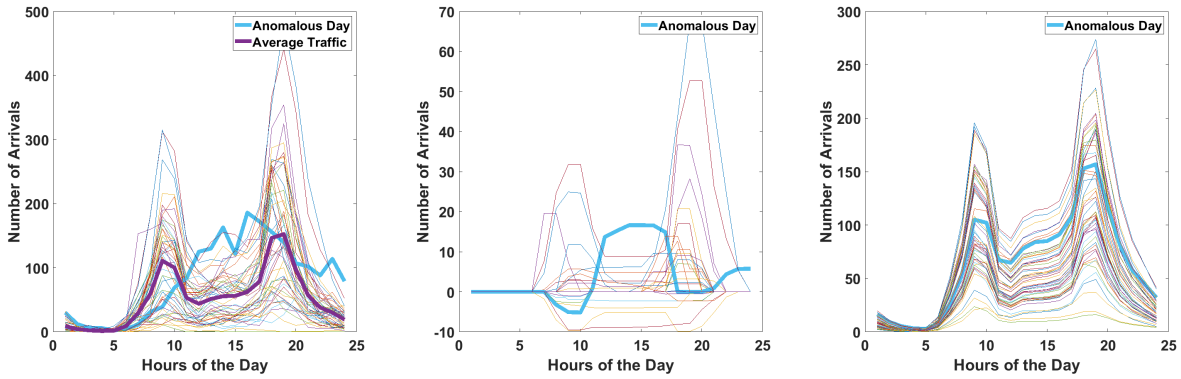
Table 3.5: Results for 2018 NYC Yellow Taxi Data. Columns indicate the percentage of selected points with top anomaly scores. The table entries correspond to the number of events detected at the corresponding percentage.

%	0.014	0.07	0.14	0.3	0.7	1	2	3
EE	0	0	1	3	9	9	16	18
LOF	0	0	0	1	1	2	4	5
OCSVM	0	0	2	5	8	11	15	16
HoRPCA	0	5	11	14	**20**	**20**	**20**	**20**
WHoRPCA	0	0	1	3	9	9	16	18
LRSCP	0	1	1	1	5	7	10	11
BATF	0	1	5	9	14	15	**20**	**20**
LOSS	**3**	**9**	12	**15**	16	17	19	**20**
LOGSS	0	0	1	6	10	12	18	18
GLOSS-EE	1	8	**13**	**15**	18	18	19	**20**
GLOSS-LOF	2	8	12	14	18	18	19	19
GLOSS-SVM	0	3	3	9	17	18	**20**	**20**

In Fig. 3.5, we illustrate the bike data for July 4th at Hudson River banks, and the low-rank and sparse

Table 3.6: Results on 2018 NYC Bike Trip Data.

%	0.3	1	2	3	4.2	7	9.7	12.5
EE	0	0	0	1	2	2	2	3
LOF	1	1	1	1	2	2	4	6
OCSVM	0	0	1	2	2	2	3	3
HoRPCA	**2**	**3**	4	4	6	8	11	14
WHoRPCA	1	1	2	7	**11**	12	12	12
LRSCP	0	1	2	3	3	4	6	10
BATF	2	3	4	6	9	10	11	13
LOSS	0	1	1	1	4	13	**15**	16
LOGSS	1	**3**	4	6	9	11	14	**17**
GLOSS-EE	0	0	3	7	9	11	**15**	16
GLOSS-LOF	1	2	2	2	2	6	8	11
GLOSS-SVM	1	2	**5**	**8**	10	**15**	**15**	15



(a) (b) (c)

Figure 3.5: Bike Activity data, the extracted sparse part and low-rank part across for July 4th Celebrations at Hudson River banks. (a) Real Data where the traffic for 52 Wednesdays is shown along with the traffic on Independence Day and average traffic; (b) Sparse tensor where the curve corresponding to the anomaly is highlighted; (c) Low-rank tensor with the curve corresponding to the Independence Day highlighted.

parts extracted by GLOSS. It can be seen that as the data varies across different weeks, the low-rank part can explain this variance well by fitting a pattern to days with varying amplitudes. Thus, the proposed method does not get affected by events such as the weather as it can capture both low and high traffic days in the low-rank part which can be seen in Fig 3.5c. The deviations from the daily pattern, rather than the actual traffic volume, is captured by the sparse part, which is then input to the anomaly scoring algorithms. Thus, our method is able to extract the events at a fairly low K.

3.8 Conclusions

In this chapter, we proposed robust tensor decomposition based anomaly detection methods for urban traffic data. The proposed methods extract a low-rank component using a weighted nuclear norm and imposes the sparse component to be temporally smooth to better model the anomaly structure. In one of the methods, graph regularization is employed to preserve the geometry of the data and to account for nonlinearities in the anomaly structure. In another, low-rank tensor recovery is implemented through minimizing graph total variation on similarity graphs constructed across each mode. This approximation circumvents the need for computing a computationally expensive nuclear norm minimization. ADMM based computationally efficient and scalable algorithms are proposed to solve the resulting optimization problems. As the proposed methods focus on spatiotemporal feature extraction, the resulting features can be input to well-known anomaly detection methods such as EE, LOF and OCSVM for anomaly scoring.

The proposed methods are evaluated on both synthetic and real urban traffic data. Results on synthetic data illustrate the robustness of our methods to varying levels of missing data and their sensitivity to even low amplitude anomalies. In particular, our methods outperforms HoRPCA and WHoRPCA thanks to temporal smoothness assumption on the sparse part. Moreover, the graph regularization improves the accuracy further by ensuring that the low-rank projections preserve local geometry of the data. In real data, our methods begin to detect anomalies earlier, *i.e.* the top anomaly scores usually correspond to events of interest, than existing methods. GLOSS provides further improvement over LOSS as more events are detected for a given number of selected points. Furthermore, the results from real data show how the extracted sparse component highlights the anomalous activities. For both synthetic and real data, LOGSS outperforms other methods in terms of computational efficiency with some loss in AUC compared to GLOSS. Experiments on synthetic data reveal that when anomalies are longer in duration, the proposed methods perform better. LOGSS also outperforms LOSS when anomalies are shorter in duration. Although LOSS and GLOSS perform better in real data, LOGSS shows similar performance with shorter run time.

In future work, a statistical tensor anomaly scoring method will be explored instead of scoring each fiber individually by a separate algorithm. In recent years, deep learning based methods have also provided a promising new direction for anomaly detection [33, 28]. Some examples include long short-term memory (LSTM) neural network used to predict traffic flow and detect anomalies [94] and fully connected neural networks used to decompose traffic data into normal and anomalous parts [205]. Future work will consider

extensions to tensor type data and application of 3D-CNNs [33]. Another possible extension of the proposed work is online anomaly detection. The method proposed in this paper cannot achieve real-time anomaly detection as for the current application the data has been collected and stored offline. However, it is possible to extend low-rank tensor models to an online setting. Recently, methods for online tensor subspace tracking have been introduced [87, 141, 8, 110]. Our method can be extended to the online setting using the methods outlined in these papers. Applications and extensions of the proposed method on network data and other spatiotemporal data with different characteristics such as fMRI will also be considered.

CHAPTER 4

GEOMETRIC TENSOR LEARNING

4.1 Introduction

Most high-dimensional data have a low-dimensional structure and principal component analysis (PCA) is the fundamental approach to extract this low-dimensional structure. A major drawback of PCA is that it is sensitive to grossly corrupted or outlying observations, which are ubiquitous in real-world data. Robust PCA (RPCA) addresses this issue by decomposing the observed data matrix into a low-rank and sparse part [30]. However, RPCA can only handle two-way matrix data while real data is usually multi-way in nature and stored in arrays known as tensors. In recent years, different extensions of RPCA have been introduced to deal with tensor type data based on the different tensor models. Some examples include simple low-rank tensor completion (SiLRTC) [120], higher-order RPCA (HoRPCA) based on the Tucker model [70], tensor RPCA (TRPCA) based on the t-SVD model [125] and tensor-train based tensor RPCA [53, 197].

Different tensor models employ different definitions of rank and result in different interpretations of what a low-rank tensor is. Tucker rank, optimized by minimizing sum of nuclear norms (SNN), cannot appropriately capture the global correlation in a tensor as each mode represents the matrix row in an unbalanced matricization scheme [17]. Tensor tubal rank, optimized by minimizing tensor nuclear norm (TNN), on the other hand, characterizes the correlations along the first and second modes while that along the third mode is encoded by the embedded circular convolution. Moreover, definition of TNN is usually limited to third-order tensors [210]. Tensor-train rank, optimized using tensor train nuclear norm (TTNN), can capture the global correlation of all tensor entries, by providing the mean of the correlation between two sets of modes as it is based on a canonical unfolding [17].

While the robust low-rank tensor representations capture the global structure of tensor data, they do not preserve the local geometric structure. Manifold learning addresses this issue and has been successfully implemented for tensors [164, 161]. However, current manifold learning approaches typically focus on only one mode of the data. Yet for many data matrices and tensors, correlations exist across all modalities. Several recent papers [67, 156, 155, 128, 127, 157, 22] exploit this coupled relationship to co-organize matrices and infer underlying row and column embeddings.

Inspired by the success of low-rank embedding and manifold learning, in this chapter, we propose to integrate them into a unified framework for simultaneously capturing the global low-rank and the local geometric structure. In particular, we propose a graph regularized robust low-rank tensor-train decomposition. The proposed model is based on robust tensor-train decomposition introduced in [197]. Unlike previous graph regularized tensor decompositions, the proposed method introduces graph regularization across each canonical unfolding to leverage the underlying geometry on each tensor mode. The resulting optimization problem is shown to be computationally expensive due to the size of the graphs across each canonical unfolding. An equivalence between these computationally expensive graph regularization terms and regular tensor unfolding is derived and a computationally efficient implementation of graph regularized robust tensor-train decomposition is proposed.

4.2 Tensor Train Robust PCA on Graphs

Given an observed tensor with missing entries and gross corruption $\mathcal{P}_\Omega[\mathcal{Y}]$, the objective of tensor RPCA is to extract a low-rank tensor, $\mathcal{X}$, and a sparse tensor, $\mathcal{S}$, corresponding to gross outliers such that $\mathcal{P}_\Omega[\mathcal{Y}] = \mathcal{P}_\Omega[\mathcal{X} + \mathcal{S}]$. Robust tensor-train PCA model was proposed in [197] as follows:

$$\text{minimize}_{\mathcal{X},\mathcal{S}} \sum_{n=1}^{N-1} \alpha_n \|\mathcal{X}_{[n]}\|_* + \lambda \|\mathcal{S}\|_1, \qquad \text{s.t. } \mathcal{P}_\Omega[\mathcal{X} + \mathcal{S}] = \mathcal{P}_\Omega[\mathcal{Y}].$$

While nuclear norm minimization can capture the global structure of the tensor, incorporating graph regularization can capture the local geometry and non-linear structures within the data. Adding graph regularization on the mode-n canonical unfoldings of the low-rank part, $\mathcal{X}_{[n]} \in \mathbb{R}^{I_1 \dots I_n \times I_{n+1} \dots I_N}$, the modified objective can be rewritten as:

$$\underset{\mathcal{X},\mathcal{S}}{\text{minimize}} \sum_{n=1}^{N-1} \alpha_n \|\mathcal{X}_{[n]}\|_* + \sum_{n=1}^{N} \theta_n \text{tr}(\mathcal{X}_{[n]}^\top \Phi_n \mathcal{X}_{[n]}) + \lambda \|\mathcal{S}\|_1, \qquad \text{s.t.} \quad \mathcal{P}_\Omega[\mathcal{X} + \mathcal{S}] = \mathcal{P}_\Omega[\mathcal{Y}], \tag{4.1}$$

where $\Phi_n \in \mathbb{R}^{I_1 \dots I_n \times I_1 \dots I_n}$ is the mode-n graph Laplacian. The solution to (4.1) will be referred to as tensor train robust PCA with graph regularization (TTRPCA-G).

4.2.1 Kronecker Structured Graphs

For each mode n, the graph regularization proposed in (4.1) computes the similarity over all the modes from the first to the nth mode. Thus, it potentially utilizes the same local geometric information multiple times. In this section, we propose modeling each Φ_n with a Kronecker structure where $\Phi_n = \hat{\Phi}_n \otimes \mathbf{I}$ with

$\hat{\Phi}_n \in \mathbb{R}^{I_n \times I_n}$. This structure imposes a manifold on only mode n, rather than the set of all modes n' with $n' \leq n$. In the following, we show that solving such a system is equivalent to replacing the trace norm of the canonical unfolding with the trace norm of mode-n unfolding. Moreover, this structure highly reduces the computational complexity of TTRPCA-G.

Lemma 2. *Let $B \in \mathbb{R}^{I \times J}$, $A \in \mathbb{R}^{K \times I}$ and $C \in \mathbb{R}^{L \times J}$, then*

$$vec(ABC^\top) = (C \otimes A)vec(B),$$

where $vec(.)$ stacks all elements of a tensor into a vector.

Lemma 3. *Given a third-order tensor $\mathcal{B} \in \mathbb{R}^{I \times J \times \mathcal{M}}$, and its third mode slices $B_i \in \mathbb{R}^{I \times J}$,*

$$\|C\mathcal{B}_{(2)}\|_F^2 = \|(C \otimes \mathbf{I})\mathcal{B}_{[2]}\|_F^2,$$

where $C \in \mathbb{R}^{L \times J}$ is any matrix and $\mathbf{I} \in \mathbb{R}^{I \times I}$ is the identity.

Proof. Let $\mathbf{b}_i = \text{vec}(B_i) \in \mathbb{R}^{IJ \times 1}$, then,

$$\text{vec}(\mathcal{B}_{(2)}^\top C^\top) = \begin{bmatrix} \text{vec}(B_1 C^\top) \\ \text{vec}(B_2 C^\top) \\ \vdots \\ \text{vec}(B_M C^\top) \end{bmatrix} = \begin{bmatrix} (C \otimes \mathbf{I})\mathbf{b}_1 \\ (C \otimes \mathbf{I})\mathbf{b}_2 \\ \vdots \\ (C \otimes \mathbf{I})\mathbf{b}_M \end{bmatrix}$$

$$= (\mathbf{I} \otimes (C \otimes \mathbf{I}))[\mathbf{b}_1^\top, \mathbf{b}_2^\top, \ldots, \mathbf{b}_M^\top]^\top = (\mathbf{I} \otimes (C \otimes \mathbf{I}))\text{vec}(\mathcal{B}_{[2]}) = \text{vec}((C \otimes \mathbf{I})\mathcal{B}_{[2]}),$$

where the second equality follows from Lemma 1. Thus,

$$\|C\mathcal{B}_{(2)}\|_F^2 = \|\mathcal{B}_{(2)}^\top C^\top\|_F^2 = \|\text{vec}(\mathcal{B}_{(2)}^\top C^\top)\|_F^2 = \|\text{vec}((C \otimes \mathbf{I})\mathcal{B}_{[2]})\|_F^2 = \|(C \otimes \mathbf{I})\mathcal{B}_{[2]}\|_F^2. \qquad \square$$

By reshaping any tensor with N modes to third-order tensors, where mode n is mapped to the second mode , it can be shown that this result can be generalized for any mode n and order N.

Theorem 1. *Let $\Phi_n = \hat{\Phi}_n \otimes \mathbf{I}$, where $\hat{\Phi}_n \in \mathbb{R}^{I_n \times I_n}$ and $\mathbf{I} \in \mathbb{R}^{I_1 I_2 \ldots I_{n-1} \times I_1 I_2 \ldots I_{n-1}}$. The term $\sum_{n=1}^N \theta_n \text{tr}(\mathcal{X}_{[n]}^\top \Phi_n \mathcal{X}_{[n]})$ is equivalent to $\sum_{n=1}^N \theta_n \text{tr}(\mathcal{X}_{(n)}^\top \hat{\Phi}_n \mathcal{X}_{(n)})$.*

Proof. Since $\hat{\Phi}_n$ is symmetric, let $\hat{\Phi}_n = \hat{P}_n^\top \hat{P}_n$. Then $\Phi_n = (\hat{P}_n^\top \hat{P}_n) \otimes \mathbf{I} = P_n^\top P_n$, where $P_n = (\hat{P}_n \otimes \mathbf{I})$. Thus, the following equalities hold:

$$\begin{aligned}\text{tr}(\mathcal{X}_{[n]}^\top \Phi_n \mathcal{X}_{[n]}) &= \|P_n \mathcal{X}_{[n]}\|_F^2 = \|(\hat{P}_n \otimes \mathbf{I})\mathcal{X}_{[n]}\|_F^2 = \|\hat{P}_n \mathcal{X}_{(n)}\|_F^2 = \text{tr}(\mathcal{X}_{(n)}^\top \hat{P}_n^\top \hat{P}_n \mathcal{X}_{(n)}) \\ &= \text{tr}(\mathcal{X}_{(n)}^\top \hat{\Phi}_n \mathcal{X}_{(n)}),\end{aligned}$$

where the third equality follows from Lemma 2. □

By Theorem 1, the graph regularization term in (4.1) is equivalent to $\sum_{n=1}^N \theta_n \text{tr}(\mathcal{X}_{(n)}^\top \hat{\Phi}_n \mathcal{X}_{(n)})$. Thus, we can rewrite (4.1) as:

$$\underset{\mathcal{X},\mathcal{S}}{\text{minimize}} \sum_{n=1}^{N-1} \alpha_n \|\mathcal{X}_{[n]}\|_* + \sum_{n=1}^{N} \theta_n \text{tr}(\mathcal{X}_{(n)}^\top \hat{\Phi}_n \mathcal{X}_{(n)}) + \lambda \|\mathcal{S}\|_1, \quad \text{s.t.} \quad \mathcal{P}_\Omega[\mathcal{X} + \mathcal{S}] = \mathcal{P}_\Omega[\mathcal{Y}]. \tag{4.2}$$

The solution to (4.2) will be referred to as tensor train robust PCA with mode-n graph regularization (TTRPCA-nG).

4.2.2 Optimization

The objective functions (4.1) and (4.2) are optimized using an Alternating Direction Method of Multipliers (ADMM) scheme as ADMM has been previously utilized for solving similar convex problems [70, 154, 156]. In this section, we will give a detailed derivation of the update steps for optimizing (4.1). While the update steps will be similar for (4.2), we will note when they differ.

To separate the nuclear norm and graph regularization terms and to isolate functions of each mode, we introduce auxiliary variables $\mathfrak{L}^n$ and $\mathfrak{G}^n$. (4.1) is then rewritten as:

$$\begin{gathered}\underset{\{\mathfrak{L}\},\{\mathfrak{G}\},\mathcal{X},\mathcal{S}}{\text{minimize}} \sum_{n=1}^{N-1} \alpha_n \|\mathfrak{L}_{[n]}^n\|_* + \sum_{n=1}^{N} \theta_n \text{tr}(\mathfrak{G}_{[n]}^{n\top} \Phi_n \mathfrak{G}_{[n]}^n) + \lambda \|\mathcal{S}\|_1, \\ \text{s.t. } \mathcal{P}_\Omega[\mathcal{X} + \mathcal{S}] = \mathcal{P}_\Omega[\mathcal{Y}],\ \mathcal{X} = \mathfrak{L}^n,\ \mathcal{X} = \mathfrak{G}^n.\end{gathered}$$

The corresponding augmented Lagrangian is given by:

$$\begin{gathered}\sum_{n=1}^{N-1} \alpha_n \|\mathfrak{L}_{[n]}^n\|_* + \sum_{n=1}^{N} \theta_n \text{tr}(\mathfrak{G}_{[n]}^{n\top} \Phi_n \mathfrak{G}_{[n]}^n) + \lambda \|\mathcal{S}\|_1 + \\ \frac{\beta_1}{2} \|\mathcal{P}_\Omega[\mathcal{Y} - \mathcal{X} - \mathcal{S} - \Lambda_1]\|_F^2 + \frac{\beta_2}{2} \sum_{n=1}^{N-1} \|\mathcal{X} - \mathfrak{L}^n - \Lambda_2^n\|_F^2 + \frac{\beta_3}{2} \sum_{n=1}^{N} \|\mathcal{X} - \mathfrak{G}^n - \Lambda_3^n\|_F^2,\end{gathered} \tag{4.3}$$

where $\Lambda_1, \Lambda_2^n, \Lambda_3^n \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ are the Lagrange multipliers. Using (4.3), each variable can be updated iteratively.

1. $\mathcal{X}$ update: The update of low-rank variable $\mathcal{X}$ is given by:

$$\mathcal{P}_\Omega[\mathcal{X}^{t+1}] = \mathcal{P}_\Omega\left[\beta_1\mathcal{T}_1 + \mathcal{T}_2\right]/(\beta_1 + (N-1)\beta_2 + N\beta_3),$$
$$\mathcal{P}_{\Omega^\perp}[\mathcal{X}^{t+1}] = \mathcal{P}_{\Omega^\perp}[\mathcal{T}_2]/((N-1)\beta_2 + N\beta_3), \tag{4.4}$$

where $\mathcal{T}_1 = \mathcal{Y} - \mathcal{S}^t - \Lambda_1^t$, $\mathcal{T}_2 = \beta_2 \sum_{n=1}^{N-1}(\mathfrak{L}^{n,t} + \Lambda_2^{n,t}) + \beta_3 \sum_{n=1}^{N}(\mathfrak{G}^{n,t} + \Lambda_3^{n,t})$.

2. $\mathfrak{L}^n$ update: The update of $\mathfrak{L}^n$ is solved by a soft thresholding operator on singular values of $\left(\mathcal{X}^{t+1} - \Lambda_2^{n,t}\right)_{[n]}$ with a threshold of α_n/β_2.

3. $\mathfrak{G}^n$ update: The variables $\mathfrak{G}^n$ can be updated by:

$$\mathfrak{G}_{[n]}^{n,t+1} = \beta_3 G_{inv}\left(\mathcal{X}^{t+1} - \Lambda_3^{n,t}\right)_{[n]}, \tag{4.5}$$

where $G_{inv} = (2\theta_n\Phi_n + \beta_3\mathbf{I})^{-1}$ exists for any Φ_n for which the set of eigenvalues do not contain $\frac{\beta_3}{2\theta_n}$, and can be computed outside the loop for faster update.

The update rule for (4.2) is given by:

$$\mathfrak{G}_{(n)}^{n,t+1} = \beta_3 G_{inv}\left(\mathcal{X}^{t+1} - \Lambda_3^{n,t}\right)_{(n)}, \tag{4.6}$$

where $G_{inv} = \left(2\theta_n\hat{\Phi}_n + \beta_3\mathbf{I}\right)^{-1}$.

4. $\mathcal{S}$ update: The variable $\mathcal{S}$ can be updated by soft thresholding $\mathcal{P}_\Omega[\mathcal{Y} - \mathcal{X}^{t+1} - \Lambda_1^t]$ with a threshold of $\frac{\lambda}{\beta_1}$.

5. Dual updates: Finally, dual variables $\Lambda_1, \Lambda_2^n, \Lambda_3^n$ are updated using:

$$\Lambda_1^{t+1} = \Lambda_1^t + \mathcal{P}_\Omega[\mathcal{X}^{t+1} + \mathcal{S}^{t+1} - \mathcal{Y}], \tag{4.7}$$
$$\Lambda_2^{n,t+1} = \Lambda_2^{n,t} + (\mathfrak{L}^{n,t+1} - \mathcal{X}^{t+1}), \tag{4.8}$$
$$\Lambda_3^{n,t+1} = \Lambda_3^{n,t} + (\mathfrak{G}^{n,t+1} - \mathcal{X}^{t+1}). \tag{4.9}$$

The algorithms for both TTRPCA-G and TTRPCA-nG are outlined in Algorithm 4.1.

4.2.3 Computation and Memory Complexity for Graphs

In (4.1), the size of each Φ_n is $\prod_{i=1}^n I_i \times \prod_{i=1}^n I_i$. Thus, the memory requirement for TTRPCA-G is $O(I^2)$, where $I = \prod_{i=1}^N I_i$. On the other hand, TTRPCA-nG requires only $O(I_n^2)$ parameters for each $\hat{\Phi}_n$.

Algorithm 4.1: TTRPCA-G/nG

Input: $\mathcal{Y}$, Ω, Φ_n, parameters θ_n, α_n, γ, β_1, β_2, β_3, T.
Output: $\mathcal{X}$: Low-rank tensor; $\mathcal{S}$: Sparse tensor.
Initialize $\mathcal{S}^0 = 0$, $\mathfrak{L}^{n,0} = 0$, $\mathfrak{G}^{n,0} = 0$, $\Lambda_1^0 = 0$, $\Lambda_2^{n,0} = 0$, $\Lambda_3^{n,0} = 0$, $\forall n \in \{1, \ldots, N\}$,
$G_{inv} \leftarrow (2\theta_n \Phi_n + \beta_3 \mathbf{I})^{-1}$ (TTRPCA-G) or
$G_{inv} \leftarrow \left(2\theta_n \hat{\Phi}_n + \beta_3 \mathbf{I}\right)^{-1}$ (TTRPCA-nG).
for $t = 1$ **to** T **do**
Update $\mathcal{X}$ using (4.4).
Update $\mathfrak{L}^n$s using optimization step 2.
Update $\mathfrak{G}^n$s using (4.5) (for TT-PCA-G) or (4.6) (for TT-PCA-nG).
Update $\mathcal{S}$ using optimization step 4.
Update Lagrange multipliers using (4.7), (4.8) and (4.9).
end for

Computation of G_{inv} in (4.5), for $n = N$ has $O\left(I^4\right)$ complexity. On the other hand, the computational complexity of G_{inv} in TTRPCA-nG is $O(I_n^2)$. Overall, the computational complexities for TTRPCA-G and TTRPCA-nG are $O(I^4)$ and $O(I^{3/2})$, respectively.

4.3 Experiments

The proposed method was compared against Tucker based HoRPCA [70] and TT based TTRPCA [197] on data completion and denoising tasks, on both synthetic and real tensor data [1]. The results are reported in terms of peak signal to noise ratio (PSNR), structural similarity index measure (SSIM), and residual squared error (RSE), i.e., $\frac{\|\hat{\mathcal{Y}} - \mathcal{Y}_0\|_F}{\|\mathcal{Y}_0\|_F}$, where $\mathcal{Y}_0$ corresponds to the true underlying low-rank data.

Table 4.1: Denoising performance for synthetic data against varying levels of gross noise $c\%$ for various methods.

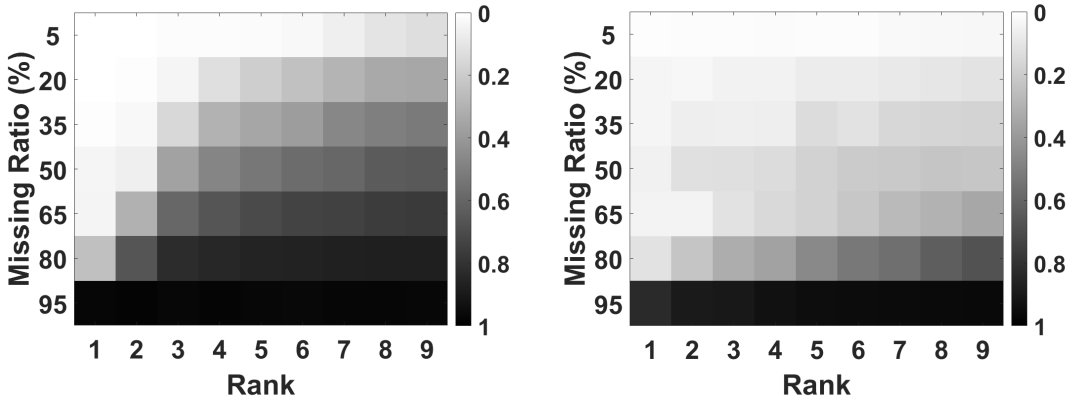
	5			20			35			50		
	RSE	PSNR	SSIM	RSE	PSNR	SSIM	RSE	PSNR	SSIM	RSE	PSNR	SSIM
Observed	1.95	17.78	0.71	3.91	11.73	0.24	5.18	9.3	0.07	6.22	7.71	0.01
HoRPCA	0.44	31.15	0.86	0.63	28.12	0.59	0.86	25.42	0.29	0.98	24.29	0.26
TTRPCA	0.07	47.52	**0.99**	0.23	36.86	0.96	0.44	31.22	0.81	0.93	24.77	0.30
TTRPCA-G	**0.04**	50.06	**0.99**	**0.15**	40.08	**0.97**	**0.33**	32.82	**0.87**	**0.86**	**25.67**	**0.35**
TTRPCA-nG	**0.04**	**50.11**	**0.99**	**0.15**	**40.07**	**0.97**	**0.33**	**32.81**	**0.87**	0.87	**25.67**	**0.35**

Following [17], we set $\alpha_k = \frac{\delta_k}{\sum_{k'=1}^{N-1} \delta_{k'}}$, where $\delta_k = \min(\prod_{n=1}^{k} I_n, \prod_{n=k+1}^{N-1} I_n)$. θ_ks are set proportional to the size of the corresponding mode $\frac{I_k}{\prod_{k'=1}^{N} I'_k}$ for TTRPCA-nG, and are set proportional to α_k for TTRPCA-G. The remaining parameters are optimized for all methods such that the best results for each method are reported.

[1] You can find our code in github.com/mrsfgl/ttrpca_g

Table 4.2: Denoising performance for real data against varying levels of gross noise $c\%$ for various methods.

	5			20			35			50		
	RSE	PSNR	SSIM	RSE	PSNR	SSIM	RSE	PSNR	SSIM	RSE	PSNR	SSIM
Observed	0.414	17.57	0.785	0.8276	11.56	0.392	1.09	9.13	0.205	1.306	7.59	0.11
HoRPCA	0.105	29.53	0.928	0.211	23.42	0.791	0.288	20.74	0.667	0.383	18.25	0.53
TTRPCA	0.106	29.42	**0.929**	0.199	23.49	0.83	0.277	21.07	0.747	0.378	18.37	0.63
TTRPCA-nG	**0.103**	**29.89**	0.926	**0.164**	**25.63**	**0.852**	**0.237**	**22.42**	**0.77**	**0.331**	**19.52**	**0.65**



(a) HoRPCA (b) TTRPCA

(c) TTRPCA-G (d) TTRPCA-nG

Figure 4.1: Phase diagrams for missing data recovery.

4.3.1 Synthetic Data

Tensors with TT structure were generated by simulating each tensor factor $\mathcal{U}_n \forall n \in \{1, \ldots, N\}$, as i.i.d. Gaussian with mean 0 and covariance $\mathbf{I}$, and merging them. For the sake of simplicity, all modes have the same size and rank, i.e., $I_n = I, \forall n$, $r_n = r, \forall n \neq N$. In our experiments, N is selected to be 4 and $I = 10$, i.e., $\mathcal{Y} \in \mathbb{R}^{10\times10\times10\times10}$ and r is varied in the range of 1 to 9.

After generating synthetic data, we simulate missing data by setting $m\%$ of the entries to zero. In

Fig. 4.1, we demonstrate the phase diagrams for tensor completion with various methods. By varying the rank r of the simulated tensors and m, we illustrate the robustness of all of the methods against increasing ranks and missing data. It can be seen that TTRPCA outperforms HoRPCA as the underlying structure is better explained by TTNN. The proposed TTRPCA-G and TTRPCA-nG further improve the performance by incorporating the underlying manifold information.

Next, we evaluate the robustness of the proposed method against sparse outliers by replacing a constant $c\%$ of the entries by random values sampled uniformly from $[0, 1]$. For this set of experiments, $m = 0\%$ and $r = 4$. In Table 4.1, we summarize the results. It can be seen that graph regularization improves the performance at all outlier levels. In particular, TTRPCA is better than HoRPCA as TTNN captures the low-rank structure better than SNN and graph regularization further improves the results by taking the local geometry into account. Moreover, the results show that the performances of TTRPCA-G and TTRPCA-nG are close to each other which indicates that the two methods capture the same geometry.

4.3.2 Real Data

The algorithms are also tested on a subset of 40 objects from COIL dataset [133]. The color images are converted to grayscale and downsampled to a size of 16×16. For each object, each sample image corresponds to a different pose angle ranging from 0 to 360 degrees with increments of 10 degrees [133]. Thus, we create tensors of size $16 \times 16 \times 36 \times 40$. We then corrupt the tensor by randomly selecting $c\%$ of all entries and setting them to 0 or 1. Note that when using this data the adjacency matrix of the fourth mode canonical graph is of size 368640×368640. This makes TTRPCA-G computationally expensive, so we only use TTRPCA-nG.

From Table 4.2, it can be seen that the proposed method outperforms other methods in denoising for varying levels of gross noise. TTRPCA-nG can capture the data structure better than the other methods as it simultaneously minimizes TTNN and considers the underlying manifold across each mode.

4.4 Conclusions

In this chapter, we proposed two graph regularized robust tensor train principal component analysis methods. In the first method, we utilized canonical unfoldings to construct the mode-n graphs while in the second method mode-n unfoldings are used. We derived an equivalence between mode-n and canonical graphs with the assumption that the canonical graph has a specific Kronecker structure.

The proposed methods outperformed both robust Tucker and tensor-train decomposition methods in denoising and completion tasks. Experiments on synthetic data show that the performances of graph regularization with canonical unfolding and mode-n unfolding were similar to each other while mode-n graphs provided much lower memory requirements and computational complexity. Future work will consider capturing low-rank structure through graph total variation minimization as suggested in [154] to reduce the computational complexity of low-rank tensor recovery task.

CHAPTER 5

COUPLED SUPPORT TENSOR MACHINE

5.1 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). Their strong connections to the patients' biological status and disease pathology suggest 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 both at the voxel- and Region-of-Interest (ROI) level and develop efficient biomarkers that predict disease status. For example, [9] proposes a classification model using functional connectivity MRI for autism disease and reaches 89% diagnostic accuracy for subjects under 20. [152] utilizes network models and brain imaging data to develop novel biomarkers for Parkinson's disease. Many works in Alzheimer's disease research such as [129, 122, 89, 66, 124, 54, 116] 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 a single modality such as individual MRI sequences are known to have limited predictive capacity, especially in the early phases of the disease. For instance, [116] uses brain MRI volumes from regions of interest to identify patients in early-stage Alzheimer's disease. They use one-year MRI data from Alzheimer's Disease Neuroimaging Initiative (ADNI) and obtain 77% prediction accuracy. Although such a performance is favorable compared to other existing approaches, the diagnostic accuracy is relatively low due to the limited information from MRI data. In recent years, it has been common to acquire multiple neuroimaging modalities in clinical studies such as simultaneous EEG-fMRI or MRI and fMRI. Even though each modality measures different biological signals, they are interdependent and mutually informative. Learning from multimodal neuroimaging data may help integrate information from multiple sources and facilitate biomarker developments in clinical studies. It also raises the need for novel supervised learning techniques for multimodal data in statistical learning literature.

Existing statistical approaches to multimodal data science are dominated by unsupervised learning

methods. These methods analyze multimodal neuroimaging data by performing joint matrix decomposition and extracting common information across different modalities. During optimization, the decomposed factors bridging two or more modalities are estimated to interpret the connections between different modalities. Examples of these methods include matrix-based joint Independent Component Analysis (jICA) [29, 72, 107, 121, 165, 6] which assume bilinear correlations between factors in different modalities. However, these matrix-vector based models cannot preserve the multilinear nature of original data and the spatiotemporal correlations across modes as most neuroimaging modalities are naturally in tensor format. Recently, various coupled matrix-tensor decomposition methods have been introduced to address this issue [4, 5, 6, 37, 36, 86, 130]. These methods impose different soft or hard multilinear constraints between factors from different modalities providing more flexibility in data modeling.

Current supervised learning approaches for multimodal data mostly concatenate data modalities as extra features without exploring their interdependence. For example, [211, 114] build generalized regression models by appending tensor and vector predictors linearly for image prediction and classification. [142] develops a discriminant analysis by including tensor and vector predictors in a linear fashion. [111] proposes an integrative factor regression for multimodal neuroimaging data assuming that data from different modalities can be decomposed into latent factors. More recently, [65] proposed multiple tensor-on-tensor regression for multimodal data, which combines tensor-on-tensor regression from [123] with traditional additive linear model. Another type of integration utilizes kernel tricks and combines information from multimodal data with multiple kernels. [71] provides a survey on various multiple kernel learning techniques for multimodal data fusion and classification with support vector machines. Combining kernels linearly or non-linearly instead of original data in different modalities provides more flexibility in information integration. [12] 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 imaging 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 such as integrate information primarily by data or feature concatenation without explicitly considering the possible correlations between different modalities. This lack of consideration of interdependence may cause issues like overfitting and parameter identifiability. Third, even though methods from [111,

65] have considered latent structures for multimodal data, these models are designed primarily for linear regression and are not directly applicable to classification problems. Fourth, the aforementioned multimodal analysis methods are mainly vector based methods, which cannot handle large-size multi-dimensional data encountered in contemporary data science. As discussed in [21], tensors provide a powerful tool for analyzing multi-dimensional data in statistics. As a result, developing a novel multimodal tensor-based statistical framework for supervised learning can be of great interest. 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 solid statistical framework for multimodal data analysis makes it impossible to interpret the generalization ability of a certain statistical model.

In this chapter, we propose a two-stage Coupled Support Tensor Machine (C-STM) for multimodal tensor-based neuroimaging data classification. The proposed model addresses the current issues in multimodal data science 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 chapter are:

1. Individual and common latent factors are extracted from multimodal tensor data, for each sample or subject, using Advanced Coupled Matrix Tensor Factorization (ACMTF) [4, 3]. The extracted components are then utilized in a statistical framework. Most of the work on ACMTF do not focus on each subject separately and the extracted factors are utilized for a signal analysis rather than a subsequent statistical learning framework. Specifically, the work on supervised approaches with CMTF is limited.

2. Building a novel Coupled Support Tensor Machine with both the coupled and non-coupled tensor CP factors for classification. In this regard, multiple kernel learning approaches are adopted to integrate components from multi-modal data.

3. For the validation of our work, we provide both theoretical and empirical evidence. We provide theoretical results such as classification consistency for statistical guarantee. A thorough numerical study has been conducted, including a simulation study and experiments on real data to illustrate the usefulness of the proposed methodology.

A Matlab package is also provided in the supplemental material, including all functions for C-STM classifi-

cation. The source codes are available at our Github repository [1].

5.2 Related Work

In this section, we review some background and prior work on coupled tensor decomposition and multiple kernel learning.

5.2.1 Coupled Matrix Tensor Factorization

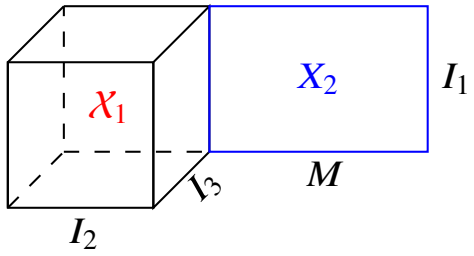


Figure 5.1: Illustration of Coupled Tensor Matrix Model

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.

During latent factor estimation, CMTF solves an objective function that approximates a CP decomposition for the tensor modality and a singular value decomposition for the second modality with the assumption that the factors from one mode of each modality are the same. Given $\mathcal{X}_1 \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_d}$ and $X_2 \in \mathbb{R}^{I_1 \times J_2}$, without loss of generality assume that the factors from the first mode of the tensor $\mathcal{X}_1$ span the column space of the matrix X_2. CMTF then tries to estimate all factors by minimizing:

$$\mathbf{Q}(\mathfrak{U}_1, \mathbf{V}) = \frac{1}{2}\|\mathcal{X}_1 - [[X_1^{(1)}, X_1^{(2)}, \ldots X_1^{(d)}]]\|_F^2 + \frac{1}{2}\|X_2 - X_2^{(1)} X_2^{(2)T}\|_F^2, \quad \text{s.t. } X_1^{(1)} = X_2^{(1)}, \tag{5.1}$$

where $X_p^{(m)}$ are the factor matrices for modality p and mode m. The factor matrices $X_1^{(1)} = X_2^{(1)}$ are the coupled factors between tensor and matrix data. An illustration of this coupling is given in Figure

[1] https://github.com/PeterLiPeide/Coupled_MatrixTensor_SupportTensor_Machine

5.1. These factor matrices can also be represented in Kruskal form, $\mathfrak{U}_1 = [[X_1^{(1)}, X_1^{(2)}, \dots X_1^{(d)}]]$ and $\mathfrak{U}_2 = [[X_2^{(1)}, X_2^{(2)}]]$. By minimizing the objective function $\mathbf{Q}(\mathfrak{U}_1, \mathfrak{U}_2)$, CMTF estimates latent factors for 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 (5.1). Although this model is formulated for the joint decomposition of a dth order tensor and a matrix, extensions to two or more tensors with couplings across multiple modes are possible.

In real data, couplings across different modalities might include shared or modality-specific (individual) components. Shared components correspond to those columns of the factor matrices that contribute to the decomposition of both modalities, while individual components carry information unique to the corresponding modality. Although CMTF provides a successful framework for joint data analysis, it often fails to obtain a unique estimation for shared or individual components. As a result, any further statistical analysis and learning from CMTF estimation will suffer from the 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 (5.1), and restricting the norm of the columns of the factors to be unity to provide uniqueness up to a permutation. This modification provides a more precise estimation for latent factors compared to CMTF ([3, 5]). In our framework, we utilize ACMTF to extract the latent factors which are in turn used to build a classifier for multimodal data.

5.2.2 CP-STM for Tensor Classification

CP-STM has been previously studied by [169, 76, 77] and uses CP tensor to construct STM types of model. Assume there is a collection of data $T_n = \{(\mathcal{X}_1, y_1), (\mathcal{X}_2, y_2), \dots, (\mathcal{X}_n, y_n)\}$, where $\mathcal{X}_t \in \mathcal{X} \subset \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$ are d-way tensors. $\mathcal{X}$ is a compact tensor space, which is a subspace of $\mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$. $y_t \in \{-1, 1\}$ are binary labels. CP-STM assumes the tensor predictors are in CP format, and can be classified by the function which minimizes the objective function

$$\min_f \quad \lambda ||\mathbf{f}||^2 + \frac{1}{n} \sum_{t=1}^{n} \mathcal{L}(\mathbf{f}(\mathcal{X}_t), y_t). \tag{5.2}$$

By using tensor kernel function

$$K(\mathcal{X}_1, \mathcal{X}_2) = \sum_{l,m=1}^{r} \prod_{j=1}^{d} K^{(j)}(\mathbf{x}_{1,l}^{(j)}, \mathbf{x}_{2,m}^{(j)}), \tag{5.3}$$

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

$$\mathbf{f}(\mathcal{X}) = \sum_{t=1}^{n} \alpha_t y_t K(\mathcal{X}_t, \mathcal{X}) = \alpha^T \mathbf{D}_y \mathbf{K}(\mathcal{X}) \tag{5.4}$$

where $\mathcal{X}$ is a new d-way rank-r tensor, $\alpha = [\alpha_1, \ldots, \alpha_n]^T$ is the coefficient vector, $\mathbf{D}_y$ is a diagonal matrix whose diagonal elements are $y_1, .., y_n$ and $\mathbf{K}(\mathcal{X}) = [K(\mathcal{X}_1, \mathcal{X}), \ldots, K(\mathcal{X}_n, \mathcal{X})]^T$ is a column vector, whose values are kernel values computed between training and test data. We denote the collection of functions in the form of (5.4) with $\mathcal{H}$, which is a functional space also known as Reproducing Kernel Hilbert Space (RKHS). The optimal classifier CP-STM $\mathbf{f} \in \mathcal{H}$ can be estimated by plugging function (5.4) into objective function (5.2), and minimize it with Hinge or Squared Hinge loss. The coefficients of the optimal CP-STM model is denoted by α^*. The classification model is statistically consistent if the tensor kernel function satisfying the universal approximating property, which is shown by [109].

5.2.3 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 ([71]). A more important and 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 $\{K_1(\cdot,\cdot), \ldots K_m(\cdot,\cdot)\}$, a new kernel function can be constructed by

$$K(\cdot,\cdot) = \mathbf{f}_\eta(\{K_1(\cdot,\cdot), \ldots K_m(\cdot,\cdot)\}|\eta) \tag{5.5}$$

where $\mathbf{f}_\eta$ is a linear or non-linear function and η is a vector whose elements are the weights for the kernel combination. Linear combination methods are the most popular in multiple kernel learning, where the kernel function is parameterized as

$$\begin{aligned} K(\cdot,\cdot) &= \mathbf{f}_\eta(\{K_1(\cdot,\cdot), \ldots K_m(\cdot,\cdot)\}|\eta) \\ &= \sum_{l=1}^{m} \eta_l K_l(\cdot,\cdot). \end{aligned} \tag{5.6}$$

The weight parameters η_l can be simply assumed to be the same (unweighted) ([145, 16]), or be determined by looking at some performance measures for each kernel or data representation ([167, 147]). There are few more advanced approaches such as optimization-based, Bayesian approaches, and boosting approaches that can also be adopted ([103, 64, 176, 92, 69, 41, 19]). In this chapter, we only consider linear combination (5.6), and select the weight parameters in a heuristic data-driven way to construct our C-STM model.

5.3 Methods

Let $T_n = \{(\mathcal{X}_{1,1}, X_{1,2}, y_1), \ldots, (\mathcal{X}_{n,1}, X_{n,2}, y_n)\}$ be training data, where each sample $t \in \{1, \ldots, n\}$ has two data modalities $\mathcal{X}_{t,1}, X_{t,2}$ and a corresponding binary label $y_t \in \{1, -1\}$. In this chapter, following [2], we assume that the first data modality is a third-order tensor, $\mathcal{X}_{t,1} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, and the other is a matrix, $X_{t,2} \in \mathbb{R}^{I_4 \times I_3}$. The third mode of $\mathcal{X}_{t,1}$ and the second mode of $X_{t,2}$ 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, consists of two stages: Multimodal tensor factorization, i.e., ACMTF, and coupled support tensor machine as illustrated in Figure 5.2. In this section, we present both stages and the corresponding procedures.

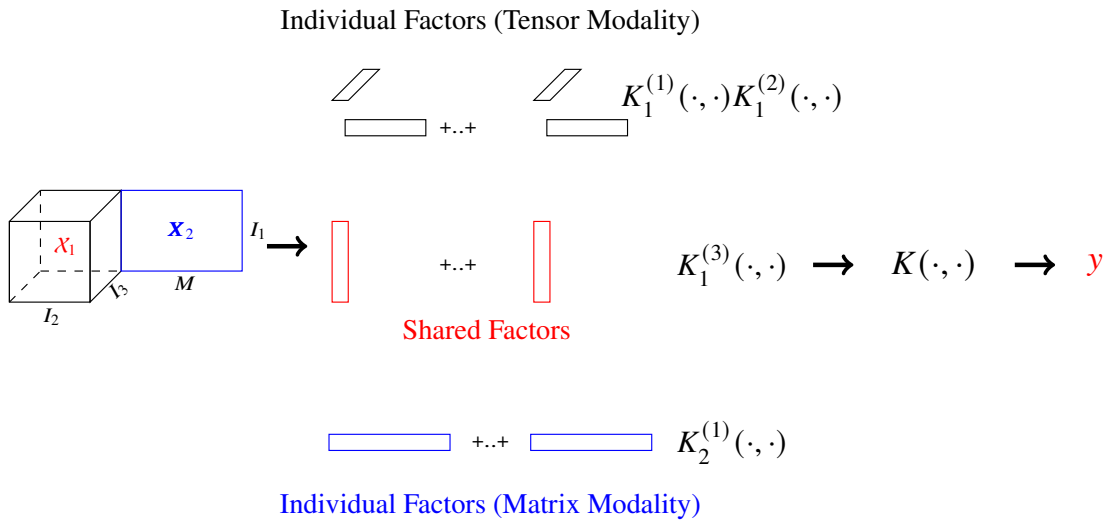


Figure 5.2: C-STM Model Pipeline

5.3.1 Multimodal Tensor Factorization

In this chapter, the first aim is to perform a joint factorization across two modalities for each training sample, t. Let $\mathfrak{U}_{t,1} = [[\zeta; X_{t,1}^{(1)}, X_{t,1}^{(2)}, X_{t,1}^{(3)}]]$ denote the Kruskal tensor of $\mathcal{X}_{t,1}$, and $\mathfrak{U}_{t,2} = [[\sigma; X_{t,2}^{(1)}, X_{t,2}^{(2)}]]$ denote the singular value decomposition of $X_{t,2}$. The weights of the columns of each factor matrix $X_{t,p}^{(m)}$, where p is the index for modality and m denotes the mode, are denoted by ζ and σ and the norms of these columns are constrained to be 1 to avoid redundancy. The objective function of ACMTF [4, 3] is then given by:

$$\begin{aligned}\mathbf{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2}) = \gamma\|\mathcal{X}_{t,1} - [[\zeta; X_{t,1}^{(1)}, X_{t,1}^{(2)}, X_{t,1}^{(3)}]]\|_F^2 + \gamma\|X_{t,2} - X_{t,2}^{(1)}\mathbf{\Sigma}X_{t,2}^{(2)^\top}\|_F^2 \\ + \beta\|\zeta\|_0 + \beta\|\sigma\|_0 \\ \text{s.t.} \quad X_{t,1}^{(3)} = X_{t,2}^{(2)} \\ \|\mathbf{x}_{t,1,k}^{(1)}\|_2 = \|\mathbf{x}_{t,1,k}^{(2)}\|_2 = \|\mathbf{x}_{t,1,k}^{(3)}\|_2 = \|\mathbf{x}_{t,2,k}^{(1)}\|_2 = \|\mathbf{x}_{t,2,k}^{(2)}\|_2 = 1, \\ \forall k \in \{1, \ldots, r\}\end{aligned} \tag{5.7}$$

where $\mathbf{\Sigma}$ is a diagonal matrix whose elements are the singular values σ of the matrix $X_{t,2}$ and $\mathbf{x}_{t,m,k}^{(j)} \in \mathbb{R}^{I_j}$ denotes the columns of the factor matrices for the object $\mathcal{X}_{t,m}$. The objective function in (5.7) includes penalties for the number of non-zero weights in both tensor and matrix decomposition. Thus, the model identifies the shared and individual components. These factors are then considered as different data representations for multimodal data, and used to predict the labels y_t in C-STM classifier.

5.3.2 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 similarity, i.e., 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 measures through a non-linear combination. This combination should be able to take individual and shared components into account separately for better adaptability depending on the size and corruptions on the data as the coupled modes are likely to be better estimated than the individual modes. Thus, we use tensor kernels for individual modes of each modality and combine these with the kernels of the coupled modes as illustrated in Figure 5.2. The kernel function for

C-STM is defined as

$$K\big((\mathcal{X}_{t,1}, X_{t,2}), (\mathcal{X}_{i,1}, X_{i,2})\big) = K\big((\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2}), (\mathfrak{U}_{i,1}, \mathfrak{U}_{i,2})\big)$$
$$= \sum_{k,l=1}^{r} w_1 K_1^{(1)}(\mathbf{x}_{t,1,k}^{(1)}, \mathbf{x}_{i,1,l}^{(1)}) K_1^{(2)}(\mathbf{x}_{t,1,k}^{(2)}, \mathbf{x}_{i,1,l}^{(2)}) + w_2 K_1^{(3)}(\mathbf{x}_{t,1,k}^{(3)*}, \mathbf{x}_{i,1,l}^{(3)*}) + w_3 K_2^{(1)}(\mathbf{x}_{t,2,k}^{(1)}, \mathbf{x}_{i,2,l}^{(1)}) \tag{5.8}$$

for two pairs of decomposed tensor matrix factors $(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})$ and $(\mathfrak{U}_{i,1}, \mathfrak{U}_{i,2})$. $\mathbf{x}_{t,1,k}^{(3)*}$ is the average of the estimated shared factors $\frac{1}{2}[\mathbf{x}_{t,1,k}^{(3)} + \mathbf{x}_{t,2,k}^{(2)}]$. This kernel is inspired by the idea of Multiple Kernel Learning with linear combination of multiple kernels for multimodal data. Few more details regarding chosing such kernel combination are provided in the section 5.5.2.1. w_1, w_2, and w_3 are the three weight parameters combining the three kernel functions. As discussed in [71], there is no unique choice for determining these weights, in this chapter, we adopt a cross-validation approach as explained in the Appendix C.2.

With the kernel function in (5.8), C-STM model tries to estimate a bivariate decision function $\mathbf{f}$ from a collection of functions $\mathcal{H}$ such that

$$\mathbf{f} = \arg\min \quad \lambda \cdot ||\mathbf{f}||^2 + \frac{1}{n}\sum_{t=1}^{n} \mathcal{L}(\mathbf{f}(\mathcal{X}_t), y_t), \tag{5.9}$$

where $\mathcal{L}(\mathcal{X}_t, y_t) = \max\big(0, 1 - \mathbf{f}(\mathcal{X}_t) \cdot y_t\big)$ is Hinge loss. $\mathcal{H}$ is defined as the collection of all functions in the form of

$$\begin{aligned} \mathbf{f}(\mathcal{X}_1, X_2) &= \sum_{t=1}^{n} \alpha_t y_t K((\mathcal{X}_{t,1}, X_{t,2}), (\mathcal{X}_1, X_2)) \\ &= \alpha^T \mathbf{D}_y \mathbf{K}(\mathcal{X}_1, X_2) \end{aligned} \tag{5.10}$$

due to the well-known representer theorem ([10]) for any pair of testing data $(\mathcal{X}_1, X_2)$ and for $\alpha \in \mathbb{R}^n$. For all possible values of α, equation (5.10) defines the data collection $\mathcal{H}$. $\mathbf{D}_y$ is a diagonal matrix whose diagonal elements are labels from the training data T_n. $\mathbf{K}(\mathcal{X}_1, X_2)$ is a $n \times 1$ vector whose t-th element is $K\big((\mathcal{X}_{t,1}, X_{t,2}), (\mathcal{X}_1, X_2)\big)$. The optimal C-STM decision function, denoted by $\mathbf{f_n} = \alpha^{*T} \mathbf{D}_y \mathbf{K}(\mathcal{X}_1, X_2)$, can be estimated by solving the quadratic programming problem

$$\begin{aligned} \min_{\alpha \in \mathbb{R}^n} \quad & \frac{1}{2}\alpha^T \mathbf{D}_y \mathbf{K} \mathbf{D}_y \alpha - \mathbf{1}^T \alpha, \\ \text{S.T.} \quad & \alpha^T \mathbf{y} = 0, \\ & 0 \preceq \alpha \preceq \frac{1}{2n\lambda}, \end{aligned} \tag{5.11}$$

where $\mathbf{K}$ is the kernel matrix constructed by function (5.8). Problem (5.11) is the dual problem of (5.9), and its optimal solution α^* also minimizes the objective function (5.9) when plugging functions in the form of (5.10). For a new pair of test points $(\mathcal{X}_1, X_2)$, the class label is predicted as sign $(\mathbf{f_n}(\mathcal{X}_1, X_2))$.

5.4 Model Estimation

In this section, we first present the estimation procedure for coupled tensor matrix decomposition (5.7), and then combine it with the classification procedure to summarize the algorithm for C-STM.

To satisfy the constraints in the objective function (5.7), the function $\mathbf{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})$ is converted to a differentiable and unconstrained form given by:

$$\begin{aligned}\mathbf{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2}) =& \gamma\|\mathcal{X}_{t,1} - [[\zeta; X_{t,1}^{(1)}, X_{t,1}^{(2)}, X_{t,1}^{(3)}]]\|_F^2 + \gamma\|X_{t,2} - X_{t,2}^{(1)}\mathbf{\Sigma}X_{t,2}^{(2)^\top}\|_F^2 \\ &+ \xi\|X_{t,1}^{(3)} - X_{t,2}^{(2)}\|_F^2 \\ &+ \sum_{k=1}^{r}\Bigg[\beta\sqrt{\zeta_r^2+\epsilon} + \beta\sqrt{\sigma_r^2+\epsilon} + \theta\big[(\|\mathbf{x}_{t,1,k}^{(1)}\|_2 - 1)^2 + (\|\mathbf{x}_{t,1,k}^{(2)}\|_2 - 1)^2 \\ &+ (\|\mathbf{x}_{t,1,k}^{(3)}\|_2 - 1)^2 + (\|\mathbf{x}_{t,2,k}^{(1)}\|_2 - 1)^2 + (\|\mathbf{x}_{t,2,k}^{(2)}\|_2 - 1)^2\big]\Bigg],\end{aligned} \tag{5.12}$$

where ℓ_1 norm penalties in (5.7) are replaced with differentiable approximations; ξ and θ are Lagrange multipliers and $\epsilon > 0$ is a very small number. This unconstrained optimization problem can be solved by nonlinear conjugate gradient descent ([2, 4, 130]).

Let $\mathcal{T}_t$ be the full (created by converting Kruskal tensor, or the factor matrices into multidimensional array form) tensor of $\mathfrak{U}_{t,1}$, and $\mathbf{M}_t = X_{t,2}^{(1)}\mathbf{\Sigma}X_{t,2}^{(2)^\top}$, the partial derivative of each latent factor can be derived as follows:

$$\frac{\delta\mathbf{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta X_{t,1}^{(1)}} = \gamma(\mathcal{T}_t - \mathcal{X}_{t,1})_{(1)}(\zeta^\top \odot X_{t,1}^{(3)} \odot X_{t,1}^{(2)}) + \theta(X_{t,1}^{(1)} - \bar{X}_{t,1}^{(1)}), \tag{5.13}$$

$$\gamma\frac{\delta\mathbf{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta X_{t,1}^{(2)}} = (\mathcal{T}_t - \mathcal{X}_{t,1})_{(2)}(\zeta^\top \odot X_{t,1}^{(3)} \odot X_{t,1}^{(1)}) + \theta(X_{t,1}^{(2)} - \bar{X}_{t,1}^{(2)}), \tag{5.14}$$

$$\gamma\frac{\delta\mathbf{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta X_{t,1}^{(3)}} = (\mathcal{T}_t - \mathcal{X}_{t,1})_{(3)}(\zeta^\top \odot X_{t,1}^{(2)} \odot X_{t,1}^{(1)}) + \xi(X_{t,1}^{(3)} - X_{t,2}^{(2)}) + \theta(X_{t,1}^{(3)} - \bar{X}_{t,1}^{(3)}), \tag{5.15}$$

$$\gamma\frac{\delta\mathbf{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta X_{t,2}^{(1)}} = (\mathbf{M}_t - X_{t,2})X_{t,2}^{(2)}\mathbf{\Sigma} + \theta(X_{t,2}^{(1)} - \bar{X}_{t,2}^{(1)}), \tag{5.16}$$

$$\gamma\frac{\delta\mathbf{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta X_{t,2}^{(2)}} = (\mathbf{M}_t - X_{t,2})^\top X_{t,2}^{(1)}\mathbf{\Sigma} + \tau(X_{t,2}^{(2)} - X_{t,1}^{(3)}) + \theta(X_{t,2}^{(2)} - \bar{X}_{t,2}^{(2)}), \tag{5.17}$$

$$\frac{\delta\mathbf{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta\sigma_k} = \mathbf{x}_{t,2,k}^{(1)^\top}(\mathbf{M}_t - X_{t,2})\mathbf{x}_{t,2,k}^{(2)} + \frac{\beta}{2}\frac{\sigma_k}{\sqrt{\sigma_k^2+\epsilon}},\ k \in \{1, \ldots, r\}, \tag{5.18}$$

$$\frac{\delta\mathbf{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta\zeta_k} = \mathrm{vec}(\mathcal{T}_t - \mathcal{X}_{t,1})^\top\left(\mathbf{x}_{t,1,k}^{(3)} \odot \mathbf{x}_{t,1,k}^{(2)} \odot \mathbf{x}_{t,1,k}^{(1)}\right) + \frac{\beta}{2}\frac{\zeta_k}{\sqrt{\zeta_k^2+\epsilon}},\ k \in \{1, \ldots, r\}, \tag{5.19}$$

Algorithm 5.1: ACMTF Decomposition

Input: Multimodal data $(\mathcal{X}_1, X_2)$, r, η, S (Upper limit for the number of iterations)
Output: $\mathfrak{U}^*_{t,1}, \mathfrak{U}^*_{t,2}$
$\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2} = \mathfrak{U}^0_{t,1}, \mathfrak{U}^0_{t,2}$ ▷ Initial value
$\mathbf{\Delta}_0 = -\nabla\mathbf{Q}(\mathfrak{U}^0_{t,1}, \mathfrak{U}^0_{t,2})$
$\varphi_0 = \arg\min_\varphi \mathbf{Q}\big[(\mathfrak{U}^0_{t,1}, \mathfrak{U}^0_{t,2}) + \varphi\mathbf{\Delta}_0\big]$
$\mathfrak{U}^1_{t,1}, \mathfrak{U}^1_{t,2} = (\mathfrak{U}^0_{t,1}, \mathfrak{U}^0_{t,2}) + \varphi_0\mathbf{\Delta}_0$
$\mathbf{g}_0 = \mathbf{\Delta}_0$
while s < S and $\|\mathbf{Q}(\mathfrak{U}^s_{t,1}, \mathfrak{U}^s_{t,2}) - \mathbf{Q}(\mathfrak{U}^{s-1}_{t,1}, \mathfrak{U}^{s-1}_{t,2})\| \geqslant \eta$ **do**
    $\mathbf{\Delta}_{s+1} = -\nabla\mathbf{Q}(\mathfrak{U}^s_{t,1}, \mathfrak{U}^s_{t,2})$
    $\mathbf{g}_{s+1} = \mathbf{\Delta}_{s+1} + \frac{\mathbf{\Delta}^\top_{s+1}(\mathbf{\Delta}_{s+1}-\mathbf{\Delta}_s)}{-\mathbf{g}^\top_s(\mathbf{\Delta}_{s+1}-\mathbf{\Delta}_s)}\mathbf{g}_s$
    $\varphi_{s+1} = \arg\min_\varphi \mathbf{Q}\big[(\mathfrak{U}^s_{t,1}, \mathfrak{U}^s_{t,2}) + \varphi\mathbf{g}_{s+1}\big]$
    $\mathfrak{U}^{s+1}_{t,1}, \mathfrak{U}^{s+1}_{t,2} = (\mathfrak{U}^s_{t,1}, \mathfrak{U}^s_{t,2}) + \varphi_{s+1}\mathbf{g}_{s+1}$
end while

where $(vec(.))$ is a vectorization operator that stacks all elements of the operand in a column vector, $\mathcal{T}_{(j)}$ denotes the mode-j unfolding of a tensor $\mathcal{T}$, and $\odot$ denotes Khatri-Rao product. $\bar{\mathbf{M}}$ is a normalized matrix whose columns have unit ℓ_2 norms.

By combining all of the partial derivatives, the partial derivative of the objective function is given by:

$$\nabla\mathbf{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2}) = \Bigg[\frac{\delta\mathbf{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta X^{(1)}_{t,1}}, \frac{\delta\mathbf{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta X^{(2)}_{t,1}}, \frac{\delta\mathbf{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta X^{(3)}_{t,1}}, \frac{\delta\mathbf{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta X^{(2)}_{t,2}}, \frac{\delta\mathbf{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta\zeta_1}, \ldots \frac{\delta\mathbf{Q}(\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2})}{\delta\sigma_1}, \ldots\Bigg]^\top \tag{5.20}$$

which is a $2r + 5$ dimensional vector. As mentioned in [4], a nonlinear conjugate gradient method with Hestenes-Stiefel updates is used to optimize (5.12). The procedure is described in Algorithm 5.1.

Once the factors for all data pairs in the training set T_n are extracted, we can create the kernel matrix using the kernel function in (5.8). By solving the quadratic programming problem (5.11), we can obtain the optimal decision function $\mathbf{f_n}$. This two-stage procedure for C-STM estimation is summarized in Algorithm 5.2.

Algorithm 5.2: Coupled Support Tensor Machine

procedure C-STM
  Input: Training set $T_n = \{(\mathcal{X}_{1,1}, X_{1,2}, y_1), \ldots, (\mathcal{X}_{n,1}, X_{n,2}, y_n)\}$, $\mathbf{y}$, kernel function K, r, λ, η, S
  for t = 1, 2,. . . n **do**
    $\mathfrak{U}^*_{t,1}, \mathfrak{U}^*_{t,2}$ = ACMTF$\big((\mathcal{X}_{t,1}, X_{t,2}), r, \eta, \mathrm{S}\big)$
  end for
  Create initial matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$
  for t = 1,. . . ,n **do**
    for i = 1,. . . ,n **do**
      $\mathbf{K}[i,t] = K\Big((\mathfrak{U}_{t,1}, \mathfrak{U}_{t,2}), (\mathfrak{U}_{i,1}, \mathfrak{U}_{i,2})\Big)$ ▷ Kernel values
      $\mathbf{K}[i,t] = \mathbf{K}[t,i]$
    end for
  end for
  Solve the quadratic programming problem (5.11) and find the optimal α^*.
  Output: α^*
end procedure

5.5 Experiments

5.5.1 Parameter Selection

5.5.1.1 Multimodal Tensor Factorization

The proposed model requires the selection of three different parameters, namely, γ, β, and rank r. To select these parameters, we closely follow best practices outlined in previous work on CMTF [2], ACMTF [4] and CCMTF [130]. First of all, one of these parameter can be set to 1 as a pivot, and following previous work, we set $\gamma = 1$. The selection of rank r is directly related to the selection of β. As β enforces sparsity over the singular values, it directly minimizes the rank. With sufficiently large r, we can estimate the low-rank part through optimization. For the selection of r in real data, we set $r = 5$ following the work of [130] where it was shown through CORCONDIA tests [85] that $r = 3$ is sufficiently large for oddball data. In the case of the simulation study, $r = 5$ is again sufficiently large as the data were generated from rank $r = 3$ factors. Finally, based on our empirical results and the results presented in [130] we set $\beta = 0.001$ using k-fold cross validation.

5.5.1.2 C-STM

The parameters in C-STM include kernel weights w_1, w_2, w_3 and regularization parameter λ in the optimization. The weight parameters, normalized such that the ℓ_2-norm is equal to 1, and λ are selected using 5 fold cross-validation. The overall classification accuracy in our validation set serves as the performance metric and helps us determine the best combination of weights and λ.

The selection of weight parameters w_1, w_2, w_3 is indeed a problem of how to combine kernels from different modalities. It is straightforward to calculate kernels from every data modality, however, combining them appropriately and effectively would be challenging unless we can find out the weight for each kernel. This problem has been widely studied in the literature of Multiple Kernel Learning. In [71], the authors summarize that the existing methods of kernel weight selection can be divided into five categories, including fixed rules, heuristic approaches, optimization approaches, Bayesian approaches, and boosting approaches. As there is no consensus on the best way to choose the weights, we adopt a cross-validation approach as explained in the Appendix of the revised manuscript to identify the kernel weights. The overall classification accuracy in our validation set serves as the performance metric and helps us determine the best combination of weights.

The generalization of our method to more than two modalities would be straightforward for tuning the weights. This is because the tuning problem has been widely studied in multiple kernel learning (MKL) research. There is no restriction on the number of kernels one can include in MKL framework. The weight selection techniques in MKL can be adapted to our framework.

The optimization problem defined in (5.9) is an ordinary SVM problem once the kernel values are calculated through equation (5.8). Thus, for more information about the estimation procedure, λ selection, as well as the consistency results readers are referred to existing Support Vector Machine literature [163].

5.5.2 Simulated Data

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 multi-modalities in C-STM, we include CP-STM from [76], Constrained Multilinear Discriminant Analysis (CMDA), and Direct General Tensor Discriminant Analysis (DGTDA) from [112] as competitors. These existing approaches can only take a single tensor / matrix as the feature for classification. As a result, they are not able to enjoy the multi-modalities in the

simulated data. We apply these approaches on every single data modality in our simulated data, and compare their classification performance with C-STM which uses multimodal data.

We generate synthetic data using the idea from [61]. Suppose the two data modalities in our classification problems are

$$\begin{aligned} \mathcal{X}_{t,1} &= \sum_{k=1}^{3} \mathbf{x}_{k,t,1}^{(1)} \circ \mathbf{x}_{k,t,1}^{(2)} \circ \mathbf{x}_{k,t,1}^{(3)} \\ X_{t,2} &= \sum_{k=1}^{3} \mathbf{x}_{k,t,2}^{(1)} \circ \mathbf{x}_{k,t,2}^{(2)} \end{aligned} \tag{5.21}$$

where $\mathcal{X}_{t,1}$ are three-way tensors in the size of 30 by 20 by 10. $X_{t,2}$ are matrices in the size of 50 by 10. Both of them have CP ranks equal to 3. To generate data for the simulation study, we first generate the latent factors (vectors) from various multivariate normal distributions, and then convert these factors into full tensors $\mathcal{X}_{t,1}$ and matrices $X_{t,2}$ using equation (5.21). The multivariate normal distributions we used to generate columns of the latent factors in equation (5.21) are specified in Table 5.1 below. In Table 5.1, we use $c = 1, 2$ to denote data from two different classes.

Table 5.1: Distribution Specifications for Simulation Study; **MVN** stands for multivariate normal distribution. **I** are identity matrices. Bold numbers are vectors whose elements are all equal to the numbers.

		Tensor Factors		Shared Factors	Matrix Factors
Simulation	c	$\mathbf{x}_{k,t,1}^{(1)}$	$\mathbf{x}_{k,t,1}^{(2)}$	$\mathbf{x}_{k,t,1}^{(3)} = \mathbf{x}_{k,t,2}^{(2)}$	$\mathbf{x}_{k,t,2}^{(1)}$
Case 1	1	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$
	2	$\mathbf{MVN}(\mathbf{1.5}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1.25}, \mathbf{I})$
Case 2	1	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$
	2	$\mathbf{MVN}(\mathbf{1.5}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1.5}, \mathbf{I})$
Case 3	1	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$
	2	$\mathbf{MVN}(\mathbf{1.5}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1.75}, \mathbf{I})$
Case 4	1	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$
	2	$\mathbf{MVN}(\mathbf{1.5}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{2}, \mathbf{I})$
Case 5	1	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$
	2	$\mathbf{MVN}(\mathbf{1.5}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{2.25}, \mathbf{I})$
Case 6	1	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$
	2	$\mathbf{MVN}(\mathbf{2}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$
Case 7	1	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$
	2	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{2}, \mathbf{I})$
Case 8	1	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$
	2	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{2}, \mathbf{I})$	$\mathbf{MVN}(\mathbf{1}, \mathbf{I})$

There are eight different cases in our simulation study. In case 1 - 5, one of the tensor factors and the matrix factors are generated from different multivariate normal distributions for data in different classes. This means the tensor and matrix data both contain certain class information (discriminant power) which are different in different data modalities. Notice that the discriminant power in one of the tensor factor remains the same among case 1 - 5, while the power in the matrix factor increases. Case 6 and 7 assume the class information exists only in a single data modality. In case 6, only one of the tensor factors are generated from different distributions for data in different classes. This factor then becomes the matrix factor in class 7. In case 8, the shared factors are sampled from different distributions, meaning that both tensor and matrix data modalities have class information. However, such class information are from the shared factors are the same between different modalities.

For each simulation case, we generate 50 pairs of tensor and matrix from both class, collecting 100 pairs of observations in total. We then perform a random training and testing set separation by randomly choosing 20 samples as the testing 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 training and testing set separation is repeated for 50 times and the average prediction accuracy of these 50 repetitions for all the cases are reported in Figure 5.3. Additionally, the standard deviations are illustrated by the error bars in the figure. The results of CP-STM, CMDA, and DGTDA with tensor data are denoted by CPSTM1, CMDA1, and DGTDA1 respectively in the figure. The results using matrix data are denoted by CPSTM2, CMDA2, and DGTDA2.

From the Figure 5.3, we can conclude that our C-STM has a more favorable performance in this multimodal classification problem comparing with other competitors. Its accuracy rates are significantly larger than other methods in most cases. Particularly, we can see that the accuracy rates of C-STM (orange) are increasing from case 1 to case 5, while the accuracy rates of CP-STM using tensor data remain the same. This is because the difference between class mean vectors for the first tensor factor does not change from case 1 to case 5. However, the gap between class mean vectors in matrix factor increases. Due to this fact, both C-STM and CP-STM (yellow) which utilize matrix data are getting better performance from case 1 to case 5. More importantly, C-STM always outperforms CP-STM with matrix data as it enjoys the extra class information from multimodalities. In case 6 and case 7 where class information are in single data

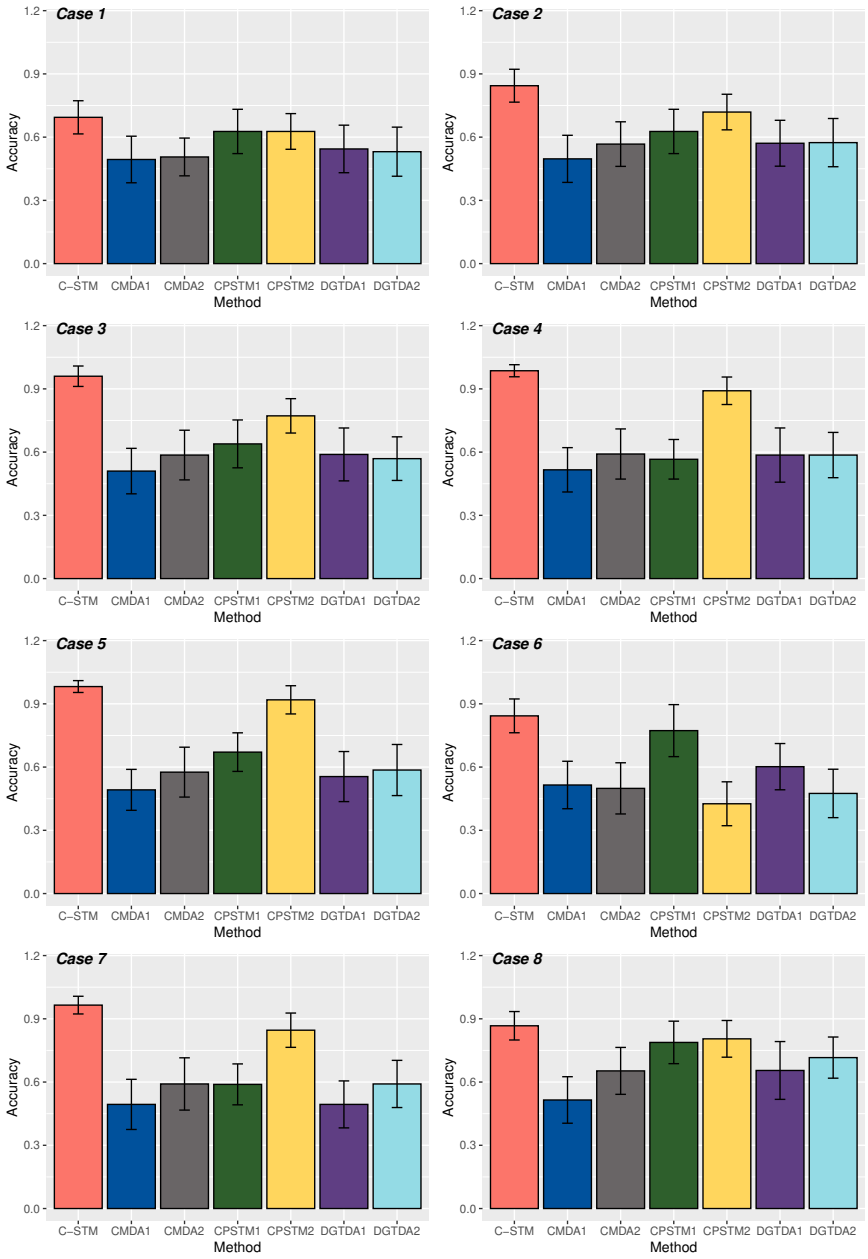


Figure 5.3: Simulation Result: Average accuracy rates shown in bar plots; Standard deviation of accuracy rates shown by error bars

modalities, the advantage of C-STM is not as significant as the previous cases, though its performances are slightly better than CP-STM. This indicates C-STM can provide robust classification results even when extra data modalities do not provide any other class information, as it can extract more accurate estimates of the factors in the decomposition step. In case 8 where the class information is from the shared factors, C-STM recovers the shared factors accurately and provides significantly better classification accuracy. Through this simulation, we showed that C-STM has a clear advantage of using multimodal data in classification problems,

and is robust to redundant data modalities.

5.5.2.1 Kernel Selection

In this section, we evaluate and justify the choice of the kernel function presented in (5.8). In this formulation, the individual and coupled modes for each modality are first separated and then the individual modes of each modality are combined as a tensor kernel function. The kernels from the individual modes are then added to those from the coupled modes to obtain the final form where the weights for the individual and coupled parts can be optimized as discussed in the Appendix. This kernel formulation separates the coupled and individual information and integrates them as a linear combination. Although this is not the only way to integrate kernels, the relatively simple structure of such combination provides us with several benefits such as interpretability, convenient parameter tuning, and generalizability for multimodal data. With equation (5.8), it is possible to explain the contribution of different data modalities to discrimination power by looking at the weights parameters. Further, with linear combination of kernels, the weight parameters can be tuned with the different approaches introduced in [71] such as Group Lasso. Even though we do not adopt these tuning techniques in this chapter, it still shows the advantage of choosing such a combination and can be the foundation for future work. Lastly, this kernel combination can be extended for data with more modalities easily since kernels are appended linearly.

Besides the aforementioned reasons, we also provide numerical experiments to illustrate the performance of our choice against other kernel combination choices. In these experiments the factor sizes are the same ($\mathcal{X}_1 \in \mathbb{R}^{40\times40\times40}$, and $X_2 \in \mathbb{R}^{40\times40}$, $r = 3$) so that the kernels are balanced across the modes. We consider two cases, i.e., Case 8 in Table 5.1, and Case 9 where the columns of the latent factors corresponding to all individual and coupled modes of the second class are from the distribution $\mathbf{MVN}(\mathbf{2}, \mathbf{I})$. Although there can be many different kernel combinations, we select four particular formulations for comparison as they can be a basis for other choices. The particular formulations are the weighted combination of individual kernels from all modes (K2), the weighted combination of the tensor kernels corresponding to the two modalities (K3) and the tensor kernel corresponding to all modes across modalities (K4). The formulations for the different kernels are given in Table 5.2. We report average classification accuracy across 50 simulations, where the simulated tensors are randomly initialized.

In Table 5.3, we can see that the kernel selection schemes K1 and K2 perform the best. K3 performs

Table 5.2: Various kernel combination schemes. Note that $K_2^{(2)} = K_1^{(3)}$.

	Combination Scheme
K1	$w_1 K_1^{(1)} K_1^{(2)} + w_2 K_1^{(3)} + w_3 K_2^{(1)}$
K2	$w_1 K_1^{(1)} + w_2 K_1^{(2)} + w_3 K_1^{(3)} + w_4 K_2^{(1)}$
K3	$w_1 K_1^{(1)} K_1^{(2)} K_1^{(3)} + w_2 K_2^{(1)} K_2^{(2)}$
K4	$K_1^{(1)} K_1^{(2)} K_1^{(3)} K_2^{(1)}$

slightly worse as it is not as flexible as the previous two. Finally, K4 performs the worst as it is affected by all modes simultaneously, and cannot generalize well. While the difference in performance between the kernels is not significant, K3 and K4 cannot determine whether the observed class differences are due to an individual mode or a coupled mode. Thus, K1 and K2 are better in terms of explaining the results. For Case 8, in most cases, cross-validation across a range of weight parameters for K1 and K2 yields $w_2 = 1$ and $w_3 = 1$, respectively, and the remaining weights are equal to zero. This directly identifies the source of discriminability and allows for better interpretability, which is not possible for K3 and K4. Finally, K1 has less number of parameters than K2 and this can be advantageous in cases with high number of modalities. The smaller number of parameters make cross-validation simpler, while still allowing for some interpretability.

Table 5.3: Classification accuracy using different kernel combinations.

	K1	K2	K3	K4
Case 9	0.91 ± 0.036	0.9 ± 0.043	0.87 ± 0.038	0.82 ± 0.045
Case 8	0.90 ± 0.039	0.9 ± 0.038	0.88 ± 0.036	0.83 ± 0.06

5.5.3 EEG-fMRI Data

In this section, we present the application of the proposed method on simultaneous EEG-fMRI data. The simultaneous electroencephalography (EEG) with functional magnetic resonance imaging (fMRI) is one of the most popular non-invasive multimodal brain imaging techniques to study human brain function. EEG records electrical activity from the scalp resulting from ionic current within the neurons of the brain. Its millisecond temporal resolution makes it possible to record event-related potentials that occur in response to visual, auditory and sensory stimuli ([172, 1]). While EEG provides high temporal resolution, its spatial resolution is limited by the number of electrodes placed on the scalp and thus provides less spatial resolu-

tion compared to other neuroimaging modalities such as magnetic resonance imaging (MRI) and Positron Emission Tomography (PET). As a result, it has been commonplace to record EEG data in conjunction with a high spatial resolution modality. As another powerful tool in studying human brain function, blood oxygenation level dependent (BOLD) functional magnetic resonance imaging (fMRI) provides signals with much higher spatial resolution to reflect hemodynamic changes in blood oxygenation level at all voxels related to neuronal activities ([138, 15, 101, 62]). Recording simultaneous EEG and fMRI can provide high resolution information at both the spatial and temporal dimensions at the same time. Thus, developing novel machine learning techniques to utilize such multimodal data is of great significance. In this application, we apply our C-STM model to a binary trial classification problem on a simultaneous EEG-fMRI data.

The data is obtained from the study [178]. In this study, there are seventeen individuals (six females, average age 27.7) participated in three runs each of analogous visual and auditory oddball paradigms. The 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 stimuli and make responses. In the visual task, a large red circle on isoluminant gray backgrounds was considered as the target stimuli, and a small green circle were the standard stimuli. 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 EEG and fMRI 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 class stimulus types in all the trials. Through our numerical study, we want to demonstrate the fact our C-STM model enjoys the advantage of data multimodality and provides more accurate class predictions. The data from Subject 4 are dropped since its fMRI data are corrupted.

We pre-process both the EEG and fMRI data with Statistical Parametric Mapping (SPM 12) ([11]) and Matlab. The EEG data is collected by a custom built MR-compatible EEG system with 49 channels. [178] provides a version of re-referenced EEG data with 34 channels which are used in our experiment. This version of EEG data are sampled at 1,000 Hz, and are downsampled to 200 Hz at the beginning of pre-processing. We then remove both low-frequency and high-frequency noise in the data using SPM filter functions. As the last step of EEG pre-processing, we define trials from Brain Imaging Data Structure (BIDS) files [136] and extract EEG data epochs recorded within the trial-related time windows. The time window for each trial is considered to go from 100 ms before the stimulus onset until 500 ms after the stimulus. For each trial,

we construct a three-mode tensor corresponding to the EEG data for all subjects where the modes represent channel × time × subject. We denote it as $\mathcal{X}_{t,1} \in \mathbb{R}^{34\times121\times16}$. The fMRI data is collected by 3T Philips Achieva MR Scanner with 170 volumes (TR = 2s) per session. Each 3D volume contains 32 slices. The voxel size in the image is 3 x 3 x 4 mm. For each subject, we realign all the fMRI volumes from multiple sessions to the mean volume, and co-register the participant's T1 weighted anatomical scan to the mean fMRI volume. Next, we normalize all the fMRI volumes to match the MNI brain template ([102]) by creating segments from co-registered T1 weighted scan, and keep the voxel size as 3 x 3 x 4 mm. All normalized fMRI volumes are then smoothed by 3D Gaussian kernels with full width at half maximum (FWHM) parameter being $8 \times 8 \times 8$. After the pre-processing, we further perform a regular statistical analysis ([119, 190]) to extract fMRI volumes from visual and auditory stimulus related voxels. Such data are also known as Region of Interest (ROI) data. We extract fMRI volumes from 178 voxels (in Figure 5.4a) for auditory oddball tasks, and 112 voxels for auditory tasks. As a result, fMRI data are modeled by matrices whose rows and columns stand for voxels and subjects: $\mathcal{X}_{t,2} \in \mathbb{R}^{16\times178}$ for auditory task data, and $\mathcal{X}_{t,2} \in \mathbb{R}^{16\times112}$ for visual task data. There is no time mode in fMRI data because the trial duration is less than the repetition time of fMRI (time for obtaining a single 3D volume fMRI). For each trial, there is only one 3D scan of fMRI collected from a single subject. The ROI data then becomes a vector for this subject in the trial as we extract volumes from the regions of interest.

To classify trials with oddball and standard stimulus, we collect 140 multimodal data samples $(\mathcal{X}_{t,1}, \mathcal{X}_{t,2})$ from auditory tasks, and 100 samples from visual tasks. For both types of tasks, the 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. Like the procedures in our 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), and specificity (true negative rate) of classifiers are calculated using the test set at each experiment. The experiment is repeated multiple times, and the average accuracy, precision, sensitivity, and specificity, and their standard deviations (in subscripts) are reported in Table 5.4. The single mode classifiers CPSTM, CMDA, and DGTDA are also applied on either EEG or fMRI data as a comparison. The single mode classifiers applied on EEG data are denoted by appending the number "1" after their names, and those applied on fMRI data are denoted by appending the number "2". The area under the curve (AUC) for all the classifiers are also reported in Table 5.4.

The results in Table 5.4 show that the trial classification accuracy for C-STM using multimodal data

Table 5.4: Real Data Result: Simultaneous EEG-fMRI Data Trial Classification (Mean of Performance Metrics with Standard Deviations in Subscripts)

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	$0.64_{0.10}$	$0.60_{0.13}$	$0.86_{0.10}$	$0.44_{0.18}$	$0.64_{0.07}$

is better 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 classification of both auditory and visual tasks. This observation agrees to the conclusion from our simulation study. Similar to our simulation study, the tensor discriminant analysis does not work as well as CP-STM and C-STM. In addition, it is obvious that the performance of tensor discriminant analysis using fMRI data are better than using EEG data. This is within our expectation, since the regions we extracted from fMRI data are identified by group level fMRI statistical analysis. The data in these regions have already shown significant differences between different trials in the traditional study, and thus are easy to classify. On the other hand, there is no prior analysis and feature extraction procedure applied on EEG data, leaving a low signal to noise ratio in EEG data. However, C-STM still can take advantage of using EEG data and further increase the classification accuracy, highlighting its robustness and potential in processing noisy mulitmodal tensor data.

5.6 Conclusion

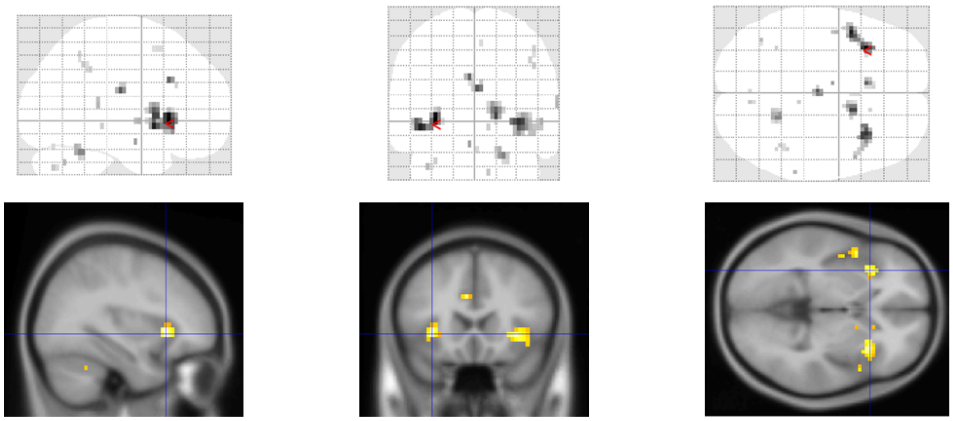
In this chapter, we have proposed a novel coupled support tensor machine classifier for multimodal data by combining the advanced coupled matrix tensor factorization (ACMTF) and support tensor machine (STM). The most distinctive feature of this classifier is its ability to integrate features across different modalities and structures. The proposed approach can simultaneously take matrix- and tensor-shaped data for classification

and can be easily extended to inputs with more than two modality. The coupled tensor matrix decomposition unveils the intrinsic correlation structure between data across different modalities, making it possible to integrate information from multiple sources efficiently. Such a decomposition also makes the whole method robust and applicable to large-scale noisy data with missing values. The newly designed kernel functions in C-STM provide feature-level information fusion, combining discriminant information from different modalities. Moreover, the kernel formulation makes it possible to utilize the most discriminative features from each modality by tuning the weight parameters in the function.

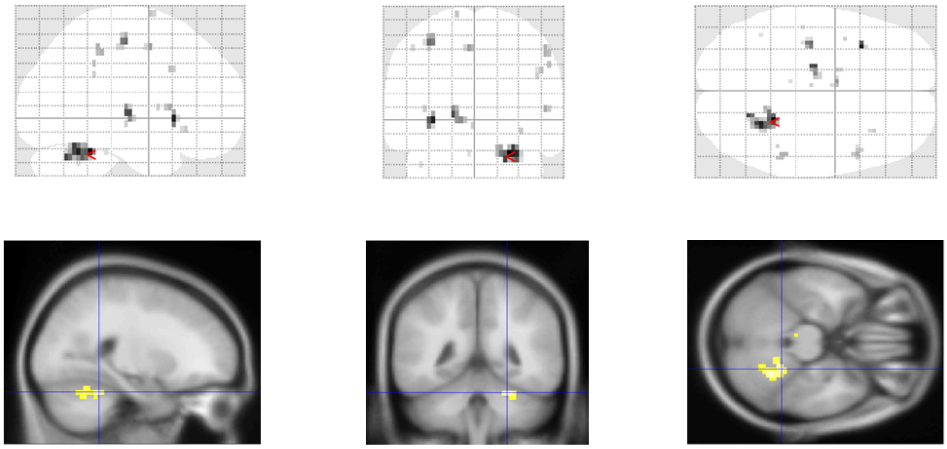
The most important theoretical extension of our current approach would be the development of excess risk for C-STM. In particular, we are looking for an explicit expression for the excess risk in terms of data factors from multiple modalities to quantify the contribution of every single modality in minimizing 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 factors estimation and their impact on the excess risk will build the foundation to the next level.

Future work will focus on learning the weight parameters in the kernel function via optimization. As [71] introduced, the weights in the kernel function can be further estimated by including a group lasso penalty in the objective function. This procedure allows the identification of the most significant components and reduce the cost of parameter selection. Finally, the proposed method can be extended to multimodal data with more than two modalities, and for regression problems.

In conclusion, we believe C-STM offers many encouraging possibilities for multimodal data integration and analysis. Its capability of handling multimodal tensor inputs will make it appropriate in many advanced data applications in neuroscience and medical research. We anticipate that this method will play an important role in a variety of applications.



(a) Auditory Task



(b) Visual Task

Figure 5.4: Region of Interest (ROI)

CHAPTER 6

CONCLUSIONS

In this thesis, we introduced new tensor based machine learning models using various data structures. In particular, we utilized tensor network structures, geometric models and multi-modal coupling for efficient tensor based unsupervised and supervised learning. The proposed methods in the thesis contribute significantly to the tensor learning literature by improving existing structures and using tensors to model new problems.

In Chapter 2, we proposed a tensor decomposition structure, i.e., Multi Branch Tensor Network, that provides improved storage and computational efficiency compared to existing architectures, without sacrificing representation power of the low-dimensional approximations. We explored two applications of multi-branch structure in supervised and unsupervised settings and provided theoretical analysis of convergence, computational and storage complexities. In the supervised setting, a multi-branch implementation of LDA was introduced. The proposed approach reduced the computational complexity by orders of magnitude, while improving the classification accuracy compared to vector-, Tucker- or TT-based methods. We also demonstrated that the proposed approach is more general compared to Tucker based MDA methods, TT-based BTT and MPS, and provided a way of selecting the optimal structure depending on data size. The proposed methods show robustness against small-sample-size (SSS) problem, which is considered to be the main drawback of LDA [63], as it learns smaller number of parameters due to the efficient multi-branch structure. The proposed Multi-branch methods can be further utilized in any Ritz pair (extremal eigenvalue-eigenvector pair) computation. In the unsupervised setting, the proposed structure was used for a graph regularized optimization problem to reduce the computational complexity and learn more effective subspaces for dimensionality reduction. Low-dimensional projections, or features of the data are then used for clustering. Using multi branch structure reduced the computational complexity for the optimization by orders of magnitude, especially when the projected space was larger in dimension. This is a crucial as, often, using a small number of features does not provide acceptable clustering quality or other learning objective functions. The proposed method outperformed Tucker-based graph regularized method in terms of clustering quality, hence, suggesting that multi-branch decompositions could learn subspaces that better fit the data compared to Tucker based methods even in a manifold learning setting.

In Chapter 3, we present a framework for robust tensor decomposition for anomaly detection in spa-

tiotemporal data. In particular, we focus on urban traffic monitoring applications where the anomalies exhibit themselves as temporally contiguous events. Motivated by this application, we model spatiotemporal data as a low-rank plus sparse tensor where the low-rank part corresponds to underlying data and sparse part corresponds to anomalies. The continuous nature of the anomalies are taken into account with the additional constraint that the sparse part is temporally continuous (LOSS). We explore two extensions of this model. First, to capture the underlying local geometric relations in the data, we consider graph regularization across each mode (GLOSS). This type of regularization is motivated by the assumption that the data lies on a product graph, which holds true for many real world scenarios, since real world data is often generated over graphs with such structures. Specifically, for traffic data, this structure is fitting by design. Second, we modify our objective function by minimizing the graph total variation instead of nuclear norm for low-rank approximation to reduce the computational complexity (LOGSS). Finally, we incorporate a tensor completion framework in all of these methods to address missing data. The proposed methods are shown to improve anomaly detection performance compared to baseline methods and have higher accuracy than existing robust tensor decomposition methods. In particular, using graph regularization in the objective improves the results significantly even when the nuclear norm penalties are dropped. This shows that using geometric structure instead of global structure such as rank can be used to obtain highly accurate and low complexity tensor models.

In Chapter 4, we propose incorporating geometric structure in addition to global structure into tensor denoising and recovery formulations. Geometric tensor learning allows for better modeling of underlying relations of data compared to purely algebraic measures. We propose a TT based robust PCA model where a graph smoothness penalty is applied to each mode-n canonical unfolding. Since this formulation results in computationally intractable matrix inversion problems, we propose an extension where we impose a Kronecker structure on the mode-n canonical graph. This structure is designed such that the the redundancy across the different mode unfoldings is minimized. We prove the equivalence between the proposed graph smoothness penalty with mode-n unfolding based graph smoothness penalty. We show the advantage of using geometric approaches by comparing these two methods with purely algebraic objective functions. The proposed methods outperform the existing methods in missing data imputation, and denoising tasks. Moreover, using the Kronecker structured graph rather than the canonical graph provided similar results with improved computational efficiency. The results in this chapter illustrate that with additional regularizations on topological structure, TT-based models can be further improved.

Finally, in Chapter 5, we utilize coupled tensor decomposition in a supervised setting. Heterogenous data collected by sensing the same physical phenomena are generally coupled. Coupled tensor decomposition methods have been utilized extensively for unsupervised learning tasks such as denoising, recovery and clustering such heterogeneous data. In this chapter, we showed that these models can also be utilized in supervised settings, for robust dimensionality reduction and feature extraction. To combine representations from different sources, we proposed using multiple kernel learning methods. MKL uses a combination of single kernels, that can take inputs from various data sources, to obtain better results and avoid potential bias from kernel selection [71]. We propose a two-step model where in the first step, factor matrices for all samples are extracted by coupling heterogeneous data sources. In the next step, we feed the extracted features, i.e., factor matrices, to a kernelized support tensor machine. The proposed method shows better classification accuracy compared to supervised learning based on the individual data sources. Coupled decomposition extracts the features from each sample using information from both data sources, which provides better features for the subsequent STM and improves the accuracy compared to STM trained on the features of individual modalities. The proposed method also outperforms methods that specifically learn discriminative factors from individual modalities, which illustrates the advantage of coupling.

6.1 Future Work

The work in this thesis suggests new research directions. In this section, we suggest areas of future work for each chapter.

6.1.1 Multi-Branch Tensor Learning

The proposed tensor decomposition structure is a hybrid between Tucker and TT decompositions, hence, is generally applicable in any tensor decomposition problem. Thus, the structure can be utilized in many other supervised and unsupervised tasks, such as regression, STMs, dictionary learning, manifold learning, data recovery and compression. It can also be utilized in improving tensor-based deep learning methods, by either compressing the data, or the network parameters as it provides an optimal way of decomposing large tensors.

Theoretical aspects of the representation power of this structure is also of interest as it would lead to a deeper understanding of tensor decompositions. Since Tucker and TT decompositions are special cases of the multi-branch structure, this analysis would also reveal a concrete reasoning for the choice of the

decomposition structure depending on the problem, and data at hand.

6.1.2 Tensor Methods for Anomaly Detection

Chapter 3 reveals the utility of tensor based robust learning architectures on the problem of anomaly detection, specifically for spatiotemporal data. However, there are still some open questions regarding the theoretical recovery guarantees for the proposed algorithms. Future work can explore recovery bounds for anomalies depending on the data structure. Furthermore, the proposed methods rely on the proper selection of the regularization parameters. Although we have shown empirically that the performance is robust within a wide range of parameter choices, these methods still require some tuning for desirable performance. A fully Bayesian extension of the proposed approach could be considered as future work to automatically estimate these parameters depending on the data.

Anomaly detection part in this chapter was implemented by scoring each fiber individually by a separate algorithm, which may result in loss of information. Future work will consider a statistical tensor anomaly scoring method to avoid this simplification. Finally, since the specific application in this chapter is spatiotemporal data, a natural extension would be online anomaly detection. Online subspace tracking or functional data analysis literature may provide the necessary tools for such an extension.

6.1.3 Geometric Tensor Learning

In Chapter 4, we illustrate how using geometric relations within data improve robust tensor learning. Future work would quantify these improvements in terms of theoretical bounds for recovery when such a structure is utilized. Although we utilize a combination of global and local structure in our methods, it is also of interest to use only the local structure, i.e., geometric relations for data recovery. Recovery guarantees for such an approach is also of great interest as it might allow milder conditions compared to algorithms that use global structure. In this chapter, we estimated the underlying graphs using a k-NN approach. It is known in graph learning literature that with noisy and missing data, this generally produces noisy estimates of the underlying graph. As such, future work will focus on simultaneous graph learning with data recovery.

Finally, as the use of graphs corresponding to the canonical mode-n unfolding requires excessive memory and computational resources, a multi-branch structure could be utilized to reduce these costs without approximating the graphs with a Kronecker structure. This would also pave the way for a fully geometric

robust multi-branch decomposition.

6.1.4 Supervised Coupled Tensor Learning

In Chapter 5, coupled tensor factorization was utilized as a feature extraction step. However, without using the class label information, the extracted features might not be suitable for the subsequent classification task. In this approach, we propose to employ a supervised coupled factorization to address this. Specifically, we extend Multilinear Discriminant Analysis to coupled factorization by solving:

$$\text{minimize}_{U_n^{(i)}\,\forall n,i} \sum_{i=1}^{2} \sum_{n=1}^{d_i-1} \frac{tr(U_n^{(i),\top} S_W^{i,n} U_n^{(i)})}{tr(U_n^{(i),\top} S_B^{i,n} U_n^{(i)})} + \sum_{i=1}^{2} \sum_{n\in\mathfrak{N}} \|A_n^i \hat{U}_i - V_n\|_F^2, \tag{6.1}$$

where $S_W^{i,n}$, $S_B^{i,n}$ are within and between class scatters of modality i and mode-n; $\mathfrak{N}$ is the set of coupled modes, and A_n's are transformations through which the couplings are defined. The transformations are used to explain the difference in properties and resolutions across different modalities even when they correspond to similar phenomenon.

To classify a test sample, one can use Mahalanobis distance with respect to class means and covariance. Another approach would be to train a classifier using the sample mode factors Y that are learned through a least squares optimization and similar procedures with Approach 1 can be utilized for test cases.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Rodolfo Abreu, Alberto Leal, and Patrícia Figueiredo. "EEG-informed fMRI: a review of data analysis methods". In: *Frontiers in human neuroscience* 12 (2018), p. 29.

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

[3] Evrim Acar et al. "ACMTF for fusion of multi-modal neuroimaging data and identification of biomarkers". In: *2017 25th European Signal Processing Conference (EUSIPCO)*. IEEE. 2017, pp. 643–647.

[4] Evrim Acar et al. "Structure-revealing data fusion". In: *BMC bioinformatics* 15.1 (2014), pp. 1–17.

[5] Evrim Acar et al. "Tensor-based fusion of EEG and FMRI to understand neurological changes in schizophrenia". In: *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2017, pp. 1–4.

[6] Evrim Acar et al. "Unraveling diagnostic biomarkers of schizophrenia through structure-revealing fusion of multi-modal neuroimaging data". In: *Frontiers in neuroscience* 13 (2019), p. 416.

[7] Hemant Kumar Aggarwal and Angshul Majumdar. "Hyperspectral image denoising using spatio-spectral total variation". In: *IEEE Geoscience and Remote Sensing Letters* 13.3 (2016), pp. 442–446.

[8] Anima Anandkumar et al. "Tensor vs. matrix methods: Robust tensor decomposition under block sparse perturbations". In: *Artificial Intelligence and Statistics*. PMLR. 2016, pp. 268–276.

[9] Jeffrey S Anderson et al. "Functional connectivity magnetic resonance imaging classification of autism". In: *Brain* 134.12 (2011), pp. 3742–3754.

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

[11] John Ashburner et al. "SPM12 manual". In: *Wellcome Trust Centre for Neuroimaging, London, UK* 2464 (2014).

[12] Francis R Bach. "Consistency of the group lasso and multiple kernel learning." In: *Journal of Machine Learning Research* 9.6 (2008).

[13] Mohammad Taha Bahadori, Qi Rose Yu, and Yan Liu. "Fast Multivariate Spatio-temporal Analysis via Low Rank Tensor Learning." In: *NIPS*. Citeseer. 2014, pp. 3491–3499.

[14] Richard H. Bartels and George W Stewart. "Solution of the matrix equation AX+ XB= C [F4]". In: *Communications of the ACM* 15.9 (1972), pp. 820–826.

[15] JW Belliveau et al. “Functional mapping of the human visual cortex by magnetic resonance imaging”. In: *Science* 254.5032 (1991), pp. 716–719.

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

[17] Johann A Bengua et al. “Efficient tensor completion for color image and video recovery: Low-rank tensor train”. In: *IEEE Transactions on Image Processing* 26.5 (2017), pp. 2466–2479.

[18] Johann A Bengua et al. “Matrix product state for higher-order tensor compression and classification”. In: *IEEE Transactions on Signal Processing* 65.15 (2017), pp. 4019–4030.

[19] 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*. 2002, pp. 24–31.

[20] Gouri Sankar Bhunia et al. “Spatial and temporal variation and hotspot detection of kala-azar disease in Vaishali district (Bihar), India”. In: *BMC infectious diseases* 13.1 (2013), p. 64.

[21] Xuan Bi et al. “Tensors in statistics”. In: *Annual Review of Statistics and Its Application* 8 (2020).

[22] Amit Boyarski, Sanketh Vedula, and Alex Bronstein. “Deep matrix factorization with spectral geometric regularization”. In: *arXiv preprint arXiv: 1911.07255* (2019).

[23] Markus M Breunig et al. “LOF: identifying density-based local outliers”. In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 2000, pp. 93–104.

[24] Laura F Bringmann et al. “Changing dynamics: Time-varying autoregressive models using generalized additive modeling.” In: *Psychological methods* 22.3 (2017), p. 409.

[25] Laura F Bringmann et al. “Modeling nonstationary emotion dynamics in dyads using a time-varying vector-autoregressive model”. In: *Multivariate behavioral research* 53.3 (2018), pp. 293–314.

[26] Rasmus Bro. “PARAFAC. Tutorial and applications”. In: *Chemometrics and Intelligent Laboratory Systems* 38.2 (1997), pp. 149–171.

[27] Rasmus Bro, Claus A Andersson, and Henk AL Kiers. “PARAFAC2—Part II. Modeling chromatographic data with retention time shifts”. In: *Journal of Chemometrics: A Journal of the Chemometrics Society* 13.3-4 (1999), pp. 295–309.

[28] Saikiran Bulusu et al. “Anomalous example detection in deep learning: A survey”. In: *IEEE Access* 8 (2020), pp. 132330–132347.

[29] Vince D Calhoun et al. “Method for multimodal analysis of independent source differences in schizophrenia: combining gray matter structural and auditory oddball functional data”. In: *Human brain mapping* 27.1 (2006), pp. 47–62.

[30] E. J. Candès et al. “Robust principal component analysis?” In: *Journal of the ACM (JACM)* 58.3

(2011), p. 11.

[31] Clayson Celes, Azzedine Boukerche, and Antonio AF Loureiro. "Crowd Management: A New Challenge for Urban Big Data Analytics". In: *IEEE Communications Magazine* 57.4 (2019), pp. 20–25.

[32] Mohammadhossein Chaghazardi and Shuchin Aeron. "Sample, computation vs storage tradeoffs for classification using tensor subspace models". In: *arXiv preprint arXiv:1706.05599* (2017).

[33] Raghavendra Chalapathy and Sanjay Chawla. "Deep learning for anomaly detection: A survey". In: *arXiv preprint arXiv:1901.03407* (2019).

[34] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM computing surveys (CSUR)* 41.3 (2009), pp. 1–58.

[35] Venkat Chandrasekaran et al. "Rank-sparsity incoherence for matrix decomposition". In: *SIAM Journal on Optimization* 21.2 (2011), pp. 572–596.

[36] Christos Chatzichristos et al. "Early soft and flexible fusion of EEG and fMRI via tensor decompositions". In: *arXiv preprint arXiv:2005.07134* (2020).

[37] Christos Chatzichristos et al. "Fusion of EEG and fMRI via soft coupled tensor decompositions". In: *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE. 2018, pp. 56–60.

[38] Cong Chen et al. "A support tensor train machine". In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2019, pp. 1–8.

[39] Longbiao Chen et al. "Fine-grained urban event detection and characterization based on tensor cofactorization". In: *IEEE Transactions on Human-Machine Systems* 47.3 (2016), pp. 380–391.

[40] Xinyu Chen et al. "Missing traffic data imputation and pattern discovery with a Bayesian augmented tensor factorization model". In: *Transportation Research Part C: Emerging Technologies* 104 (2019), pp. 66–77.

[41] Mario Christoudias, Raquel Urtasun, Trevor Darrell, et al. "Bayesian localized multiple kernel learning". In: *Univ. California Berkeley, Berkeley, CA* (2009).

[42] A. Cichocki. "Era of big data processing: A new approach via tensor networks and tensor decompositions". In: *arXiv preprint arXiv:1403.2048* (2014).

[43] A. Cichocki et al. "Tensor decompositions for signal processing applications: From two-way to multiway component analysis". In: *IEEE Signal Processing Magazine* 32.2 (2015), pp. 145–163.

[44] Andrzej Cichocki et al. "Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions". In: *Foundations and Trends® in Machine Learning* 9.4-5 (2016), pp. 249–429.

[45] Andrzej Cichocki et al. "Tensor networks for dimensionality reduction and large-scale optimization:

Part 2 applications and future perspectives". In: *Foundations and Trends® in Machine Learning* 9.6 (2017), pp. 431–673.

[46] Lieven De Lathauwer and Joséphine Castaing. "Blind identification of underdetermined mixtures by simultaneous matrix diagonalization". In: *IEEE Transactions on Signal Processing* 56.3 (2008), pp. 1096–1105.

[47] Lieven De Lathauwer, Josphine Castaing, and Jean-Franois Cardoso. "Fourth-order cumulant-based blind identification of underdetermined mixtures". In: *IEEE Transactions on Signal Processing* 55.6 (2007), pp. 2965–2973.

[48] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. "A multilinear singular value decomposition". In: *SIAM journal on Matrix Analysis and Applications* 21.4 (2000), pp. 1253–1278.

[49] 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". In: *SIAM journal on Matrix Analysis and Applications* 21.4 (2000), pp. 1324–1342.

[50] Dingxiong Deng et al. "Latent space model for road networks to predict time-varying traffic". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 1525–1534.

[51] Lei Deng et al. "Graph Spectral Regularized Tensor Completion for Traffic Data Imputation". In: *IEEE Transactions on Intelligent Transportation Systems* (2021).

[52] Wei Deng and Wotao Yin. "On the global and linear convergence of the generalized alternating direction method of multipliers". In: *Journal of Scientific Computing* 66.3 (2016), pp. 889–916.

[53] Renwei Dian, Shutao Li, and Leyuan Fang. "Learning a low tensor-train rank representation for hyperspectral image super-resolution". In: *IEEE transactions on neural networks and learning systems* 30.9 (2019), pp. 2672–2683.

[54] Yiming Ding et al. "A deep learning model to predict a diagnosis of Alzheimer disease by using 18F-FDG PET of the brain". In: *Radiology* 290.2 (2019), pp. 456–464.

[55] Youcef Djenouri et al. "A survey on urban traffic anomalies detection algorithms". In: *IEEE Access* 7 (2019), pp. 12192–12205.

[56] Sergey V Dolgov et al. "Computation of extreme eigenvalues in higher dimensions using block tensor train format". In: *Computer Physics Communications* 185.4 (2014), pp. 1207–1216.

[57] Haishun Du et al. "Sparse representation-based robust face recognition by graph regularized low-rank sparse representation recovery". In: *Neurocomputing* 164 (2015), pp. 220–229.

[58] James H Faghmous et al. "A parameter-free spatio-temporal pattern mining model to catalog global ocean dynamics". In: *2013 IEEE 13th International Conference on Data Mining*. IEEE. 2013, pp. 151–160.

[59] Hadi Fanaee-T and João Gama. "Tensor-based anomaly detection: An interdisciplinary survey". In: *Knowledge-Based Systems* 98 (2016), pp. 130–147.

[60] Hadi Fanaee-T and Joao Gama. "Event detection from traffic tensors: A hybrid model". In: *Neurocomputing* 203 (2016), pp. 22–33.

[61] Hadi Fanaee-T and Joao Gama. "SimTensor: A synthetic tensor data generator". In: *arXiv preprint arXiv:1612.03772* (2016).

[62] Massimo Filippi, Roland Bammer, et al. *MR imaging in white matter diseases of the brain and spinal cord*. Springer, 2005.

[63] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.

[64] Glenn Fung et al. "A fast iterative algorithm for fisher discriminant using heterogeneous kernels". In: *Proceedings of the twenty-first international conference on Machine learning*. 2004, p. 40.

[65] Mostafa Reisi Gahrooei et al. "Multiple tensor-on-tensor regression: an approach for modeling processes with heterogeneous sources of data". In: *Technometrics* 63.2 (2021), pp. 147–159.

[66] Giovana Gavidia-Bovadilla et al. "Early prediction of Alzheimer's disease using null longitudinal model-based classifiers". In: *PloS one* 12.1 (2017), e0168011.

[67] Matan Gavish and Ronald R Coifman. "Sampling, denoising and compression of matrices by coherent matrix organization". In: *Applied and Computational Harmonic Analysis* 33.3 (2012), pp. 354–369.

[68] Xiurui Geng et al. "A high-order statistical tensor based algorithm for anomaly detection in hyperspectral imagery". In: *Scientific reports* 4 (2014), p. 6869.

[69] 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*. Vol. 19. MIT Press. 2007, p. 465.

[70] Donald Goldfarb and Zhiwei Qin. "Robust low-rank tensor recovery: Models and algorithms". In: *SIAM Journal on Matrix Analysis and Applications* 35.1 (2014), pp. 225–253.

[71] Mehmet Gönen and Ethem Alpaydın. "Multiple kernel learning algorithms". In: *The Journal of Machine Learning Research* 12 (2011), pp. 2211–2268.

[72] Adrian R Groves et al. "Linked independent component analysis for multimodal data fusion". In: *Neuroimage* 54.3 (2011), pp. 2198–2217.

[73] Weiwei Guo, Irene Kotsia, and Ioannis Patras. "Tensor learning for regression". In: *IEEE Transactions on Image Processing* 21.2 (2011), pp. 816–827.

[74] Xian Guo et al. "Support tensor machines for classification of hyperspectral remote sensing imagery". In: *IEEE Transactions on Geoscience and Remote Sensing* 54.6 (2016), pp. 3248–3264.

[75] Zhifeng Hao et al. “A linear support higher-order tensor machine for classification”. In: *IEEE Transactions on Image Processing* 22.7 (2013), pp. 2911–2920.

[76] Lifang He et al. “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*. SIAM. 2014, pp. 127–135.

[77] Lifang He et al. “Kernelized support tensor machines”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 1442–1451.

[78] Xiaofei He, Deng Cai, and Partha Niyogi. “Tensor subspace analysis”. In: *Advances in neural information processing systems*. 2006, pp. 499–506.

[79] Victoria Hodge and Jim Austin. “A survey of outlier detection methodologies”. In: *Artificial intelligence review* 22.2 (2004), pp. 85–126.

[80] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. “On manifolds of tensors of fixed TT-rank”. In: *Numerische Mathematik* 120.4 (2012), pp. 701–731.

[81] Yuwang Ji et al. “A Survey on Tensor Techniques and Applications in Machine Learning”. In: *IEEE Access* 7 (2019), pp. 162950–162990.

[82] Bo Jiang et al. “Image representation and learning with graph-laplacian tucker tensor decomposition”. In: *IEEE transactions on cybernetics* 49.4 (2018), pp. 1417–1426.

[83] Taisong Jin et al. “Low-rank matrix factorization with multiple hypergraph regularizer”. In: *Pattern Recognition* 48.3 (2015), pp. 1011–1022.

[84] Vassilis Kalofolias et al. “Matrix completion on graphs”. In: *arXiv preprint arXiv:1408.1717* (2014).

[85] Maja H Kamstrup-Nielsen, Lea G Johnsen, and Rasmus Bro. “Core consistency diagnostic in PARAFAC2”. In: *Journal of Chemometrics* 27.5 (2013), pp. 99–105.

[86] Esin Karahan et al. “Tensor analysis and fusion of multimodal brain images”. In: *Proceedings of the IEEE* 103.9 (2015), pp. 1531–1559.

[87] Hiroyuki Kasai. “Fast online low-rank tensor subspace tracking by CP decomposition using recursive least squares from incomplete observations”. In: *Neurocomputing* 347 (2019), pp. 177–190.

[88] Hiroyuki Kasai, Wolfgang Kellerer, and Martin Kleinsteuber. “Network volume anomaly detection and identification in large-scale networks based on online time-structured traffic tensor tracking”. In: *IEEE Transactions on Network and Service Management* 13.3 (2016), pp. 636–650.

[89] 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”. In: *Brain imaging and behavior* 10.3 (2016), pp. 799–817.

[90] Boris N Khoromskij. “O (dlog N)-quantics approximation of N-d tensors in high-dimensional

numerical modeling". In: *Constructive Approximation* 34.2 (2011), pp. 257–280.

[91] Tae-Kyun Kim, Shu-Fai Wong, and Roberto Cipolla. "Tensor canonical correlation analysis for action classification". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2007, pp. 1–8.

[92] Marius Kloft et al. "Efficient and accurate lp-norm multiple kernel learning." In: *NIPS*. vol. 22. 22. 2009, pp. 997–1005.

[93] Tamara G Kolda and Brett W Bader. "Tensor decompositions and applications". In: *SIAM review* 51.3 (2009), pp. 455–500.

[94] Xiangjie Kong et al. "HUAD: Hierarchical urban anomaly detection based on spatio-temporal data". In: *IEEE Access* 8 (2020), pp. 26573–26582.

[95] Jean Kossaifi et al. "T-net: Parametrizing fully convolutional nets with a single high-order tensor". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 7822–7831.

[96] Jean Kossaifi et al. "Tensor regression networks". In: *arXiv preprint arXiv:1707.08308* (2017).

[97] Irene Kotsia, Weiwei Guo, and Ioannis Patras. "Higher rank support tensor machines for visual recognition". In: *Pattern Recognition* 45.12 (2012), pp. 4192–4203.

[98] Daniel Kressner, Michael Steinlechner, and André Uschmajew. "Low-rank tensor methods with subspace correction for symmetric eigenvalue problems". In: *SIAM Journal on Scientific Computing* 36.5 (2014), A2346–A2368.

[99] J. B. Kruskal. "Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics". In: *Linear Algebra and its Applications* 18.2 (1977), pp. 95–138. ISSN: 0024-3795. DOI: https://doi.org/10.1016/0024-3795(77)90069-6. URL: http://www.sciencedirect.com/science/article/pii/0024379577900696.

[100] Alp Kut and Derya Birant. "Spatio-temporal outlier detection in large databases". In: *Journal of computing and information technology* 14.4 (2006), pp. 291–297.

[101] Kenneth K Kwong et al. "Dynamic magnetic resonance imaging of human brain activity during primary sensory stimulation." In: *Proceedings of the National Academy of Sciences* 89.12 (1992), pp. 5675–5679.

[102] Jack L Lancaster et al. "Bias between MNI and Talairach coordinates analyzed using the ICBM-152 brain template". In: *Human brain mapping* 28.11 (2007), pp. 1194–1205.

[103] Gert RG Lanckriet et al. "Learning the kernel matrix with semidefinite programming". In: *Journal of Machine learning research* 5.Jan (2004), pp. 27–72.

[104] Gisela Lechuga et al. "Discriminant analysis for multiway data". In: *International Conference on Partial Least Squares and Related Methods*. Springer. 2014, pp. 115–126.

[105] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. "Trajectory clustering: a partition-and-group framework". In: *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. 2007, pp. 593–604.

[106] Namgil Lee et al. "Nonnegative Tensor Train Decompositions for Multi-domain Feature Extraction and Clustering". In: *International Conference on Neural Information Processing*. Springer. 2016, pp. 87–95.

[107] 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". In: *Journal of integrative neuroscience* 11.03 (2012), pp. 313–337.

[108] Li Li et al. "Trend modeling for traffic time series analysis: An integrated study". In: *IEEE Transactions on Intelligent Transportation Systems* 16.6 (2015), pp. 3430–3439.

[109] Peide Li and Taps Maiti. "Universal Consistency of Support Tensor Machine". In: *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE. 2019, pp. 608–609.

[110] Ping Li et al. "Online robust low-rank tensor modeling for streaming data analysis". In: *IEEE transactions on neural networks and learning systems* 30.4 (2018), pp. 1061–1075.

[111] Quefeng Li and Lexin Li. "Integrative factor regression and its inference for multimodal data analysis". In: *arXiv preprint arXiv:1911.04056* (2019).

[112] Qun Li and Dan Schonfeld. "Multilinear discriminant analysis for higher-order tensor data classification". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.12 (2014), pp. 2524–2537.

[113] Shuangjiang Li et al. "Low-rank tensor decomposition based anomaly detection for hyperspectral imagery". In: *2015 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2015, pp. 4525–4529.

[114] Xiaoshan Li et al. "Tucker tensor regression and neuroimaging analysis". In: *Statistics in Biosciences* 10.3 (2018), pp. 520–545.

[115] Xutao Li et al. "MR-NTD: Manifold regularization nonnegative tucker decomposition for tensor data dimension reduction and representation". In: *IEEE transactions on neural networks and learning systems* 28.8 (2016), pp. 1787–1800.

[116] Yingjie Li et al. "Early prediction of Alzheimer's disease using longitudinal volumetric MRI data from ADNI". in: *Health Services and Outcomes Research Methodology* 20.1 (2020), pp. 13–39.

[117] Ziyue Li et al. "Tensor completion for weakly-dependent data on graph for metro passenger flow prediction". In: *arXiv preprint arXiv:1912.05693* (2019).

[118] Chaoguang Lin et al. "Anomaly detection in spatiotemporal data via regularized non-negative tensor analysis". In: *Data Mining and Knowledge Discovery* 32.4 (2018), pp. 1056–1073.

[119] Martin A Lindquist et al. “The statistical analysis of fMRI data”. In: *Statistical science* 23.4 (2008), pp. 439–464.

[120] Ji Liu et al. “Tensor completion for estimating missing values in visual data”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.1 (2012), pp. 208–220.

[121] Jingyu Liu et al. “Combining fMRI and SNP data to investigate connections between brain function and genetics using parallel ICA”. in: *Human brain mapping* 30.1 (2009), pp. 241–255.

[122] Siqi Liu et al. “Early diagnosis of Alzheimer’s disease with deep learning”. In: *2014 IEEE 11th international symposium on biomedical imaging (ISBI)*. IEEE. 2014, pp. 1015–1018.

[123] Eric F Lock. “Tensor-on-tensor regression”. In: *Journal of Computational and Graphical Statistics* 27.3 (2018), pp. 638–647.

[124] Xiaojing Long et al. “Prediction and classification of Alzheimer disease based on quantification of MRI deformation”. In: *PloS one* 12.3 (2017), e0173372.

[125] Canyi Lu et al. “Tensor robust principal component analysis with a new tensor nuclear norm”. In: *IEEE transactions on pattern analysis and machine intelligence* 42.4 (2019), pp. 925–938.

[126] Haiping Lu, Konstantinos N Plataniotis, and Anastasios N Venetsanopoulos. “MPCA: Multilinear principal component analysis of tensor objects”. In: *IEEE Transactions on Neural Networks* 19.1 (2008), pp. 18–39.

[127] Gal Mishne, Eric Chi, and Ronald Coifman. “Co-manifold learning with missing data”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 4605–4614.

[128] Gal Mishne et al. “Data-driven tree transforms and metrics”. In: *IEEE transactions on signal and information processing over networks* 4.3 (2017), pp. 451–466.

[129] John C Morris et al. “Pittsburgh compound B imaging and prediction of progression from cognitive normality to symptomatic Alzheimer disease”. In: *Archives of neurology* 66.12 (2009), pp. 1469–1475.

[130] Raziyeh Mosayebi and Gholam-Ali Hossein-Zadeh. “Correlated coupled matrix tensor factorization method for simultaneous EEG-fMRI data fusion”. In: *Biomedical Signal Processing and Control* 62 (2020), p. 102071.

[131] Kevin P Murphy. “Switching kalman filters”. In: (1998).

[132] Atsuhiro Narita et al. “Tensor factorization using auxiliary information”. In: *Data Mining and Knowledge Discovery* 25.2 (2012), pp. 298–324.

[133] Sameer A Nene, Shree K Nayar, and Hiroshi Murase. *Columbia Object Image Library (COIL-100)*.

[134] Luong Ha Nguyen and James-A Goulet. “Anomaly detection with the switching kalman filter for structural health monitoring”. In: *Structural Control and Health Monitoring* 25.4 (2018), e2136.

[135] Yongming Nie et al. "Graph-regularized tensor robust principal component analysis for hyperspectral image denoising". In: *Applied optics* 56.22 (2017), pp. 6094–6102.

[136] Guiomar Niso et al. "MEG-BIDS, the brain imaging data structure extended to magnetoencephalography". In: *Scientific data* 5.1 (2018), pp. 1–5.

[137] Alexander Novikov et al. "Tensorizing neural networks". In: *Advances in Neural Information Processing Systems*. 2015, pp. 442–450.

[138] Seiji Ogawa et al. "Brain magnetic resonance imaging with contrast dependent on blood oxygenation". In: *proceedings of the National Academy of Sciences* 87.24 (1990), pp. 9868–9872.

[139] Ivan V Oseledets. "Approximation of 2ˆd\times2ˆd matrices using tensor decomposition". In: *SIAM Journal on Matrix Analysis and Applications* 31.4 (2010), pp. 2130–2145.

[140] Ivan V Oseledets. "Tensor-train decomposition". In: *SIAM Journal on Scientific Computing* 33.5 (2011), pp. 2295–2317.

[141] Alp Ozdemir, Edward M Bernat, and Selin Aviyente. "Recursive tensor subspace tracking for dynamic brain network analysis". In: *IEEE Transactions on Signal and Information Processing over Networks* 3.4 (2017), pp. 669–682.

[142] Yuqing Pan, Qing Mai, and Xin Zhang. "Covariate-Adjusted Tensor Classification in High Dimensions". In: *Journal of the American Statistical Association* (2018), pp. 1–15.

[143] Evangelos Papalexakis, Konstantinos Pelechrinis, and Christos Faloutsos. "Spotting misbehaviors in location-based social networks using tensors". In: *Proceedings of the 23rd International Conference on World Wide Web*. 2014, pp. 551–552.

[144] Evangelos E Papalexakis, Alex Beutel, and Peter Steenkiste. "Network anomaly detection using co-clustering". In: *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. IEEE. 2012, pp. 403–410.

[145] Paul Pavlidis et al. "Gene functional classification from heterogeneous data". In: *Proceedings of the fifth annual international conference on Computational biology*. 2001, pp. 249–255.

[146] Nathanaël Perraudin and Pierre Vandergheynst. "Stationary signal processing on graphs". In: *IEEE Transactions on Signal Processing* 65.13 (2017), pp. 3462–3477.

[147] Shibin Qiu and Terran Lane. "A framework for multiple kernel support vector regression and its applications to siRNA efficacy prediction". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 6.2 (2008), pp. 190–199.

[148] Yuning Qiu et al. "A generalized graph regularized non-negative tucker decomposition framework for tensor data representation". In: *IEEE transactions on cybernetics* (2020).

[149] Matthew Roughan et al. "Spatio-temporal compressive sensing and internet traffic matrices (extended version)". In: *IEEE/ACM Transactions on Networking* 20.3 (2011), pp. 662–676.

[150] Peter J Rousseeuw and Katrien Van Driessen. "A fast algorithm for the minimum covariance determinant estimator". In: *Technometrics* 41.3 (1999), pp. 212–223.

[151] Aliaksei Sandryhaila and Jose MF Moura. "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure". In: *IEEE Signal Processing Magazine* 31.5 (2014), pp. 80–90.

[152] Katharina A Schindlbeck and David Eidelberg. "Network imaging biomarkers: insights and clinical applications in Parkinson's disease". In: *The Lancet Neurology* 17.7 (2018), pp. 629–640.

[153] Bernhard Schölkopf et al. "Support vector method for novelty detection". In: *Advances in neural information processing systems*. 2000, pp. 582–588.

[154] Nauman Shahid, Francesco Grassi, and Pierre Vandergheynst. "Tensor Robust PCA on Graphs". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 5406–5410.

[155] Nauman Shahid et al. "Fast robust PCA on graphs". In: *IEEE Journal of Selected Topics in Signal Processing* 10.4 (2016), pp. 740–756.

[156] Nauman Shahid et al. "Robust principal component analysis on graphs". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2812–2820.

[157] Abhishek Sharma and Maks Ovsjanikov. "Geometric Matrix Completion: A Functional View". In: *arXiv preprint arXiv:2009.14343* (2020).

[158] Lei Shi, Aryya Gangopadhyay, and Vandana P Janeja. "STenSr: Spatio-temporal tensor streams for anomaly detection and pattern discovery". In: *Knowledge and Information Systems* 43.2 (2015), pp. 333–353.

[159] Nicholas D Sidiropoulos et al. "Tensor decomposition for signal processing and machine learning". In: *IEEE Transactions on Signal Processing* 65.13 (2017), pp. 3551–3582.

[160] Age Smilde, Rasmus Bro, and Paul Geladi. *Multi-way analysis: applications in the chemical sciences*. John Wiley & Sons, 2005.

[161] Seyyid Emre Sofuoglu and Selin Aviyente. "Graph Regularized Tensor Train Decomposition". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 3912–3916.

[162] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. "UCF101: A dataset of 101 human actions classes from videos in the wild". In: *arXiv preprint arXiv:1212.0402* (2012).

[163] Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

[164] Yuting Su et al. "Graph regularized low-rank tensor representation for feature selection". In: *Journal of Visual Communication and Image Representation* 56 (2018), pp. 234–244.

[165] Jing Sui et al. “Discriminating schizophrenia and bipolar disorder by fusing fMRI and DTI in a multimodal CCA+ joint ICA model”. In: *Neuroimage* 57.3 (2011), pp. 839–855.

[166] Jimeng Sun, Dacheng Tao, and Christos Faloutsos. “Beyond streams and graphs: dynamic tensor analysis”. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2006, pp. 374–383.

[167] Hiroaki Tanabe et al. “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*. IEEE. 2008, pp. 71–78.

[168] Dacheng Tao et al. “General tensor discriminant analysis and gabor features for gait recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.10 (2007).

[169] Dacheng Tao et al. “Supervised tensor learning”. In: *Fifth IEEE International Conference on Data Mining (ICDM’05)*. IEEE. 2005, 8–pp.

[170] Liang Tao et al. “Low rank approximation with sparse integration of multiple manifolds for data representation”. In: *Applied Intelligence* 42.3 (2015), pp. 430–446.

[171] Joshua B Tenenbaum, Vin De Silva, and John C Langford. “A global geometric framework for nonlinear dimensionality reduction”. In: *science* 290.5500 (2000), pp. 2319–2323.

[172] Michal Teplan et al. “Fundamentals of EEG measurement”. In: *Measurement science review* 2.2 (2002), pp. 1–11.

[173] Ryota Tomioka, Kohei Hayashi, and Hisashi Kashima. “On the extension of trace norm to tensors”. In: *NIPS Workshop on Tensors, Kernels, and Machine Learning*. Vol. 7. 2010.

[174] Ledyard R Tucker. “Implications of factor analysis of three-way matrices for measurement of change”. In: *Problems in measuring change* 15 (1963), pp. 122–137.

[175] Ledyard R Tucker et al. “The extension of factor analysis to three-dimensional matrices”. In: *Contributions to mathematical psychology* 110119 (1964).

[176] Manik Varma and Debajyoti Ray. “Learning the discriminative power-invariance trade-off”. In: *2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007, pp. 1–8.

[177] Ulrike Von Luxburg. “A tutorial on spectral clustering”. In: *Statistics and computing* 17.4 (2007), pp. 395–416.

[178] Jennifer M Walz et al. “Simultaneous EEG-fMRI reveals temporal evolution of coupling between supramodal cortical attention networks and the brainstem”. In: *Journal of Neuroscience* 33.49 (2013), pp. 19212–19222.

[179] Kaidong Wang et al. “Hyperspectral and Multispectral Image Fusion via Nonlocal Low-Rank Tensor Decomposition and Spectral Unmixing”. In: *IEEE Transactions on Geoscience and Remote Sensing* 58.11 (2020), pp. 7654–7671.

[180] Qi Wang et al. “Robust bi-stochastic graph regularized matrix factorization for data clustering”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[181] Wenqi Wang, Vaneet Aggarwal, and Shuchin Aeron. “Principal Component Analysis with Tensor Train Subspace”. In: *arXiv preprint arXiv:1803.05026* (2018).

[182] Wenqi Wang, Vaneet Aggarwal, and Shuchin Aeron. “Principal component analysis with tensor train subspace”. In: *Pattern Recognition Letters* 122 (2019), pp. 86–91.

[183] Wenqi Wang, Vaneet Aggarwal, and Shuchin Aeron. “Tensor train neighborhood preserving embedding”. In: *IEEE Transactions on Signal Processing* 66.10 (2018), pp. 2724–2732.

[184] Xudong Wang and Lijun Sun. “Diagnosing spatiotemporal traffic anomalies with low-rank tensor autoregression”. In: *IEEE Transactions on Intelligent Transportation Systems* (2021).

[185] Xudong Wang et al. “A probabilistic tensor factorization approach to detect anomalies in spatiotemporal traffic activities”. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2019, pp. 1658–1663.

[186] Yao Wang et al. “Hyperspectral image restoration via total variation regularized low-rank tensor decomposition”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 11.4 (2017), pp. 1227–1243.

[187] Yu Wang, Wotao Yin, and Jinshan Zeng. “Global convergence of ADMM in nonconvex nonsmooth optimization”. In: *Journal of Scientific Computing* 78.1 (2019), pp. 29–63.

[188] *Weizmann Facebase*. `http://www.wisdom.weizmann.ac.il/~/vision/FaceBase/`.

[189] Zaiwen Wen and Wotao Yin. “A feasible method for optimization with orthogonality constraints”. In: *Mathematical Programming* 142.1-2 (2013), pp. 397–434.

[190] Keith J Worsley et al. “A general statistical analysis for fMRI data”. In: *Neuroimage* 15.1 (2002), pp. 1–15.

[191] Elizabeth Wu, Wei Liu, and Sanjay Chawla. “Spatio-temporal outlier detection in precipitation data”. In: *International Workshop on Knowledge Discovery from Sensor Data*. Springer. 2008, pp. 115–133.

[192] Kun Xie et al. “Graph based tensor recovery for accurate internet anomaly detection”. In: *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE. 2018, pp. 1502–1510.

[193] Ming Xu et al. “Anomaly detection in road networks using sliding-window tensor factorization”. In: *IEEE Transactions on Intelligent Transportation Systems* 20.12 (2019), pp. 4704–4713.

[194] Ming Yan and Wotao Yin. “Self equivalence of the alternating direction method of multipliers”. In: *Splitting Methods in Communication, Imaging, Science, and Engineering*. Springer, 2016, pp. 165–194.

[195] Shuicheng Yan et al. “Discriminant analysis with tensor representation”. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE. 2005, pp. 526–532.

[196] Shuicheng Yan et al. “Multilinear discriminant analysis for face recognition”. In: *IEEE Transactions on Image Processing* 16.1 (2006), pp. 212–220.

[197] Jing-Hua Yang et al. “Low-rank tensor train for tensor robust principal component analysis”. In: *Applied Mathematics and Computation* 367 (2020), p. 124783.

[198] Jieping Ye, Ravi Janardan, and Qi Li. “Two-dimensional linear discriminant analysis”. In: *Advances in Neural Information Processing Systems*. 2005, pp. 1569–1576.

[199] Tatsuya Yokota, Qibin Zhao, and Andrzej Cichocki. “Smooth PARAFAC decomposition for tensor completion”. In: *IEEE Transactions on Signal Processing* 64.20 (2016), pp. 5423–5436.

[200] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S Dhillon. “Temporal Regularized Matrix Factorization for High-dimensional Time Series Prediction.” In: *NIPS*. 2016, pp. 847–855.

[201] Rose Yu and Yan Liu. “Learning from multiway data: Simple and efficient tensor regression”. In: *International Conference on Machine Learning*. PMLR. 2016, pp. 373–381.

[202] Stefanos Zafeiriou. “Discriminant nonnegative tensor factorization algorithms”. In: *IEEE Transactions on Neural Networks* 20.2 (2009), pp. 217–235.

[203] Huichu Zhang, Yu Zheng, and Yong Yu. “Detecting urban anomalies using multiple spatio-temporal data sources”. In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2.1 (2018), pp. 1–18.

[204] Junyu Zhang, Zaiwen Wen, and Yin Zhang. “Subspace methods with local refinements for eigenvalue computation using low-rank tensor-train format”. In: *Journal of Scientific Computing* 70.2 (2017), pp. 478–499.

[205] Mingyang Zhang et al. “A decomposition approach for urban anomaly detection across spatiotemporal data”. In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press. 2019, pp. 6043–6049.

[206] Mingyang Zhang et al. “Urban Anomaly Analytics: Description, Detection and Prediction”. In: *IEEE Transactions on Big Data* (2020).

[207] Xing Zhang, Gongjian Wen, and Wei Dai. “A tensor decomposition-based anomaly detection algorithm for hyperspectral image”. In: *IEEE Transactions on Geoscience and Remote Sensing* 54.10 (2016), pp. 5801–5820.

[208] Zemin Zhang et al. “Novel Methods for Multilinear Data Completion and De-noising Based on Tensor-SVD”. in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2014.

[209] Qibin Zhao et al. “Tensor ring decomposition”. In: *arXiv preprint arXiv:1606.05535* (2016).

[210] Yu-Bang Zheng et al. “Tensor N-tubal rank and its convex relaxation for low-rank tensor recovery”. In: *Information Sciences* 532 (2020), pp. 170–189.

[211] Hua Zhou, Lexin Li, and Hongtu Zhu. “Tensor regression with applications in neuroimaging data analysis”. In: *Journal of the American Statistical Association* 108.502 (2013), pp. 540–552.