Summer 8-11-2020

Learning Interpretable Features of Graphs and Time Series Data

Shah Muhammad Hamdi
Georgia State University

LEARNING INTERPRETABLE FEATURES OF GRAPHS AND TIME SERIES DATA

by

SHAH MUHAMMAD HAMDI

Under the Direction of Rafal A. Angryk, PhD

ABSTRACT

Graphs and time series are two of the most ubiquitous representations of data of modern time. Representation learning of real-world graphs and time-series data is a key component for the downstream supervised and unsupervised machine learning tasks such as classification, clustering, and visualization. Because of the inherent high dimensionality, representation learning, i.e., low dimensional vector-based embedding of graphs and time-series data is very challenging. Learning interpretable features incorporates transparency of the feature roles, and facilitates downstream analytics tasks in addition to maximizing the performance of the downstream machine learning models. In this thesis, we leveraged tensor (multidimensional array) decomposition for generating interpretable and low dimensional feature space of graphs and time-series data found from three domains: social networks, neuroscience, and heliophysics. We present the theoretical models and empirical results on node embedding of social networks, biomarker embedding on fMRI-based brain networks, and prediction and visualization of multivariate time-series-based flaring and non-flaring solar events.

INDEX WORDS: Data Mining, Graphs, Time Series, Tensor Decomposition

LEARNING INTERPRETABLE FEATURES OF GRAPHS AND TIME SERIES DATA

by

SHAH MUHAMMAD HAMDI

A Dissertation Submitted in Partial Fulfillment for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2020

LEARNING INTERPRETABLE FEATURES OF GRAPHS AND TIME SERIES DATA

by

SHAH MUHAMMAD HAMDI

DEDICATION

This work is dedicated to my parents, Maa: AFROZA BEGUM NAZMA and Abbu: MD SHAHZAHAN ALI SARKER. Your only son could never achieve this goal without your continuous support and priceless inspiration that you provided him throughout his PhD from thousands of miles away.

ACKNOWLEDGEMENTS

First of all, I would like to pay my heartiest gratitude to most merciful, almighty Allah for giving me the opportunity to accomplish this dissertation.

I wish to express my sincere gratitude to the best advisor one could ask for, Dr. Rafal Angryk. Dr. Angryk's confidence in my abilities, his encouragement, supervision, and the push he provided helped me tremendously in the completion of this goal.

I would also like to thank each of my committee members – Dr. Yubao Wu, Dr. Raj Sunderraman, and Dr. Petrus Martens, for their continual support and guidance in my graduate studies.

I appreciate all the discussions, feedback, assistance, and friendship from the members of our lab over the years– Ahmet, Azim, Berkay, Dustin, Max, Mike, Ruizhe, Soukaina, Sushant, and so on.

I would like to mention the support of some of my very close friends such as Soukaina, Nadir Bhai, Sifat, Adil, Fuad, and Muzaheed during my graduate student life.

In addition to my parents, I would like to mention the names of some of my family members who directly helped me in achieving this goal such as uncles Dr. Shamsul Alam, Wahidur Rahman Faruk, Shaheenur Rahman, and Towfiqur Rahman Babu, aunts Nazmin Nahar, and Salma Begum Runa, and cousin Arifur Rahman. I also thank the rest of my family including other uncles, aunts, and cousins whose names I could not write due to the space constraints.

I would like to pay my heartiest tribute to the memory of my family members who left the world: grandparents Dilua Khatun and Golam Mowla, grandparents Sadhu Sarker and Bayela Khatun, uncles Obaidur Rahman and Mahabubur Rahman Matin, and cousins Farhana Rahman Suborna and Silma Subah Nipun. You are in my thoughts and prayers forever.

Finally, to all my friends and family living in Bangladesh and USA, thank you, I am honored to have you all in my life.

FUNDING ACKNOWLEDGEMENT

This project has been supported in part by funding from the Division of Advanced Cyberinfrastructure within the Directorate for Computer and Information Science and Engineering, the Division of Astronomical Sciences within the Directorate for Mathematical and Physical Sciences, and the Division of Atmospheric and Geospace Sciences within the Directorate for Geosciences, under NSF awards #1443061, #1812964, #1936361 and #1931555. It was also supported in part by funding from NASA, through the Heliophysics' Living With a Star Science Program, under NASA award #NNX15AF39G, as well as through the direct contract from Space Radiation Analysis Group (SRAG). In addition to that, the work has been in part sponsored by state funding from Georgia State University's Second Century Initiative, and Next Generation Program.

TABLE OF CONTENTS

List of Tables

List of Figures

1 INTRODUCTION

Interpretability is an important question in machine learning [2]. Representation learning, one of the most important fields of machine learning [3], has been used in building highly accurate models for image classification [4], object detection from images [5], speech recognition [6], word embedding [7], graph embedding [8], and so on. Given the high dimensional vector-based representation of real-world objects such as images, speeches, texts, graphs, etc, representation learning models such as manifold learning [9] and deep learning [3] generate low-dimensional vector-based representations of the objects, so that the accuracy of the downstream machine learning tasks such as classification and clustering is maximized. The lack of explainability of the latent features generated by the black-box models such as deep neural networks results in an uninterpretable knowledge, even if they are fed into transparent classifiers such as decision tree and random forest. Moreover, when the latent features are considered for feature selection, the feature ranking produced by feature scoring algorithms such as Fisher [10], mutual information [11], minimal Redundancy Maximal Relevance (mRMR) [11], etc also makes less sense.

In order to generate interpretable features for graphs and time series data, we leverage tensor decomposition. Tensors are multidimensional arrays, whose low-rank decomposition can result in low-dimensional, multi-view-based, and interpretable representations. In particular, we used tensor decomposition-based learning for node embedding for social networks [12], biomarker embedding for brain networks [13], and magnetic field parameters embedding for multivariate time series-based solar event data. We showed how tensor decomposition can be used for capturing the directionality and proximity of the nodes in arbitrary types of networks in order to generate an interpretable feature space for the nodes. Modeling network data in third-order tensors, and decomposing the tensors into factor matrices can result in meaningful features, that can be utilized by any supervised/unsupervised learning models, especially tree-based classifiers and feature scoring algorithms [14]. We also demonstrated the representation learning of the biomarkers, i.e., the discriminative brain regions and their connections that distinguish subjects with brain-related diseases and normal controls, using the decomposition of tensors. Finally, we leveraged tensor decomposition for visualizing multivariate time series-based flaring and nonflaring solar events in two dimensional space. For demonstrating the interdisciplinary nature of our work, we applied our models in three application areas of social networks, fMRI-based brain networks, multivariate time series-based flaring and nonflaring solar event data.

1.1 *Motivation*

Graphs (a.k.a networks) are one of the most ubiquitous data structures used in computer science and related fields. By capturing the interactions between individual entities, graphs facilitate discovering the underlying complex structure of a system. Mining real-life graphs plays an important role in studying the network behavior of different domains such as social sciences (social network), linguistics (word co-occurrence network), biology (protein-protein interaction network), neuroscience (brain network) and so on. Recently, there has been a surge of research interest in embedding graph structures, such as nodes, edges, subgraphs, and the whole graph in a low dimensional vector space (e.g., Fig. 1.1). Among them, representation learning of the nodes is most widely studied, which facilitates downstream machine learning tasks, such as network reconstruction, link prediction, node classification, and graph classification.

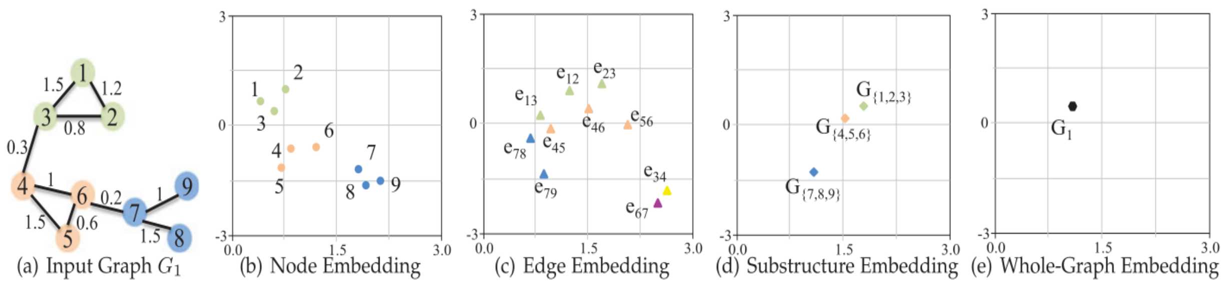


Figure 1.1: Embedding of different graph structures in 2D space [1]

In the network reconstruction, the ability to reconstruct the adjacency matrix of the original network from the embedding matrix is assessed. In link prediction, the embedding algorithm is trained after hiding some edges/links, and the prediction accuracy of the hidden links is assessed from the reconstructed adjacency matrix derived from the embedding matrix. In node classification, the learned node embeddings are provided with the node labels, and embedding performance is evaluated by training and testing a downstream classifier. Dimensionality reduction of the embedding matrix facilitates the 2D/3D visualization of the nodes. In the multi-graph setting, the whole graph can be represented by a low-dimensional vector, and graph level classification/clustering can be performed.

Graph embedding aims to represent the graph structures (e.g., nodes, edges, subgraphs, or whole graphs) in low-dimensional space so that the performance of downstream machine learning tasks can be maximized. The problem of graph embedding is related to two areas : (1) graph analytics, and (2) representation learning. Graph analytics deals with querying the graphs and leveraging statistical features of the graphs structures to mine useful information depending on the application. Graph representation learning embeds the graph structures in fixed-dimensional vector space without the constraint of low-dimensional embedding. For example, [15] represents each node as a vector with dimensionality equals the number of nodes in the input graph. Every dimension denotes the geodesic distance of a node to each other node in the graph. Graph embedding lies in the overlapping area of graph analytics and graph representation learning [1].

While most of the graph embedding models proposed in the literature emphasize more on the representation learning area so that downstream machine learning tasks are facilitated, optimizing the embedding matrix to facilitate graph analytics remains less explored. Graph analytics require the representations to be interpretable so that querying on the feature space gives meaningful insights of different properties such as neighborhood, higher-order proximity, directionality (if the graph is directed), clustering coefficient, closeness centrality, betweenness centrality, PageRank, etc.

Therefore, we are motivated to design one graph embedding model, which will shed the light of interpretability on the embedding space, so that the objectives of both graph analytics and graph representation learning are emphasized. We proposed the data model for the graphs using third-order tensors, which capture higher-order proximities among nodes. By decomposing the tensor(s), we get the node representations that emphasize both directionality and proximity.

Beyond graphs, our another motivation came from the problem of visualization of multivariate time series data in 2D space [16]. Multivariate time series data is high dimensional, and classical dimensionality algorithms such as PCA and t-SNE [17] suffers from curse of dimensionality, if raw data containing thousands of features are provided directly as input [18]. Therefore, we leveraged tensor decomposition to reduce the dimensionality of the multivariate time series data before applying classical dimensionality reduction algorithms. For experiments, we chose flaring and nonflaring solar active region data as the source of multivariate time series data. Solar flares are considered as very intense solar events, which can cause serious damage in the life and property in space and ground [19]. Predicting solar flares is one of the most prioritized objectives in solar weather analysis. In our work, we presented the time series classification model for solar flare prediction, and proposed 2D visualization technique following the interpretable and low dimensional feature space generation by tensor decomposition.

1.2 *Challenges*

In node embedding, preserving the similarity of the nodes in the embedding space is hard due to the subjectiveness of the notion *similarity*. Two nodes can be similar in terms of the direct neighborhood (first-order proximity), multihop relations (higher-order proximity), community structure, etc. Therefore, we face the following challenges in node embedding.

1. **Structural property preserving:** The learned representations of the nodes should preserve the structural properties of the graph. From the literature, we know about different structural features of nodes such as degree, clustering coefficient, PageRank score, closeness centrality, betweenness centrality, etc. The node embedding algorithms should preserve these properties. For example, if the PageRank scores of two nodes are similar, the distance of their vector representations should be small. If the embedding space properly preserves the structure properties, then the feature space becomes interpretable, and graph analytics tasks are facilitated.

2. **Performance in downstream machine learning tasks:** Node representations should be able to maximize the performance of downstream machine learning tasks such as network reconstruction, link prediction, node classification, clustering, and visualization.

3. **Scalability:** Real-world networks can have millions to billions of nodes. The node embedding algorithm should be scalable for large graphs. This challenge is very hard when we consider the global structural proximities of the nodes.

4. **Less hyperparameter depending:** Recent node embedding algorithms especially random walk-based and deep learning-based methods require optimized tuning of a lot of hyperparameters. Tuning a large set of hyperparameters increase the cost of

using the algorithm in real-world scenarios. Therefore, node embedding algorithms should be easy to use in terms of hyperparameter tuning.

5. **Generalization:** Node embedding algorithms should be generic enough to capture the node similarities in different types of graphs such as (un)-directed, (un)-weighted, sparse/dense, and small/large graphs.

In this work, we considered static graphs only, and the graphs with auxiliary information such as node/edge attributes/labels, knowledge graphs, multi-layer networks, and dynamic networks are out of the scope of this work.

1.3 *Contributions*

The contributions of this thesis are listed below.

1. We present two novel tensor decomposition-based node embedding algorithms that can operate on arbitrary types of graphs, leverage local and global structural similarities of nodes, require fewer hyperparameters, and generate an interpretable feature space for nodes (see details at chapter 2 and our papers [12] and [20]).
2. We demonstrate the use of tensor decomposition in representation learning of the biomarkers from the fMRI-based brain network data and show the use of feature selection algorithms in discriminative subnetwork mining (see details at chapter 3 and 4, and our papers [13], [21], and [22]).
3. We discuss the modeling of solar flare prediction problem as multivariate time series classification, and present tensor decomposition-based dimensionality reduction and 2D visualization of flaring and nonflaring solar events (see details at chapter 5 and our papers [23], [24], and [25]).

4. We present our experimental findings on datasets of three domains: social science (social networks), neuroscience (brain networks), and heliophysics (multivariate time series of solar weather parameters) (see details at chapter 6).

1.4 *Outline*

This thesis is organized as follows. Chapter 2 presents node embedding algorithms that generate interpretable embedding space by capturing directionality and proximity of nodes. In Chapter 3, we present the comparative analysis of different tensor models on fMRI-based functional connectivity data. In Chapter 4, we discuss the representation learning of the biomarkers of the fMRI-based brain networks using tensor decomposition-based models. In Chapter 5, we discuss the modeling of the solar flare prediction problem by multivariate time series, and 2D visualization of flaring and nonflaring solar events by different dimensionality reduction techniques. In Chapter 6, we present the experimental findings. In Chapter 7, we conclude the thesis and present future work.

2 GRAPH EMBEDDING BY TENSOR DECOMPOSITION

Node embedding algorithms have earned considerable attention from the graph mining community in recent years. The relying on the tuning of a lot of hyperparameters, and computationally expensive matrix decompositions make the existing algorithms complex to use and perform poor in real-life graphs. Moreover, most of the algorithms produce latent features for embedding graph structures, whose roles are not easily understandable. In this chapter, we present two tensor decomposition-based node embedding algorithms, which are able to produce interpretable features for nodes in a graph. Both algorithms can work on different types of graphs such as (un)directed, (un)weighted, and sparse/dense. They leverage k-step transition probability matrices of graphs to preserve local and global structural similarities. The k-step transition probability matrices are used to construct one or more third-order tensors, and factor matrices found from CANDECOMP/PARAFAC (CP) decomposition of the tensor(s) produce an interpretable and low dimensional feature space for the nodes. We have experimentally evaluated the algorithms using different types of real-life graphs found from different domains such as social networks and fMRI-based brain networks. Our algorithms have proven superiority in terms of interpretability of the learned features, network reconstruction, link prediction, node classification, and graph classification.

2.1 *Overview*

In recent years, a good number of node embedding algorithms have been proposed. They can be roughly divided into three categories - matrix decomposition-based approaches, multihop similarity-based approaches, and random walk-based approaches. Most matrix decomposition-based approaches decompose various matrix representations of graphs by eigendecomposition or Singular Value Decomposition (SVD). Multihop similarity-based approaches consider the higher-order proximities of the nodes, and use matrix factorization for decomposing higher-order proximity matrices (e.g., GraRep [26], AROPE [27]). Random walk-based approaches consider the input graph as a set of random walks from each node (e.g., Node2vec [8], DeepWalk [28]). These random walks are considered as sentences, where the nodes are considered as words in a Natural Language Processing (NLP) model. Finally, the Skip-gram model [7] is used to find the node embeddings.

While eigendecomposition on the large real-world networks is very expensive, random walk-based methods are comparatively scalable. But, the random walk-based approaches require the tuning of a number of hyperparameters, some of which are NLP-based. For example, Node2vec requires tuning of several hyperparameters such as context size, walks per node, walk length, return parameter and in-out parameter. Moreover, almost all the node embedding algorithms represent the nodes as d-dimensional vectors, and do not provide any direction to the interpretability of the features.

In this work, we propose two algorithms: Tensor Decomposition-based Node Embedding (TDNE) [12] and Tensor Decomposition-based Node Embedding per Slice (TDNEpS) [20]. TDNE uses higher-order transition probability matrices of a graph to construct one third-order tensor, while TDNEpS considers each transition probability matrix as one third-order tensor. Both algorithms use higher-order transition probability matrices of a graph to construct one or more third-order tensors, and perform CP decomposition to get the representations of the nodes and the representations of the transition steps.

The algorithms do not rely on eigendecomposition of large matrices, or tuning of the NLP-based hyperparameters such as context size.

The main contributions of this work are:

1. To the best of our knowledge, this work is the first attempt to learn embeddings of the transition steps (one kind of pairwise proximity [26]).

2. Our method provides interpretability by creating a feature space for the nodes, where the role of each feature is understandable.

3. When we have a set of graphs, and each graph consists of same labeled node set, we use learned representations of the nodes of each graph for embedding the whole graphs. Therefore, in addition to evaluate our algorithms in single graph-based tasks such as node classification and link prediction, we have evaluated our algorithms in multi-graph-based tasks such as graph classification.

2.2 *Related Work*

Early works on node embedding were basically dimensionality reduction techniques, which required the matrix factorization of the first-order proximity matrix or adjacency matrix. Laplacian Eigenmaps [9] and Locally Linear Embedding (LLE) [29] can be viewed as those early approaches. After creating a knn graph from the feature space of the data, Laplacian Eigenmaps embeds the nodes by eigendecomposition of the graph Laplacian. LLE considers that each node is a linear combination of its neighbors, and finds the solution by singular value decomposition of a sparse matrix, which is calculated by subtracting the normalized adjacency matrix from the same-sized identity matrix. The later approaches such as GraRep [26] and Higher Order Proximity preserved Embedding (HOPE) [30] considered higher-order proximities of the nodes. GraRep utilizes multihop neighborhood of the nodes by incorporating higher powers of the adjacency matrix and generates node embedding by successive singular value decomposition of the powers of the log-transformed, probabilistic adjacency matrix. HOPE measures overlap between node neighborhoods, where Jaccard similarity, Adamic-Adar score, Katz score or Personalized PageRank score can be used as overlap calculating functions. Asymmetric transitivity preserving nature of HOPE enables embedding of nodes of a directed graph. The relying on eigendecomposition or singular value decomposition of large matrices makes all the matrix factorization-based approaches computationally expensive, and results in the compromise of the performance due to poor approximation.

Being inspired by the Skip-gram model [7], which learns word embeddings by employing a fixed sliding window so that words in the similar context have similar representations, DeepWalk [28] considered the network as a "document". By applying truncated random walk, DeepWalk sampled sequence of nodes (similar to the words of a document) and used Stochastic Gradient Descent (SGD) optimization to learn the representation of each node so that it is similar to the representations of its neighbor nodes. Node2vec [8] later increased the flexibility of node sampling by incorporating a biased random walk. Although both methods are able to achieve more scalability than the matrix factorization-based methods, dependence on local neighborhood window refrains them from achieving the global optimal solution.

To capture the highly non-linear structures of the graphs, deep learning has been used by Structural Deep Network Embedding (SDNE) [31], Deep Neural Networks for Learning Graph Representation (DNGR) [32], and Graph Convolutional Network (GCN) [33]. SDNE and DNGR use deep autoencoder to learn node representation from its global neighborhood vector. GCN becomes comparatively more scalable by defining a convolution operator on graph, which iteratively aggregates embeddings of the local neighborhood to reach the global optima. Although deep learning-based models result in high accuracy, the scalability is compromised because of their high training time.

While all the previous node embedding algorithms produce node features that are not easily interpretable, our tensor decomposition-based node embedding algorithms use arbitrary-order proximity to generate an interpretable feature space for the nodes.

2.3 *Graph notations*

Definition 1. (Graph) A graph with n nodes is defined as $G = (V, E)$, where $V = \{v_1, v_2, v_3, \dots, v_n\}$ is the set of nodes, and $E = \{e_{ij}\}_{i,j=1}^{n}$ is the set of edges, which are the relationships between the nodes. The adjacency matrix S of the graph has n rows and n columns. For unweighted graphs, $S_{ij} = 1$, if there exists an edge between nodes i and j, and $S_{ij} = 0$ otherwise. For weighted graphs, $S_{ij} \neq 0$ represents the positive/negative weight of the relationship between nodes i and j, while $S_{ij} = 0$ means no relationship between them. For undirected graphs, adjacency matrix S is symmetric, i.e., $S_{ij} = S_{ji}$. For directed graphs, adjacency matrix S is not symmetric, i.e., $S_{ij} \neq S_{ji}$.

Definition 2. (1-step transition probability matrix) The 1-step transition probability between nodes i and j for both directed and undirected graphs is defined as the normalized edge weight between those nodes. Therefore, the 1-step transition probability matrix is found by normalizing each row of the adjacency matrix S.

$$A_{ij} = \frac{S_{ij}}{\sum_j S_i}$$

Definition 3. (k-step transition probability matrix) For preserving the global structural similarity, we use k-step transition probability matrix A^k, which is the k-th power of the 1-step transition probability matrix. In this matrix, A^k_{ij} represents the transition probability from node i to node j in exactly k steps.

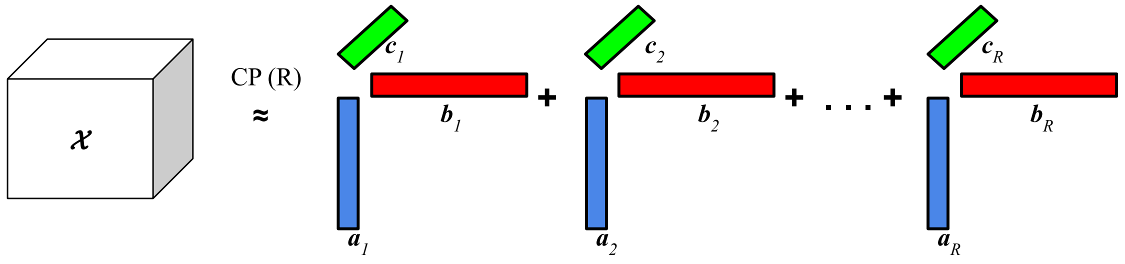


Figure 2.1: CP decomposition of a third-order tensor.

2.4 *Preliminaries of Tensor Decomposition*

Tensors are multidimensional arrays. In this work, we consider only the third-order tensors and CP decomposition. In this section, we briefly review the CP decomposition.

CP decomposition: CP decomposition factorizes the tensor into a sum of rank one tensors [34]. Given a third-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, where I, J and K denote the indices of tensor elements in three of its modes, CP decomposition factorizes the tensor in the following way.

$$\mathcal{X} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \tag{2.1}$$

Here, $\circ$ denotes the outer product of the vectors, R is the tensor rank which is a positive integer, $\mathbf{a}_r, \mathbf{b}_r$, and $\mathbf{c}_r$ are vectors, where $\mathbf{a}_r \in \mathbb{R}^I$, $\mathbf{b}_r \in \mathbb{R}^J$, and $\mathbf{c}_r \in \mathbb{R}^K$ for $r = 1, 2, 3, \ldots R$. After stacking those vectors, we can get the factor matrices $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots \mathbf{a}_R]$, $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots \mathbf{b}_R]$, and $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \ldots \mathbf{c}_R]$, where $\mathbf{A} \in \mathbb{R}^{I \times R}$, $\mathbf{B} \in \mathbb{R}^{J \times R}$, and $\mathbf{C} \in \mathbb{R}^{K \times R}$. Fig. 2.1 is a visualization of the CP decomposition of a third-order tensor.

The matricized forms of the tensor $\mathcal{X}$ is given by,

$$\mathbf{X}_{(1)} \approx \mathbf{A}(\mathbf{C} \odot \mathbf{B})^{\mathsf{T}}$$
$$\mathbf{X}_{(2)} \approx \mathbf{B}(\mathbf{C} \odot \mathbf{A})^{\mathsf{T}}$$
$$\mathbf{X}_{(3)} \approx \mathbf{C}(\mathbf{B} \odot \mathbf{A})^{\mathsf{T}}$$

where $\odot$ represents Khatri-Rao product of two matrices.

2.4.1 ALS Solution of CP Decomposition

CP decomposition can be solved by Alternating Least Squares [35]. The cost function of CP decomposition can be formulated as,

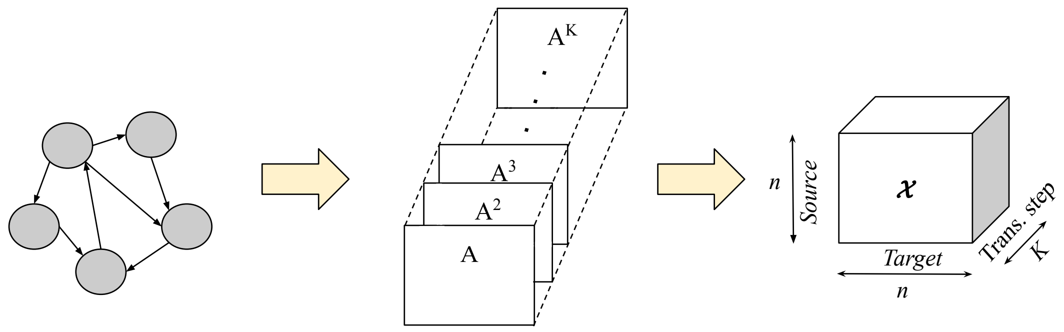
$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \left\| \mathcal{X} - \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \right\|_F^2 \tag{2.2}$$

where $\|.\|_F^2$ is the tensor Frobinius norm, which is the sum of squares of all elements of the tensor. By initializing **B** and **C** with random values, ALS updates **A** by following rule.

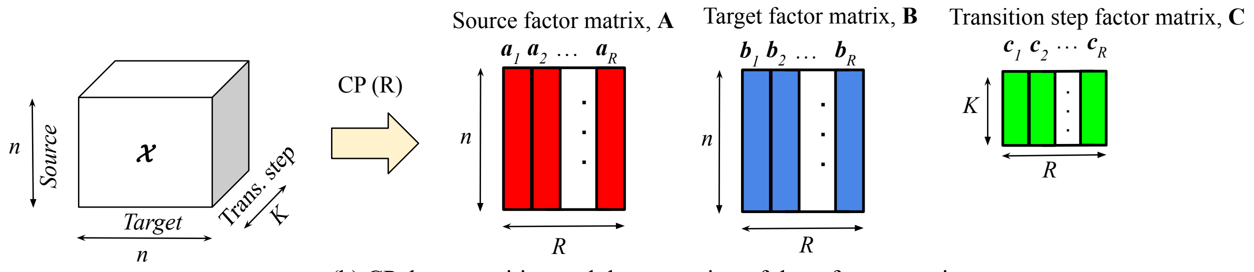
$$\mathbf{A} \leftarrow \arg\min_{\mathbf{A}} \left\| \mathbf{X}_{(1)} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})^{\mathsf{T}} \right\|_F^2 \tag{2.3}$$

Then by fixing **A** and **C**, it updates **B** by,

$$\mathbf{B} \leftarrow \arg\min_{\mathbf{B}} \left\| \mathbf{X}_{(2)} - \mathbf{B}(\mathbf{C} \odot \mathbf{A})^{\mathsf{T}} \right\|_F^2 \tag{2.4}$$



(a) Making of a third order tensor from powers of the 1-step transition probability matrix



(b) CP decomposition and the extraction of three factor matrices

Figure 2.2: CP decomposition-based representation learning of source nodes, target nodes, and transition steps

Finally, by fixing **A** and **B**, it updates **C** by,

$$\mathbf{C} \leftarrow \arg\min_{\mathbf{C}} \left\| \mathbf{X}_{(3)} - \mathbf{C}(\mathbf{B} \odot \mathbf{A})^{\mathsf{T}} \right\|_{\mathsf{F}}^{2} \tag{2.5}$$

Equations 2.3, 2.4 and 2.5 are repeated until the convergence of equation 2.2.

2.5 *Interpretable Feature Learning of Graphs*

In this section, we discuss two algorithms for tensor decomposition-based node embedding: TDNE and TDNEpS.

2.5.1 Tensor Decomposition-based Node Embedding

Fig. 2.2 describes our model of tensor decomposition-based node embedding (TDNE). Without loss of generality, we use an example of a directed graph in the figure. In TDNE, a third-order tensor $\mathcal{X} \in \mathbb{R}^{n \times n \times K}$ is constructed by stacking the k-step transition probability matrices for $k = 1, 2, 3, \ldots, K$. The objects represented by the three modes of this tensor are: nodes (as sources), nodes (as targets), and transition steps. Then CP decomposition is performed with a given rank R. CP decomposition results in vectors $\mathbf{a}_r \in \mathbb{R}^n$, $\mathbf{b}_r \in \mathbb{R}^n$, and $\mathbf{c}_r \in \mathbb{R}^K$ for $r = 1, 2, 3, \ldots R$. These vectors are stacked together to form three factor matrices, $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots \mathbf{a}_R]$, $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots \mathbf{b}_R]$, and $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \ldots \mathbf{c}_R]$, where $\mathbf{A} \in \mathbb{R}^{n \times R}$, $\mathbf{B} \in \mathbb{R}^{n \times R}$, and $\mathbf{C} \in \mathbb{R}^{K \times R}$.

In factor matrix $\mathbf{A} \in \mathbb{R}^{n \times R}$, each row is an R-dimensional representation of the *source role* played by the corresponding node. In factor matrix $\mathbf{B} \in \mathbb{R}^{n \times R}$, each row is an R-dimensional representation of the *target role* played by the corresponding node. In factor matrix $\mathbf{C} \in \mathbb{R}^{K \times R}$, each row i is an R-dimensional representation of the i-th *transition step*, where $1 \leqslant i \leqslant K$.

After we find the source factor matrix $\mathbf{A}$, target factor matrix $\mathbf{B}$, and transition factor matrix $\mathbf{C}$, we can compute the projection of source embedding of node i on the transition embedding j, where $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant K$, and get *source-transition* embedding matrix $\mathbf{ST} \in \mathbb{R}^{n \times K}$. Similarly, we can get a *target-transition* embedding matrix $\mathbf{TT} \in \mathbb{R}^{n \times K}$ that reflects the projection of target embeddings on transition step embeddings. Finally, we get the node embedding matrix $\mathbf{Z} \in \mathbb{R}^{n \times 2K}$ by concatenating $\mathbf{ST}$ and $\mathbf{TT}$. First K columns of $\mathbf{Z}$ represent *source role* of a node with varying transition steps, and last K columns of $\mathbf{Z}$ represent *target role* of a node with varying transition steps. TDNE is shown in Algorithm 2.1.

$$\mathbf{ST} = \mathbf{A} * \mathbf{C}^{T}$$

$$\mathbf{TT} = \mathbf{B} * \mathbf{C}^{T}$$

$$\mathbf{Z} = [\mathbf{ST}, \mathbf{TT}]$$

Algorithm 2.1 TDNE: Tensor Decomposition-based Node Embedding

Input: 1-step transition probability matrix A
Maximum transition step K
CP decomposition rank R
Output: Node embedding matrix $\mathbf{Z}$

```
 1: $n = \mathrm{count\_rows}(A)$
 2: $\mathcal{X} = \mathrm{tensor}(n, n, K)$
 3: for $k$ in 1 to $K$ do
 4:     $\mathcal{X}(:, :, k) = A^{k}$
 5: end for
 6: $[\mathbf{A}, \mathbf{B}, \mathbf{C}] \Leftarrow \mathrm{CP\_ALS}(\mathcal{X}, R)$
 7: $\mathbf{ST} = \mathbf{A} * \mathbf{C}^{T}$
 8: $\mathbf{TT} = \mathbf{B} * \mathbf{C}^{T}$
 9: $\mathbf{Z} = [\mathbf{ST}, \mathbf{TT}]$
10: return $\mathbf{Z}$
```

2.5.2 Tensor Decomposition-based Node Embedding per Slice

In TDNEpS (Algorithm 2.2), we consider each transition probability matrix A^{k} as a third-order tensor with a single slice. Therefore, instead of having a single third-order tensor, we have K third-order tensors, where each tensor $\mathcal{X}^{(k)} \in \mathbb{R}^{n \times n \times 1}$. Then for each tensor $\mathcal{X}^{(k)}$, CP decomposition is performed with a given rank R, and three factor matrices are found, which are source factor matrix regarding k^{th} transition step $\mathbf{A}^{(k)} \in \mathbb{R}^{n \times R}$, target factor matrix regarding k^{th} transition step $\mathbf{B}^{(k)} \in \mathbb{R}^{n \times R}$, and k^{th} transition factor matrix $\mathbf{C}^{(k)} \in \mathbb{R}^{1 \times R}$. We compute *source-transition* embedding matrix $\mathbf{ST}^{(k)} \in \mathbb{R}^{n \times 1}$ and *target-transition* embedding matrix $\mathbf{TT}^{(k)} \in \mathbb{R}^{n \times 1}$, and finally concatenate $\mathbf{ST}^{(k)}$'s and $\mathbf{TT}^{(k)}$'s for $1 \leqslant k \leqslant K$ to get the node embedding matrix $\mathbf{Z} \in \mathbb{R}^{n \times 2K}$.

$$\begin{aligned}
\mathbf{ST}^{(k)} &= \mathbf{A}^{(k)} * \mathbf{C}^{(k)^T} \\
\mathbf{TT}^{(k)} &= \mathbf{B}^{(k)} * \mathbf{C}^{(k)^T} \\
\mathbf{Z} = [\mathbf{ST}^{(1)}, \mathbf{ST}^{(2)}, &\ldots \mathbf{ST}^{(K)}, \\
\mathbf{TT}^{(1)}, \mathbf{TT}^{(2)}, &\ldots \mathbf{TT}^{(K)}]
\end{aligned}$$

Algorithm 2.2 TDNEpS: Tensor Decomposition-based Node Embedding per Slice

```
Input: 1-step transition probability matrix $A$
Maximum transition step $K$
CP decomposition rank $R$
Output: Node embedding matrix $\mathbf{Z}$
 1: $n = \mathrm{count\_rows}(A)$
 2: for $k$ in 1 to $K$ do
 3:     $\mathcal{X}^{(k)} = \mathrm{tensor}(n, n, 1)$
 4:     $\mathcal{X}^{(k)} = A^k$
 5:     $[\mathbf{A}^{(k)}, \mathbf{B}^{(k)}, \mathbf{C}^{(k)}] \Leftarrow \mathrm{CP\_ALS}(\mathcal{X}^{(k)}, R)$
 6:     $\mathbf{ST}^{(k)} = \mathbf{A}^{(k)} * \mathbf{C}^{(k)^T}$
 7:     $\mathbf{TT}^{(k)} = \mathbf{B}^{(k)} * \mathbf{C}^{(k)^T}$
 8: end for
 9: $\mathbf{Z} = [\mathbf{ST}^{(1)}, \mathbf{ST}^{(2)}, \ldots \mathbf{ST}^{(K)}, \mathbf{TT}^{(1)}, \mathbf{TT}^{(2)} \ldots \mathbf{TT}^{(K)}]$
10: return $\mathbf{Z}$
```

The meaning of each column of **Z** is same for both TDNE and TDNEpS. Although both algorithms have similar complexities and output, our experimental findings suggest that TDNEpS has less performance variance and high accuracy in comparison with TDNE.

Both TDNE and TDNEpS can be easily extended to other matrix-based graph representations such as considering the adjacency matrices of the line graphs for edge embedding.

3 TENSOR DECOMPOSITION ON BRAIN NETWORKS

A multidimensional array is also known as a tensor. An N-th order tensor is the tensor product of N vector spaces, where each vector space has its own coordinate system. Decomposing higher order tensor into lower order tensors is a prominent research problem in mathematics. There are several tensor decomposition algorithms such as CANDECOMP/PARAFAC (CP), Tucker, INDSCAL, PARAFAC2, CANDELINC, DEDICOM, and PARATUCK2 [34]. In this chapter, we consider third order tensors for fMRI data and consider CP and Tucker as the methods of tensor decomposition. Both CP and Tucker decomposition can be solved by Alternating Least Squares (ALS) algorithms. In this chapter, we briefly discuss Tucker decomposition and its ALS optimization algorithms. The contents in this chapter are presented from [22].

3.1 *Tucker Decomposition*

Tucker decomposition is a form of higher order Principal Component Analysis (PCA). A tensor is decomposed into a core tensor, which is multiplied by a matrix along its each mode. Tucker decomposition of a third order tensor $\chi \in \mathbb{R}^{I \times J \times K}$ is given by,

$$\chi \approx g \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} = [\![g; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \tag{3.1}$$

Here, $\times_n$ denotes mode-n tensor product. $\mathbf{A} \in \mathbb{R}^{I \times P}$, $\mathbf{B} \in \mathbb{R}^{J \times Q}$, and $\mathbf{C} \in \mathbb{R}^{K \times R}$ are the factor matrices. These factor matrices can be thought as the principal components along

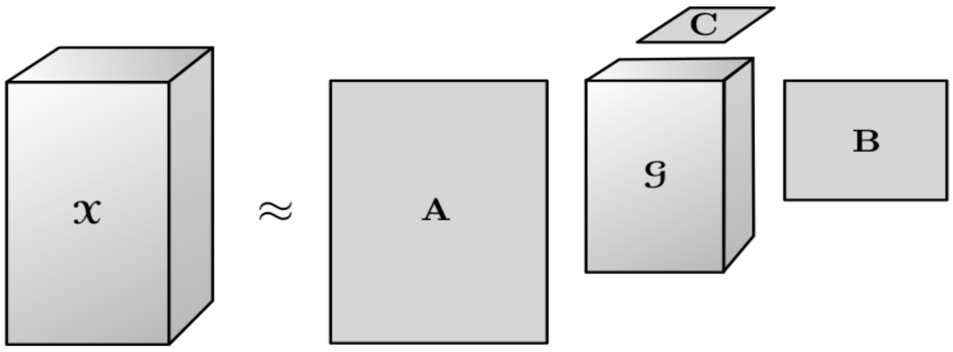


Figure 3.1: Tucker decomposition of a third order tensor.

each mode. The $g \in \mathbb{R}^{P \times Q \times R}$ is the core tensor and the elements of this tensor represents the interaction between those principal components. Unlike CP, where each factor matrix has the same number of columns, in Tucker decomposition P, Q and R are the number of columns of the factor matrix **A**, **B** and **C**. When $P < I$, $Q < J$ and $R < K$, g can be thought as a compressed representation of χ. Tucker decomposition is a generalization of CP, because if $P = Q = R$, then the decomposition becomes CP. Fig. 3.1 is a visualization of the Tucker decomposition of a third order tensor.

3.1.1 ALS Solution of Tucker Decomposition

The cost function for ALS optimization for Tucker decomposition is given by,

$$\min_{g,\mathbf{A},\mathbf{B},\mathbf{C}} \|\chi - [\![g; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|_F^2 \tag{3.2}$$

It is shown by Kolda et al. in [34], that

$$\|\chi - [\![g; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|_F^2 = \|\chi\|_F^2 - \left\|\chi \times_1 \mathbf{A}^T \times_2 \mathbf{B}^T \times_3 \mathbf{C}^T\right\|_F^2 \tag{3.3}$$

Therefore, Equation 3.2 can be rewritten as,

$$\max_{\mathbf{A},\mathbf{B},\mathbf{C}} \left\| \chi \times_1 \mathbf{A}^T \times_2 \mathbf{B}^T \times_3 \mathbf{C}^T \right\|_F^2 \tag{3.4}$$

Similar to CP, ALS optimizes Equation 3.4 by updating **A** while keeping **B** and **C** fixed, then updating **B** by keeping **A** and **C** fixed, then updating **C** by keeping **A** and **B** fixed, and iterating until convergence.

3.2 *Application of Tensor Decomposition in Brain Network Representation*

Brain Informatics, enriched by the advances of neuroimaging technologies such as Magnetic Resonance Imaging (MRI), Positron Emission Tomography (PET), and Electroencephalography (EEG), pose many challenges to data mining. These imaging modalities are noninvasive methods used to diagnose and investigate neurological and neurodevelopmental disorders. fMRI is a popular brain imaging technique, that records the change in Blood Oxygenation Level Dependent (BOLD) signals in different brain regions over time. Resting-state fMRI-based data analysis has facilitated diagnosis of several neurological and neurodevelopmental diseases such as Alzheimer's, Schizophrenia, Bipolar disorder, Attention-deficit/hyperactivity disorder (ADHD), Autism, and Dyslexia [36], [37], [38], [39].

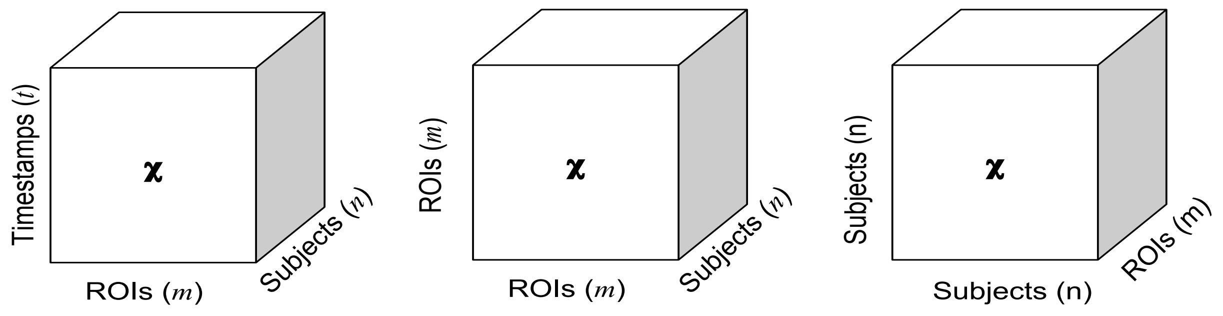
fMRI data can be represented in various forms, e. g., the sequence of 3D brain volumes over time, multivariate time series, and functional connectivity graphs. Given a training set of fMRI data representations of some human subjects and the associated labels of healthy/diseased, the task of binary classification aims to maximize classification accuracy on test data. Because of the advances in graph mining algorithms, most of the supervised learning studies on fMRI data take functional connectivity graphs (binarized by thresholding) as the inputs and transform the problem into graph classification [36], [37], [40], [41], [38]. Graph classification can be addressed by two approaches: structure-based approach and subgraph pattern-based approach. In structure-based approach, node-based features such as degree, PageRank score, and clustering coefficient [36], [38] are calculated and each graph is transformed into a vector. In subgraph pattern-based approach [40], discriminative subgraphs are used as features.

When the fMRI data is represented by undirected and unweighted graphs, one big challenge is the correct representation of the graphs. Since the graphs are made by thresholding the functional connectivity matrices (each matrix element denotes the correlation of BOLD time series of two regions of interest or, ROIs), the sparsity of the generated graphs depends on the threshold value. Sparsity affects the performance of both graph classification approaches discussed above. Though most of the data mining papers disregard the edges with negative weights, it is still debated in the neuroscience community whether to keep or discard negative correlations [42].

To address this problem, some recent studies emerged with the idea of tensor-based modeling of fMRI data [43], [44]. By stacking the functional connectivity matrices or the multivariate time series of the ROIs of all subjects, a third order tensor can be formed. By decomposing the tensor, we can identify the discriminative representations of subjects, so that subjects with neurological disease and normal controls can be easily separated.

3.3 *Modeling the fMRI data in Tensors*

In this section, we describe five tensorization schemata for the fMRI data. In Fig. 3.2, we visualize five models of the tensor. Among these five models, Tensor Model 3 was previously used in the literature [43], while we designed other four for the purpose of comparison. All the models of the tensors are third order. After tensorizing the



(a) Tensor model 1 (b) Tensor model 2 and 3 (c) Tensor model 4 and 5

Figure 3.2: Dimensions for fMRI data using third order tensor

data, we use CP and Tucker decomposition for computing the factor matrices. For the healthy/disabled prediction, we use only the factor matrix found in *subjects* mode. Factor matrices in other modes such as *ROIs* and *Timestamps* are out of the scope of this work.

3.3.1 Tensor Model 1: Stacked Multivariate Time Series

We have the dataset $D = \{A_1, A_2, \ldots, A_n\}$, where each matrix $A_i \in \mathbb{R}^{m \times t}$ is a multivariate time series, and their corresponding labels of healthy/disabled, which are given by $y_i = \{-1, +1\}$. Here, m denotes the number of ROIs, n denotes the number of subjects, and t denotes the number of time samples. In this tensorization scheme, we simply stack all A_i's together. Therefore, $\mathcal{X} = [A_1; A_2; \ldots, A_n]$ and $\mathcal{X} \in \mathbb{R}^{m \times t \times n}$ (Fig. 3.2a). After CP decomposition, we get three factor matrices **A**, **B** and **C**, where $\mathbf{A} \in \mathbb{R}^{m \times R}$, $\mathbf{B} \in \mathbb{R}^{t \times R}$ and $\mathbf{C} \in \mathbb{R}^{n \times R}$. After Tucker decomposition, we get three factor matrices of different number of columns, where $\mathbf{A} \in \mathbb{R}^{m \times P}$, $\mathbf{B} \in \mathbb{R}^{t \times Q}$ and $\mathbf{C} \in \mathbb{R}^{n \times R'}$. Therefore, **C** is the factor matrix in the *subject* space, where each row is a vector-based representation of each subject. Then, we split the rows (subjects) into train and test set, concatenate corresponding class label of each training subject, and train a classifier. Finally, we can evaluate the classification performance by predicting the class labels of the test subjects.

3.3.2 Tensor Model 2: Stacked Functional Connectivity Matrices

For each multivariate time series matrix A_i, we calculate Pearson correlation coefficient between each pair of time series. It gives us functional connectivity matrices $C_1, C_2, \dots, C_n$, where $C_i \in \mathbb{R}^{m \times m}$. Each matrix C_i is symmetric and can be thought of as an adjacency matrix of an edge-weighted complete graph K_m. By stacking the C_i's one after another, we get a tensor $\mathcal{X} \in \mathbb{R}^{m \times m \times n}$ (Fig. 3.2b). After CP decomposition we get three factor matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, where $\mathbf{A} \in \mathbb{R}^{m \times R}$, $\mathbf{B} \in \mathbb{R}^{m \times R}$ and $\mathbf{C} \in \mathbb{R}^{n \times R}$. Since two modes are the same in the third order tensor, after CP decomposition we get two identical factor matrix, i. e., $\mathbf{A} = \mathbf{B}$. The similar case is also found in Tucker decomposition. In this tensor modeling scheme, $\mathbf{C}$ is the necessary subject factor matrix.

3.3.3 Tensor Model 3: Stacked Non-negative Functional Connectivity Matrices

Here, the matrices $C_1, C_2, \dots C_n$ are thresholded by keeping only the non-negative matrix elements. Therefore, C_i's do not represent edge-weighted complete graphs, rather they denote weighted and undirected sparse graphs. The shape of the tensor is the same as Tensor Model 2 and the tensor is given by $\mathcal{X} \in \mathbb{R}^{m \times m \times n}$(Fig. 3.2b). The factor matrices that are found after CP and Tucker decomposition in this tensorization scheme is similar to the Tensor Model 2. Factor matrix $\mathbf{C}$, which is defined in the *subject* space is the necessary factor matrix.

3.3.4 Tensor Model 4: Node-wise Jaccard Kernel on Functional Connectivity Matrices

In this tensor model, we consider each C_i as an edge-weighted complete graph. For each pair of complete graphs, weighted Jaccard is calculated using the vectors represented by

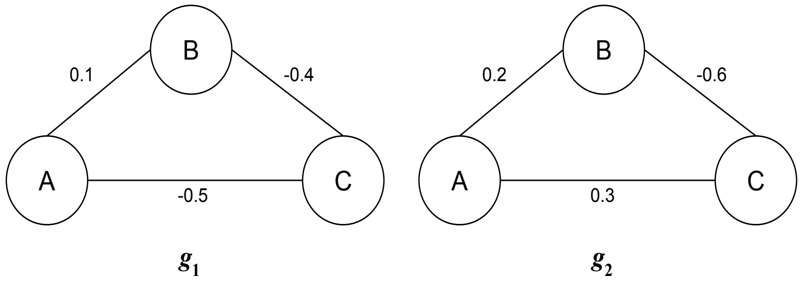


Figure 3.3: Two edge-weighted complete graphs.

each node (vector of each node is found from the weights associated with the adjacent edges of that node). Given two vectors S and T, weighted Jaccard between them is [45]:

$$J(S,T) = \frac{\sum_k min(S_k, T_k)}{\sum_k max(S_k, T_k)} \tag{3.5}$$

In Fig. 3.3, we show two example edge-weighted complete graphs. If g_i^j denotes the node j of graph g_i, then the calculation of node-wise Jaccard between these two edge-weighted complete graphs is as follows.

$$\begin{aligned} J(g_1, g_2) &= [J(g_1^A, g_2^A), J(g_1^B, g_2^B), J(g_1^C, g_2^C)] \\ &= [\frac{0.1 - 0.5}{0.2 + 0.3}, \frac{0.1 - 0.6}{0.2 - 0.4}, \frac{-0.5 - 0.6}{0.3 - 0.4}] \\ &= [-0.8, 2.5, 11] \end{aligned}$$

By calculating the node-wise Jaccard between each pair of complete graphs, we get a tensor $\mathcal{X} \in \mathbb{R}^{n \times n \times m}$ (Fig. 3.2c). After tensor decomposition, we get two identical factor matrices in the subjects space.

3.3.5 Tensor Model 5: Node-wise Jaccard Kernel on Non-negative Functional Connectivity Matrices

In this model, we ignore the negative weighted edges in the complete graphs. Similar to the calculation of node-wise Jaccard described for Tensor Model 4, we get a tensor $\mathcal{X} \in \mathbb{R}^{n \times n \times m}$ (Fig. 3.2c). We similarly get two identical factor matrices in the subject space after tensor decomposition.

We discuss the classification performances of the brain network representations modeled by above five tensor models found after CP and Tucker decomposition in the experiments chapter.

4 BIOMARKER EMBEDDING FROM BRAIN NETWORKS

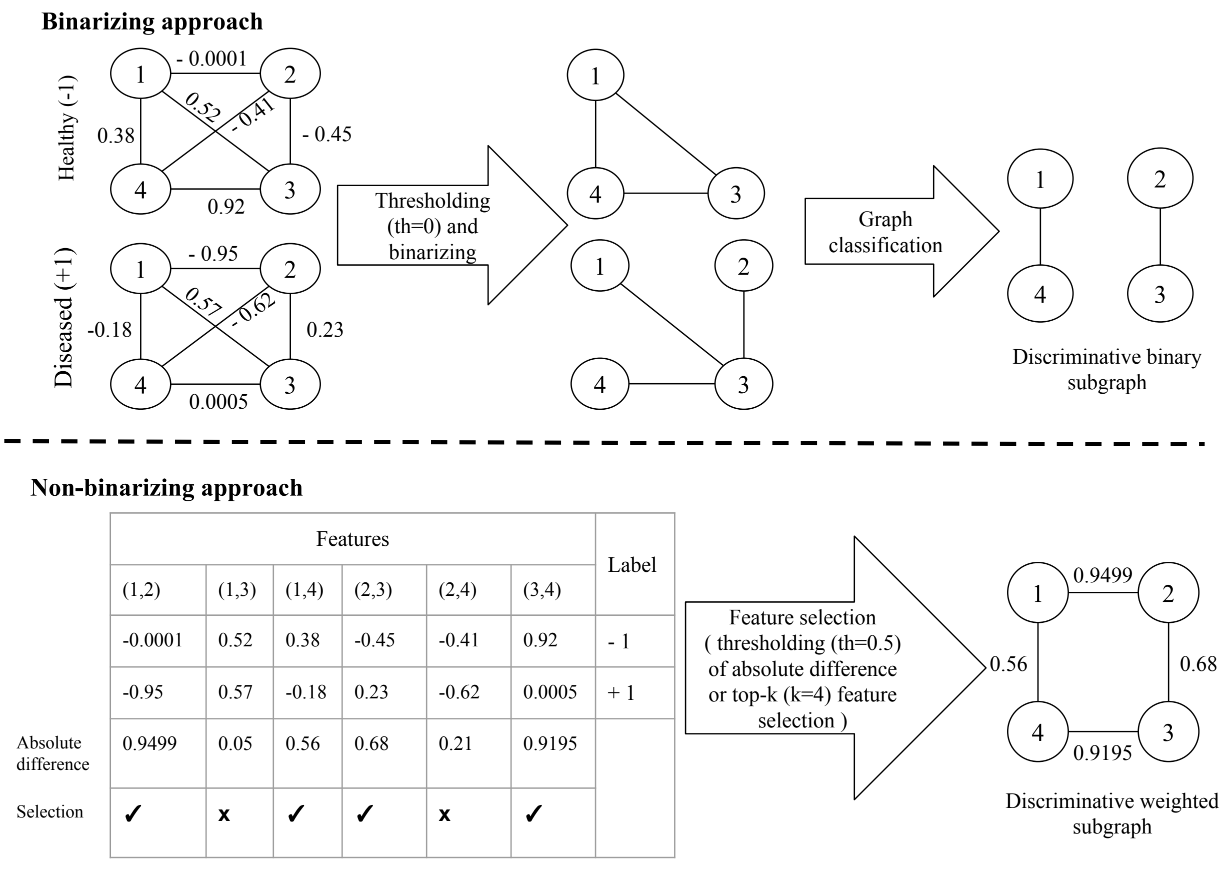
The comprehensive set of neuronal connections of the human brain, which is known as the human connectomes, has provided valuable insight into neurological and neurodevelopmental disorders. Functional Magnetic Resonance Imaging (fMRI) has facilitated this research by capturing regionally specific brain activity. fMRI-based functional connectomes are used to extract the complete functional connectivity networks, which are edge-weighted complete graphs. In the complete functional connectivity networks, each node represents one brain region or Region of Interest (ROI), and each edge weight represents the functional similarity of the adjacent ROIs. In order to leverage from the graph mining methodologies, these complete graphs are often made sparse by applying thresholds on weights. This approach can result in losing discriminative information, while addressing the issue of biomarkers detection, i.e., finding discriminative ROIs and connections, given the data of healthy and diseased population. We present a framework for representing the complete functional connectivity networks in a threshold-free manner and finding the biomarkers by using feature selection algorithms. Additionally, for computing meaningful representations of the discriminative ROIs and connections, we apply tensor decomposition techniques. The contents of this section are based on the papers [13] and [21].

4.1 *Overview*

Enriched by the neuroimaging technologies such as Magnetic Resonance Imaging (MRI), Positron Emission Tomography (PET), Electroencephalography (EEG), and Diffusion Tensor Imaging (DTI), *brain informatics* has been playing a key role in the investigation of neurological and neurodevelopmental disorders. The complexity, heterogeneity, and scarcity of brain informatics data pose a great challenge to data mining. Among the data collection modalities, functional MRI (fMRI) is a popular one, which measures the functional activities of different brain regions. fMRI is used to construct the functional connectivity network, a complete, edge-weighted graph, where the nodes represent brain regions (ROIs) and the edge-weights represent similarity/dissimilarity of two ROIs in their Blood Oxygenation Level Dependent (BOLD) time series. BOLD time series represents the aggregated activation of the neurons of the ROI over the scan period. The analysis of functional connectivity networks has facilitated the early diagnosis of several neurological and neurodevelopmental diseases such as Alzheimer's [36], Schizophrenia [46], Bipolar disorder [37], Attention-deficit/hyperactivity disorder (ADHD) [47], Autism [48], and Dyslexia [49].

Given a set of functional connectivity networks and associated healthy/diseased class labels, one of the most important research questions asked by the neuroscientist community is, "How can one find the biomarkers that distinguish two classes?" Data mining people, considering the problem as binary classification, mostly give two types of solutions - graph classification and tensor decomposition.

Because of the advances of graph mining algorithms, most supervised learning studies transform the complete functional connectivity networks into sparse graphs by thresholding and binarizing [36], [37], [40], [50], [38]. Then, they extract structure-based features such as degree, clustering coefficient and PageRank score of each node, and/or subgraph-based features such as gSpan-based frequent subgraphs [51] to construct a feature space.



	Features						Label
	(1,2)	(1,3)	(1,4)	(2,3)	(2,4)	(3,4)	
	-0.0001	0.52	0.38	-0.45	-0.41	0.92	- 1
	-0.95	0.57	-0.18	0.23	-0.62	0.0005	+ 1
Absolute difference	0.9499	0.05	0.56	0.68	0.21	0.9195	
Selection	✓	x	✓	✓	x	✓	

Binarizing approach
Healthy (-1)
Diseased (+1)
- 0.0001
0.52
- 0.41
0.38
- 0.45
0.92
- 0.95
0.57
- 0.62
-0.18
0.23
0.0005
Thresholding (th=0) and binarizing
Graph classification
Discriminative binary subgraph
Non-binarizing approach
Feature selection (thresholding (th=0.5) of absolute difference or top-k (k=4) feature selection)
0.9499
0.56
0.68
0.9195
Discriminative weighted subgraph

Figure 4.1: Non-binarizing approach can result in producing more significant biomarkers

While structure-based features are hand-engineered and require a lot of domain knowledge, subgraph-based features are computationally very expensive. Another disadvantage of thresholded and binarized graphs is that they can lose some discriminative information. Fig. 4.1 shows a motivating example, where two complete functional connectivity networks of different class labels are shown. Each network has four nodes and six weighted edges, where signed weights represent positive/negative correlation of the BOLD time series of the adjacent nodes (ROIs). For making the sparse graphs, the edges are positively thresholded and binarized. As a result, both representations fail to find the discriminative edges (1,2) and (3,4), which distinguish two classes by a big difference in same-signed weights.

This approach also can not weight the edges of the discriminative subgraph in terms of discrimination score. On the contrary, while thresholding and binarization are not applied and the complete graphs are represented by fixed-length vectors (since each complete graph has the same number of nodes and edges), simple feature selection scheme such as thresholding on absolute difference of edge weights and selecting the edges (features) with top-k absolute difference of weights, can produce more discriminative edges, e.g. (1,2), (2,3), (3,4) and (4,1) along with their discrimination scores. The unsupervised feature selection approach (shown for simplicity in this example) also uses apriori knowledge such as k in top-k feature selection, but still, this apriori knowledge is less expensive than the apriori knowledge about the sparsity of the graph [42].

Multi-dimensional arrays or tensor-based approaches stack the adjacency matrices of the thresholded functional connectivity networks to construct one third-order tensor and decompose the tensor into one subject factor matrix and two identical ROI factor matrices (since the adjacency matrices make partially symmetric tensor) [43], [52], [44]. Each column of the factor matrices represents one latent feature of different entities (subjects or ROIs). After tensor decomposition, the subject factor matrix is used for classifier training and testing. While this approach may result in high classification accuracy, the latent features, i.e., the columns of the subject factor matrix are difficult to interpret, because the relationship of these latent features with the discriminative ROIs and connections are not yet established [43].

In this work, we address the problem of graph thresholding by using vector representation and applying feature selection algorithms to find discriminative features, which eventually become the edges of the discriminative subgraph. We also address the issue of interpretability of tensor decomposition-based approach by constructing the tensor with ROI-connection-based incidence matrices of the induced discriminative subgraphs of the complete functional networks of the subjects, instead of the adjacency matrices of the thresholded functional connectivity networks.

The main contributions of this work are:

1. Transforming the fMRI-based biomarker detection problem from the graph space to vector space, i.e., solving the problem with functional connectivity vectors instead of thresholded graphs.

2. Using feature selection algorithms for constructing the discriminative subgraph, whose nodes and edges represent the biomarkers.

3. Proposing a novel tensor construction scheme with the computed discriminative subgraphs so that tensor decomposed factor matrices can be easily manipulated for determining the influence of the biomarkers on the human subjects.

4.2 *Existing Approaches*

Data mining research on fMRI-based functional connectivity data can be divided into two categories : discriminative subgraph-based graph classification and discriminative latent feature-based tensor decomposition. In this section, we discuss both approaches, and distinguish our approach from them.

Graph classification-based approaches: In these approaches, binary or weighted graphs are generated by applying a threshold on functional connectivity matrices so that all the nodes of the graphs are connected. Then, structure-based and/or subgraph pattern-based features are extracted for representing the graphs in feature space. Wee et al. [36] used functional connectivity matrices found from fMRI and DTI modalities to distinguish Mildly Cognitive Impaired (MCI) subjects from the healthy controls (MCI is the early stage of Alzheimer's disease). Their approach is an example of a structure-based graph classification approach. Given a functional network, weighted local clustering coefficient of each node is calculated, and the graph is represented by a vector consisting of these local connectivity measures. Then SVM classifier is applied to this vector space. This model gives a ranking of the ROIs in terms of how well they are clustered with respect to other ROIs, which provides a good step towards the interpretability of the biomarkers. Jie et al. [38] presented another structure-based graph classification approach, where they used Weisfeiler-Lehman graph kernel [53] for computing the global connectivity features of each graph, which are used along with a local connectivity feature of weighted local clustering coefficient of each node. Considering the thresholded edge weights as the probabilities of the link between two nodes, Kong et al. [50] presented a discriminative subgraph feature selection algorithm based on dynamic programming to compute the probability distribution of the discrimination scores for each subgraph pattern. In some cases of studies of neurological and neurodevelopmental disorders, along with the neuroimaging data, there may be additional clinical, serologic and cognitive measures data from each subject. Cao et al. [40] presented a discriminative subgraph mining algorithm for brain networks which leverages such multiple side views-based data.

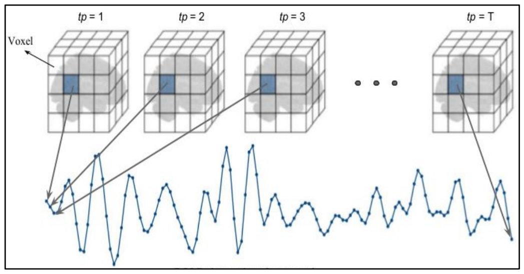
Tensor decomposition-based approaches: Tensor decomposition is used to extract the latent discriminative features of each subject. In [43], tensors are formed by stacking the non-negative connectivity matrices of all subjects. The resulting tensor is decomposed with several constraints such as symmetry of the factor matrix representing the ROI space and orthogonality of the factor matrix representing the subject space in order to maximize the discrimination among the subjects of different classes. In [44], the time-sliced non-negative connectivity matrices are used to create the tensors in order to discover the latent factors of the time windows. Both studies [43] and [44] modeled the problem as constrained CANDECOMP/PARAFAC (CP) decomposition and used Alternating Direction Method of Multipliers (ADMM) as the optimization framework.

Although tensor decomposition-based approaches represent each functional connectivity network in latent feature space, graph mining-based approaches can find meaningful discriminative patterns or biomarkers, i.e., discriminative ROIs and connections. However, the imposed threshold-based sparsity make these approaches lose some discriminative information. Moreover, some of these approaches rely on the exhaustive enumeration of the induced subgraphs, which can be computationally very expensive in terms of both space and time.

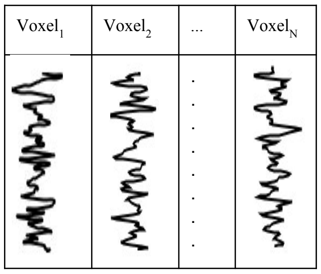
The model we present in this work does not use thresholds for imposing sparsity. By leveraging the completeness and equal number of nodes of the functional connectivity networks, our model enumerates the edges, and represent each functional connectivity network as fixed-length vector instead of enumerating the subgraphs. The edges whose weights are maximally dependent on the class label can be found by applying feature selection algorithms such as Fisher and minimal Redundancy Maximal Relevance (mRMR) criterion. The selected edges (features) form the discriminative subgraph. Therefore, our model for fMRI-based biomarker detection does a paradigm shift from frequent/discriminative subgraph mining to feature selection.

4.3 *Finding Discriminative Subgraph from Complete Functional Connectivity Networks*

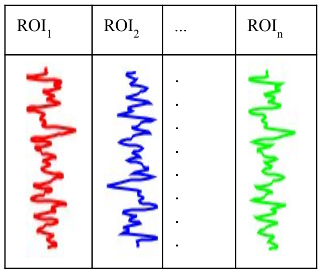
In this section, we discuss how to represent a complete functional connectivity network using a fixed-length functional connectivity vector. Although any feature selection algorithm can be plugged into our framework, for simplicity and brevity we discuss two well-known feature selection algorithms from the literature - Fisher [10], and mRMR [11], which can be used to find discriminative connections from vector represented complete functional connectivity networks.



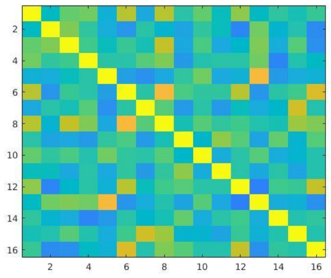
(a) BOLD time series extraction of the voxels by sequentially capturing whole brain volume over a period of time



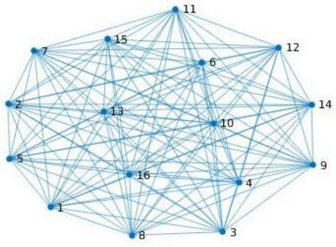
(b) Multivariate time series (voxel-level)



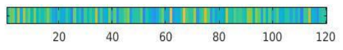
(c) Multivariate time series (ROI-level)



(d) Functional connectivity matrix found after pairwise Pearson correlation calculation on the ROI-based multivariate time series



(e) Complete functional connectivity network of 16 nodes and 120 edges



(f) Functional connectivity vector found by flattening the upper/lower triangular portion of functional connectivity matrix; each feature represents an edge of the complete graph

Figure 4.2: Extraction of functional connectivity vector

4.3.1 Functional Connectivity Vector

In its raw form, fMRI data is four-dimensional. The scanner captures a sequence of whole brain volumes of the subject in a regular time interval (Fig. 4.2a). Therefore, three dimensions represent spatial information and one dimension represents temporal information. In fMRI scan, a voxel is the unit of the brain volume. After a series of preprocessing steps, the BOLD time series of each voxel is found as the change of activation over a time period (Fig. 4.2b). The voxels are grouped into regions of interest (ROIs), where the regions can be defined manually by the neuroscientists targeting a specific disease relevant area or by some standard brain atlas such as Harvard-Oxford Atlas for whole brain analysis. A time series is found for each ROI, which is the mean of the time series across the voxels of that ROI. One representation of the fMRI data is the multivariate time series, where the variables are represented by the ROIs (Fig. 4.2c). From these ROI time series, pairwise Pearson correlation coefficients are calculated.

Then, Fisher's r-to-z transformation is applied on the elements of the correlation matrix to improve the normality of the correlation coefficients as

$$z = \frac{1}{2}\ln\left(\frac{1+r}{1-r}\right),$$

where r is the Pearson correlation coefficient. This z-map is used as another data representation, and called the functional connectivity matrix (Fig. 4.2d). Functional connectivity matrix is symmetric and can be considered as the adjacency matrix of a complete graph (Fig. 4.2e), where the entries of the matrix define the edge weights. A complete graph with n nodes has $e = n(n-1)/2$ edges. By considering each edge as a feature, the complete functional connectivity network of a subject is represented by an e dimensional vector (Fig. 4.2f). Functional connectivity vector is found by flattening the upper/lower triangular portion of the functional connectivity matrix. A dataset of n_s subjects, where each subject's functional connectivity network has n nodes with e edges is represented by $\{(X^{(i)}, y^{(i)})\}_{i=1}^{n_s}$, where $X^{(i)} \in \mathbb{R}^e$ and $y^{(i)} \in \{+1, -1\}$. The feature space is denoted by $X = \{x_1, x_2, \ldots, x_e\}$.

4.3.2 Mining Discriminative Subgraph by Feature Selection Algorithms

After extracting the functional connectivity vectors from the complete functional connectivity networks, we get a high dimensional feature space. Since each feature represents an edge or connection between two ROIs, selection of the edge features that are most statistically relevant to the class label results in a subgraph that can distinguish healthy and diseased classes. In the resultant discriminative subgraph, each edge is assigned a weight, which is the score assigned to its corresponding feature by a feature selection algorithm.

Feature selection algorithms can be divided into three categories - filters, wrappers and embedding methods [54]. Filter methods use the intrinsic property of the data and rank the features before feeding the reduced feature space into a classifier for learning. Wrappers use the classifier performance to evaluate the feature subset. Wrapper models tend to give better results, but they are classifier dependent and computationally more expensive than the filters. Embedding methods inject the feature selection process into the learning step of the classifier.

Because of being classifier independent, supervised filter-based feature selection algorithms can be used for discriminative subgraph mining from vector-represented functional connectivity data. Depending on whether the label information is used in feature selection, filter methods are divided into supervised (e.g., Fisher and mRMR), and unsupervised (e.g., maximum variance and Laplacian score) approaches. Since we consider supervised case in this work, i.e., biomarker detection from labeled functional connectivity networks, we propose supervised filter-based feature selection algorithms. Whether the edges of the discriminative subgraph depend on each other in distinguishing the classes, two variants of supervised filter-based feature selection algorithms can be applied - univariate and multivariate.

Univariate feature selection

Univariate feature selection algorithms consider each feature independently and ignore any correlation between them. Each edge of the discriminative subgraph selected by such algorithms is individually discriminating, and independent with other selected edges. Fisher scoring is an example of univariate feature ranking. Fisher score or F-score is the ratio of the inter-class distance and intra-class distance of a given feature [10]. Fisher score of a feature x_j is defined as,

$$F(x_j) = \frac{\sum_{k=1}^{c} n_k(\mu_j^k - \mu_j)^2}{(\sigma_j)^2},$$

where c denotes the number of classes, n_k denotes the number of samples of class k, μ_j and σ_j denote the mean and standard deviation of feature x_j, and μ_j^k denotes the mean of k-th class, corresponding to the feature x_j.

Multivariate feature selection

Multivariate feature selection algorithms select features whose combination ensures higher discrimination ability, although the individual discrimination ability of the selected features might be poor. The edges of the discriminative subgraph selected by such algorithms are combinedly discriminating. An example of multivariate feature selection is minimal Redundancy Maximal Relevance (mRMR) [11]. mRMR selects features sequentially, and the selected features are expected to have maximal relevance with the class labels while having minimal redundancy with already selected features.

If $f-1$ features are already selected by mRMR and the already selected feature set is denoted by S_{f-1}, then the algorithm finds the f^{th} feature from the set $X - S_{f-1}$ by optimizing the following condition:

$$\max_{x_j \in X - S_{f-1}} \left[I(x_j; y) - \frac{1}{f-1} \sum_{x_i \in S_{f-1}} I(x_j; x_i) \right],$$

where y is the class label variable and $I(p; q)$ is the mutual information between two random variables p and q. Mutual information between p and q is defined in terms of their probability density function $Pr(p)$, $Pr(q)$, and $Pr(p, q)$.

$$I(p; q) = \iint Pr(p, q) \log \frac{Pr(p, q)}{Pr(p)Pr(q)} dp dq$$

mRMR score of the f^{th} selected feature x_j is found by,

$$mRMR\_score(x_j) = I(x_j; y) - \frac{1}{f-1} \sum_{x_i \in S_{f-1}} I(x_j; x_i)$$

After scoring each feature by univariate/multivariate feature selection algorithm, best features can be selected by either one of the following two strategies.

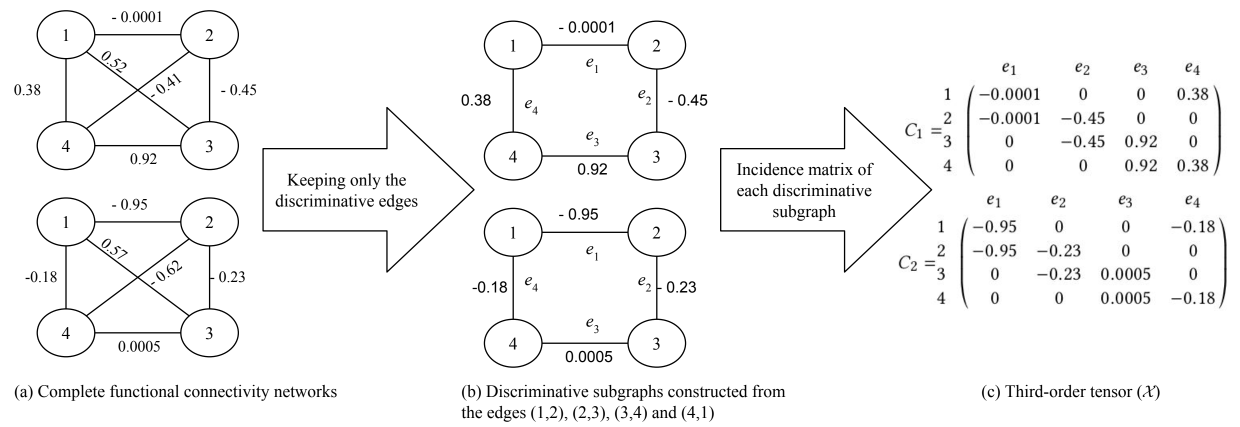
1. **Feature score thresholding:** All the features whose scores are below the given threshold, are removed.

2. **Top-k feature selection:** Features are sorted in descending order of their scores. Top-k features with maximum scores are selected.

Finally, the selected features provide the edges of the discriminative subgraph, where the feature scores become the weights of the edges. In the discriminative subgraph, the nodes (ROIs) and the edges (connections) can be considered as the discriminative patterns or biomarkers of a neurological or neurodevelopmental disease. Visualization of the edge-weighted discriminative subgraph can help the neuroscientists in their analysis of these diseases.

Table 4.1: Notations

Symbol	Definition	Symbol	Definition
n	Number of nodes in each complete graph	$\mathcal{X}$	Tensor
e	Number of edges in each complete graph	$\mathbf{S}$	Subject factor matrix
n_s	Number of subjects in the dataset	$\mathbf{N}$	Discriminative nodes factor matrix
n_{dn}	Number of discriminative nodes in the dataset	$\mathbf{E}$	Discriminative edges factor matrix
n_{de}	Number of discriminative edges in the dataset	$\mathbf{I}_k$	Identity matrix of size k
$\mathbf{C}$	Incidence matrix	P, Q, R	Ranks of tensor decomposition


- 0.0001
0.52
- 0.41
0.38
- 0.45
0.92
- 0.95
0.57
- 0.62
-0.18
0.23
0.0005
Keeping only the discriminative edges
Incidence matrix of each discriminative subgraph
e_1 e_2 e_3 e_4
(a) Complete functional connectivity networks
(b) Discriminative subgraphs constructed from the edges (1,2), (2,3), (3,4) and (4,1)
(c) Third-order tensor ($\mathcal{X}$)

$$C_1 = \begin{array}{c|cccc} & e_1 & e_2 & e_3 & e_4 \\ \hline 1 & -0.0001 & 0 & 0 & 0.38 \\ 2 & -0.0001 & -0.45 & 0 & 0 \\ 3 & 0 & -0.45 & 0.92 & 0 \\ 4 & 0 & 0 & 0.92 & 0.38 \end{array}$$

$$C_2 = \begin{array}{c|cccc} & e_1 & e_2 & e_3 & e_4 \\ \hline 1 & -0.95 & 0 & 0 & -0.18 \\ 2 & -0.95 & -0.23 & 0 & 0 \\ 3 & 0 & -0.23 & 0.0005 & 0 \\ 4 & 0 & 0 & 0.0005 & -0.18 \end{array}$$

Figure 4.3: Constructing third-order tensor from induced discriminative subgraph of the complete functional connectivity networks

4.4 *Representation Learning for the Biomarkers*

Given a set of biomarkers, i.e., discriminative ROIs and connections, how does a subject's functional connectivity network interact with them? Good representation of biomarkers facilitates the computation of the impact of them on the functional connectivity network of a subject. We address this issue by tensor decomposition technique. In this section, we discuss the procedure of tensor construction, tensor decomposition techniques, and utilizing the resultant biomarker factor matrices for computing the biomarker impacts on healthy and diseased subjects. The important notations used in this section are summarized in Table 4.1.

4.4.1 Tensor Construction

A third-order tensor is constructed by stacking the weighted incidence matrices of the discriminative subgraphs induced in the complete functional connectivity network of each subject. Three modes of the tensor are the subjects, discriminative nodes, and discriminative edges. The set of discriminative edges, which is computed by a feature selection algorithm after vectorizing all the complete functional connectivity networks of the dataset, can be used to construct the discriminative subgraph of each subject. Incidence matrix of the discriminative subgraph has a row for each discriminative node and a column for each discriminative edge. The weighted incidence matrix $\mathbf{C} \in \mathbb{R}^{n_{dn} \times n_{de}}$ is defined as,

$$\mathbf{C}_{i,j} = \begin{cases} w_j, & \text{if node } i \text{ is end-node of edge } j \\ 0, & \text{otherwise} \end{cases}$$

for $1 \leqslant i \leqslant n_{dn}$ and $1 \leqslant j \leqslant n_{de}$. The weighted incidence matrices of the discriminative subgraph of all subjects are stacked to construct a third-order tensor $\mathcal{X} \in \mathbb{R}^{n_s \times n_{dn} \times n_{de}}$. An example is shown in Fig. 4.3. In this example, the functional connectivity networks are the same as the functional connectivity networks of Fig. 4.1, and the discriminative edges are (1,2), (2,3), (3,4), and (4,1).

4.4.2 Tensor Decomposition in Biomarker Embedding

Tensor decomposition is performed to acquire the latent factor-based representations of the objects defined in each mode of a higher order tensor. Two widely used tensor decomposition techniques are CP decomposition and Tucker decomposition [34]. In

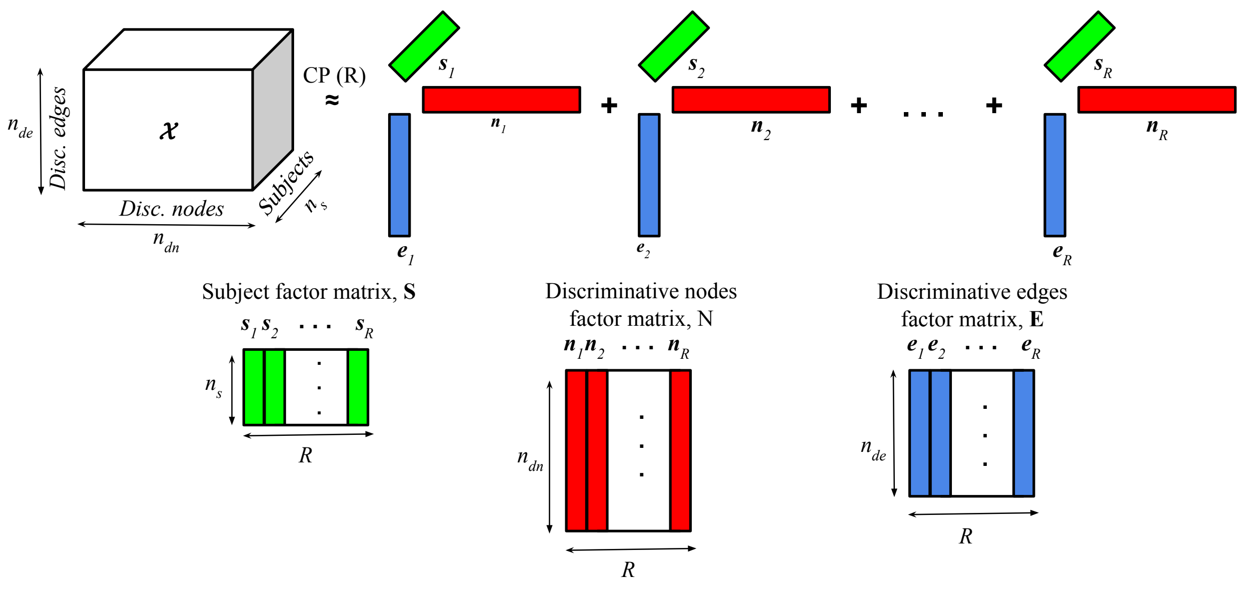


Figure 4.4: CP decomposition of the *subjects-discriminative nodes-discriminative edges*-based tensor.

this subsection, we briefly discuss them in terms of our *subjects-discriminative nodes-discriminative edges*-based tensor.

CP Decomposition

CP decomposition factorizes the tensor into a sum of rank one tensors. Given a third order tensor $\mathcal{X} \in \mathbb{R}^{n_s \times n_{dn} \times n_{de}}$, CP decomposition factorizes the tensor as follows.

$$\mathcal{X} \approx \sum_{r=1}^{R} \mathbf{s}_r \circ \mathbf{n}_r \circ \mathbf{e}_r = [\![\mathbf{S}, \mathbf{N}, \mathbf{E}]\!]$$

Here, **o** denotes the outer product of the vectors. R is a positive integer and also called the tensor rank. $\mathbf{s}_r, \mathbf{n}_r$, and $\mathbf{e}_r$ are vectors, where $\mathbf{s}_r \in \mathbb{R}^{n_s}$, $\mathbf{n}_r \in \mathbb{R}^{n_{dn}}$, and $\mathbf{e}_r \in \mathbb{R}^{n_{de}}$ for $r = 1, 2, 3, \ldots, R$. After stacking those vectors, we can get the factor matrices $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \ldots \mathbf{s}_R]$, $\mathbf{N} = [\mathbf{n}_1, \mathbf{n}_2, \ldots \mathbf{n}_R]$, and $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \ldots \mathbf{e}_R]$. We show a visualization of the CP decomposition of our *subjects-discriminative nodes-discriminative edges*-based tensor in Fig. 4.4. Each row of a factor matrix is a R-dimensional representation of an object, i.e., subject, discriminative node, or discriminative edge. In CP decomposition, there are no imposed orthogonality constraints for the factor matrices. Nevertheless, we can compute the impact of the discriminative node j on subject i by the inner product:

$$\mathbf{S}(i,:) * \mathbf{N}(j,:)^T$$

Similarly, we can compute the impact of the discriminative edge j on subject i by the inner product:

$$\mathbf{S}(i,:) * \mathbf{E}(j,:)^T$$

Tucker decomposition

Tucker decomposition is a form of higher order Principal Component Analysis (PCA). A tensor is decomposed into a core tensor, which is multiplied by a matrix along its each mode. Tucker decomposition of a third order tensor $\mathcal{X} \in \mathbb{R}^{n_s \times n_{dn} \times n_{de}}$ is given by,

$$\mathcal{X} \approx \mathcal{G} \times_1 \mathbf{S} \times_2 \mathbf{N} \times_3 \mathbf{E} = [\![\mathcal{G}; \mathbf{S}, \mathbf{N}, \mathbf{E}]\!] \tag{4.1}$$

Here, $\times_n$ denotes mode-n tensor product. $\mathbf{S} \in \mathbb{R}^{n_s \times P}$, $\mathbf{N} \in \mathbb{R}^{n_{dn} \times Q}$, and $\mathbf{E} \in \mathbb{R}^{n_{de} \times R}$ are the factor matrices. These factor matrices can be thought as the principal components along each mode. $\mathcal{G} \in \mathbb{R}^{P \times Q \times R}$ is the core tensor and the elements of this tensor represents the interaction between those principal components. We show a visualization of the Tucker decomposition of our *subjects-discriminative nodes-discriminative edges*-based tensor in Fig. 4.5.

Since we do the projection of the rows of the subject factor matrix on the rows of factor matrices of the biomarkers, we require each biomarker factor matrix to be orthogonal.

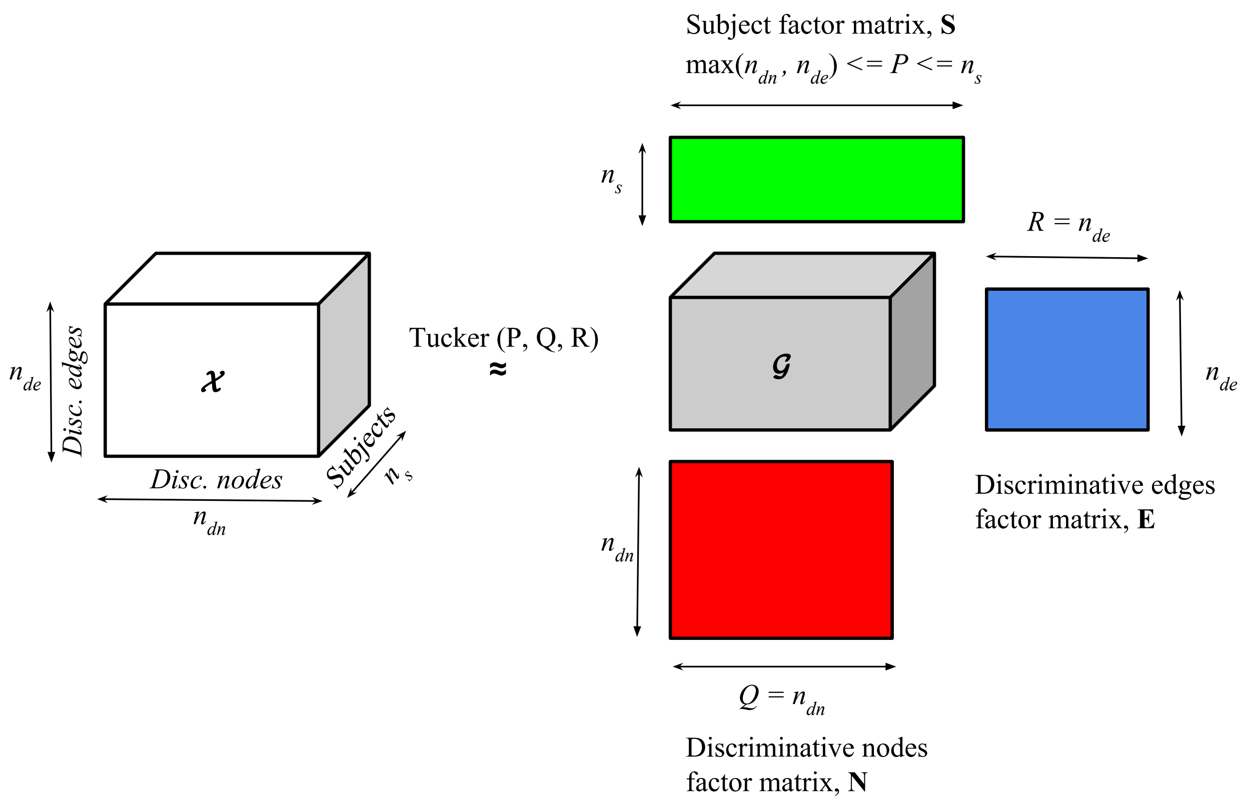


Figure 4.5: Tucker decomposition of the *subjects-discriminative nodes-discriminative edges*-based tensor.

Additionally, we require the columns of the subject factor matrix to be pairwise orthogonal, because orthogonal columns (features) of subject factor matrix ensures meaningful representation of each row. If $n_s > n_{dn}$ and $n_s > n_{de}$, by setting $max(n_{dn}, n_{de}) \leqslant P \leqslant n_s$, $Q = n_{dn}$, and $R = n_{de}$, Tucker decomposition ensures follwing constraints to be held.

- $\mathbf{N}\mathbf{N}^T = \mathbf{N}^T\mathbf{N} = \mathbf{I}_{n_{dn}}$, i.e., discriminative nodes factor matrix, $\mathbf{N}$ is an orthogonal matrix.
- $\mathbf{E}\mathbf{E}^T = \mathbf{E}^T\mathbf{E} = \mathbf{I}_{n_{de}}$, i.e., discriminative edges factor matrix, $\mathbf{E}$ is an orthogonal matrix.
- $\mathbf{S}^T\mathbf{S} = \mathbf{I}_P$, i.e., columns of the subject factor matrix $\mathbf{S}$ are pairwise orthogonal.
- $\mathbf{S}\mathbf{S}^T \neq \mathbf{I}_{n_s}$, i.e., rows of the subject factor matrix $\mathbf{S}$ are not pairwise orthogonal.

Now, we can compute the impact of the discriminative node j on subject i by the inner product:

$$\mathbf{S}(i, 1 : n_{dn}) * \mathbf{N}(j, :)^T$$

Similarly, we can compute the impact of the discriminative edge j on subject i by the inner product:

$$\mathbf{S}(i, 1 : n_{de}) * \mathbf{E}(j, :)^T$$

Both CP and Tucker decomposition can be solved by Alternating Least Squares (ALS) optimization. After a random initialization of all factor matrices, ALS updates one factor matrix while keeping other two as fixed until convergence. The details of ALS optimization for CP and Tucker decomposition can be found in [34].

Now we summarize our proposed framework, which combines the feature selection-based discriminative subgraph mining method with the tensor decomposition-based representation learning of the biomarkers.

1. Firstly, we represent the complete functional connectivity networks of the dataset as fixed-length functional connectivity vectors.
2. We apply univariate/multivariate filter-based supervised feature selection algorithm for finding top-k discriminative features. By incrementally selecting features, and feeding the reduced feature space to a classifier, while train and test examples are selected by a predefined cross-validation scheme, we can observe the classification accuracies found from the selected feature sets. The feature set resulting in the best cross-validation accuracy is considered to be the edge set of the discriminative subgraph, where the edges are weighted by their corresponding feature scores.
3. After we find the discriminative subgraph of the dataset, we compute the impact of its nodes and edges on a given subject by decomposing the third-order tensor, whose three modes represent subjects, discriminative nodes, and discriminative edges.

5 TIME SERIES-BASED SOLAR FLARE PREDICTION AND VISUALIZATION

As an application of time series-based feature learning, we discuss solar flare prediction in this chapter. In [23], [24], and [25], we presented a multivariate time series classification-based approach for solar flare prediction. Solar flare prediction is an important task because of their potential impacts on both space and terrestrial infrastructure [55]. This prediction task can be modeled as a binary classification between flaring and non-flaring Active Regions. Previous works on flare prediction focused on representing flaring and non-flaring Active Region examples in vector space, where the feature space was found from the Active Region magnetic field parameters. We extract time series samples of these Active Region parameters and present a flare prediction method based on the k-NN classification of the univariate time series. We find that, for our classification task, using a statistical summarization on the time series of a single Active Region parameter, called *total unsigned current helicity*, outperforms the use of all Active Region parameters at a single instant of time. Additionally, we present a data model of the flaring/non-flaring Active Regions using multivariate time series. In the end of this chapter, we discuss tensor decomposition-based flaring and non-flaring solar event visualization.

5.1 *Overview*

Solar flares are sudden bursts of radiation from Sun's surface. Solar events such as flares and Coronal Mass Ejections (CMEs) can have hazardous impacts on infrastructures both in space and on the ground. X-rays and UV radiation of large flares can cause radio blackout. Energetic particle flux from flares can cause solar radiation storm, which can have negative health effects on astronauts, aircrew and airline passengers, as well as negative technological impacts on electronic devices of the satellites, aircraft, and even the devices located on the ground [56]. Therefore, precise forecasting and prediction of severe space weather conditions such as M-class and X-class flares can save infrastructures in space and on the ground, whose replacement/repairing cost might be trillions of dollars [57]. Fig. 5.1 is an example of an M-class flare [58].

Since theoretical models of solar flare occurrence, such as the relationship between the photospheric and coronal magnetic field of the Sun during the flare occurrence, are not fully understood, heliophysics community relies on data-driven approaches for flare prediction. As most flares occur in the Active Regions of the Sun, flare prediction can be modeled as a supervised learning problem of machine learning, specifically the binary classification between flaring and non-flaring Active Regions (AR), where flaring Active Regions are considered to be in the positive class and non-flaring Active Regions are considered to be in the negative class. In this work, as positive class examples, we consider the Active Regions that have one or more M-class or X-class flares during their crossing of the observable solar disk. The Active Regions that have never flared during the disk crossing (not even C-class flares) are considered as negative class examples.

While the previous studies on flare prediction focused on the vector-based representation of flaring and non-flaring Active Regions, where the feature space is formed by the magnetic field-based AR parameters formulated by solar physicists [19, 59, 60], we focus on the time series properties of the AR parameters. These time series are extracted based on two time windows : *lookback* (the time window before which the flare happens), and *span* (the time window during which the AR parameter values are calculated).

Time series representation of the AR parameters can be used to rank them based on the time series quality in classification. When the time series of each AR parameter is considered, the problem becomes the multivariate time series classification problem. By

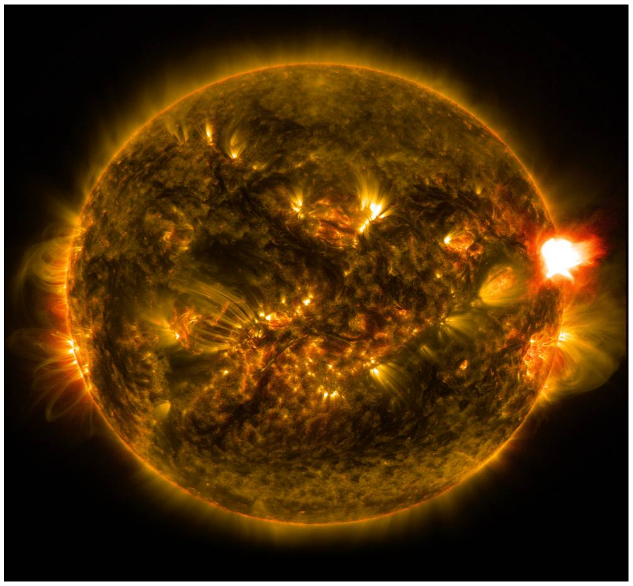


Figure 5.1: Image of an M-class flare which erupts from the right side of the Sun at 11:24 p.m. EST on Jan. 12, 2015. Credit: NASA/SDO.

finding the AR parameter whose time series exhibit the best classification performance, the problem can be simplified into the single-variate time series classification. If time series data of a single AR parameter is utilized for flare prediction, then time series classification algorithms which are successfully applied on other domains such as stock market behavior prediction [61], handwriting recognition [62] and so on, can similarly be applied to flare prediction.

The complexity of time series classification greatly depends on the length of the time series, because each time step increase in the time series length is an increase in the dimension of the input vector space, leading to the curse of dimensionality. To overcome this problem, we propose to use a statistical summarization of the time series. The summarized time series of the best AR parameter is considered as the vector-based representation of flaring/non-flaring AR, and k-nearest neighbors (k-NN) classifier is used on this vector space. By considering the time series of only one AR parameter, which is selected based on the classification performance on different datasets, our model reduces the cost of calculating multiple AR parameters, while exhibiting better classification performance than the models where all AR parameters are used with the values at a single instant of time.

The contributions made by this work are listed below.

1. Data modeling of the flaring/non-flaring Active Regions using multivariate time series based on lookback and span time windows.

2. Finding the best AR parameter in terms of its time series quality in classification.

3. Making the summarized representation of the time series of the best AR parameter to form a new vector space of flaring and non-flaring Active Regions. The performance of k-NN classifier on this vector space is better than the state-of-the-art flare prediction models.

4. Experimentally validating that the consideration of C-class flares in positive class does not improve classification performance.

5. Leveraging tensor decomposition for unsupervised visualization of flaring and non-flaring active regions in 2D scatterplots.

5.2 *Related Work*

While most of the current methods of flare prediction are data-driven approaches, the earliest flare prediction system was THEO [63], which was an expert system that required human input. The system was adopted by Space Environment Center (SEC) of National Oceanic and Atmospheric Administration (NOAA) in 1987. It used a set of sunspot and magnetic field properties to predict different flare classes.

Later efforts of flare prediction are mostly based on data-driven approaches rather than on purely theoretical modeling. Data-driven approaches are divided into two categories -

linear statistical and nonlinear statistical (mostly machine learning). These two categories can be subdivided into two subcategories - line-of-sight magnetogram-based models and vector magnetogram-based models.

Active Regions are parameterized either by photospheric magnetic field data that contain only the line-of-sight component of the magnetic field or by the full-disk photospheric vector magnetic field. After the launch of Solar Dynamics Observatory (SDO) by NASA in 2010, its instrument Helioseismic and Magnetic Imager (HMI) has been mapping the full-disk vector magnetic field every 12 minutes [64]. Although the continuous stream of vector magnetogram is a better means for parameterizing the Active Regions, it was not easily available before 2010 and people had to use line-of-sight magnetic data for flare prediction.

Linear statistical studies focus on identifying the AR magnetic properties that are correlated with the flares. Cui et al. [65] and Jing et al. [66] used line-of-sight magnetogram to parameterize Active Regions and studied correlation-based statistical relationships between those AR parameters and flare occurrences. Leka and Barnes [59] calculated vector magnetogram-based AR parameters for the first time and used linear discriminant analysis (LDA) for classification. They collected vector magnetogram data from Mees Solar Observatory Imaging Vector Magnetograph on the summit of Mount Haleakala.

Nonlinear statistical models mainly use machine learning-based classifiers. After parameterizing the Active Regions with line-of-sight magnetograms, Ahmed et al. [67] used the artificial neural network, Yu et al. [68] used C4.5 decision tree, Song et al. [69] used logistic regression, and Al-Ghraibah et al. [70] used relevance vector machine as classification models. Qahwaji et al. [71] considered McIntosh classification of sunspot groups and solar cycle data and used support vector machine (SVM) and Cascade-Correlation Neural Networks (CCNN) for prediction. Bobra et al. [19] used SVM on the AR parameters derived from vector magnetograms. Nishizuka et al. [60] used both line-of-sight and vector magnetograms and compared the performance of three classifiers - k-NN, SVM, and extremely randomized tree (ERT).

Almost all of the abovementioned works focussed on the parameterization of the Active Regions by line-of-sight or vector magnetograms but did not consider the impact of the time series of the individual AR parameters that can be extracted for a particular duration of time before the occurrence of the flare. In this work, we evaluated the vector magnetogram-based AR parameters based on their time series quality to distinguish flaring and non-flaring Active Regions.

5.3 *Data Modeling of the Active Regions*

In this section, we define some terminologies and present a formal data model for our flaring and non-flaring Active Regions (Fig. 5.2). Each Active Region instance is initially represented by six data fields.

$$event = \langle id, timestamp, lookback, span, mvts, label \rangle$$

Here, id is the NOAA Active Region number and $timestamp$ is the occurrence time of the flare (for flaring Active Regions) and the sampling time before which the parameter data are collected (for non-flaring Active Region). The $lookback$ represents the time window before the occurence of the flare. The $span$ is the time window for sampling AR patches from SDO/HMI images in 12 minutes cadence. These AR patches are used to get timewise magnetic field values $B_1, B_2, \ldots, B_T$, where $B_i = [B_{\phi i}, B_{\theta i}, B_{ri}]$ is formed from the components of the vector magnetic field data (see [19] for details). The magnetic field values are used to calculate N parameter values of the Active Region. The $mvts$ is a collection of time series $\{P_1, P_2, \ldots, P_N\}$, where P_j represents the time series of j-th

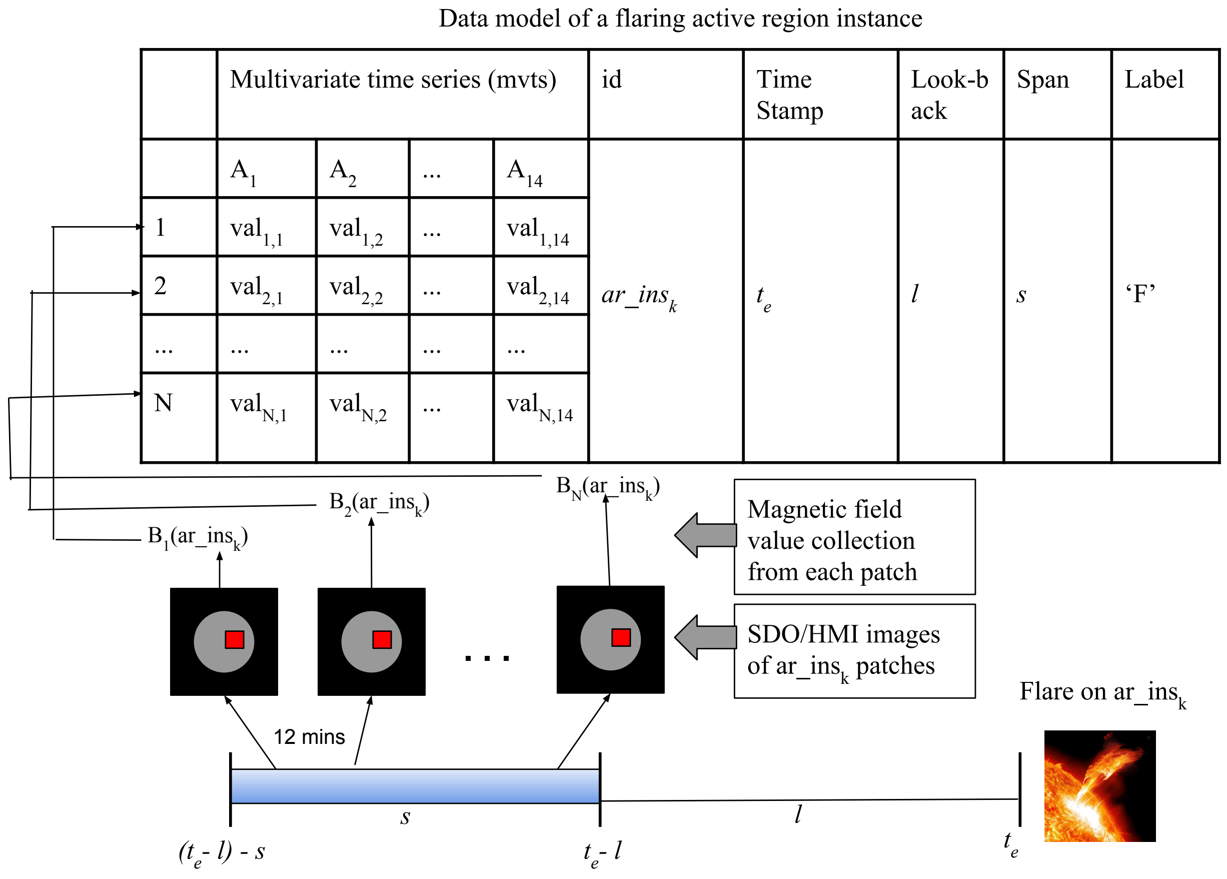


Figure 5.2: Data model of a flaring Active Region in terms of id, generation time, lookback time, span time, multivariate time series, and label.

parameter. Each time series has fixed length T. If the time unit of $span$ is hours, and the time unit of $cadence$ is minutes, then $T = \frac{span \times 60}{cadence}$. Therefore, P_j is a vector of length T and represented by $P_j = [v_{1,j}, v_{2,j}, \dots, v_{T,j}]$, where $v_{k,j}$ is the k-th value of the time series P_j. If an M-class or X-class flare occurs in time t_e of an Active Region, then the $label$ of this example is $+1$. If no M-class or X-class flare occurs during the disk crossing of the Active Region, then the $label$ of that example is -1. The data model of the flaring Active Regions is depicted in Fig. 5.2.

5.4 *Flare Classification Model*

5.4.1 Problem Definition

Given lookback l hours and span s hours, a dataset of M events is represented by $\{(X_i, y_i)\}_{i=1}^{M}$, where $X_i = mvts_i \in \mathbb{R}^{T \times N}$ and $y_i \in \{+1, -1\}$. In binary classification, if Tr is the number of labeled events, i.e., training examples, we train a classifier with Tr labeled examples $\{(X_i, y_i)\}_{i=1}^{Tr}$ and use the classification model to label $M - Tr$ test examples $\{(X_i)\}_{i=Tr+1}^{M}$. In this work, we aim to find a single AR parameter P_j, where $1 \leqslant j \leqslant N$, so that its corresponding time series can give the best classifying features. By utilizing only one parameter time series, we reduce the data to $\{(X_i(:, P_j), y_i)\}_{i=1}^{M}$, where $X_i(:, P_j) \in \mathbb{R}^{T}$, so that the dimensionality decreases and the classification performance increases in comparison with other representations $\{(X_i(:, P_{j'}), y_i)\}_{i=1}^{M}$, where $j' \neq j$.

5.4.2 Summarization of Time Series

We represent each time series of length T using 8 summary statistics [72], [73]. First four of them are mean (μ), standard deviation (σ), skewness ($SKEW$), and kurtosis ($KURT$) of the time series. Formulas of these statistics on the time series $P = [v_1, v_2, \ldots, v_T]$ are as follows.

$$\mu(P) = \frac{\sum_{i=1}^{T} v_i}{T} \tag{5.1}$$

$$\sigma(P) = \sqrt{\frac{\sum_{i=1}^{T}(v_i - \mu(P))^2}{T}} \tag{5.2}$$

$$SKEW(P) = \frac{\sum_{i=1}^{T}(v_i - \mu(P))^3}{T\sigma(P)^3} \tag{5.3}$$

$$KURT(P) = \frac{\sum_{i=1}^{T}(v_i - \mu(P))^4}{T\sigma(P)^4} - 3 \tag{5.4}$$

Then we calculate the first derivative of the time series P, which is given by $P' = [v'_1, v'_2, \ldots, v'_{T-1}]$.

$$v'_i = v_{i+1} - v_i, 1 \leqslant i \leqslant T - 1$$

Finally, we calculate the same statistics, i.e., mean, standard deviation, skewness, and kurtosis of P'.

$$\mu(P') = \frac{\sum_{i=1}^{T-1} v_i'}{T-1} \tag{5.5}$$

$$\sigma(P') = \sqrt{\frac{\sum_{i=1}^{T-1}(v_i' - \mu(P'))^2}{T-1}} \tag{5.6}$$

$$SKEW(P') = \frac{\sum_{i=1}^{T-1}(v_i' - \mu(P'))^3}{(T-1)\sigma(P')^3} \tag{5.7}$$

$$KURT(P') = \frac{\sum_{i=1}^{T-1}(v_i' - \mu(P'))^4}{(T-1)\sigma(P')^4} - 3 \tag{5.8}$$

Equations (1) - (8) provide 8 summary statistics of a time series P of length T. These 8 numbers make a vector u which can be thought as a summarized representation of the time series P, and is given by

$$\begin{aligned} u(P) = [&\mu(P), \sigma(P), SKEW(P), KURT(P), \\ &\mu(P'), \sigma(P'), SKEW(P'), KURT(P')] \end{aligned}$$

This summarization method can be used for feature-based representation of a time series of any length.

5.4.3 Parameter Selection and Classification

To assess the classification ability of the time series of each AR parameter, we make datasets $D_j = \{(X_i(:, P_j), y_i)\}_{i=1}^{M}$ for $1 \leqslant j \leqslant N$ with given lookback l and span s. Since $D_j \in \mathbb{R}^{T \times M}$, and the performance and runtime of time series-based classifiers depend on the number of dimensions of the vector space, i.e., the length of the time series T, we summarize each time series (column of D_j) by 8 summary statistics. After reducing the data to $D_j \in \mathbb{R}^{8\times M}$, we use k-NN classifier to distinguish examples of two classes.

We divide the dataset D_j into a training dataset $D_{j\_train} \in \mathbb{R}^{8\times Tr}$ and a testing dataset $D_{j\_test} \in \mathbb{R}^{8\times(M-Tr)}$. For each test example in D_{j_test}, k nearest training examples from D_{j_train} are found by calculating the Euclidean distance in 8 dimensional space. The class label of the test example is found from the most common class label among its k neighbors of the training dataset [74], [75]. When $k = 1$, the test example is assigned the class label of its nearest training example. If k is even and there is a tie between the numbers of positive and negative nearest neighbors, then the class label of the nearest neighbor is chosen.

By varying the number of neighbors (k) in k-NN classifier, lookback window size, and span window size, we measure the performance metrics. The AR parameter whose summarized time series get consistently better score with k-NN classifier than the summarized time series of other AR parameters can be considered to be the best AR parameter in distinguishing flaring and non-flaring Active Regions.

5.5 *Solar Event Visualization*

Visualizing high dimensional data in a scatter plot of two or three dimensions is a very important data analytics task. Principal Component Analysis (PCA) and Distributed Stochastic Neighbor Embedding (t-SNE) are two popular visualization techniques. PCA transforms the raw data from its original dimensionality to first k eigenvectors by the eigendecomposition of the covariance matrix of the vector represented data. It tries to provide a minimum number of variables that keep the maximum amount of variation or information about how the original data is distributed. Contrary to PCA, t-SNE is not a mathematical technique but a probabilistic one. t-SNE minimizes the divergence

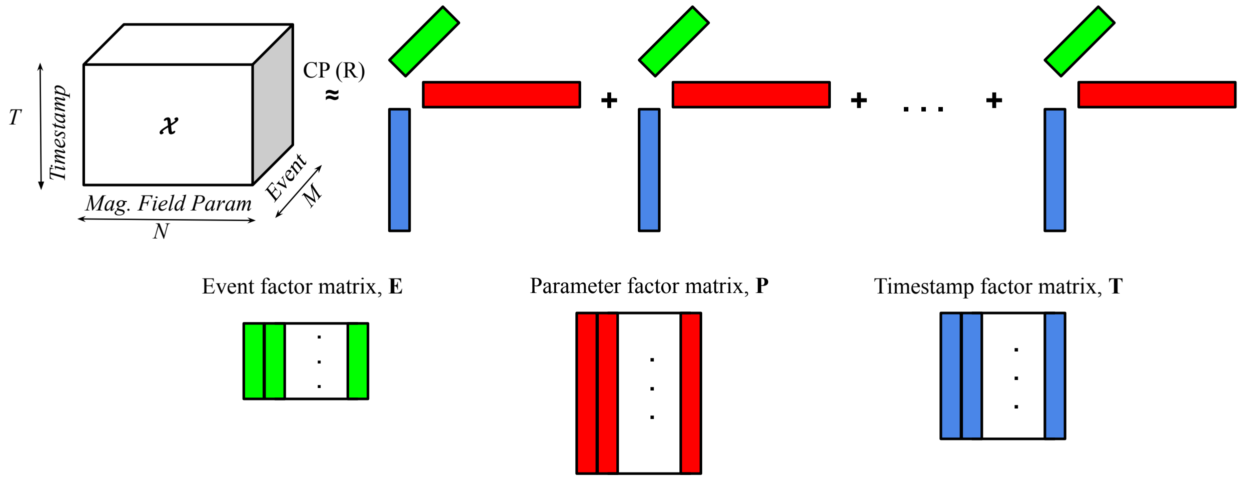

T
Timestamp
$\mathcal{X}$
Mag. Field Param
N
Event
M
CP (R)
≈
+
+ . . . +
Event factor matrix, **E**
Parameter factor matrix, **P**
Timestamp factor matrix, **T**

Figure 5.3: CP decomposition-based representation learning of solar events, timestamps, and magnetic field parameters

between two distributions: the distribution that measures pairwise similarities of the original high dimensional space, and a distribution that measures pairwise similarities of the corresponding low-dimensional representation space [17].

A dataset of M flaring and non-flaring events is represented by $\{(X_i)\}_{i=1}^{M}$, where $X_i = mvts_i \in \mathbb{R}^{T \times N}$. Here M is the number of events, T is the number of timestamps, and N is the number of magnetic field parameters (section 5.3). We avoid using the labels of flaring and non-flaring events to support the unsupervised behavior of this task.

Fig. 5.3 describes our model of tensor decomposition-based solar events embedding. A third-order tensor $\mathcal{X} \in \mathbb{R}^{M \times T \times N}$ is constructed by stacking M multivariate time series matrices $mvts_i \in \mathbb{R}^{T \times N}$ for $1 \leqslant i \leqslant M$. The objects represented by the three modes of this tensor are: solar events, timestamps, and magnetic field parameters. Then CP decomposition is performed with a given rank R. CP decomposition results in vectors $\mathbf{e}_r \in \mathbb{R}^M$, $\mathbf{t}_r \in \mathbb{R}^T$, and $\mathbf{p}_r \in \mathbb{R}^N$ for $r = 1, 2, 3, \ldots R$. These vectors are stacked together to form three factor matrices, $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \ldots \mathbf{e}_R]$, $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \ldots \mathbf{t}_R]$, and $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \ldots \mathbf{p}_R]$, where $\mathbf{E} \in \mathbb{R}^{M \times R}$, $\mathbf{T} \in \mathbb{R}^{T \times R}$, and $\mathbf{P} \in \mathbb{R}^{N \times R}$.

In factor matrix $\mathbf{E} \in \mathbb{R}^{M \times R}$, each row is an R-dimensional representation of the solar event. In factor matrix $\mathbf{T} \in \mathbb{R}^{T \times R}$, each row is an R-dimensional representation of the individual time instant of the time series. In factor matrix $\mathbf{P} \in \mathbb{R}^{N \times R}$, each row is an R-dimensional representation of the magnetic field parameter.

After we find the event factor matrix $\mathbf{E}$, timestamp factor matrix $\mathbf{T}$, and parameter factor matrix $\mathbf{P}$, we can compute the projection of embedding of event i on the embedding of timestamp j, where $1 \leqslant i \leqslant M$ and $1 \leqslant j \leqslant T$, and get *event-timestamp* embedding matrix $\mathbf{ET} \in \mathbb{R}^{M \times T}$. Similarly, we can get a *event-parameter* embedding matrix $\mathbf{EP} \in \mathbb{R}^{M \times N}$ that reflects the projection of event embeddings on parameter embeddings. Finally, we get the event embedding matrix $\mathbf{Z} \in \mathbb{R}^{M \times (T+N)}$ by concatenating $\mathbf{ET}$ and $\mathbf{EP}$. First T columns of $\mathbf{Z}$ represent the individual event with respect to each timestamp, and last N columns of $\mathbf{Z}$ represent the individual event with respect to each magnetic field parameter. By this way, we get an interpretable feature space of the solar events, where the roles of each feature understandable with respect to the timestamps and magnetic field parameters.

$$
\begin{aligned}
\mathbf{ET} &= \mathbf{E} * \mathbf{T}^T \\
\mathbf{EP} &= \mathbf{E} * \mathbf{P}^T \\
\mathbf{Z} &= [\mathbf{ET}, \mathbf{EP}]
\end{aligned}
$$

Therefore, by CP decomposition, we are able to represent the multivariate time series-based solar events with a compressed dimensionality of $T + N$, which is less than the original dimensionality of $T \times N$. Finally, we can project these compressed representations directly on 2D scatter plot though PCA or t-SNE. Only weakness of this dimensionality reduction method is: the tuning of hyperparameters such as number of iterations in CP algorithm and CP decomposition rank R. We recommend the trials of R with the range of $2, 3, ..., 20$ for the sake of low rank decomposition, and human observation of each trial result on the scatter plot.

6 EXPERIMENTAL EVALUATION

In this chapter, we present our experimental findings for the interpretable feature space extraction by tensor decomposition. At first, we present the node embedding performances of the tensor decomposition-based methods, which are based on interpretability of the feature space, network reconstruction, link prediction, node classification, and graph classification. Then, we present the classification performances found from different tensor modeling of the fMRI-based brain network data. We demonstrate the discriminative subnetwork mining, and the visualization of the learned biomarkers from a set of healthy and struggling population of reading disability-based resting-state fMRI data. Finally, we present summarized time series-based solar flare prediction performance analysis, and 2D visualization of flaring and nonflaring solar events in scatter plots. Each section contains the description of datasets and/or baselines.

6.1 *Graph Embedding by Tensor Decomposition*

In this section, we experimentally evaluate our algorithms TDNE and TDNEpS with respect to the interpretability of the feature space, and performance of the network reconstruction, link prediction, node classification, and graph classification. Among these tasks, graph classification is a multi-graph-based learning task, while others are single graph-based. We have also evaluated the performance and runtime of different node embedding algorithms while varying the number of embedding dimensions.

6.1.1 Experimental Settings

To comprehensively experiment our algorithms we have used different types of networks, such as directed and undirected, weighted and unweighted, sparse and dense, small and large networks. In Table 6.1, we list the datasets used in the experiments and their properties.

For network reconstruction, link prediction, node classification, and graph classification, we have compared our algorithms TDNE and TDNEpS with six baseline algorithms. We selected the baselines based on the categories of node embedding algorithms (section 2.2). For all of these baselines, we have used dataset and task-dependent hyperparameters suggested by their papers, and the survey paper [76].

- Laplacian Eigenmaps (LAP) [9] is a matrix decomposition-based method that performs eigendecomposition of the Laplacian of the graph.

- LLE [29] is a matrix decomposition-based method that embeds nodes by singular value decomposition, taking into consideration that each node is a linear combination of its neighbors.

- HOPE [30] is a multihop similarity-based method that performs generalized SVD on the similarity matrix found from node neighborhoods. We set the decay parameter $\beta = 0.01$ for Katz Index.

- GraRep [26] is a multihop similarity-based method that generates node embedding by successive singular value decomposition of the powers of the log-transformed, probabilistic adjacency matrix. We set maximum transition step $K = 6$, and log shifted factor $\beta = 1/n$.

- Node2Vec [8] is a random walk-based method, which is a generalization of DeepWalk [28]. Node2Vec uses biased random walk to create node sequences, and uses

Table 6.1: Dataset statistics and their use in experiments

Dataset	**Network type**	**Network properties**	$\lvert V\rvert$	$\lvert E\rvert$	**Density****	**Experiment**
Karate	social network	directed, unweighted	34	78	0.06952	interpretability of the features
BlogCatalog	social network	undirected, unweighted	10,312	333,983	0.00628	network reconstruction and link prediction
Brazilian airports	Air-traffic network	undirected, unweighted	131	1,003	0.11779	node classification
European airports	Air-traffic network	undirected, unweighted	399	5,993	0.07548	node classification
ADHD graph database*	Brain network	undirected, weighted	90	1992.897	0.4976	graph classification
Schizophrenia graph database*	Brain network	undirected, unweighted	132	5539.068	0.64065	graph classification

* For graph database, $|V|$ is the number of nodes in each graph, and $|E|$ is the mean number of edges of all graphs

** For undirected graph, density=$\frac{2 * |E|}{|V|(|V|-1)}$. For directed graph, density=$\frac{|E|}{|V|(|V|-1)}$

Skip-gram model [7] to learn node embeddings. We set walks per node $r = 80$, walk length $l = 10$, context size $k = 10$, return parameter $p = 1$, and in-out parameter $q = 1$.

- SDNE [31] is a deep learning-based method, which adopts a deep auto-encoder to preserve the first two order proximities. We use the default neural network structure and parameters in the implementation of the authors.

For node classification and graph classification, we use classification accuracy, i.e., percentage of correct predictions. For network reconstruction and link prediction, we use $\mathrm{Precision@N_p}$ and mean average precision (MAP) [27], [76].

$\mathrm{Precision@N_p}$: $\mathrm{Pr@N_p}$ is the fraction of correct predictions in top-N_p predicted node pairs. It is defined as,

$$\mathrm{Pr@N_p} = \frac{|E_{pred}(1 : N_p) \cap E_{obs}|}{N_p} \tag{6.1}$$

where $E_{pred}(1 : N_p)$ are top-N_p predicted node pairs, and E_{obs} are the observed edges. For network reconstruction, $E_{obs} = E$. For link prediction, E_{obs} is the set of hidden edges.

Mean Average Precision: MAP considers the precision of each node and computes the average over all nodes.

$$MAP = \frac{\sum_i AP(i)}{n} \tag{6.2}$$

$$AP(i) = \frac{\sum_{N_p} \mathrm{Pr@N_p}(i)\mathbb{I}\{E_{pred_i}(N_p) \in E_i\}}{|\{N_p : E_{pred_i}(N_p) \in E_i\}|}$$

$$Pr@N_p(i) = \frac{|E_{pred_i}(1:N_p) \cap E_i|}{N_p}$$

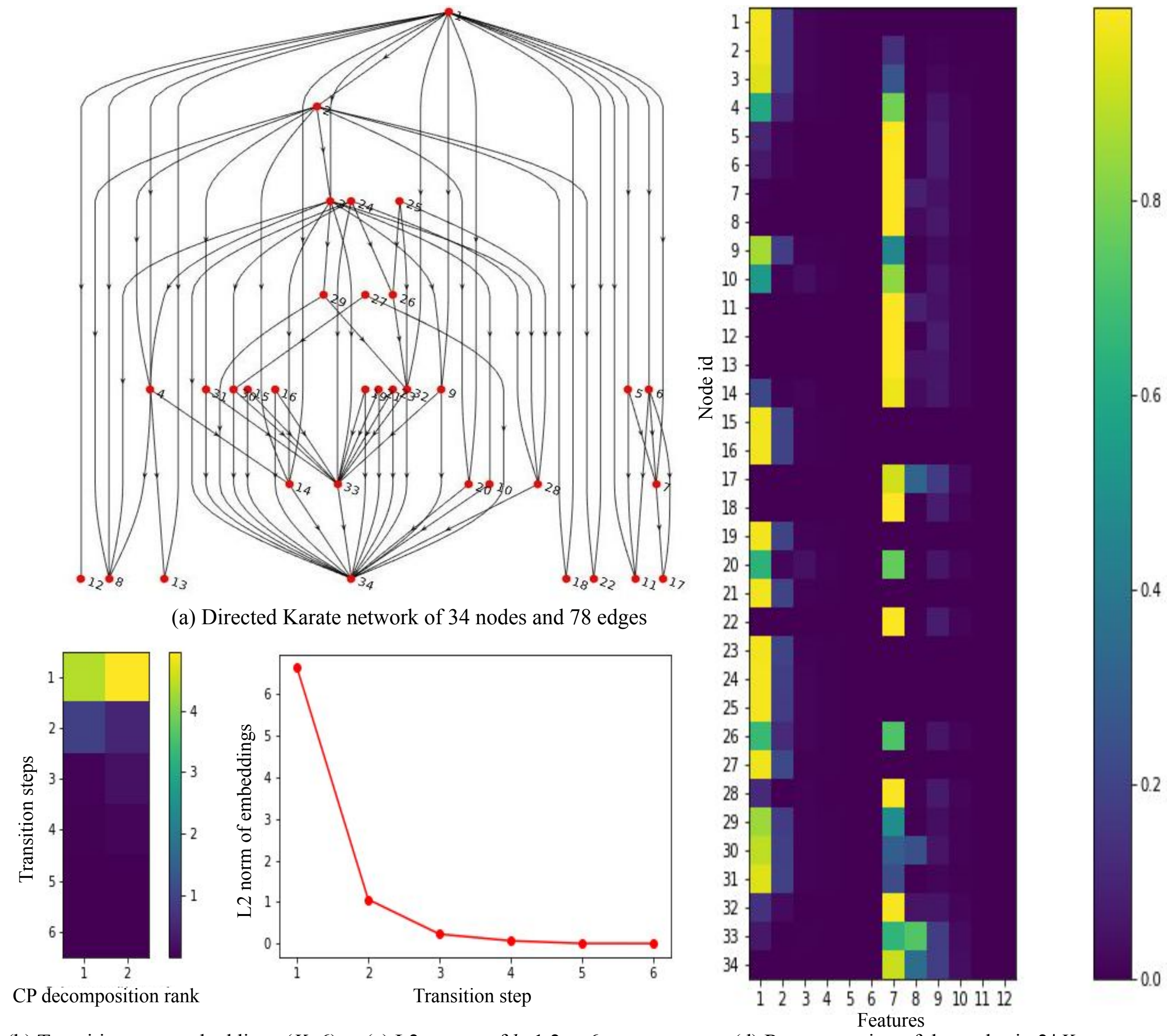
where E_{pred_i} is the set of predicted edges for node i. For network reconstruction, E_i is the set of observed edges for node i. For link prediction, E_i is the set of hidden edges for node i.

For implementing baseline algorithms and performance evaluation metrics, we used GEM (Graph Embedding Methods) library [1]. All the experiments are conducted in a single PC with Intel Core i7-6700 CPU (clock speed 3.40GHz), 16 GB RAM, and Ubuntu 16.04 operating system. Our implementation uses Tensorly library of Python [77]. We made TDNE and TDNEpS available at: https://github.com/hamdi08/TDN.

6.1.2 Interpretability of the Features

For this experiment, we used Zachary's Karate club network [78] (Fig. 6.1a), where we consider each edge as directed. The network has 34 nodes and 78 directed edges. We performed TDNE with $K = 6$ and $r = 2$. Therefore, the third-order tensor has size 34*34*6. After CP decomposition, we visualize the embeddings of each transition step (Fig. 6.1b). The L2 norms of the transition embeddings (Fig. 6.1c) show the relatively high importance of lower-order proximities compared to the higher-order proximities, which is intuitive for the social networks. In Fig. 6.1d, the final embeddings of each node is shown in a 12 (=2*6) dimensional feature space, where the first six features represent the source property of the nodes with varying transition step from one to six, and the last six features represent the target property of the nodes with varying transition step from one to six. Node 1, which has all outgoing edges and no incoming edges, is embedded in a way so

1 https://github.com/palash1992/GEM



(a) Directed Karate network of 34 nodes and 78 edges

(b) Transition step embeddings (K=6) after CP decomposition with rank 2

(c) L2-norms of k=1,2,.., 6 step transition step embeddings

(d) Representation of the nodes in 2*K dimensional space

Figure 6.1: Executing TDNE on directed Karate network

that it has high values in only source property representing features (more specifically, the features which represent source property in lower transition steps). Almost opposite embedding nature is observed in node 34, which has all incoming edges and no outgoing edges. For some nodes which have almost equal number of incoming and outgoing edges, such as node 9 and 10, we see a distribution of high values among source property representing features and target property representing features. Features representing higher-order transition steps (such as 4^{th}, 5^{th} and 6^{th}-order) of both source and target properties have no impact in this network, which supports the facts found in Fig. 6.1(c). Therefore, in order to further reduce the dimensionality, we can remove these features.

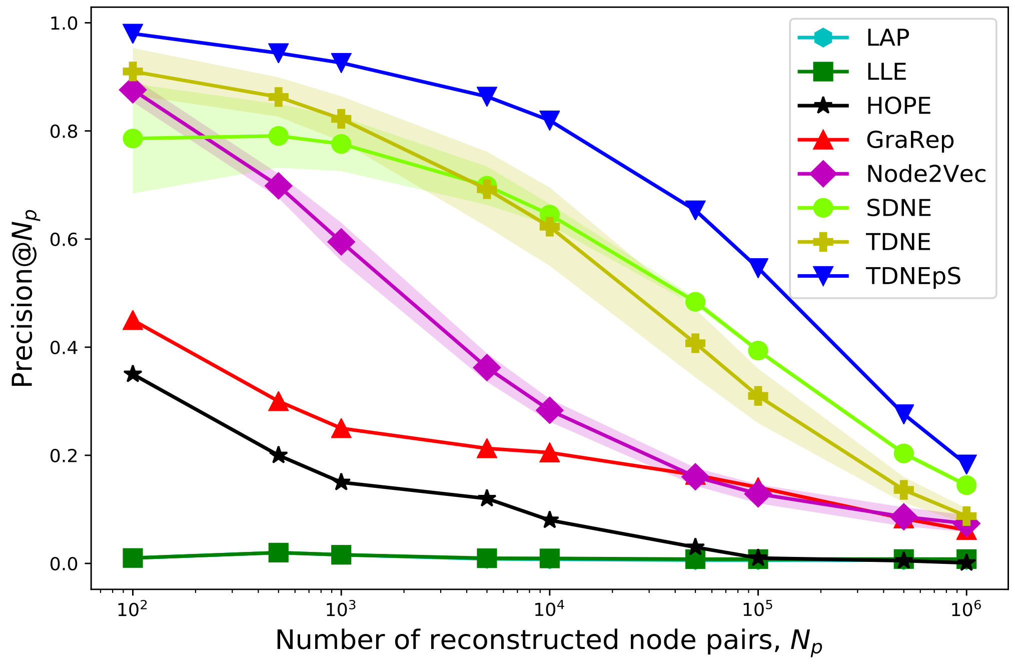
6.1.3 Network Reconstruction

Reconstruction of the network from the learned embeddings of the nodes is a common task for evaluating node embedding algorithms. The node pairs (possible edges) are ranked according to the node similarities, i.e., the inner product of two node embeddings, and equations 6.1 and 6.2 are used to determine $\mathrm{Precision}@N_p$ and MAP.

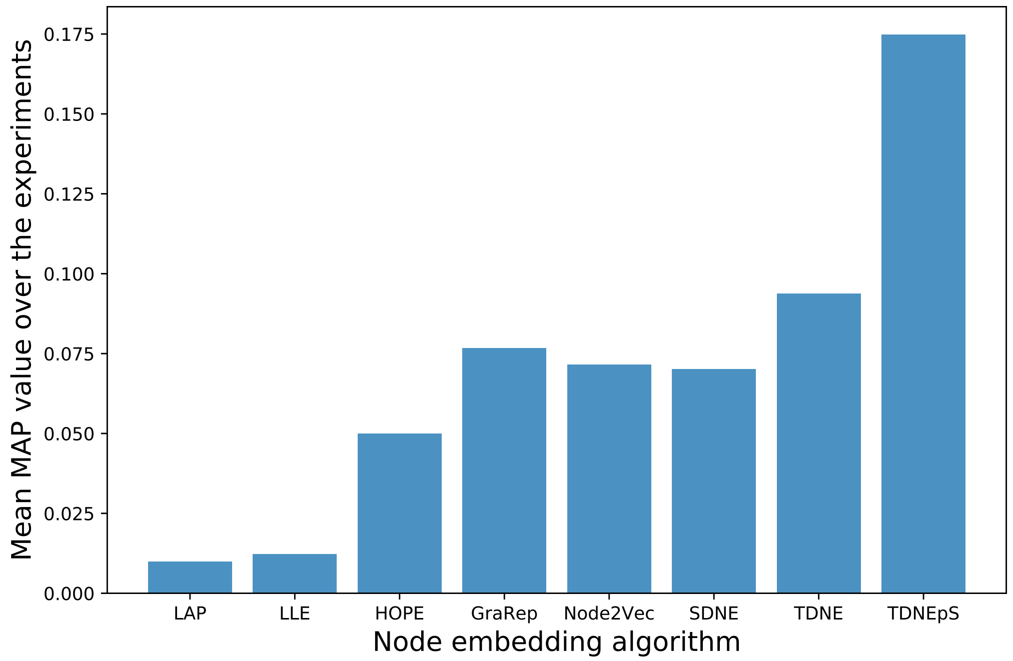
For this experiment, we have used BlogCatalog network [2] which consists of 10,312 nodes and 333,983 edges (undirected and unweighted). In this social network, nodes represent bloggers and edges represent social relationships among them.

Fig. 6.2 shows that TDNE ($K = 3, r = 4$) and TDNEpS ($K = 1, r = 1$) outperforms other baseline algorithms in terms of $\mathrm{Precision}@N_p$ (Fig. 6.2a) and MAP (Fig. 6.2b). We have used the number of dimensions $d = 128$ for the baselines. We varied the number of reconstructed node pairs from one hundred to one million, and recorded $\mathrm{Precision}@N_p$ for each given number of reconstructed node pairs. We executed each algorithm five times, and plotted the means as points and the standard deviations as shaded regions. We observe that single matrix factorization-based methods such as Laplacian Eigenmaps and

2 http://socialcomputing.asu.edu/datasets/BlogCatalog3



(a) Precision@$\mathrm{N_p}$ in network reconstruction



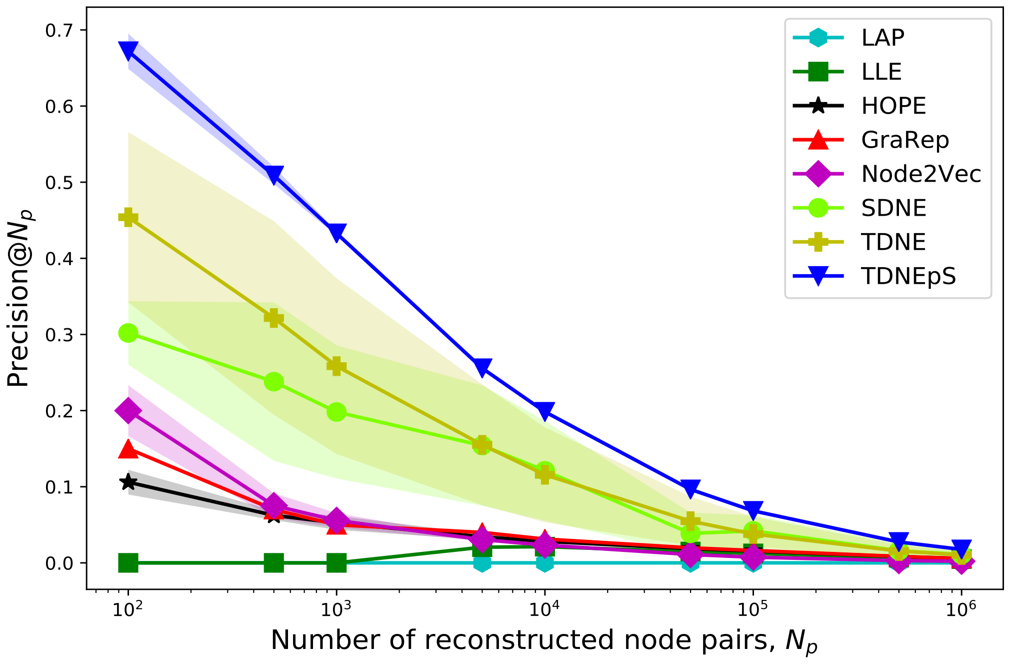
(b) Mean MAP of each baseline in network reconstruction

Figure 6.2: Network reconstruction performance of TDNE and TDNEpS with other baseline algorithms on BlogCatalog network

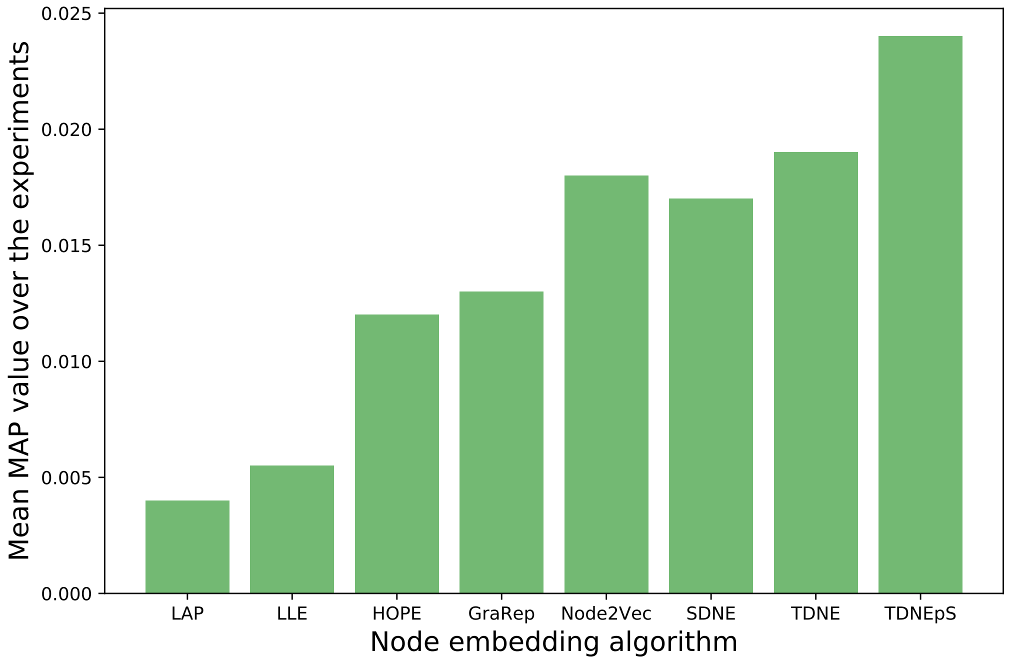
LLE perform very poorly because of their relying on the approximation of eigenvectors of large, first-order proximity matrix. Multihop similarity-based methods such as HOPE and GraRep that use higher-order proximities of nodes perform comparatively better. Random walk-based method Node2Vec performs better than them because of its flexible neighborhood sampling. Although Node2Vec's performance is better than GraRep in terms of $\mathsf{Precision@N_p}$, in terms of MAP, Node2Vec's performance is a bit inferior to GraRep's. SDNE's performance in $\mathsf{Precision@N_p}$ is better than all methods (including TDNE) except TDNEpS, but in MAP performance SDNE is inferior to both tensor decomposition-based methods. Surprisingly, TDNE with $d = 6$ and TDNEpS with $d = 2$ perform better than all the baselines, which use $d = 128$. The robust performance of these tensor decomposition-based node embedding methods can be attributed to the representation learning of the transition steps (proximities). While TDNE shows some variance over the experiments, TDNEpS performs consistently and outperform TDNE in network reconstruction. This proves the superiority of independent decomposition of the tensor slices in comparison with single decomposition of a single tensor.

6.1.4 Link Prediction

Link prediction is a typical application of node embedding that aims to predict which pairs of nodes are likely to form edges. In our experiments, we randomly hid 20% of edges (66,797 edges) of the BlogCatalog network, and executed node embedding algorithms on the remaining edges (267,186 edges) to learn node representations. We used the same number of dimensions in each node embedding algorithm as the experiment of network reconstruction. For both $\mathsf{Precision@N_p}$ and MAP, we followed exactly the same experimental settings of network reconstruction (variation of the number of reconstructed node pairs and number of executions of each algorithm).



(a) Precision@N_p in link prediction



(b) Mean MAP of each baseline in link prediction

Figure 6.3: Link prediction performance of TDNE and TDNEpS with other baseline algorithms on BlogCatalog network

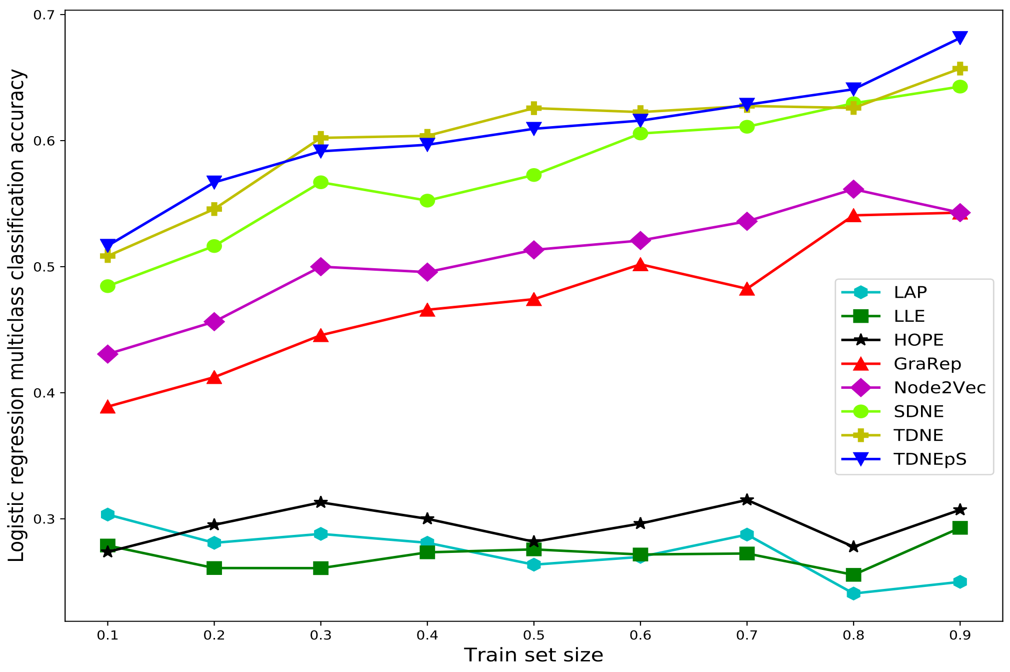
Fig. 6.3 shows that both TDNE and TDNEpS outperform other baselines in terms of both $\mathsf{Precision@N_p}$ (Fig. 6.3a) and MAP (Fig. 6.3b). Since link prediction is a harder task than network reconstruction, all algorithms give smaller values in both metrics. According to $\mathsf{Precision@N_p}$ and MAP values of link prediction, we observe a performance precedence order, which implies $\mathsf{TDNEpS} > \mathsf{TDNE} > \mathsf{SDNE} > \mathsf{Node2vec} > \mathsf{GraRep} > \mathsf{HOPE} > \mathsf{LLE} > \mathsf{LAP}$. These results support the fact that algorithms considering higher-order proximities tend to have better performance than the algorithms considering lower-order proximities, and proximity learning nature of tensor decomposition-based methods perform better in link prediction even with the small number of dimensions. Like the case of network reconstruction, TDNE shows more variance and less precision than TDNEpS.

6.1.5 Node Classification

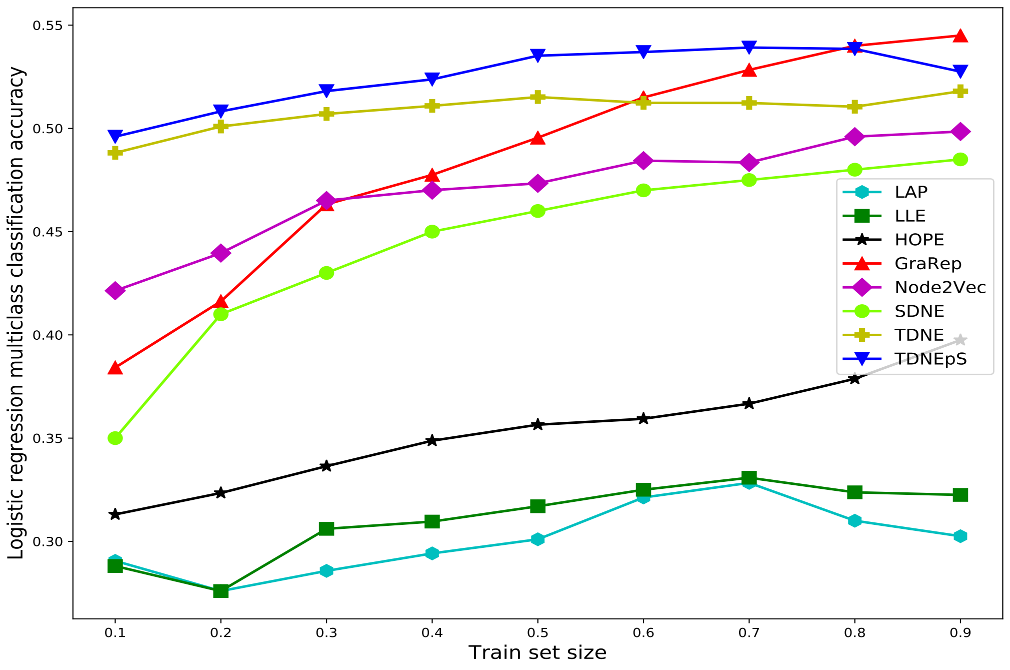
For node classification, we used the Brazilian airport network, which has 131 nodes and 1,003 edges, and Europian airport network, which has 399 nodes and 5,993 edges. Both networks are undirected and unweighted. The nodes correspond to the airports in Brazil and Europe, and edges indicate the existence of commercial flights between them. The nodes have four labels from 0 to 3, which indicate the airport activities, i.e., the number of takeoffs and landings in the year 2016, and label 0 means the highest activity level. The data is collected and labeled by Ribeiro et al. [79] from the website of the National Civil Aviation Agency (ANAC) [3], and made available [4].

We worked with two settings of number of dimensions in two airport networks. We set $d = 16$ for Brazil airport network, and $d = 32$ for Europian airport network. We executed both tensor decomposition-based methods with $R = 4$ for both networks, since there are

3 http://www.anac.gov.br/
4 https://github.com/leoribeiro/struc2vec



(a) Node classification performance in Brazilian airport network



(b) Node classification performance in Europian airport network

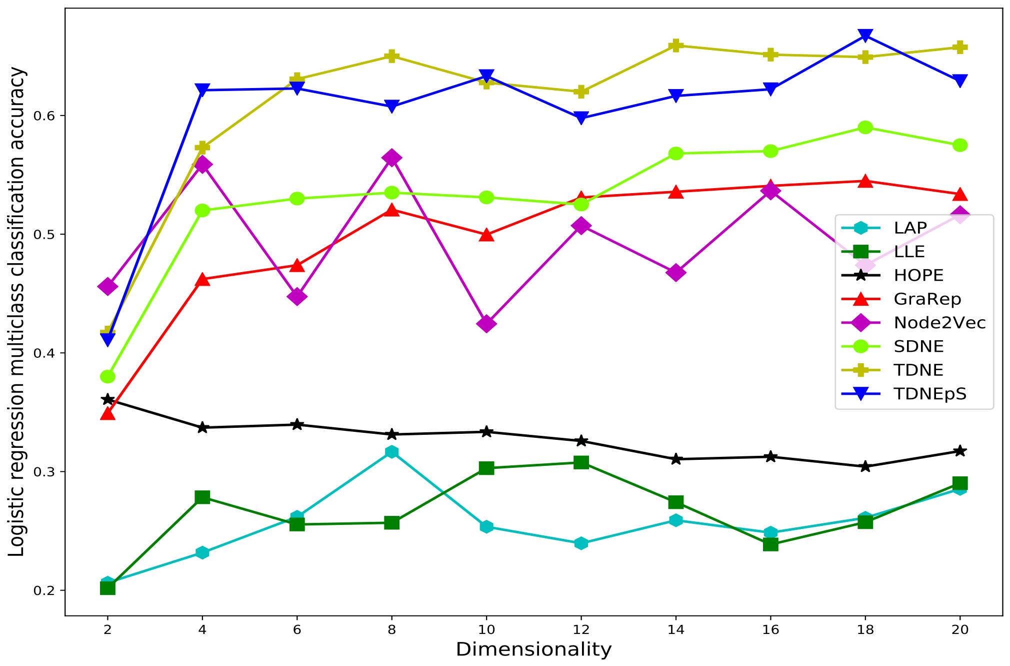
Figure 6.4: Node classification in airport networks varying training set size

four classes. As the classifier, we used one-vs-all logistic regression classifier with L2 regularization. We varied the percentage of training representations from 10% to 90% and reported mean classification accuracy over 10 trials of random sampling (Fig. 6.4). From Fig. 6.4a and 6.4b, we can see that both TDNE and TDNEpS outperform other baselines in node classification in both networks.

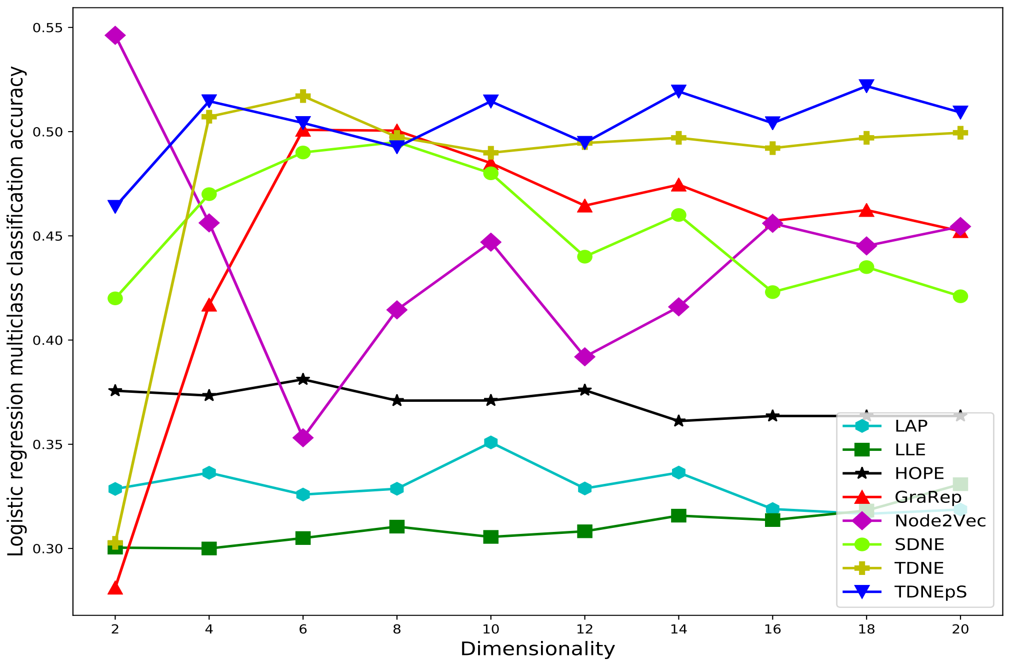
Experiments on both airport networks with respect to node classification lead to an interesting observation. The distinction of performances of three classes of algorihtms is visible. Matrix decomposition-based methods LAP, LLE, and HOPE perform poorest in both networks. Multihop similarity-based (GraRep), random walk-based (Node2Vec), and deep learning-based (SDNE) methods perform better than matrix decomposition-based methods in both datasets. Both tensor decomposition-based methods TDNE and TDNEpS perform best in both networks. SDNE has good performance in Brazilian airport network, while it performs poorer in European airport network in comparison with Node2Vec, GraRep, and tensor decomposition-based methods. Although GraRep does not perform well in Brazilian airport network with respect to the performances of SDNE, Node2Vec, and tensor decomposition-based methods, in European airport network it performs very good by gradually increasing the accuracy with respect to the increase of training set size. In both networks, with only 30% training data, the classification accuracy of TDNEpS is almost twice than that of HOPE, the best performing single matrix decomposition-based method.

6.1.6 Performance and Runtime Varying Dimensionality

We evaluated the effect of the input number of dimensions in node embedding algorithms for the task of node classification in both airport networks. We varied the number of dimensions and measured the multiclass classification accuracy of the L2 regularized logistic regression classifier after splitting the training and test node representations by



(a) Node classification performance in Brazilian airport network



(b) Node classification performance in Europian airport network

Figure 6.5: Node classification in airport networks varying number of dimensions

Table 6.2: Mean execution times of different node embedding algorithms while varying the number of dimensions

	Execution times (in seconds)	
Algorithm	Brazilian airport network	Europian airport network
LAP	0.04135	0.09955
LLE	0.03762	0.07717
HOPE	0.00862	0.02715
GraRep	0.03152	0.31382
Node2Vec	0.39355	1.08808
SDNE	16.66337	146.86828
TDNE	0.05425	0.35163
TDNEpS	0.04305	0.39544

10 fold stratified cross-validation. In both networks, TDNE and TDNEpS outperform other algorithms in almost every setting of the number of dimensions. Fig. 6.5 shows that with the increase of the number of dimensions, in the case of most algorithms, classification accuracy increases at first and then becomes stable (TDNEpS, TDNE, SDNE, and GraRep in Fig. 6.5a), or gradually decreases (SDNE and GraRep in Fig. 6.5b). The low dimensional setting, which is a deviation from the default 128-dimensional setting of Node2Vec [8], probably causes the noise in classification accuracy in both networks. Similar to Fig. 6.4, all matrix decomposition-based methods show poorer performance than other baselines.

Table 6.2 shows the execution times of different algorithms in embedding the nodes of both airport networks. We report the mean runtime while varying the number of dimensions. The slow training time of SDNE is visible while being compared with other algorithms. Matrix decomposition-based algorithms embed the nodes in the fastest time for both networks. GraRep and tensor decomposition-based algorithms have similar execution times, while Node2Vec performs slower than them in both cases.

6.1.7 Graph Classification

For graph classification, we have a graph database of labeled graphs, $D = \{G_1, G_2, \ldots, G_{|D|}\}$. We can represent each graph by a fixed dimensional vector space by different graph embedding schemes, such as computing the structural properties such as degree, clustering coefficient, etc of the nodes [36], counting the appearances of frequent/discriminative subgraph patterns [40], and so on. While structure-based features are hand-engineered and require a lot of domain knowledge, subgraph-based features are computationally very expensive [21]. To evaluate the node embedding algorithms in the light of graph classification, we take a graph database, where each graph has the same labeled node set, and apply node embedding algorithms to embed the nodes of each graph. If each graph has n nodes, then graph G_i has a node embedding matrix $\mathbf{Z}_i \in \mathbb{R}^{n \times d}$. We get the embedding of the graph G_i by reshaping its node embedding matrix by $\mathbf{Z}_i \in \mathbb{R}^{1 \times nd}$. Therefore, the embedding matrix of the graph database is $\mathbf{Z}_D \in \mathbb{R}^{|D| \times nd}$.

Brain network classification is a good example of graph classification, where each graph has the same labeled node set. In brain networks, the nodes represent brain regions defined by some standard brain atlas, and edges represent functional/structural similarity of the brain regions [38]. Non-invasive neuroimaging modalities such as Magnetic Resonance Imaging (MRI), Electroencephalography (EEG), and Diffusion Tensor Imaging (DTI) can be used to construct brain networks, that have been used by the neuroscience community to investigate different neurological disorders such as Alzheimer's, Schizophrenia, Bipolar disorder, and Attention-deficit/hyperactivity disorder (ADHD) [13]. Given a set of brain networks and associated case/control labels, we aim to maximize the classification performance. For this experiment, we have considered two resting-state fMRI (Functional Magnetic Resonance Imaging)-based brain network datasets on ADHD and Schizophrenia.

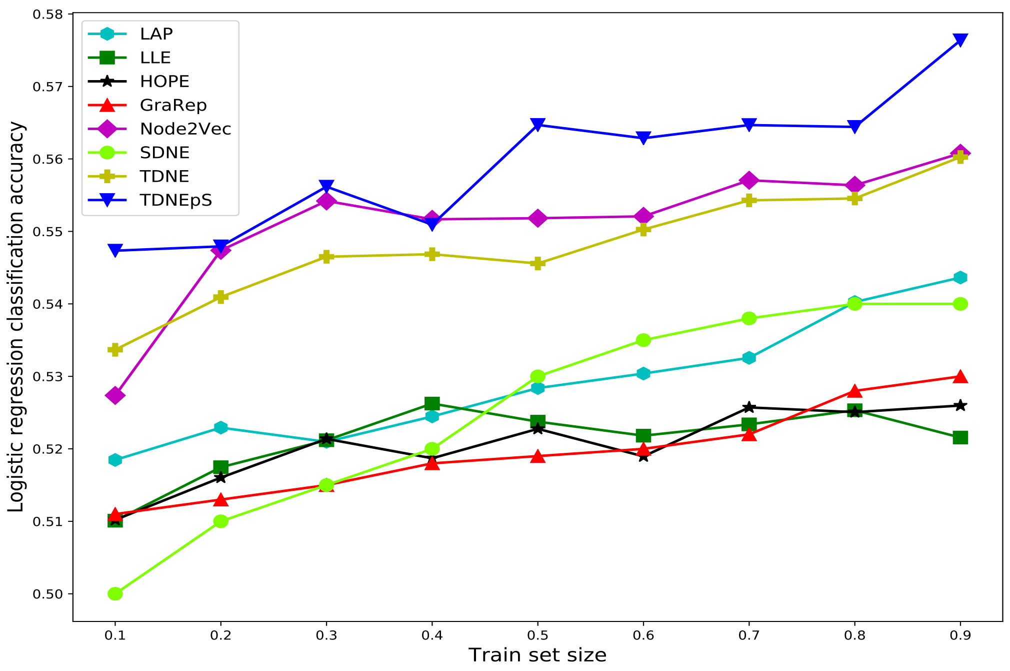
fMRI measures the functional activities of different brain regions by capturing 3D brain volumes over time. The time series of each voxel (the unit of brain volume) is calculated, which represents the change in Blood Oxygenation Level Dependent (BOLD) signal over the scan period. The voxels are grouped together to predefined brain atlas-based regions, and the mean time series is calculated for each region. The pairwise Pearson correlation coefficients between the time series of the regions give the correlation matrix. By applying threshold (usually 0) on the correlation matrix, we get the adjacency matrix of the brain network [40]. We performed the experiments of brain network classification on two datasets: ADHD and Schizophrenia.

As the first dataset, we used a brain network dataset from ADHD-200 global competition [5]. The dataset has 768 brain networks, where 280 of them are labeled as ADHD positive, and the rest are normal control. There are 90 nodes in each network, which represents 90 cerebral brain regions defined by the Automated Anatomical Labeling (AAL) parcellation on resting-state fMRI scans of the subjects. The detailed preprocessing steps of this dataset are discussed in [80]. We made the graphs sparse by removing the edges which have negative weights in the correlation matrix.

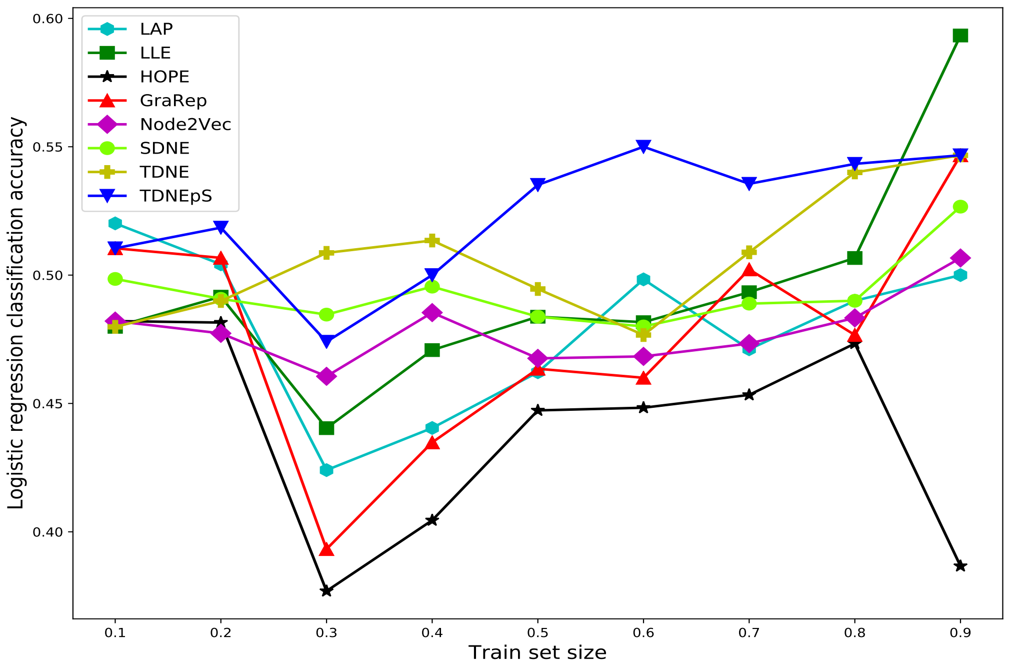
We have used $d = 16$ for all node embedding algorithms. We have used binary logistic regression classifier with L2 regularization. We report the mean classification accuracy after 10 trials of train/test sampling, while varied the train set size from 10% to 90%. From Fig. 6.6a, we see that TDNEpS outperforms all other baselines. Node2Vec performs better than TDNE in this experiment. LAP performs better than other single matrix factorization-based methods such as LLE and HOPE. The performances of SDNE and GraRep are almost same as that of matrix decomposition-based methods.

As the second dataset, we used a brain network dataset collected by The Center for Biomedical Research Excellence (COBRE) [6]. The dataset has 147 functional MRI scans, where 72 of them are diagnosed with Schizophrenia, and the rest are normal control. We

5 http://fcon_1000.projects.nitrc.org/indi/adhd200/
6 http://fcon_1000.projects.nitrc.org/indi/retro/cobre.html



(a) Graph classification performance in ADHD graph database



(b) Graph classification performance in Schizophrenia graph database

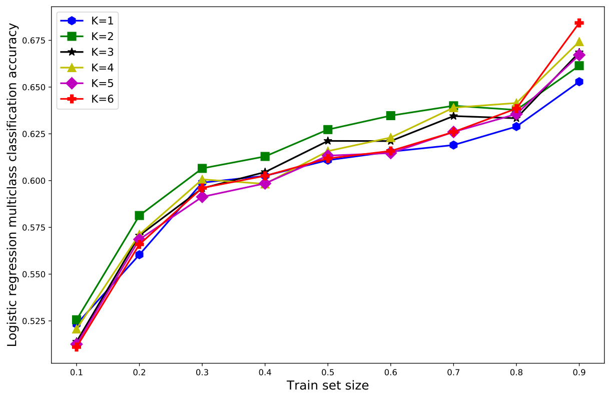
Figure 6.6: Graph classification in two brain networks datasets varying training set size

used the software Conn [81] for preprocessing the fMRI scans by head motion correction, slice timing correction, normalization to MNI template, and spatially smoothing with 8mm Gaussian kernel [40]. After default segmentation, we found the time series of 132 brain regions, which combine FSL Harvard-Oxford atlas-based cortical and subcortical regions, and AAL atlas-based cerebellar regions. Therefore, each network has 132 nodes. Similar to ADHD dataset, we added sparsity in the graphs by removing the edges with negative weights. Unlike the experiment with ADHD dataset, we considered unweighted (binary) graphs for Schizophrenia dataset. We kept other experimental settings such as number of dimensions of the node embedding algorithms, the classifier, and accuracy reporting strategy same as that of the experiment with ADHD dataset.

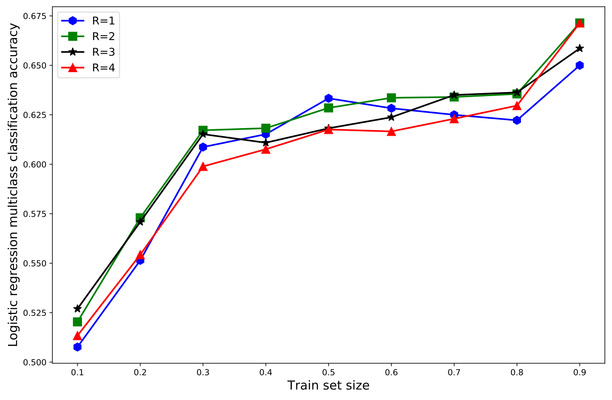
In Fig. 6.6b, we see the performances of different node embedding algorithms in Schizophrenia dataset. While the performance of almost all algorithms drop in a particular sampling (30%) of training instances, TDNEpS perform better than others in most cases. LLE, a matrix decomposition-based algorithm, performs better than all algorithms with 90% training data.

6.1.8 Parameter Sensitivity

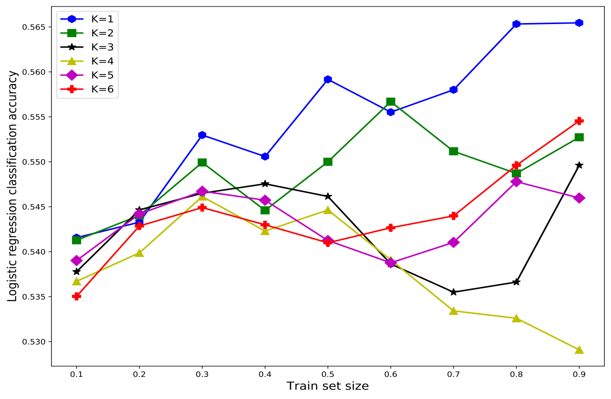
Fig. 6.7 shows the node classification and graph classification performance of TDNEpS with the change of hyperparameters K and R. The tensor decomposition rank R is set as the number of class labels in the dataset, and maximum transition step K is tuned with keeping R fixed. Therefore, for the Brazilian airport network, which has four class labels, we set $R = 4$ to see the effect of K in the node classification performance with respect to train set size (Fig. 6.7a). The setting of $K = 2$ performs well when the train set size is small. Then with fixed $K = 2$, we vary R (Fig. 6.7b), and see $R = 2$ performs consistently better with $K = 2$. We set $R = 2$ for binary labeled ADHD dataset in graph classification, and vary K with R fixed. We see from Fig. 6.7c that graph classification



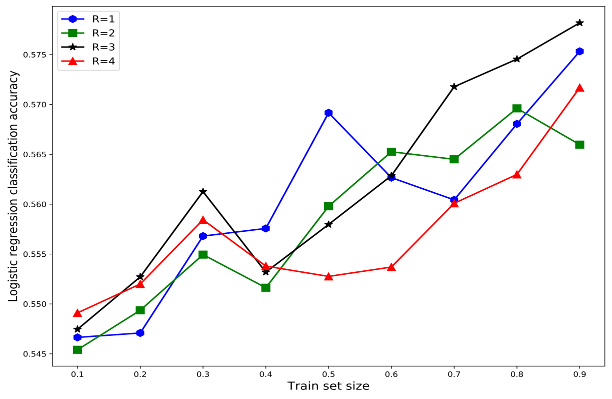
(a) Brazil airport network with fixed R(=4) and varying K



(b) Brazil airport network with fixed K(=2) and varying R



(c) ADHD dataset with fixed R(=2) and varying K



(d) ADHD dataset with fixed K(=1) and varying R

Figure 6.7: Node classification and graph classification performance of TDNEpS in different settings of K and R

performance in ADHD dataset is very sensitive with K. For some K values, the classifier might have overfit (for example, $K = 4$). We take $K = 1$ as best maximum transition step and vary R (Fig. 6.7d). Although $R = 1$ and $R = 3$ give some good results, it is difficult to find optimal R. Therefore, we recommend to use grid search for optimizing these two hyperparameters K and R.

6.2 *Tensor Modeling of the Brain Networks*

In this section, we present the performance of brain network representations after modeling the data in different tensor models (Chapter 3). We used *Tensor Toolbox* [82] of MATLAB for computing CP and Tucker decomposition with ALS optimization. In this section, we show the performance of CP and Tucker decomposition on five tensor models.

6.2.1 Data Collection

In our first dataset, the target neurodevelopmental disorder was a reading disability. For the study, we used preprocessed resting-state fMRI scans of 14 adult subjects from the local community, seven labeled as “struggling" readers (below-average reading test scores), and seven labeled “typical" (average on reading test). Because of the specific experimental setting and preprocessing requirements of this study, the number of collected samples is small. We summarize the preprocessing steps as follows.

1. Removal of time-locked physiological noise
2. Slice timing correction
3. Bulk head motion correction
4. Reorientation

5. Echo-planar Imaging (EPI)-based distortion correction
6. Linear and nonlinear coregistration to template space
7. Resampling to EPI voxel size
8. Removal of physiological noise by detrending time-shifted respiratory volume per time
9. Removal of local white matter BOLD time series
10. Skull-stripping
11. Low-pass filtering in the range of 0.001-0.1 Hz
12. Smoothing with Gaussian kernel (FWHM 6mm)

The details of the preprocessing steps for this study can be found in [83]. We are also provided 16 ROIs based on apriori reading research [39]. In Fig. 6.8, we show the visualization of these ROIs in left and right hemisphere. The preprocessing steps are done using AFNI [84], FSL [85], and FreeSurfer [86]. The subjects are scanned in one session. After preprocessing, the number of voxels in one brain volume is $53 \times 63 \times 45$, while the repetition time (time between capturing two whole brain volume) is 2 seconds. Finally, using Conn [81] the multivariate time series and functional connectivity matrices of each subject are extracted. The length of each time series of each ROI is 125.

Since the number of samples in the reading disability dataset is small, we collected another dataset from ADHD-200 global competition [7]. We used the data from NeuroIMAGE study (Radboud University Nijmegen Medical Centre, Vrije Universiteit Amsterdam, UMC Groningen), which contains data of 48 subjects (23 are normal control and rest are ADHD positive). The data of each subject is a multivariate time series, which is computed by the Automated Anatomical Labeling (AAL) parcellation on resting-state

7 `http://fcon_1000.projects.nitrc.org/indi/adhd200/`.

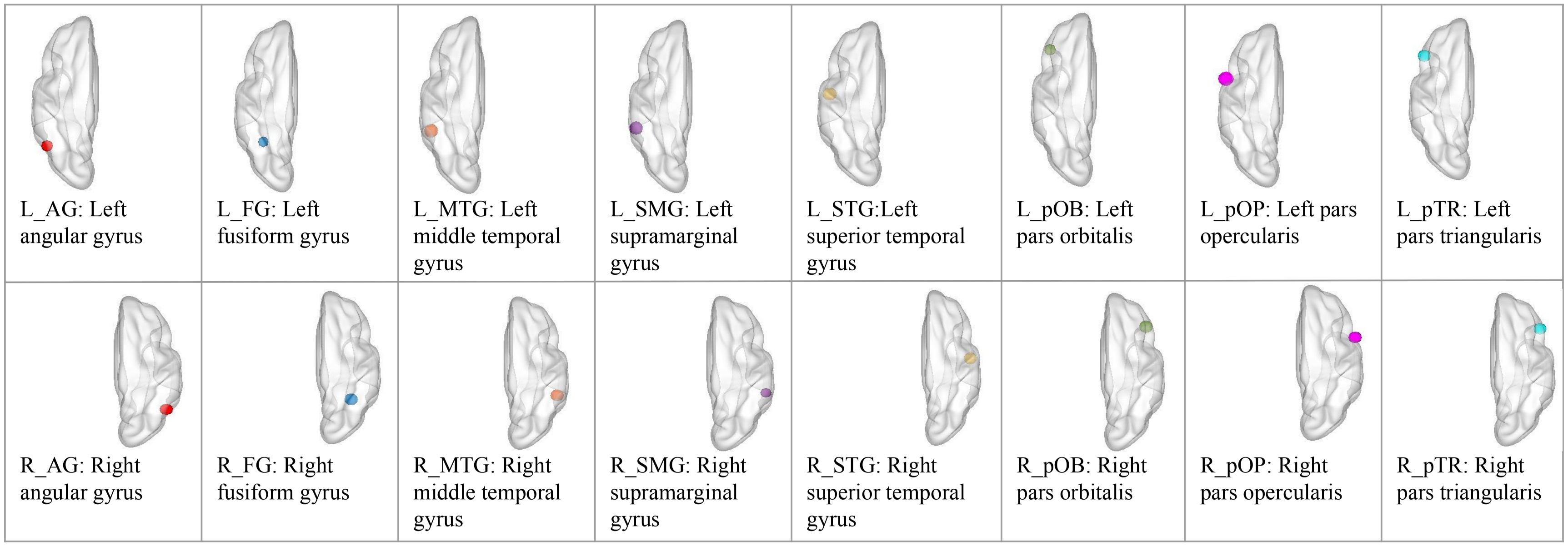

L_AG: Left angular gyrus
L_FG: Left fusiform gyrus
L_MTG: Left middle temporal gyrus
L_SMG: Left supramarginal gyrus
L_STG:Left superior temporal gyrus
L_pOB: Left pars orbitalis
L_pOP: Left pars opercularis
L_pTR: Left pars triangularis
R_AG: Right angular gyrus
R_FG: Right fusiform gyrus
R_MTG: Right middle temporal gyrus
R_SMG: Right supramarginal gyrus
R_STG: Right superior temporal gyrus
R_pOB: Right pars orbitalis
R_pOP: Right pars opercularis
R_pTR: Right pars triangularis

Figure 6.8: Visualization of 16 ROIs in left and right hemisphere

fMRI data. The length of each time series of each ROI is 257. AAL-parcellation segments the brain into 90 cerebral and 26 cerebellar ROIs. We consider all 90 cerebral regions similar to [40], [43], [52], and [44] as ROIs. Therefore, the multivariate time series of each subject has shape (257, 90).

6.2.2 Evaluation Method

After obtaining the subject factor matrix by tensor decomposition, label information is concatenated and labeled representations of the subjects are fed into a classifier with a predefined train/test splitting strategy. In our experiments, we chose Support Vector Machine (SVM) as the classifier with the radial basis function as the kernel. As the train/test splitting method, in the reading disability dataset, we chose 7-fold stratified cross validation, and in the ADHD dataset, we performed 8-fold cross validation. Therefore, in the reading disability dataset, at each iteration 12 labeled subjects are taken as training subjects and two of them are taken as test subjects, while half of the training and test subjects are labeled as struggling, and half of them are labeled as typical. In the ADHD dataset, at each iteration 40 samples are taken as training samples, and 8 of them are taken as test samples.

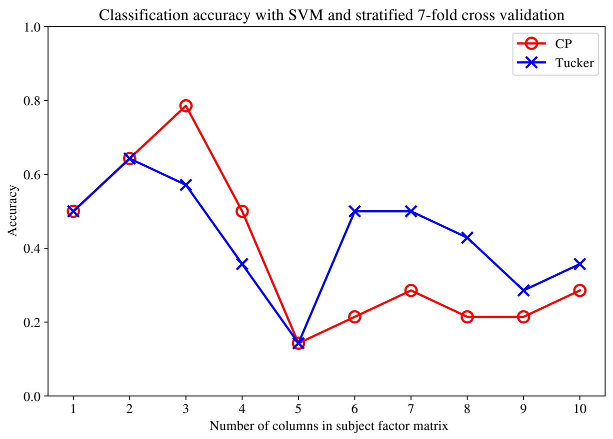
6.2.3 Performances in Different Tensor Models

We have experimented five tensor models with both CP and Tucker decompositions. For the reading disability dataset, we varied the number of columns in the subjects factor matrix from 1 to 10 (in Tucker decomposition for both datasets, we set the number of columns of other factor matrices as fixed values) and then evaluated the quality of the subject factor matrix by SVM with 7-fold stratified cross validation. For ADHD dataset, we varied the number of columns in the subjects factor matrix from 1 to 20 and used 8-fold cross validation for evaluating the classifier accuracy.

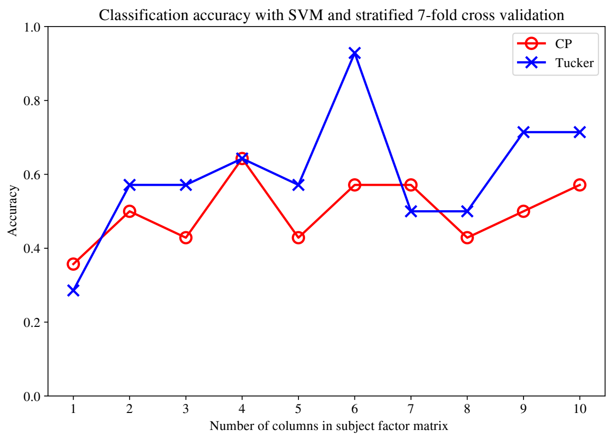
Fig. 6.9a – 6.9e show the SVM classification accuracy on the subject factor matrix found after CP and Tucker decomposition of five different tensors constructed from the reading disability dataset. Since ALS implementation of CP and Tucker is used, the resulting factor matrix depends on the initialization of other factor matrices representing ROIs and timestamps. Therefore, if we increase the number of columns of the subject factor matrix, which is equivalent to increasing the number of features in feature space, the classification accuracy may not increase linearly. In Fig. 6.9f, we summarized our findings by presenting the best accuracy rates along with the number of columns used in the factor matrix, and the mean accuracy found by varying the number of columns. In the reading disability dataset, Tensor Model 3, which was used previously in the literature [43], showed the most robust performance in both CP and Tucker decomposition with mean accuracy over 70%. The nonlinearity of accuracy with respect to the number of columns of the factor matrix is also supported by other studies [43], [52]. We also see that Tucker decomposition performs better than CP in all tensor models except Tensor Model 5. Moreover, when the number of columns in the subject factor matrix is small, e. g., 1, CP decomposition performs better than Tucker decomposition in most cases.

Experimental results on five tensor models constructed from the ADHD data are shown in Fig. 6.10a – 6.10f. In terms of best accuracy, similar to the reading disability dataset, Tucker decomposition shows better performance than CP decomposition in almost all tensor models. According to the mean accuracy, in ADHD dataset, Tensor Model 5 performs best.

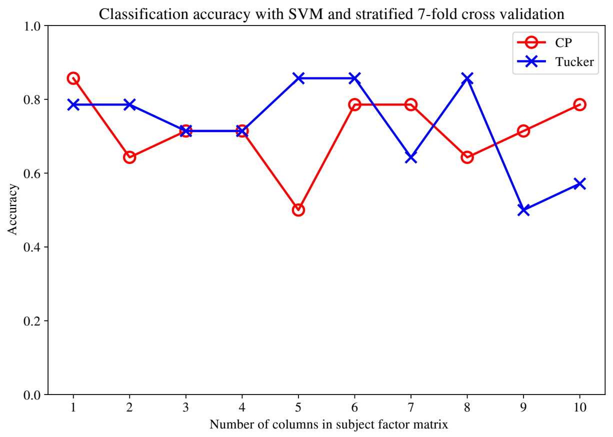
We evaluated five different tensor models, and showed the classification accuracy on the subject factor matrix generated by CP and Tucker decomposition. When we construct the tensors from a dataset of a small number of samples, such as reading disability dataset, Tensor Model 3, where the tensor is constructed by stacking the non-negative



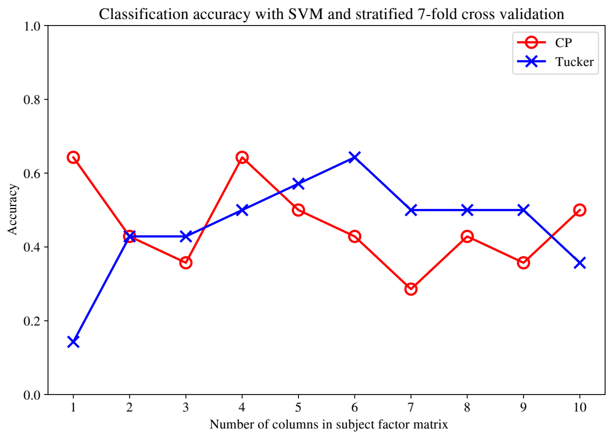
(a) Tensor model 1



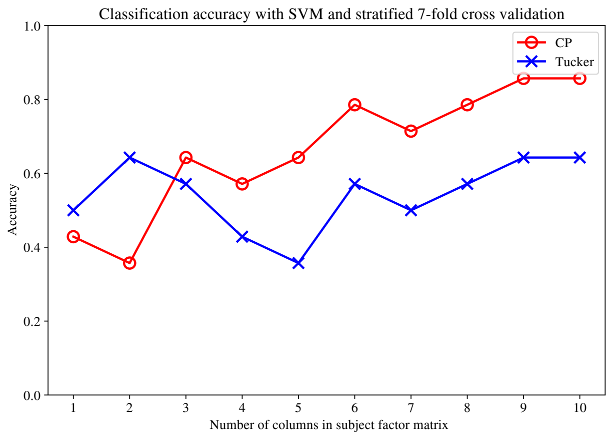
(b) Tensor model 2



(c) Tensor model 3



(d) Tensor model 4

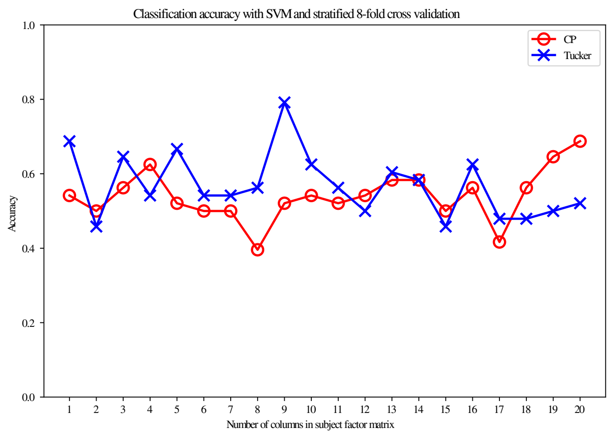


(e) Tensor model 5

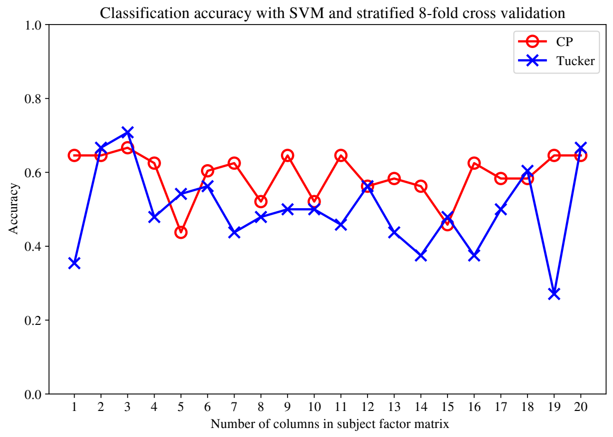
Tensor model	Tensor decomposition	Best accuracy gained	Number of columns of factor matrix when best accuracy was gained	Mean accuracy over all runs varying number of columns
1	CP	0.7857	3	0.3786
	Tucker	0.6429	2	0.4286
2	CP	0.6429	4	0.5
	Tucker	0.9286	6	0.6
3	CP	0.8571	1	0.7143
	Tucker	0.8571	5	0.7286
4	CP	0.6429	1	0.4571
	Tucker	0.6429	6	0.4571
5	CP	0.8571	9	0.6643
	Tucker	0.6429	2	0.5429

(f) Comparison of all tensor models

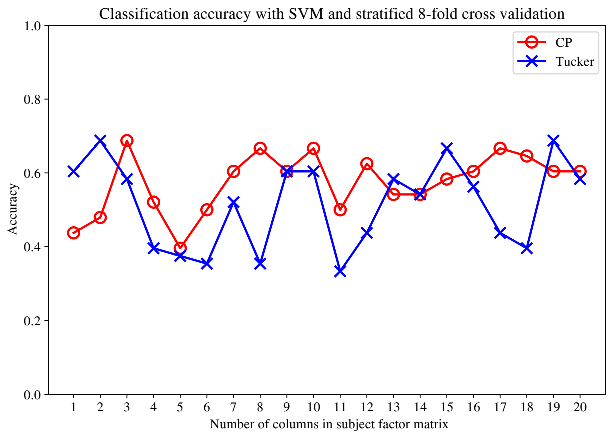
Figure 6.9: Reading disability dataset: Comparison of CP and Tucker decomposition on five tensor models on the basis of SVM classification on subject factor matrix



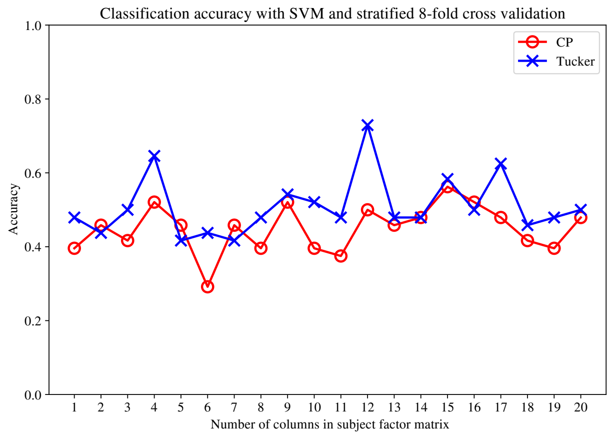
(a) Tensor model 1



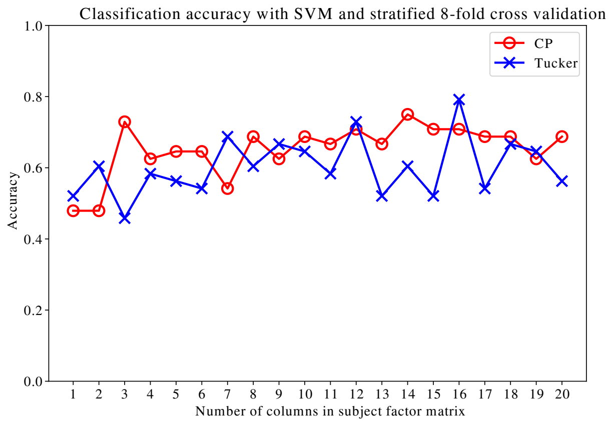
(b) Tensor model 2



(c) Tensor model 3



(d) Tensor model 4



(e) Tensor model 5

Tensor model	Tensor decomposition	Best accuracy gained	Number of columns of factor matrix when best accuracy was gained	Mean accuracy over all runs varying number of columns
1	CP	0.6875	20	0.5406
	Tucker	0.7917	9	0.5688
2	CP	0.6667	3	0.5917
	Tucker	0.7083	3	0.4979
3	CP	0.6875	3	0.574
	Tucker	0.6875	2	0.5156
4	CP	0.5625	15	0.449
	Tucker	0.7292	12	0.5094
5	CP	0.75	14	0.6521
	Tucker	0.7917	16	0.6021

(f) Comparison of all tensor models

Figure 6.10: ADHD dataset: Comparison of CP and Tucker decomposition on five tensor models on the basis of SVM classification on subject factor matrix

functional connectivity matrices, performs consistently with good classification accuracy. In a dataset of comparatively large number of samples, such as the ADHD dataset, we show that Tensor Model 5, which is constructed by node-wise Jaccard kernel on non-negative functional connectivity matrices, performs better than other tensor models.

6.3 *Biomarker Embedding from Brain Networks*

In this section, we demonstrate the experiemental findings of discriminative subnetwork mining and biomarker representation learning and visualization of the resting-state fMRI dataset of reading disability. We used *Feature Selection Toolbox* [54] for mRMR and Fisher-based feature selection, and *Tensor Toolbox* [82] for computing CP and Tucker decomposition with ALS optimization, and ran all experiments in MATLAB 2017a.

After extracting the ROI-level time series from the readling disability dataset (introduced in previous section), we calculate the Pearson correlation matrix of the ROI-based multivariate time series. Finally, we compute the functional connectivity matrix by Fisher's r-to-z transformation on the elements of the Pearson correlation matrix. Because of 16 ROIs, the resultant functional connectivity matrix has size 16×16. The functional connectivity matrix is the weighted adjacency matrix of the complete functional connectivity network. Since each complete functional connectivity network has 16 nodes and $16 * (16 - 1)/2 = 120$ weighted edges, the functional connectivity vector of each subject is 120 dimensional.

6.3.1 Generation and Visualization of Discriminative Subgraph

After representing the complete functional connectivity networks of the dataset as 120-dimensional functional connectivity vectors, we apply mRMR and Fisher-based feature

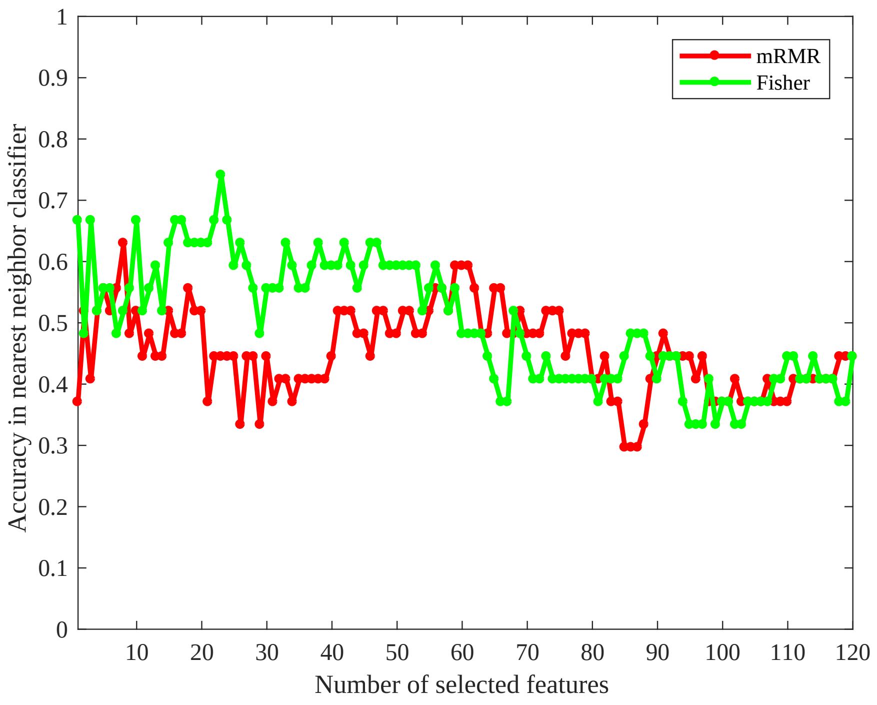


Figure 6.11: Cross-validation experiment for finding the best k features

selection algorithm to select k discriminative features. The discriminative subgraph is found after a cross-validation-based experiment. Because of the small number of examples in our dataset, we used leave-one-out (LOO) train/test splitting strategy. Leave-one-out is a special case of K-fold cross-validation while K is the number of examples. At i-th iteration of leave-one-out, all examples except the i-th one are used as training examples and the i-th example is used as the test example. For this cross-validation-based experiment, we used nearest neighbor classifier with Euclidean distance measure. At k-th iteration, we first select k features by mRMR/Fisher scoring, and report the mean accuracy of the nearest neighbor classifier, over the iterations of leave-one-out. In Fig. 6.11, we show the accuracy of nearest neighbor classifier after incrementally selecting features. Since Fisher-based top-23 selected features give maximum LOO accuracy, i.e., 74%, we get $k = 23$, and consider these 23 features as the edges of the discriminative subgraph. Each edge of the discriminative subgraph is assigned a weight, which is the Fisher score of its corresponding feature. In Fig. 6.12, we show the discriminative

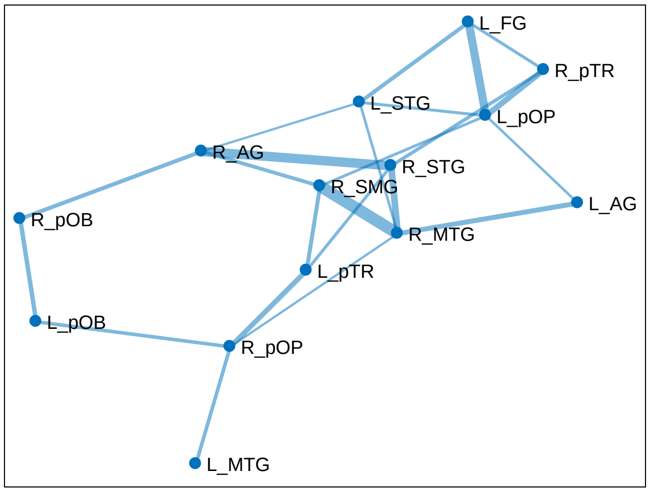
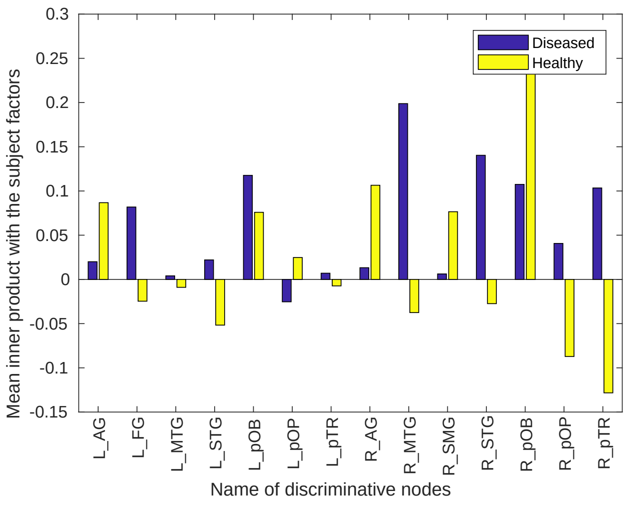


Figure 6.12: Discriminative subgraph using Fisher-selected 23 edges

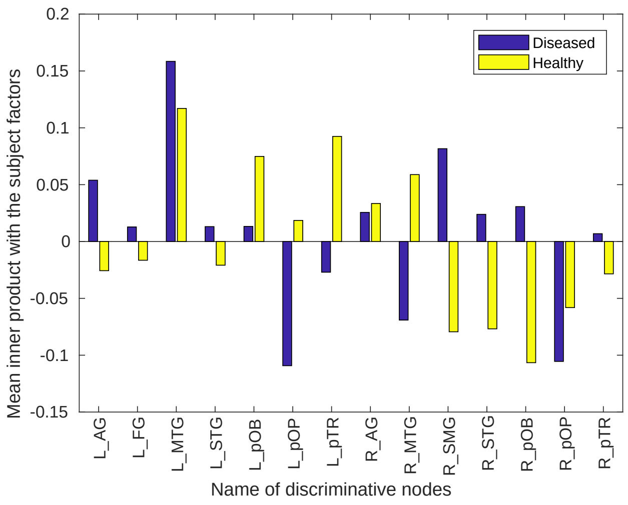
subgraph for our dataset, where edge weights are visualized by the thickness of the edge. It is easy to see that the connections (*R_AG, R_STG*) and (*R_SMG, R_MTG*) has maximum discrimination ability.

6.3.2 Interpretability of the Biomarkers

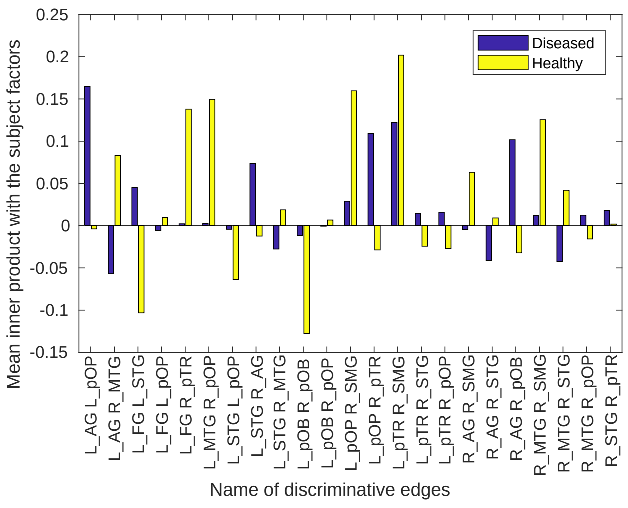
This experiment is done to verify the discrimination ability of the selected biomarkers. Firstly, Fisher-selected top 23 features are used to construct the discriminative subgraph. Then, a third-order tensor is constructed by stacking the incidence matrices of the induced discriminative subgraph of the complete functional connectivity networks. After construction of the tensor, CP and Tucker decomposition are performed separately. As a result of tensor decomposition, three factor matrices representing the subjects, discriminative nodes, and discriminative edges are found. Then, the rows of the subject factor matrix are divided into healthy and diseased classes. Inner product (cosine similarity) of each healthy subject representation and discriminative node representation is calculated and mean similarity of the healthy subjects with the discriminative nodes are calculated. The similar calculation is done to find the mean similarity of the healthy



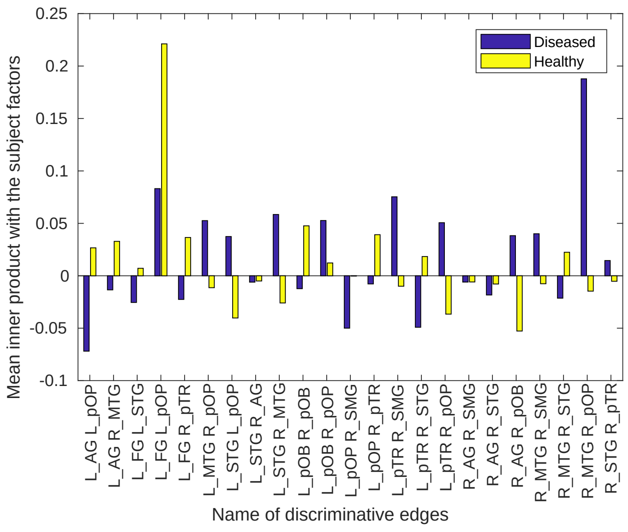
(a) Discrimination made by CP-based node factors



(b) Discrimination by Tucker-based node factors



(c) Discrimination by CP-based edge factors



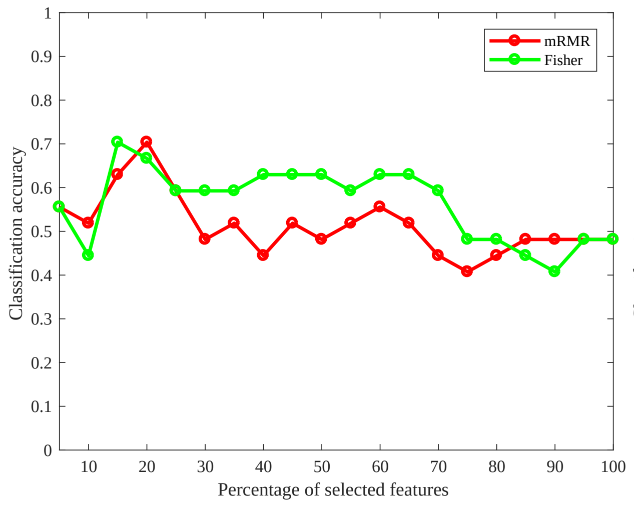
(d) Discrimination by Tucker-based edge factors

Figure 6.13: CP and Tucker decomposition-based representations of the discriminative nodes and edges

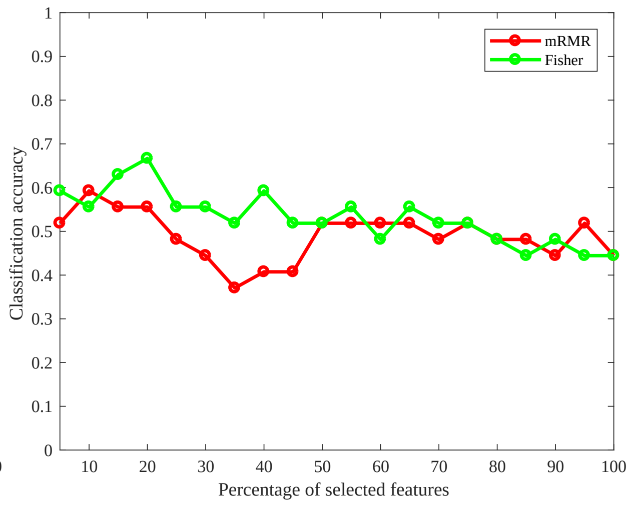
subjects with the discriminative edges. For the diseased class, mean similarity of the diseased subjects to the discriminative nodes, and mean similarity of the diseased subjects to the discriminative edges are calculated. In the discriminative subgraph (Fig. 6.12), there are 14 discriminative nodes and 23 discriminative edges. From Fig. 6.13, we see that almost all the discriminative nodes and edges show a high difference in mean similarity with the healthy and diseased class. Some less discriminative patterns are also found by this verification, such as the node *L_MTG* and the edge (*R_STG, R_PTR*). Fig. 6.12 also supports the fact that the edge (*R_STG, R_PTR*) has comparatively smaller Fisher score than other edges.

6.3.3 Performance in Different Classifiers

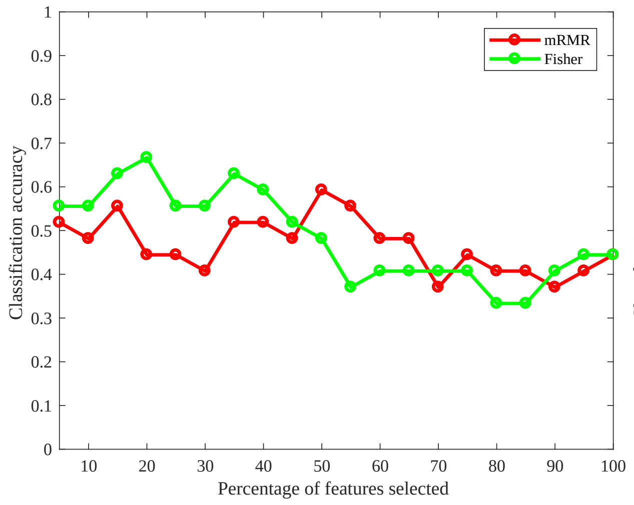
The selected features by mRMR and Fisher algorithms are tested using four classifiers - Support Vector Machine (SVM) with radial basis function kernel, Naïve Bayes, knn (number of neighbors=1), and knn (number of neighbors=10). In Fig. 6.14, we observe that in most of the cases, Fisher-based feature sets result in higher classification accuracy than mRMR-based feature sets. Around top-20% Fisher-selected features give maximum accuracy in most of the cases. The reason why Fisher-based approach gives better performance is the nature of this dataset. It proves the fact that in this dataset most features are independently relevant to the class labels. In case of other fMRI-based functional connectivity datasets, where the features are statistically dependent on each other, mRMR-based approach can perform better than Fisher-based approach. The better discrimination ability of Fisher-selected top-20% features in most of the classifiers also supports the selection of 23 features ($\approx 20\%$ of the total number of features) for constructing the discriminative subgraph for this dataset.



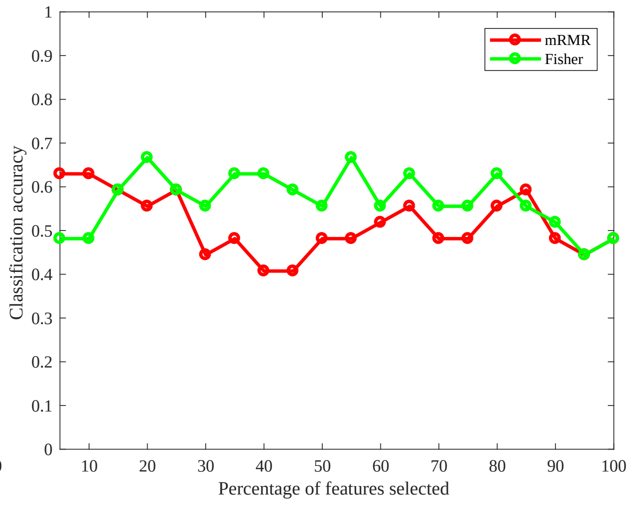
(a) SVM



(b) Naïve Bayes



(c) knn (number of neighbors = 1)



(d) knn (number of neighbors = 10)

Figure 6.14: Leave-one-out accuracies of different classifiers after selecting features in mRMR and Fisher methods

Table 6.3: Datasets

Tag *(lLsS)*	**Lookback** *L* **(hours)**	**Maximum span** *S* **(hours)**	**# of events**	**# of flares**	**# of non-flares**
l12s24	12	24	3,436	431	3,005
l24s24	24	24	3,292	403	2,889

6.4 *Solar Flare Prediction by Time Series Classification*

In this section, we demonstrate our experimental findings of multivariate time series-based solar flare prediction. We used Python's Scikit-learn library for using k-NN classifier. In all the experiments, k-NN classifier uses Euclidean distance. The code of the experiments is available at our Github repository. [8]

6.4.1 Dataset Description

In the dataset tagged by $lLsS$, the time series of a set of AR parameters are collected before L hours of M/X-class flare occurrence (or before L hours of sampling time of non-flaring Active Regions) and these time series are stacked together to make the $mvts$ of the $event$. The length of each time series of the events of the dataset $lLsS$ is $S \times 5$, since the magnetic field values of the AR patches are calculated by SHARP (Spaceweather HMI Active Region Patch) in 12 minutes cadence [87]. When $S = 24$ hours, each time series reach maximum length of $24 \times 5 = 120$. Therefore, $l12s24$ and $l24s24$ are two primitive and full datasets used in the experiments (Table 6.3). Datasets $lLsS$ with other spans, where $S < 24$ can be derived by slicing the last $S \times 5$ values from each time series of the events of $lLs24$, where $L \in \{12, 24\}$. In this work, we have used 16 AR parameters shown in Table 6.4, whose formulas can be found in [19].

8 `http://github.com/hamdi08/Flare_expts_SABID17/`

Table 6.4: AR parameters used in the experiments

Tag	Description
USFLUX	Total unsigned flux
MEANGAM	Mean angle of field from radial
MEANGBT	Mean gradient of total field
MEANGBZ	Mean gradient of vertical field
MEANGBH	Mean gradient of horizontal field
MEANJZD	Mean vertical current density
TOTUSJZ	Total unsigned vertical current
MEANALP	Mean characteristic twist parameter α
MEANJZH	Mean current helicity (B_z contribution)
TOTUSJH	Total unsigned current helicity
ABSNJZH	Absolute value of the net current helicity
SAVNCPP	Sum of the modulus of the net current per polarity
MEANPOT	Mean photospheric magnetic free energy
TOTPOT	Total photospheric magnetic free energy density
MEANSHR	Mean shear angle
SHRGT45	Fraction of Area with Shear $> 45°$

6.4.2 Train/test Splitting Methodology

In Table 6.5, we have described the splitting strategy in datasets with fixed lookback time. We trained the model with four years data, sampled from January 2011 to December 2014 and tested the model with events sampled from January 2015 to December 2016. Class imbalance ratio for training is 6.14, and for testing is 11.5. Overall 73% events of a dataset are used for training, while rest 27% are used for testing. Since the train/test splitting is done on the basis of the temporal occurrence of the events, in order to calculate the performance measures, unstratified splitting is performed once. Before running the classifier, both training and test datasets are z-normalized.

Table 6.5: Splitting datasets into train and test sets

Dataset	**Training set**				**Test set**			
	Duration	# of events	# of flares	# of non-flares	Duration	# of events	# of flares	# of non-flares
l12sS	2011-2014	2,503	349	2,154	2015-2016	933	82	851
l24sS		2,408	328	2,080		884	75	809

Predicted class

		+1	-1	Total
	+1	TP	FN	P
Actual class	-1	FP	TN	N
	Total	P′	N′	P + N

Figure 6.15: Confusion matrix

6.4.3 Performance Measures

To compare our classification results with the existing flare prediction studies ([19], [67], [68], [69], [88], [89]), we evaluate 11 performance measures: accuracy, precision (positive and negative), recall (positive and negative), F1 (positive and negative), HSS_1, HSS_2, GS and TSS. Given a set of test examples, we get a confusion matrix as a result of classification (Fig. 6.15) [75]. The confusion matrix has four entries - TP, TN, FP and FN, where TP (true positive) is the number of positive examples that are correctly labeled, TN (true negative) is the number of negative examples that are correctly labeled, FP (false positive) is the number of negative examples that are misclassified as positive, and FN (false negative) is the number of positive examples that are misclassified as negative. P and N are the numbers of actual positive and negative examples respectively. Since in flare prediction $P << N$, class imbalance problem exists and therefore accuracy is not a good performance measure. In this subsection, we briefly discuss some performance measures such as HSS_1, HSS_2, GS and TSS which are typically used in the evaluation of flare prediction performance.

Heidke Skill Score and Gilbert Score: To deal with the class imbalance problem, two versions of Heidke Skill Score HSS_1 [88] and HSS_2 [64], and Gilbert score GS [64] have been used in previous solar flare prediction literature [19].

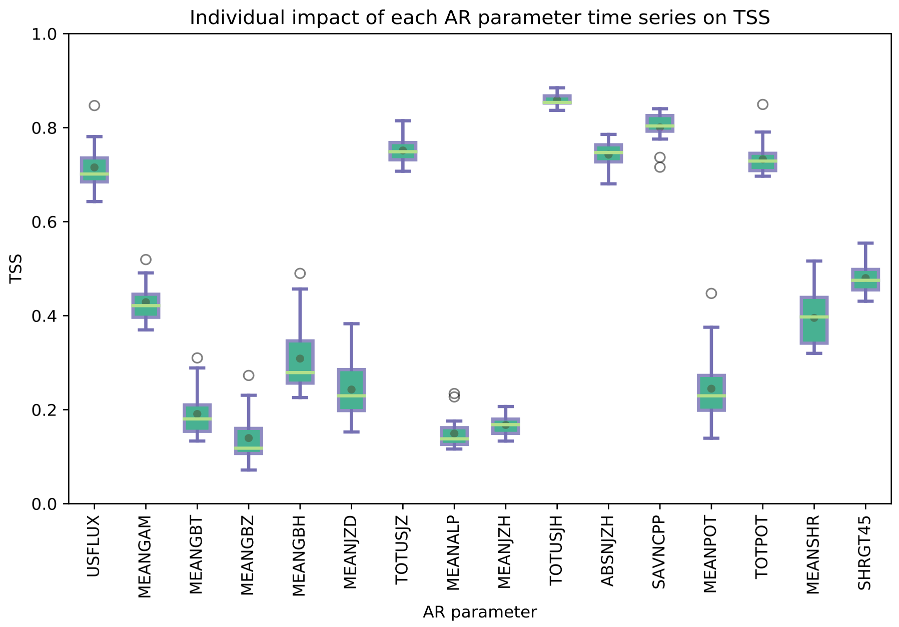
$$
\begin{aligned}
\mathsf{HSS}_1 &= \frac{\mathsf{TP} + \mathsf{TN} - \mathsf{N}}{\mathsf{P}} \\
\mathsf{HSS}_2 &= \frac{2 \times [(\mathsf{TP} \times \mathsf{TN}) - (\mathsf{FN} \times \mathsf{FP})]}{\mathsf{P} \times \mathsf{N}' + \mathsf{N} \times \mathsf{P}'} \\
\mathsf{GS} &= \frac{\mathsf{TP} \times (\mathsf{P} + \mathsf{N}) - \mathsf{P} \times \mathsf{P}'}{\mathsf{FN} \times (\mathsf{P} + \mathsf{N}) - \mathsf{N} \times \mathsf{P}'}
\end{aligned}
$$

While HSS_1 measures the improvement of the prediction over the "*always negative class*" prediction, HSS_2 measures the improvement of the prediction over random prediction. GS considers the number of TP obtained by chance.

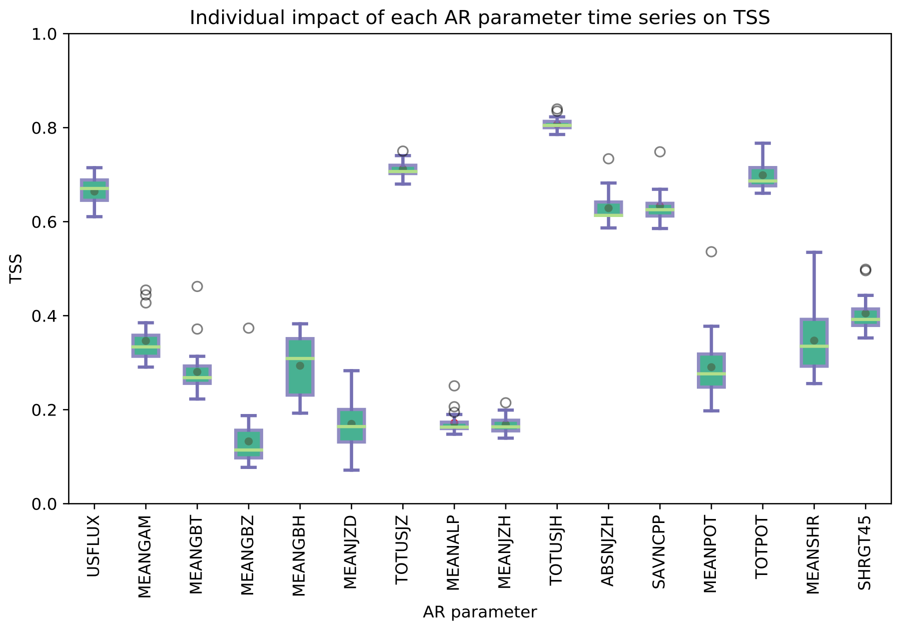
True Skill Statistic: Since $\mathsf{HSS}_1, \mathsf{HSS}_2$ and GS still show some dependence on the class imbalance ratio, Bloomfield et al. [89] defined TSS, which is independent on class imbalance ratio and defined as the difference between true positive rate and false positive rate.

$$
\mathsf{TSS} = \frac{\mathsf{TP}}{\mathsf{P}} - \frac{\mathsf{FP}}{\mathsf{N}}
$$

TSS ranges from -1 to $+1$, where random prediction scores 0, perfect prediction scores $+1$, and the prediction that is always wrong scores -1. According to Bobra et al. [19], TSS is the most meaningful measure for performance comparison of different flare prediction studies.



(a) Time series extracted with lookback 12 hours



(b) Time series extracted with lookback 24 hours

Figure 6.16: TSS distributions after 20 k-NN executions on the summarized time series of the individual AR parameters.

6.4.4 Best AR Parameter Selection

Among the 16 AR parameters, the AR parameter whose corresponding time series give maximum mean TSS with minimum variance after k-NN classification with varying k is considered to be the best AR parameter in terms of the distinguishing ability of the time series. For this experiment, we use both datasets $l12s24$ (Fig. 6.16a) and $l24s24$ (Fig. 6.16b) because of the completeness of their time series. For each event of these datasets, we collect the time series of only one parameter P_j at a time, where $1 \leqslant j \leqslant 16$ and summarized the time series by 8 summary statistics described in Section 5.4.2. Then we run k-NN classifier with varying $k = 1, 2, \ldots, 20$ and measure the TSS value for each run. From the boxplots of Fig. 6.16, it is visible that the summarized time series of *total unsigned current helicity* (TOTUSJH) achieve maximum mean TSS with minimum variance for both lookback settings. This finding is exactly the same as [19], where they found TOTUSJH as the top-1 selected AR parameter based on Fisher criterion. Like [19], our result also indicates that the parameters which calculate sums, e.g., TOTUSJH, TOTUSJZ, TOTPOT, SAVNCPP etc are better than the parameters that calculate the means, e.g., MEANGBZ, MEANGBT, MEANGBH, MEANALP etc. In our later experiments, we only used the time series of the parameter TOTUSJH.

6.4.5 Optimal k in k-NN Classifier

Since the number of neighbors (k) in k-NN classifier can affect the classification performance, we look for best performing k value in all datasets. For each lookback L where $L \in \{12, 24\}$ hours, we derive 12 datasets by increasing the span by 2 hours. The derived datasets are $\{l12s2, l12s4, \ldots, l12s24, l24s2, l24s4, \ldots, l24s24\}$. Then for each of these 24 datasets, we extract the summarized time series of the AR parameter TOTUSJH of each event. Finally, for $k = 1, 2, \ldots, 20$, we measure the TSS values in all 24 datasets. From

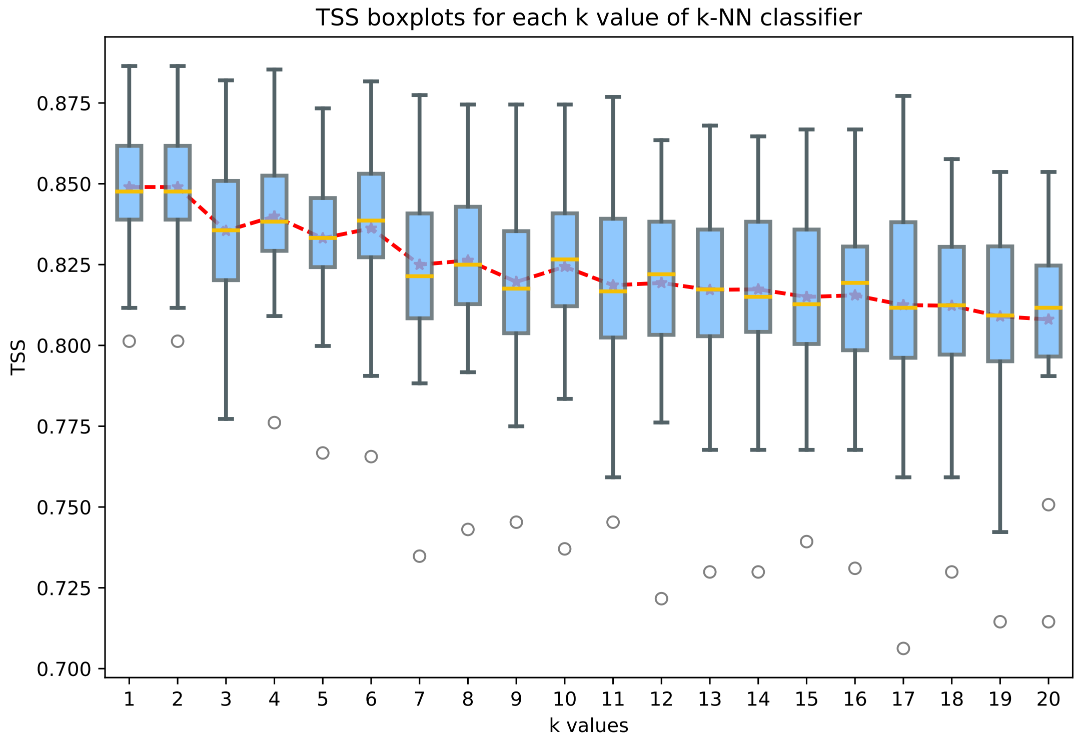


Figure 6.17: TSS distributions for different k values of k-NN classification on 24 derived datasets of summarized time series of TOTUSJH.

the boxplots of Fig. 6.17, we see that $k = 1$ and $k = 2$ has the maximum mean TSS. We consider $k = 1$ as optimal k for the later experiments.

6.4.6 Comparison with Other Baselines

In this subsection, we compare our method with two other baselines.

Baseline 1: For each $mvts_i \in \mathbb{R}^{T \times N}$, where $1 \leqslant i \leqslant M$, we collect only the last row (latest value of each of the N time series) and consider it as a vector. The resultant vector space has size $N \times M$ (recall that N is the number of AR parameters and M is the number of examples). Unlike other two baselines, this baseline does not consider any span time window. SVM with weighted class weights (cost of the positive class is set greater than that of the negative class since positive examples are comparatively rare) is applied on this vector space. A similar approach was taken in [19]. Since this baseline uses only the ***l***atest ***v***alue of ***e***ach AR ***p***arameter time series, this is tagged as *lvep* in Table 6.6.

Baseline 2: For each $mvts_i \in \mathbb{R}^{T \times N}$, where $1 \leqslant i \leqslant M$, we collect the mean value of each time series (AR parameter). Similar to baseline 1, the resultant vector space has size $N \times M$ and SVM with weighted class cost is used as the classifier. One significant difference between baseline 1 and baseline 2 is, baseline 1 does not use the time series of the AR parameters, while baseline 2 uses the mean (one summary statistic) of the time series of each AR parameter. Since this baseline uses the ***m***ean ***v***alue of ***e***ach AR ***p***arameter time series, this is tagged as *mvep* in Table 6.6.

In the proposed method, each $mvts_i$ is represented by the summarized representation of full span (24 hours) time series of the best AR parameter TOTUSJH. The resultant vector space has size $8 \times M$. Finally, k-NN classifier with $k = 1$ is run on this vector space. Since only the ***b***est AR ***p***arameter is used, we tag the proposed method as *bp* in Table 6.6.

In Table 6.6, we show the performance comparison of three methods *lvep*, *mvep*, and *bp* using 11 performance measures. For both datasets of lookback 12 and 24 hours, except

Table 6.6: Comparison of the proposed method with two other baselines in 11 performance measures

Dataset tag	l12sS			l24sS		
Baseline tag *	*lvep*	*mvep*	*bp*	*lvep*	*mvep*	*bp*
Span window used	0 hour	24 hours	24 hours	0 hour	24 hours	24 hours
AR parameters used	All	All	TOTUSJH	All	All	TOTUSJH
Classifier	SVM (weighted class cost)	SVM (weighted class cost)	k-NN (k=1)	SVM (weighted class cost)	SVM (weighted class cost)	k-NN (k=1)
Accuracy	0.919	0.919	**0.975**	0.906	0.913	**0.975**
Precision (positive)	0.520	0.521	**0.831**	0.472	0.493	**0.853**
Precision (negative)	**0.994**	0.992	0.991	**0.991**	**0.991**	0.986
Recall (positive)	**0.939**	0.927	0.902	**0.907**	**0.907**	0.853
Recall (negative)	0.917	0.918	**0.982**	0.906	0.913	**0.986**
F1 (positive)	0.670	0.667	**0.865**	0.621	0.638	**0.853**
F1 (negative)	0.954	0.954	**0.986**	0.946	0.950	**0.986**
HSS_1	0.073	0.073	**0.720**	-0.107	-0.027	**0.707**
HSS_2	0.627	0.624	**0.852**	0.573	0.594	**0.840**
GS	0.457	0.454	**0.742**	0.402	0.422	**0.724**
TSS	0.856	0.844	**0.885**	0.813	0.820	**0.840**

* *lvep*: latest value of each parameter, *mvep*: mean value of each parameter, *bp*: best parameter

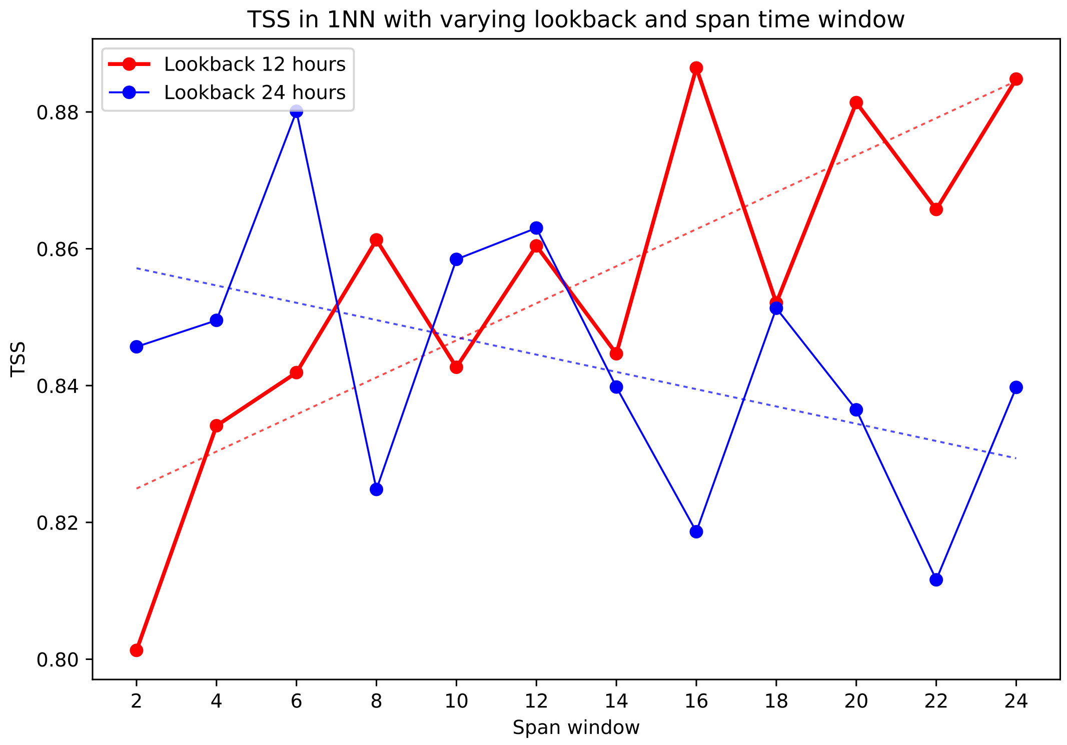


Figure 6.18: Variation of TSS after running k-NN (k=1) with different lookback and span settings on the time series of TOTUSJH

the precision (negative) and recall (positive), our proposed method *bp* performs best in all other measures. The performance of *lvep* and *mvep* are almost the same in all measures, but *lvep* performs slightly better in small lookback window while *mvep* has slightly better performance in large lookback window. Although *lvep* and *mvep* are almost as good as *bp* in some measures such as TSS, they exhibit poor performance in other measures such as HSS_1, HSS_2, and GS in comparison with *bp*.

6.4.7 Span Window-based Performance

Given a fixed lookback window L, how does the classification performance change if we change the span window size? Fig. 6.18 shows the change of TSS value after running k-NN ($k = 1$) with varying lookback and span windows of the time series of TOTUSJH. When the lookback is small such as 12 hours, the temporal proximity to the actual event is small, and we observe a linearly increasing trend (dashed red straight line) with the increase of span window. On the contrary, when the lookback is large, e.g., 24 hours, we observe a linearly decreasing trend of TSS (dashed blue straight line). Although the increase or decrease of TSS is not obvious after increasing span since the extension of time series with new values might improve/deteriorate the performance, this overall linear increasing/decreasing trend is an indication of good time series quality of the AR parameter TOTUSJH.

6.4.8 Effect of C-class Flares in Classification performance

In this experiment, we added the Active Regions with having one or more C-class flares (less intense flares in comparison with X/M-class flares) in the positive class. The number of sampled C-class flares for lookback 12 hours is 5,527 and for lookback 24 hours is 5,353. The inclusion of C-class flares changes the class imbalance ratio stated in Section 6.4.2 since the number of positive examples (X, M and C-class flares) exceeds the number of non-flares (Active Regions with no flares occurring during the disk crossing). Fig. 6.19 shows that in both lookback settings with full span window, the inclusion of C-class flares has a slightly negative impact in almost all performance metrics, while we consider k-NN ($k = 1$) classification on the summarized time series of the AR parameter TOTUSJH. This result agrees with the finding of Bloomfield et al. [89] which says that HSS_1 increases as a result of including of C-class flares in positive class, but it results in the decrease in TSS.

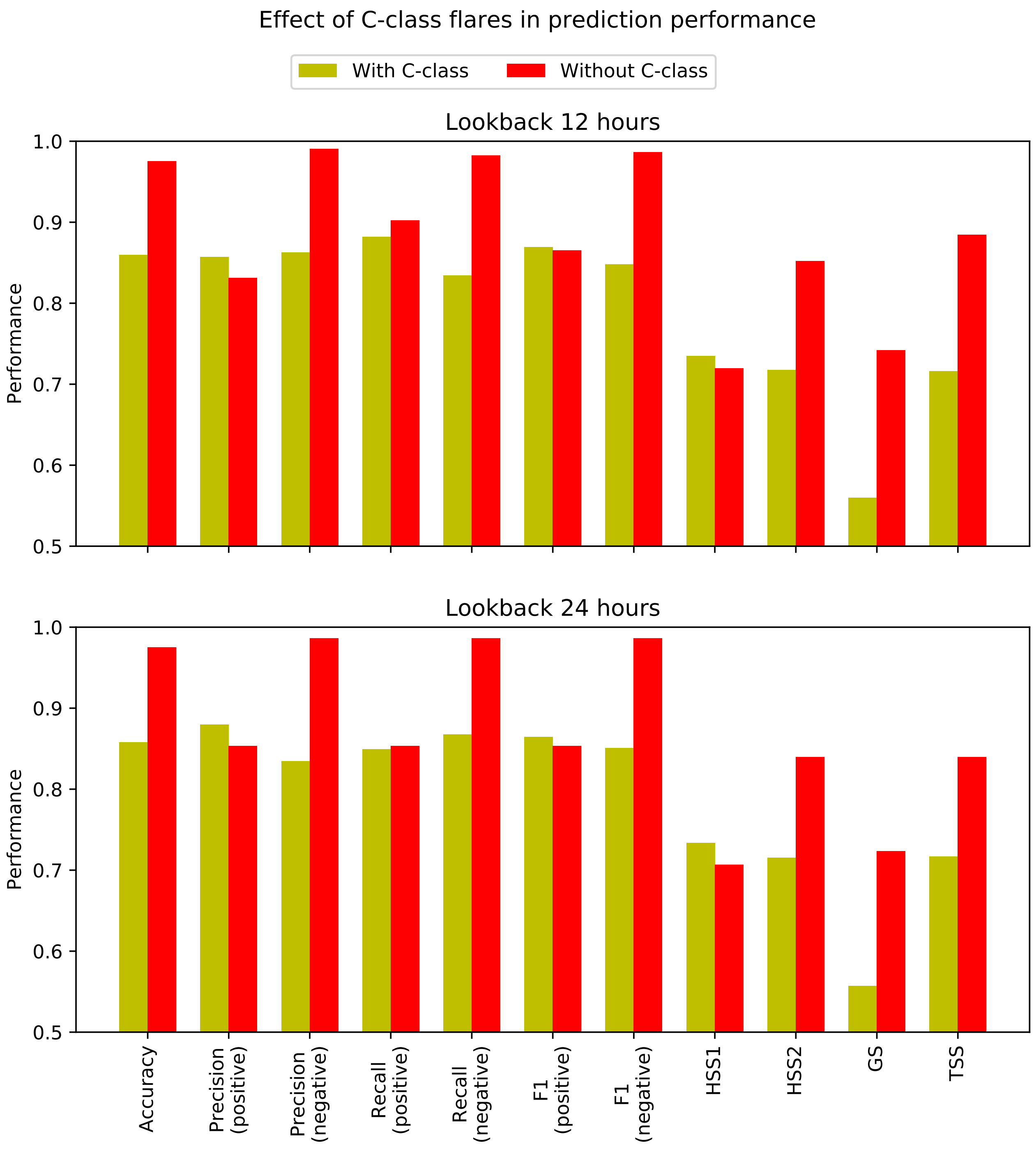


Figure 6.19: Performance after k-NN (k=1) execution on the summarized time series of TOTUSJH with/without considering C-class flares

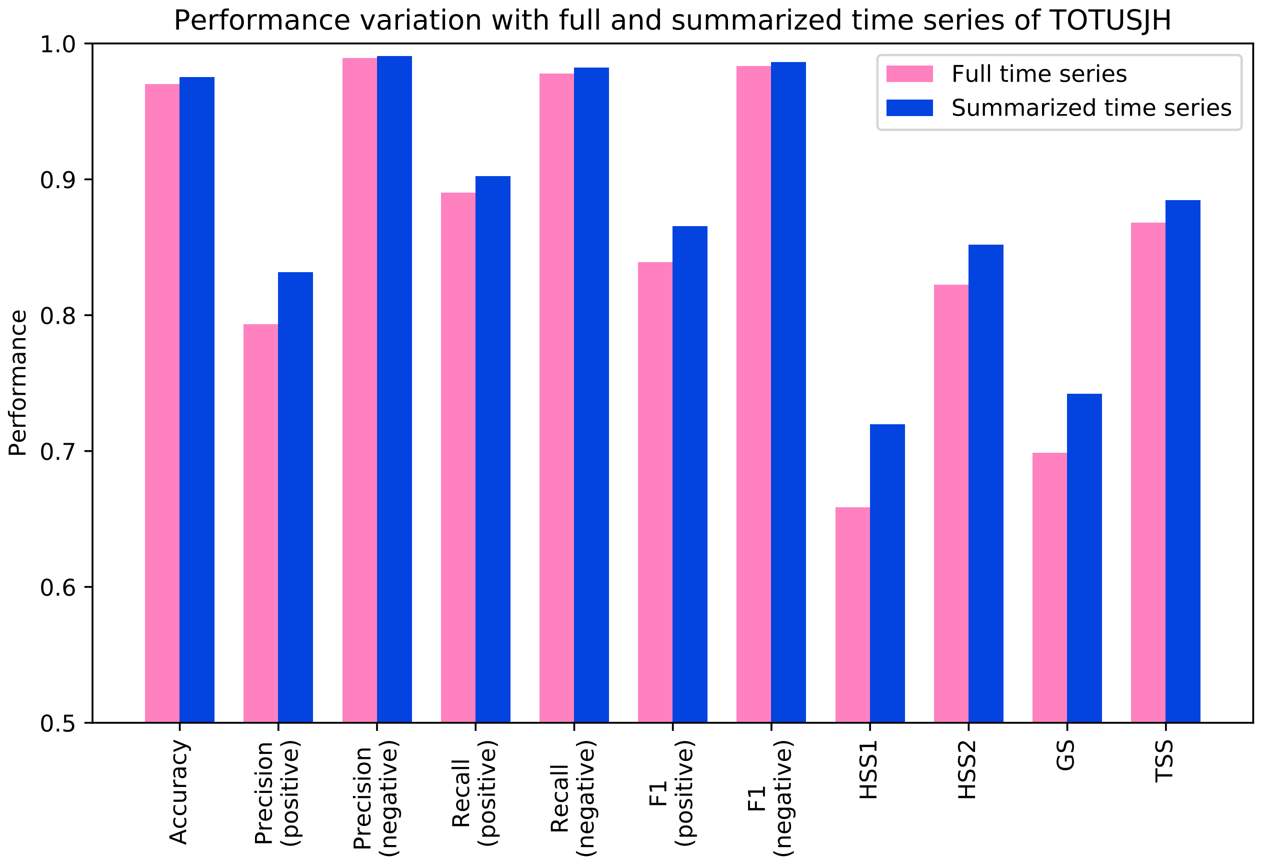


Figure 6.20: Comparison of summarized representation and full-length representation of TOTUSJH time series in classification performance

6.4.9 Effect of Time Series Summarization

Fig. 6.20 provides the rationale for our preference of summarized representation of the time series over the full-length time series. This experiment is run on the $l12s24$ dataset. It is visible that if we consider the summarized representation instead of the full-length time series of the AR parameter TOTUSJH, we get an increase in all performance metrics. The number of dimensions in the full-length time series representation is 120, while in case of the summarized representation the number of dimensions is only 8. Since k-NN classifier greatly depends on the number of dimensions in the vector space, the runtime of the classifier is more than 10 times bigger for full-length time series representation than that of summarized representation.

6.4.10 Performance using Other Classifiers

When flaring and non-flaring Active Regions are represented by the summarized representation of the time series of the AR parameter TOTUSJH, other sophisticated classifiers can achieve even better performance than k-NN ($k = 1$). Table 6.7 shows the 11 performance metrics found after running the classifiers SVM, random forest, and naïve Bayes on $l12s24$ and $l24s24$ datasets with the default settings of their Scikit-learn implementations. We also show the performance of k-NN ($k = 1$) in Table 6.7 for comparison. These classifiers and their hyperparameters (according to their Scikit-learn specifications) are listed below.

- **SVM:** C=1.0, kernel=rbf, gamma=1/8, class_weight=None.
- **Random forest:** n_estimators=10, criterion=gini, max_depth=None, class_weight=None.
- **Naïve Bayes:** priors=None.
- **k-NN:** number of neighbors (k) = 1.

Table 6.7: Performance by other classifiers on the summarized time series of TOTUSJH

Dataset tag	l12s24				l24s24			
Classifier	SVM	Random forest	Naïve Bayes	k-NN (k=1)	SVM	Random forest	Naïve Bayes	k-NN (k=1)
Accuracy	**0.99**	0.97	0.98	0.98	**0.99**	0.97	0.98	0.98
Precision(positive)	**0.96**	0.8	0.8	0.83	**0.98**	0.81	0.84	0.85
Precision(negative)	0.99	0.99	**1.0**	0.99	**0.99**	**0.99**	**0.99**	**0.99**
Recall(positive)	0.93	0.9	**0.95**	0.9	0.87	0.87	**0.91**	0.85
Recall(negative)	**1.0**	0.98	0.98	0.98	**1.0**	0.98	0.98	0.99
F1(positive)	**0.94**	0.85	0.87	0.87	**0.92**	0.84	0.87	0.85
F1(negative)	**0.99**	0.98	**0.99**	**0.99**	**0.99**	0.98	**0.99**	**0.99**
HSS_1	**0.89**	0.67	0.72	0.72	**0.85**	0.67	0.73	0.71
HSS_2	**0.94**	0.83	0.86	0.85	**0.92**	0.82	0.86	0.84
GS	**0.88**	0.71	0.75	0.74	**0.84**	0.7	0.75	0.72
TSS	0.92	0.88	**0.93**	0.89	0.87	0.85	**0.89**	0.84

Experimental results that are shown in Table 6.7 show the robustness of our data representation, i. e., the summarized representation of the time series of AR parameter TOTUSJH, in classifying flaring and non-flaring Active Regions regardless of the classifier. Although some of these classification models have better performance than k-NN, k-NN is found to be more interpretable than these classifiers with respect to lookback and span windows of the time series (Fig. 6.18).

6.5 *Solar Event Visualization*

In this section, we present the experimental findings of the solar event visualization in 2D space using different dimensionality reduction methods (see details in section 5.5).

6.5.1 Dataset Description

We used a dataset introduced in *Big Data Challenge 2019* [90]. The dataset contains 25,157 solar events in the form of multivariate time series ($mvts$) collected in the duration of 09/29/2013 - 03/19/2014. Each $mvts$ contains 25 time series of length 60 of the magnetic field parameters. Therefore, according to the problem formulation of section 5.5, $M = 25157$, $T = 60$, and $N = 25$. Among these events, 1,231 are flaring (124 X-class flares and 1,107 M-class flares), and rest 23,926 are non-flaring.

6.5.2 Baselines

We used PCA and t-SNE on four vector represented datasets.

1. **Flattened MVTS**: For each $mvts_i \in \mathbb{R}^{T \times N}$, where $1 \leqslant i \leqslant M$, we flattened as a vector of length TN. We used these vector representations in PCA and t-SNE dimensionality reduction algorithms.

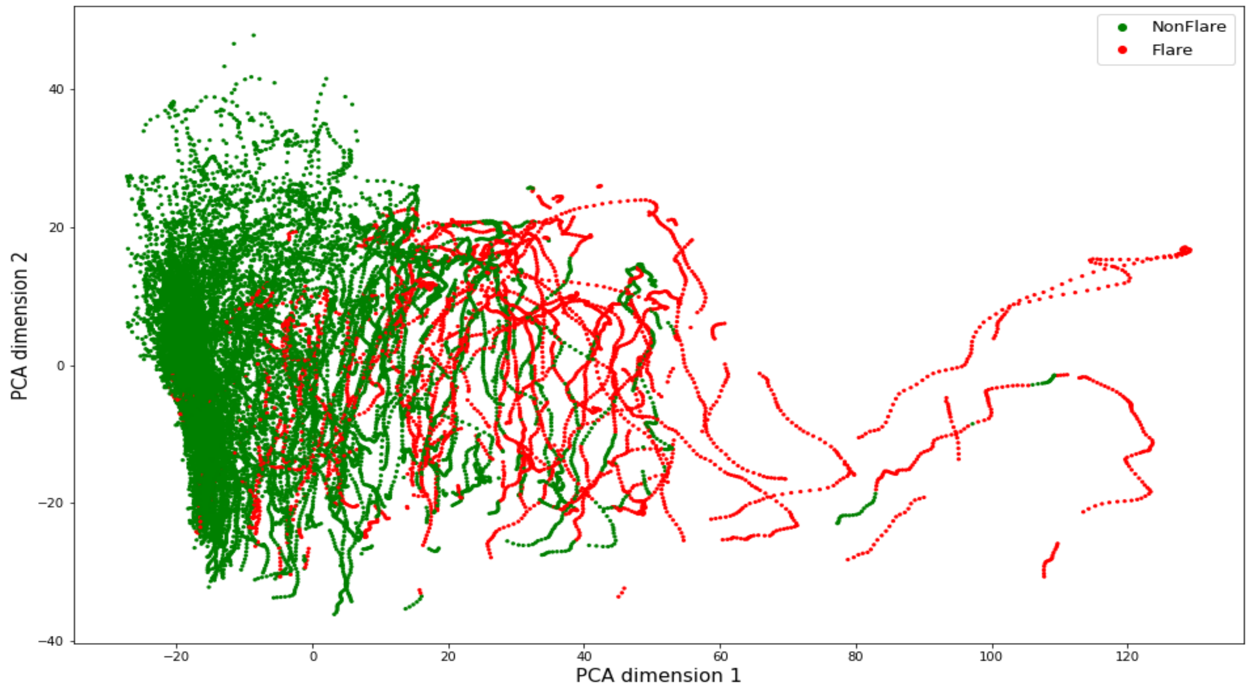
2. **Last timestamp vectors**: We extracted the last row of each $mvts$, which represents the magnetic field parameter values at the last time instant. Therefore, each event is represented by a vector of length N. Similar approach was taken in [19].

3. **Mean of each parameter**: We calculated the mean of each magnetic field parameter of each $mvts$. Each event is represented by a vector of length N.

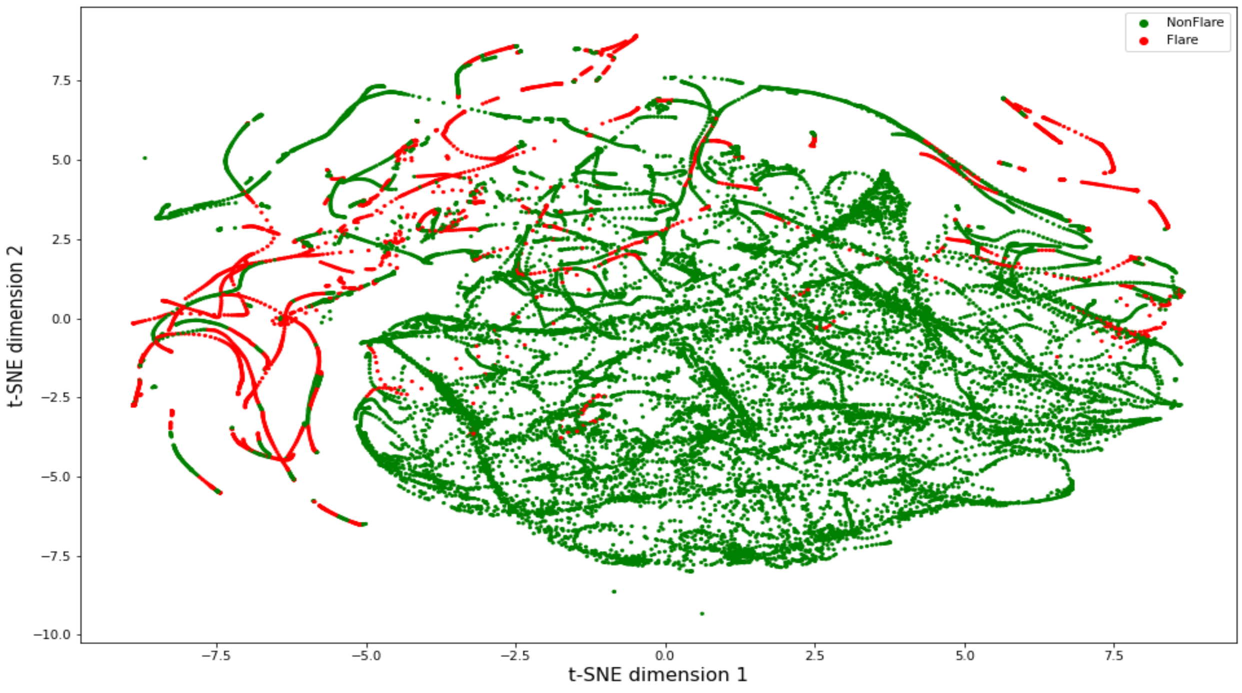
4. **Tensor decomposition**: We followed the approach discussed in section 5.5. We considered all the $mvts$'s forming a third order tensor, where the modes represent the event, timestamp, and magnetic field parameter (Fig. 5.3). We decompose the tensor by CP decomposition with a given rank $R = 2$. We find three factor matrices of events, timestamps, and magnetic field parameters. Then we project the representations of events on the representations of timestamps and magnetic field parameters separately. Finally, we perform PCA and t-SNE on the event-timestamp-parameter vectors, which are the concatenations of the event-timestamp vectors and event-parameter vectors.

6.5.3 Explanation of the 2D Visualizations

In Fig. 6.21, Fig. 6.22, Fig. 6.23, and Fig. 6.24, we show the PCA and t-SNE visualizations of the flaring and nonflaring solar events on the baseline data representations. Flaring events are projected using red points, while nonflaring events are projected using green points on the scatter plots. Although the overall performance of class separability is found better in the classical dimensionality reduction methods (PCA and t-SNE on flattened $mvts$, last timestamp vectors, and mean of magnetic field parameters), the

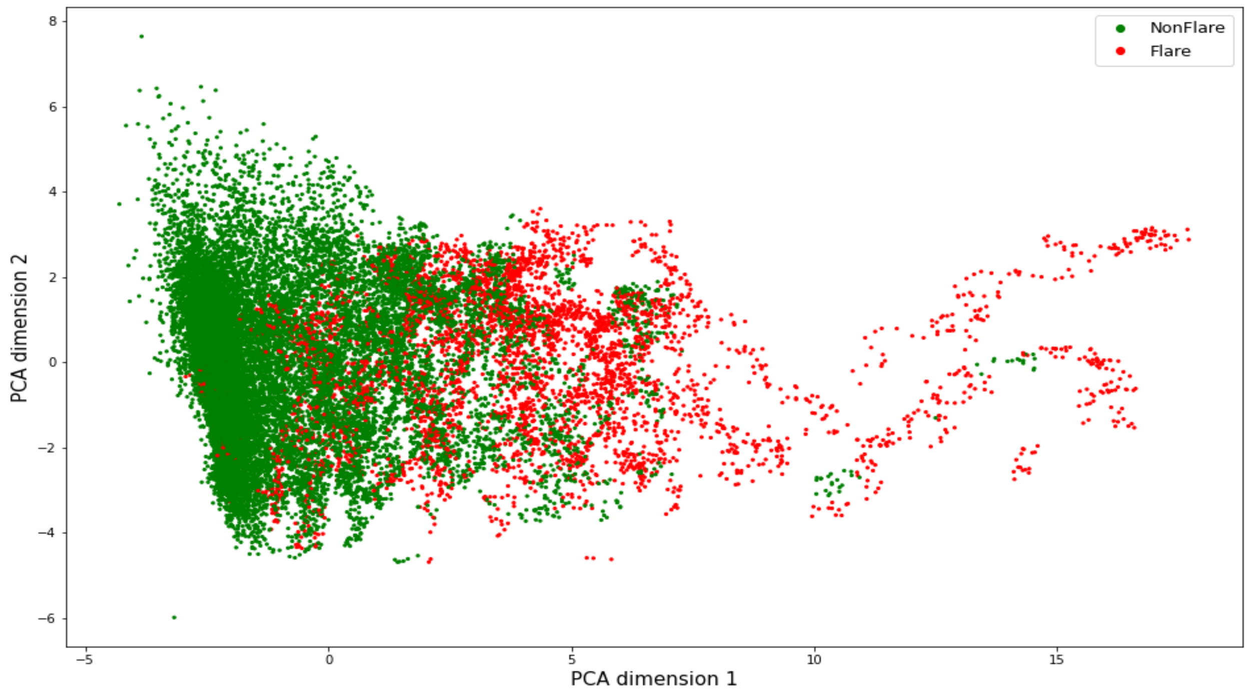


(a) PCA

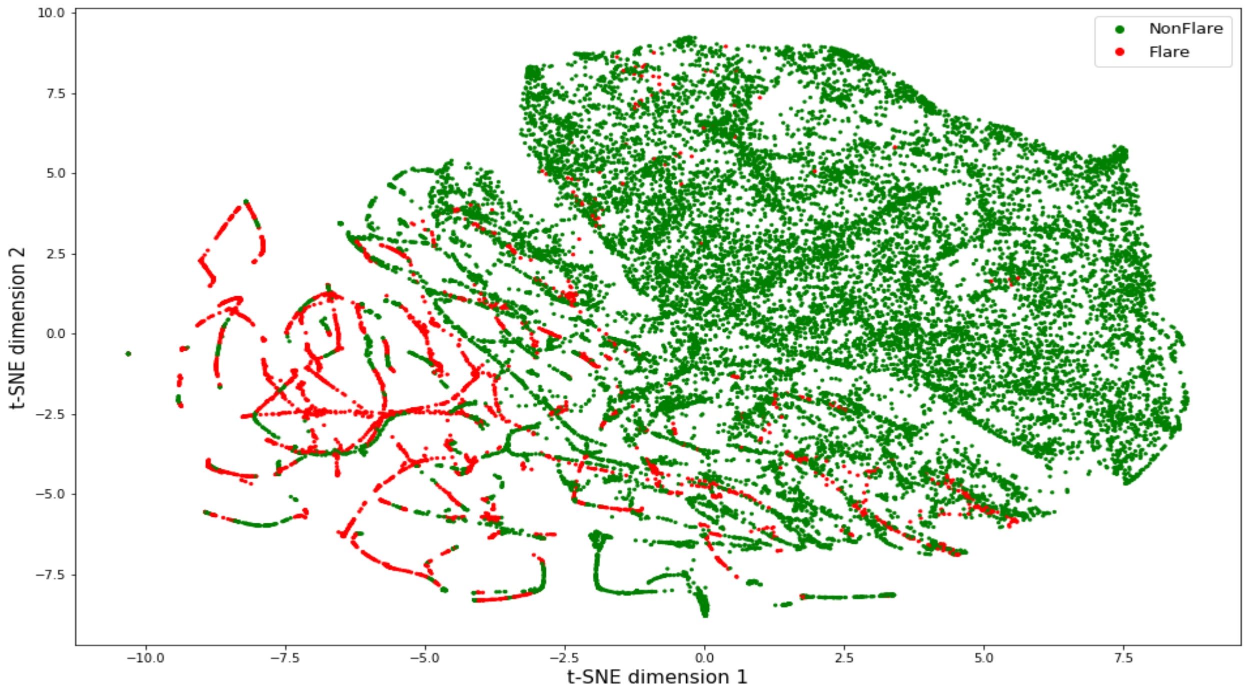


(b) t-SNE

Figure 6.21: Visualization of the flattened mvts's of the flaring and nonflaring solar events

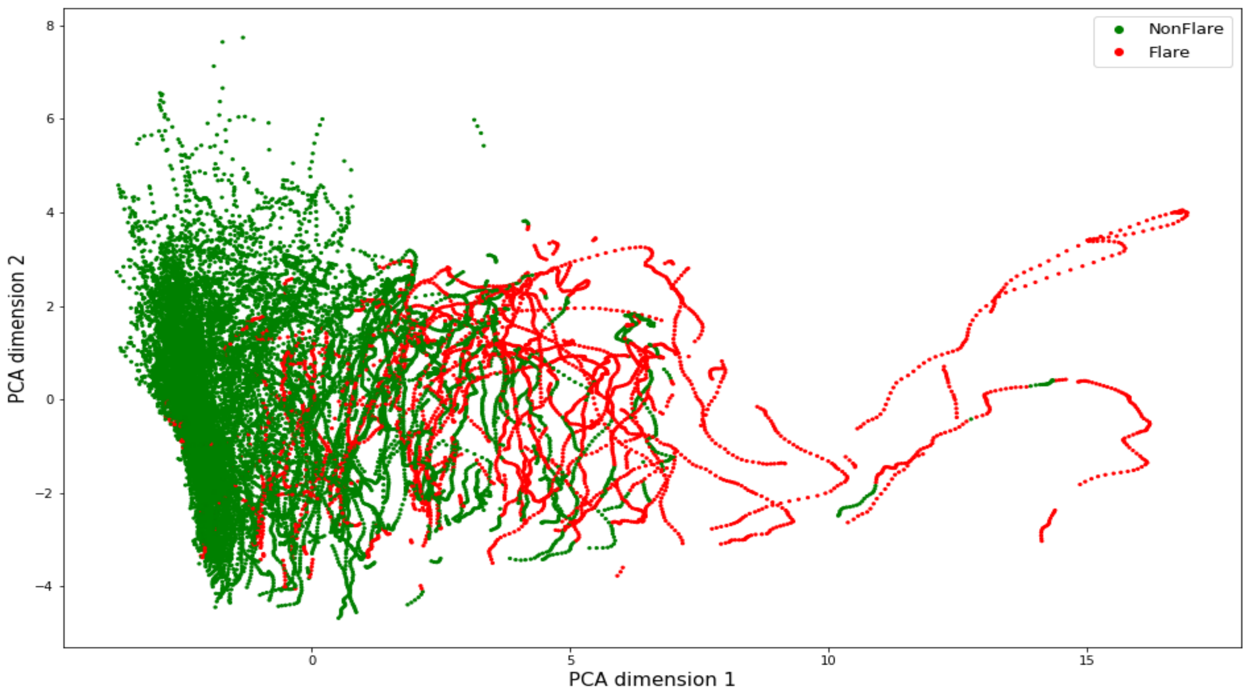


(a) PCA

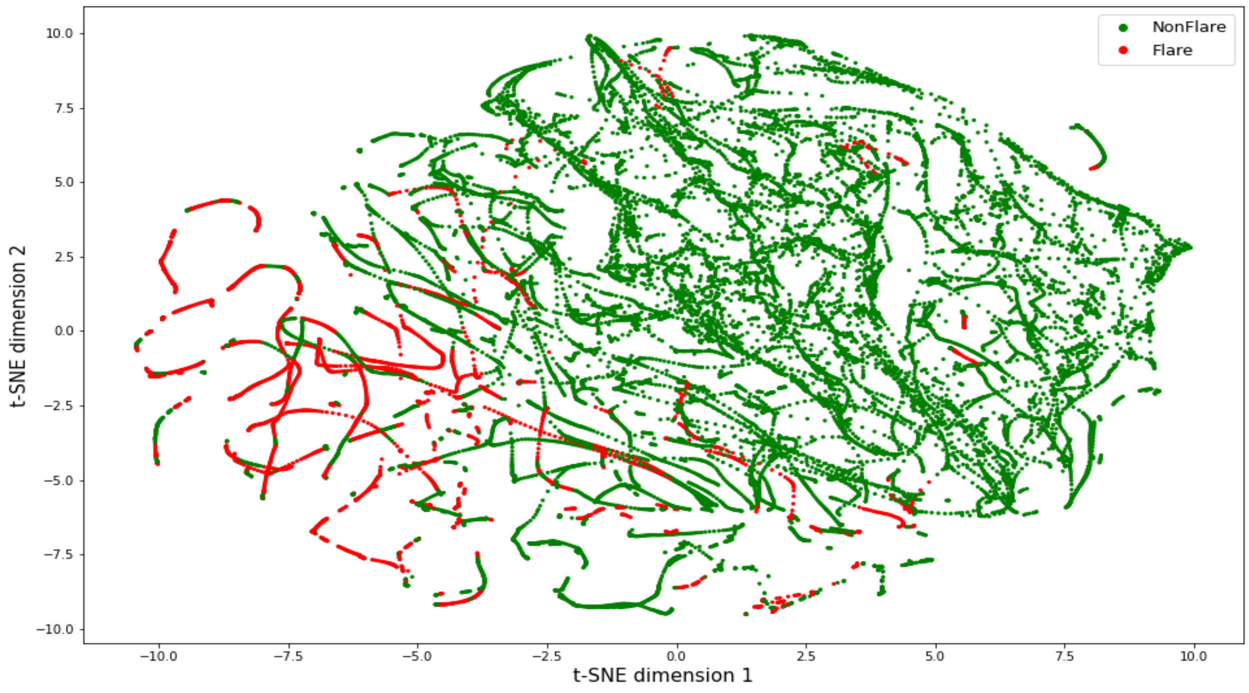


(b) t-SNE

Figure 6.22: Visualization of the last timestamp vectors of the flaring and nonflaring solar events

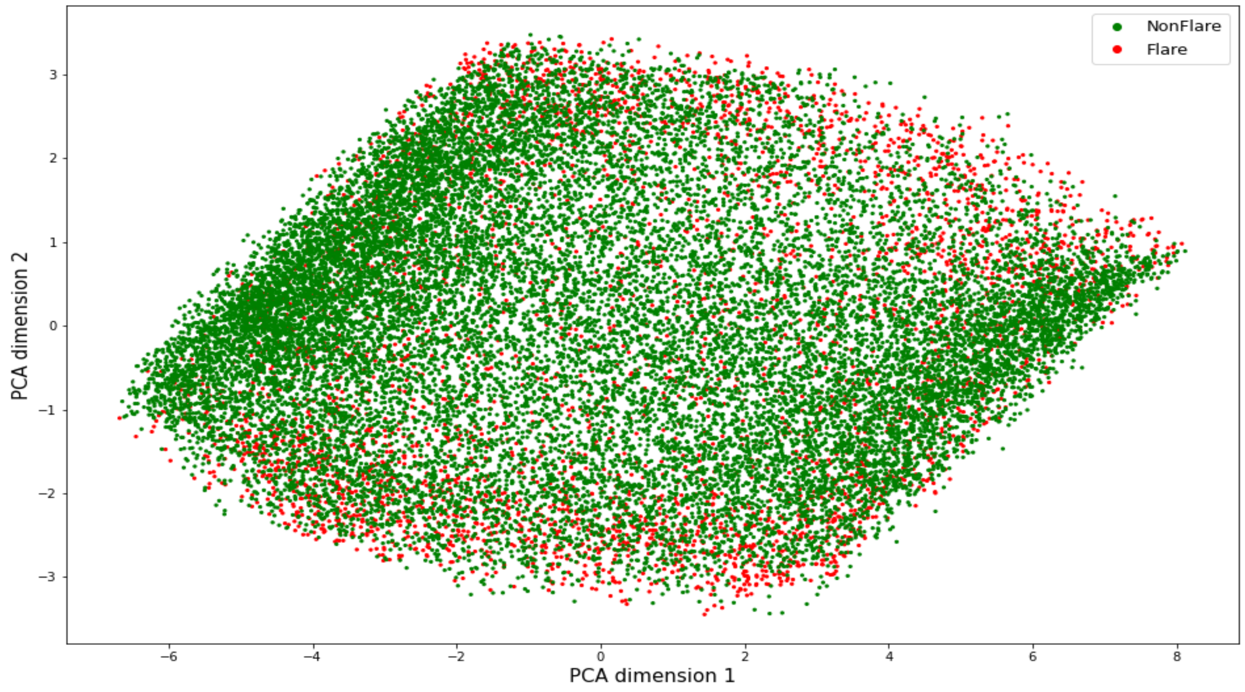


(a) PCA

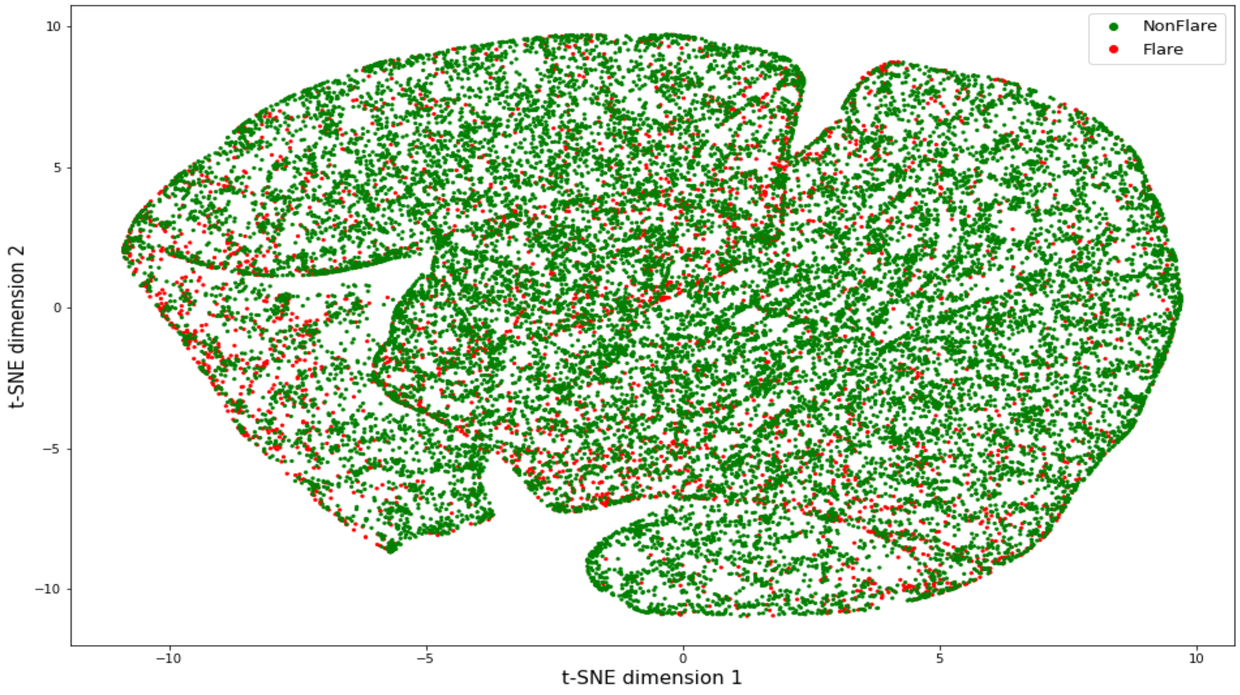


(b) t-SNE

Figure 6.23: Visualization of the mean magnetic field parameters of the flaring and nonflaring solar events



(a) PCA



(b) t-SNE

Figure 6.24: Visualization of the tensor decomposed event-timestamp-parameter projection vectors of the flaring and nonflaring solar events

tensor decomposition-based method (Fig. 6.24a and Fig. 6.24b) was performed without hyperparameter search, i.e., tuning of CP decomposition rank R, random initialization of the factor matrices, and number of iterations in the CP decomposition.

7 CONCLUSION

7.1 *Concluding Remarks*

Our prime contribution of this thesis is generating interpretable feature space leveraging tensor modeling of graph and time series data, decomposing the tensor into factor matrices, and projections of factor matrix of one entity type to another. We applied our algorithms on real world datasets of social networks, fMRI-based brain networks, and multivariate time series-based flaring and nonflaring solar events.

In the node embedding task, we presented two novel tensor decomposition- based node embedding algorithms TDNE and TDNEpS, which utilizes higher-order transition probability matrices of a graph (directed or undirected, weighted or unweighted) to construct one or more third-order tensor(s), and use CP decomposition to extract factor matrices containing the representations of the source and/or target properties of the nodes, and the transition steps. We have theoretically and experimentally shown that the node features produced by these algorithms are highly interpretable in terms of the understandability of the feature roles. Moreover, learned embeddings of the transition steps make them perform well in network reconstruction, link prediction, node classification, and graph classification.

In the application of the brain networks, we evaluated five different tensor models of fMRI data, and found the stacked non-negative functional connectivity matrices produce robust classification results. Later, we presented the method of finding discriminative patterns, that is, ROIs and connections from fMRI-based complete functional connectivity networks that can act as biomarkers for some neurological and neurodevelopmental diseases. Instead of representing the complete functional connectivity networks as thresholded graphs, and using computationally expensive frequent/discriminative subgraph mining algorithms, we represented each complete functional connectivity network by threshold-free connectivity vector and apply univariate/multivariate feature selection algorithms for finding the important features, which eventually became the edges of the discriminative subgraph. We used a tensor decomposition-based approach for finding the meaningful representations of the biomarkers and computing the biomarker impacts on the subjects.

Our third application is multivariate time series-based flaring and nonflaring solar events prediction and visualization. We showed a novel way of predicting that an Active Region of the Sun might lead to an X-class or M-class flare by leveraging the time series behavior of the magnetic field parameters. We presented a formal data model for representing flaring and non-flaring Active Regions using multivariate time series, where each time series, extracted before the lookback time of the occurrence of the event and collected throughout the span period, represents one AR parameter. We used k-NN classifier for classifying the summarized representations of the time series of the AR parameter total unsigned current helicity, and exhibit better performance than considering all parameters without time series. We also show the robustness of this representation using other classifiers. Finally, we demonstrated the visualization of multivaraite time series-based flaring and nonflaring solar events using classical dimensionality reduction techniques such as PCA and t-SNE on various data representations.

7.2 *Future Work*

We plan to extend our research on interpretable feature learning of graphs and time series data. In the following list, we specify these ideas.

1. Our presented node embedding algorithms TDNE and TDNEpS are transductive, i. e., all examples (training and test) are used in learning. When new nodes enter the network, the whole algorithm needs to be retrained. We look forward to designing an inductive algorithm, where the new nodes entering the network can use the parameters learned from the training nodes.

2. Reducing time and space complexity of the tensor decomposition-based node embedding algorithms is challenging. We look forward to reduce these complexities, and increase scalability of our models.

3. We aim to extend the node embedding algorithms for subgraph embedding. Interpretability in the multi-resolution graph embedding can be an interesting topic.

4. We look forward to work on solar event graphs. Considering the correlation matrices of the multivariate time series data of solar events as adjacency matrices of labeled graphs, and applying thresholds on edge weights can model the solar flare prediction problem as a graph classification problem. When the solar events are represented as graphs, classical graph classification algorithms such as gSpan-based [51] frequent subgraphs enumeration (as features) followed by classification, gBoost [91], and so on, and graph representation learning models such as Node2Vec [8], DeepWalk [28], and so on can be applied.

Bibliography

[1] HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Trans. Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

[2] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[3] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[4] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[6] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012.

[7] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[8] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 855–864. ACM, 2016.

[9] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pages 585–591. MIT Press, 2001.

[10] Quanquan Gu, Zhenhui Li, and Jiawei Han. Generalized fisher score for feature selection. *arXiv preprint arXiv:1202.3725*, 2012.

[11] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.

[12] Shah Muhammad Hamdi, Soukaina Filali Boubrahimi, and Rafal A. Angryk. Tensor decomposition-based node embedding. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, pages 2105–2108, 2019.

[13] Shah Muhammad Hamdi, Yubao Wu, Rafal A. Angryk, Lisa Crystal Krishnamurthy, and Robin Morris. Identification of discriminative subnetwork from fmri-based complete functional connectivity networks. *Intl. Journal of Semantic Computing*, 13(1):25–44, 2019.

[14] Soukaina Filali Boubrahimi, Ruizhe Ma, and Rafal A. Angryk. Neuro-ensemble for time series data classification. In *5th IEEE International Conference on Data Science and Advanced Analytics, DSAA 2018, Turin, Italy, October 1-3, 2018*, pages 50–59. IEEE, 2018.

[15] Arif Mahmood, Michael Small, Somaya Ali Al-Maadeed, and Nasir Rajpoot. Using geodesic space density gradients for network community detection. *IEEE Transactions on Knowledge and Data Engineering*, 29(4):921–935, 2016.

[16] Soukaina Filali Boubrahimi, Ruizhe Ma, Berkay Aydin, Shah Muhammad Hamdi, and Rafal A. Angryk. Scalable knn search approximation for time series data. In *24th International Conference on Pattern Recognition, ICPR 2018, Beijing, China, August 20-24, 2018*, pages 970–975. IEEE Computer Society, 2018.

[17] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[18] Soukaina Filali Boubrahimi and Rafal A. Angryk. Heuristics significance of neuro-ensemble-based time series classification. In *IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018*, pages 6–15. IEEE, 2018.

[19] Monica G Bobra and Sebastien Couvidat. Solar flare prediction using SDO/HMI vector magnetic field data with a machine-learning algorithm. *The Astrophysical Journal*, 798(2):135, 2015.

[20] Shah Muhammad Hamdi and Rafal A. Angryk. Interpretable feature learning of graphs using tensor decomposition. In *2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8-11, 2019*, pages 270–279. IEEE, 2019.

[21] Shah Muhammad Hamdi, Berkay Aydin, Soukaina Filali Boubrahimi, Rafal A. Angryk, Lisa Crystal Krishnamurthy, and Robin D. Morris. Biomarker detection from fmri-based complete functional connectivity networks. In *First IEEE International Conference on Artificial Intelligence and Knowledge Engineering, AIKE 2018, Laguna Hills, CA, USA, September 26-28, 2018*, pages 17–24. IEEE Computer Society, 2018.

[22] Shah Muhammad Hamdi, Yubao Wu, Soukaina Filali Boubrahimi, Rafal A. Angryk, Lisa Crystal Krishnamurthy, and Robin D. Morris. Tensor decomposition for neurodevelopmental disorder prediction. In *Brain Informatics - International Conference, BI 2018, Arlington, TX, USA, December 7-9, 2018, Proceedings*, volume 11309 of *Lecture Notes in Computer Science*, pages 339–348. Springer, 2018.

[23] Shah Muhammad Hamdi, Dustin Kempton, Ruizhe Ma, Soukaïna Filali Boubrahimi, and Rafal A Angryk. A time series classification-based approach for solar flare prediction. In *Big Data (Big Data), 2017 IEEE International Conference on*, pages 2543–2551. IEEE, 2017.

[24] Ruizhe Ma, Soukaina Filali Boubrahimi, Shah Muhammad Hamdi, and Rafal A. Angryk. Solar flare prediction using multivariate time series decision trees. In *2017 IEEE International Conference on Big Data, BigData 2017, Boston, MA, USA, December 11-14, 2017*, pages 2569–2578. IEEE Computer Society, 2017.

[25] Rafal A Angryk, Petrus C Martens, Berkay Aydin, Dustin Kempton, Sushant S Mahajan, Sunitha Basodi, Azim Ahmadzadeh, Soukaina Filali Boubrahimi, Shah Muhammad Hamdi, Michael A Schuh, et al. Multivariate time series dataset for space weather data analytics. 2019.

[26] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, pages 891–900. ACM, 2015.

[27] Ziwei Zhang, Peng Cui, Xiao Wang, Jian Pei, Xuanrong Yao, and Wenwu Zhu. Arbitrary-order proximity preserved network embedding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 2778–2786. ACM, 2018.

[28] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 701–710. ACM, 2014.

[29] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[30] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference*

on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, pages 1105–1114. ACM, 2016.

[31] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1225–1234. ACM, 2016.

[32] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 1145–1152. AAAI Press, 2016.

[33] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

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

[35] Nicholas D. Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E. Papalexakis, and Christos Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Trans. Signal Process.*, 65(13):3551–3582, 2017.

[36] Chong-Yaw Wee, Pew-Thian Yap, Daoqiang Zhang, Kevin Denny, Jeffrey N Browndyke, Guy G Potter, Kathleen A Welsh-Bohmer, Lihong Wang, and Dinggang Shen. Identification of MCI individuals using structural and functional connectivity networks. *Neuroimage*, 59(3):2045–2056, 2012.

[37] Bokai Cao, Liang Zhan, Xiangnan Kong, S Yu Philip, Nathalie Vizueta, Lori L Altshuler, and Alex D Leow. Identification of discriminative subgraph patterns in fmri brain networks in bipolar affective disorder. In *Int. Conf. on Brain Informatics and Health*, pages 105–114. Springer, 2015.

[38] Biao Jie, Daoqiang Zhang, Wei Gao, Qian Wang, Chong-Yaw Wee, and Dinggang Shen. Integration of network topological and connectivity properties for neuroimaging classification. *IEEE Trans. on Biomedical Engineering*, 2014.

[39] Anna Martin, Matthias Schurz, Martin Kronbichler, and Fabio Richlan. Reading in the brain of children and adults: A meta-analysis of 40 functional magnetic resonance imaging studies. *Human brain mapping*, 36(5):1963–1981, 2015.

[40] Bokai Cao, Xiangnan Kong, Jingyuan Zhang, Philip S. Yu, and Ann B. Ragin. Mining brain networks using multiple side views for neurological disorder identification. In *2015 IEEE Intl. Conf. on Data Mining, ICDM 2015, Atlantic City, NJ, USA, November 14-17, 2015*, pages 709–714, 2015.

[41] Xiangnan Kong and Philip S Yu. Semi-supervised feature selection for graph classification. In *Proc. of the 16th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining*, pages 793–802. ACM, 2010.

[42] Yunan Zhu and Ivor Cribben. Graphical models for functional connectivity networks: best methods and the autocorrelation issue. *bioRxiv*, page 128488, 2017.

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

[44] Bokai Cao, Chun-Ta Lu, Xiaokai Wei, S Yu Philip, and Alex D Leow. Semi-supervised tensor factorization for brain network analysis. In *Joint European Conf. on Machine Learning and Knowledge Discovery in Databases*, pages 17–32. Springer, 2016.

[45] Anna Huang. Similarity measures for text document clustering. In *Proc. of the 6th New Zealand Computer Science Research Student Conference (NZCSRSC2008), Christchurch, New Zealand*, pages 49–56, 2008.

[46] Vince D Calhoun, Paul K Maciejewski, Godfrey D Pearlson, and Kent A Kiehl. Temporal lobe and default hemodynamic brain modes discriminate between schizophrenia and bipolar disorder. *Human brain mapping*, 29(11):1265–1275, 2008.

[47] Anderson dos Santos Siqueira, Claudinei Eduardo Biazoli Junior, William Edgar Comfort, Luis Augusto Rohde, and João Ricardo Sato. Abnormal functional resting-state networks in adhd: graph theory and pattern recognition analysis of fmri data. *BioMed Research Int.*, 2014, 2014.

[48] Lucina Q Uddin, Kaustubh Supekar, Charles J Lynch, Amirah Khouzam, Jennifer Phillips, Carl Feinstein, Srikanth Ryali, and Vinod Menon. Salience network–based classification and prediction of symptom severity in children with autism. *JAMA psychiatry*, 70(8):869–879, 2013.

[49] Emily S Finn, Xilin Shen, John M Holahan, Dustin Scheinost, Cheryl Lacadie, Xenophon Papademetris, Sally E Shaywitz, Bennett A Shaywitz, and R Todd Constable. Disruption of functional networks in dyslexia: a whole-brain, data-driven analysis of connectivity. *Biological psychiatry*, 76(5):397–404, 2014.

[50] Xiangnan Kong, Philip S Yu, Xue Wang, and Ann B Ragin. Discriminative feature selection for uncertain graph classification. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 82–93. SIAM, 2013.

[51] Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 721–724. IEEE, 2002.

[52] Bokai Cao, Lifang He, Xiangnan Kong, S Yu Philip, Zhifeng Hao, and Ann B Ragin. Tensor-based multi-view feature selection with applications to brain diseases. In *Data Mining (ICDM), 2014 IEEE Int. Conf. on*. IEEE, 2014.

[53] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep):2539–2561, 2011.

[54] Giorgio Roffo, Simone Melzi, Umberto Castellani, and Alessandro Vinciarelli. Infinite latent feature selection: A probabilistic latent graph-based ranking approach. In *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[55] Ahmet Küçük, Shah Muhammad Hamdi, Berkay Aydin, Michael A. Schuh, and Rafal A. Angryk. Pg-trajectory: A postgresql/postgis based data model for spatiotemporal trajectories. In *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom), BDCloud-SocialCom-SustainCom 2016, Atlanta, GA, USA, October 8-10, 2016*, pages 81–88. IEEE Computer Society, 2016.

[56] Shah Muhammad Hamdi, Berkay Aydin, and Rafal A. Angryk. A pattern growth-based approach for mining spatiotemporal co-occurrence patterns. In *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain*, pages 1125–1132. IEEE Computer Society, 2016.

[57] JP Eastwood, E Biffis, MA Hapgood, L Green, MM Bisi, RD Bentley, R Wicks, L-A McKinnell, M Gibbs, and C Burnett. The economic impact of space weather: Where do we stand? *Risk Analysis*, 37(2):206–218, 2017.

[58] Solar flare captured by SDO. `https://www.nasa.gov/content/goddard/nasa-releases-images-of-1st-notable-solar-flare-of-2015`. [Online; accessed 30-may-2017].

[59] KD Leka and G Barnes. Photospheric magnetic field properties of flaring versus flare-quiet active regions. ii. discriminant analysis. *The Astrophysical Journal*, 595(2):1296, 2003.

[60] N Nishizuka, K Sugiura, Y Kubo, M Den, S Watari, and M Ishii. Solar flare prediction model with three machine-learning algorithms using ultraviolet brightening and vector magnetograms. *The Astrophysical Journal*, 835(2):156, 2017.

[61] Kyoung-jae Kim. Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1):307–319, 2003.

[62] Toni M Rath and Raghavan Manmatha. Word image matching using dynamic time warping. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE, 2003.

[63] Patrick S McIntosh. The classification of sunspot groups. *Solar Physics*, 125(2):251–267, 1990.

[64] James P Mason and JT Hoeksema. Testing automated solar flare forecasting with 13 years of michelson doppler imager magnetograms. *The Astrophysical Journal*, 723(1):634, 2010.

[65] Yanmei Cui, Rong Li, Liyun Zhang, Yulin He, and Huaning Wang. Correlation between solar flare productivity and photospheric magnetic field properties. *Solar Physics*, 237(1):45–59, 2006.

[66] Ju Jing, Hui Song, Valentyna Abramenko, Changyi Tan, and Haimin Wang. The statistical relationship between the photospheric magnetic parameters and the flare productivity of active regions. *The Astrophysical Journal*, 644(2):1273, 2006.

[67] Omar W Ahmed, Rami Qahwaji, Tufan Colak, Paul A Higgins, Peter T Gallagher, and D Shaun Bloomfield. Solar flare prediction using advanced feature extraction, machine learning, and feature selection. *Solar Physics*, pages 1–19, 2013.

[68] Daren Yu, Xin Huang, Huaning Wang, and Yanmei Cui. Short-term solar flare prediction using a sequential supervised learning method. *Solar Physics*, 255(1):91–105, 2009.

[69] Hui Song, Changyi Tan, Ju Jing, Haimin Wang, Vasyl Yurchyshyn, and Valentyna Abramenko. Statistical assessment of photospheric magnetic features in imminent solar flare predictions. *Solar Physics*, 254(1):101–125, 2009.

[70] Amani Al-Ghraibah, LE Boucheron, and RTJ McAteer. An automated classification approach to ranking photospheric proxies of magnetic energy build-up. *Astronomy & Astrophysics*, 579:A64, 2015.

[71] R Qahwaji and Tufan Colak. Automatic short-term solar flare prediction using machine learning and sunspot associations. *Solar Physics*, 241(1):195–211, 2007.

[72] Alex Nanopoulos, Rob Alcock, and Yannis Manolopoulos. Feature-based classification of time-series data. *International Journal of Computer Research*, 10(3):49–61, 2001.

[73] Eamonn J. Keogh. A decade of progress in indexing and mining large time series databases. In *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, page 1268, 2006.

[74] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[75] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques, 3rd edition*. Morgan Kaufmann, 2011.

[76] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowl. Based Syst.*, 151:78–94, 2018.

[77] Jean Kossaifi, Yannis Panagakis, Anima Anandkumar, and Maja Pantic. Tensorly: Tensor learning in python. *Journal of Machine Learning Research*, 20(26):1–6, 2019.

[78] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4):452–473, 1977.

[79] Leonardo Filipe Rodrigues Ribeiro, Pedro H. P. Saverese, and Daniel R. Figueiredo. *struc2vec*: Learning node representations from structural identity. In *Proc. of the 23rd ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pages 385–394, 2017.

[80] John Boaz Lee, Xiangnan Kong, Yihan Bao, and Constance M. Moore. Identifying deep contrasting networks from time series data: Application to brain network analysis. In *Proc. of the 2017 SIAM Intl. Conf. on Data Mining, Houston, Texas, USA, April 27-29, 2017.*, pages 543–551, 2017.

[81] Susan Whitfield-Gabrieli and Alfonso Nieto-Castanon. Conn: a functional connectivity toolbox for correlated and anticorrelated brain networks. *Brain Connectivity*, 2(3):125–141, 2012.

[82] Brett W. Bader, Tamara G. Kolda, et al. Matlab tensor toolbox version 2.6. Available online, February 2015.

[83] Venkatagiri Krishnamurthy, Lisa C Krishnamurthy, Dina M Schwam, Ashley Ealey, Jaemin Shin, Daphne Greenberg, and Robin D Morris. Retrospective correction of physiological noise: impact on sensitivity, specificity, and reproducibility of resting-state functional connectivity in a reading network model. *Brain connectivity*, (ja), 2017.

[84] Robert W Cox. Afni: software for analysis and visualization of functional magnetic resonance neuroimages. *Computers and Biomedical Research*, 29(3):162–173, 1996.

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

[86] Bruce Fischl. Freesurfer. *Neuroimage*, 62(2):774–781, 2012.

[87] Joint Science Operations Center (JSOC). `http://jsoc.stanford.edu/`. [Online; accessed 01-October-2017].

[88] G Barnes and KD Leka. Evaluating the performance of solar flare forecasting methods. *The Astrophysical Journal Letters*, 688(2):L107, 2008.

[89] D Shaun Bloomfield, Paul A Higgins, RT James McAteer, and Peter T Gallagher. Toward reliable benchmarking of solar flare forecasting methods. *The Astrophysical Journal Letters*, 747(2):L41, 2012.

[90] BigData Cup Challenge 2019: Flare Prediction. `http://www.kaggle.com/c/bigdata2019-flare-prediction`. [Online; accessed 14-June-2020].

[91] Hiroto Saigo, Sebastian Nowozin, Tadashi Kadowaki, Taku Kudo, and Koji Tsuda. gboost: a mathematical programming approach to graph classification and regression. *Mach. Learn.*, 75(1):69–89, 2009.