University of California
Santa Barbara

# Leveraging Network Information for Data-Driven Scientific Discovery

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Computer Science

by

Hongyuan You

Committee in charge:

Professor Ambuj Singh, Chair
Professor Yu-Xiang Wang
Professor Sang-Yun Oh
Professor Tommy Sprague

June 2021

The Dissertation of Hongyuan You is approved.

Professor Yu-Xiang Wang

Professor Sang-Yun Oh

Professor Tommy Sprague

Professor Ambuj Singh, Committee Chair

May 2021

Leveraging Network Information for Data-Driven Scientific Discovery

To my beloved parents.

## Acknowledgements

I would like to thank my advisor Professor Ambuj K. Singh for supporting my research and allowing me to learn so much as part of his group. Thanks also to Professor Yu-Xiang Wang, Professor Sang-Yun Oh and Professor Tommy Sprague for being part of my committee, contributing to the improvement of this work and providing valuable mentorships.

During my PhD studies I was fortunate to have collaborated with Scott Grafton, Xuan-Hong Dang, Petko Bogdanov, Bayyuan Hsu, Sikun Lin and Furkan Kocayusufogl. I also had a great group of labmates: Arlei, Sourav, Haraldur, Alex, Rachel, Omid, Victor, Yuanshun, Minh, Anh, Nazli, Yuning, Amy and Wei. Thanks to Greta, Tim, and Karen for their support as UCSB CS staff. My appreciation also goes out to my family and friends for their encouragement and support all through my studies.

## Curriculum Vitæ

Hongyuan You

### Education

| | |
|---|---|
| 2021 | Ph.D. in Computer Science (Expected), University of California, Santa Barbara. |
| 2013 | B.S. in Physics, Peking University, Beijing. |

### Publications

**Hongyuan You**, Sikun Lin and Ambuj K. Singh, *"Learning Interpretable Models for Coupled Networks Under Domain Constraints"*, accepted in AAAI Conference on Artificial Intelligence (AAAI), 2021.

**Hongyuan You**, Furkan Kocayusufoglu and Ambuj K. Singh, *"DANR: Discrepancy-aware Network Regularization"*, in SIAM International Conference on Data Mining (SDM), 2020.

**Hongyuan You**, Adam Liska, Vinay Shahidhar and Payel Das, *"The Underlying Brain Functional Landscape of an Individual as revealed by Graph Embedding"*, in NIPS BigNeuro workshop, Long Beach, California, USA, 2017.

**Hongyuan You**, Adam Liska, Nathan Russell and Payel Das, *"Automated Brain State Identification Using Graph Embedding Techniques"* in Pattern Recognition in NeuroImaging (PRNI), Toronto, Canada, 2017.

Adam Liska, **Hongyuan You** and Payel Das, *"Relationship between static and dynamic brain functional connectivity in autism spectrum disorders"* in International Society for Magnetic Resonance in Medicine (ISMRM), Honolulu, Hawaii, USA, 2017.

Xuan-Hong Dang, **Hongyuan You**, Ambuj K. Singh and Scott Grafton, *"Subnetwork Mining with Spatial and Temporal Smoothness"* in SIAM International Conference on Data Mining (SDM), Hoston, Texas, USA, 2017.

Xuan-Hong Dang, **Hongyuan You**, Petko Bogdanov and Ambuj K. Singh, *"Learning Predictive Substructures with Regularization for Network Data"* in International Conference on Data Mining (ICDM), Atlantic City, New Jersey, USA, 2015.

Xuan-Hong Dang, Petko Bogdanov, **Hongyuan You**, Bayyuan Hsu and Ambuj K. Singh, *"Discriminative Subnetworks with Regularized Spectral Learning for Global-state Network Data"*, in European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD), Nancy, France, 2014.

## Abstract

Leveraging Network Information for Data-Driven Scientific Discovery

by

Hongyuan You

Network is a popular format for encoding structured information in applications ranging from spatial economics to neuroimaging studies. Discovering features of local processes and structures plays a key role in understanding and interpreting the overall state of complex networks. For example, the absence or inhibition of interaction in the protein-protein network impacts the expression levels of protein pathways, which determines the presence or absence of disease; the existence of structural network fragments is significant for functional behavior in the neural system.

In this thesis, we will show that, through various regularization approaches, we can discover local substructures that affect global states or properties of network instances, or efficiently learn coherent models over networks that are robust to missing or corrupted edge weights. Second, with increases in both the amount and the modalities of neuroimaging data, there is a need for models that integrate diverse functional and structural data and that can identify plausible patterns in the complex brain architecture. We will discuss how to model both the structure and function of brain connectivities, while we place hard network constraints driven by prior knowledge and model assumptions.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Network data arises in a number of application domains ranging from Internet of Things (IoT), cloud computing, software vulnerability analysis, neuroscience, biology, geography, to social sciences [3, 4, 5, 6]. Accordingly, network analysis has emerged as a major paradigm for exploring complex processes behind observed data [7, 8, 9, 10]. Compared to high dimensional data, analysis over network data is more challenging due to the nature of inter-dependence among the entities.

In the first three sections of this thesis, we investigate the essential relationship between local network structure and global network behaviors. We consider multiple network samples, each of which is associated with a global label that reflects its current state (e.g., disease stage of the underlying brain network, drought stage of the underlying environmental network). To discover qualified sets of sub-structures, recent works on supervised subnetwork mining aims to efficiently search discriminative subnetworks for classification with respect to user-specified objective functions rather than frequency criterion. Most of the frequency-based subnetwork mining algorithms [**?**, **?**] are unsupervised and apply variations of branch-and-bound searching strategy which help avoid the in-feasibility of enumerating the full candidate set. The bottleneck of these methods lies on its redundancy of discovered subnetwork patterns without quality guarantee, and on the exponentially increasing number of subnetworks when low-frequency threshold has to be set to avoid loss of key information. For user-specified criteria such as prediction accuracy, subnetwork learning methods usually [11, 12] conduct a network-guided heuristic search or applies a rejection sampling on edit map representing the extremely large combinatorial solution space, so that subnetworks in the answer set have better classification performance with locally built network-constrained decision tree. All these search or sampling approaches make the assumption of binary interactions, although network edges are usually real-valued with the meaning of connectivity strength or correlation coefficient. Another promising approach [13, 14] is to develop shrinkage and selection methods in the regularization perspective. By penalizing local variation of coefficient vector along network edges, neighboring features on the network are encouraged to achieve similar estimation of their coefficients. We extend this approach in various real-world scenarios, and develop customized algorithms that select optimal discriminative subnetwork features in a controlled-size without irrelevant and redundant nodes, given no assumption about the statistical distribution of network data. One case is that both local network interactions and the global network state can evolve over time (Section 4). The task is to uncover a small set of local network processes that have maximum impact on the global network states such that their evolution can be used to predict the transition of the global network states. In order to support heterogeneity, we develop algorithms that decompose a set of networks each of which has a marked set of vertices into a set of common patterns that can together express the behavior of each network. The research work in this thread is related to feature selection methods. However, these ignore the interactions among network entities. Graph classification approaches do not consider temporal smoothness.

The tool that we used in above problems, network-based regularization, covers a series of general optimization problems including convex clustering [15], fused lasso [16], network-enhanced classification, and total-variation regularization [17]. Recent works [18] prefer formulating it as a large-scale convex optimization problem, which is solved in a distributed and scalable manner via alternating direction method of multipliers (ADMM) [19]. This makes it possible to perform each update independently per node or edge, with guaranteed convergence and global optimality. We provide a generalized framework of network-based regularization on dynamic networks with additional and adjustable penalties on spatial and temporal smoothness. Our formulation introduces more flexibility by allowing heterogeneous variations on every edge and between different snapshots. By incorporating the ADMM method into alternating updating steps, our convex formulation achieves global convergence and optimality, and scales well to large datasets.

Another major topic of thesis is applying network-based methods on neuroimaging data. As the amount and sources of neuroimaging data about nervous system processes increases, these new imaging data sets are often complex and difficult to analyze, and usually not reducible, thus requiring novel analysis approaches utilizing a network-based view [20, 21, 22, 23, 24, 25, 26]. Consider brain networks associated with Alzheimer's disease (AD) [27, 28, 29] as a specific case of the aforementioned local-global problem. At an early stage, the disease can be as mild as causing the patient difficulty in remembering recent events; however, as the disease progresses, disease dynamics results in more brain regions being impacted. Detecting subnetwork markers in brain connectivity that predict and evolve along with the progression of the disease is thus an important task since it not only helps to characterize the disease but further provides the means to plan the right treatments at the right stage of the disease.

While the majority of network-based studies for fMRI have focused on the brain's default mode or "resting" state [30], more recent efforts have turned to understanding brain connectivity elicited by task demands, including visual processing [21, 31], memory [32], and learning [20]. Therefore there is a need for models that integrate diverse functional and structural neuroimaging data and that can identify plausible patterns in the complex architecture that is the human brain. Global network analysis of both functional and structural connectivity has demonstrated that brain networks have characteristic topological properties, including dense modular structures and efficient long-distance paths [31, 33]. Relating functional connectivity networks from multiple tasks with an underlying structural connectivity network opens up the opportunity of examining the structure-function relationship from the perspective of different tasks and cognitive states. Existing approaches for exploring this structure-function relationship commonly compute statistics between the underlying structures and the functional activation signals. These studies show that highly structurally connected regions have higher functional correlations, but not vice versa. Others examine the possibility of predicting a functional connectivity network from structural networks: Goni et al. [34] adopt numerical simulation of fMRI activity on adjusted structural networks by a “neural mass” model. Becker et al. [35] seek to approximate a functional connectivity network, as modeled by an adjacency matrix, with a weighted sum of powers of the structural connectivity matrix, with a rotation matrix to align eigenvectors to the target functional matrix. It has also been shown that jointly modeling structural and functional connectivity improves the classification of schizophrenia [36].

## 1.1 Contributions and Organizations

- **Regularized Spectral Learning for Global-state Network Data:** Data mining practitioners are facing challenges from data with network structure. In this work, we address a specific class of *global-state* networks comprising of a set of network instances sharing a similar structure yet having different values at local nodes. Each instance is associated with a global network state which indicates the occurrence of an event. The objective is to uncover a small set of discriminative subnetworks that can optimally classify global network values. Unlike most existing studies which explore an exponential subnetwork space, we address this difficult problem by adopting a space transformation approach. Specifically, we present an algorithm that optimizes a constrained dual-objective function to learn a low-dimensional subspace that is capable of discriminating networks labelled by different global states, while reconciling with common network topology sharing across instances. Our algorithm takes an efficiency-appealing approach from spectral graph learning and we show that globally optimum solutions can be achieved via matrix eigen-decomposition.

- **Learning Predictive Substructures for Network Data:** Learning a succinct set of substructures that predicts global network properties plays a key role in understanding complex network data. Existing approaches address this problem by sampling the exponential space of all possible subnetworks to find ones of high prediction accuracy. In this work, we develop a novel framework that avoids sampling by formulating the problem of predictive subnetwork learning as node selection, subject to network-constrained regularization. Our framework involves two steps: (i) subspace learning, and (ii) predictive substructures discovery with network regularization. The framework is developed based upon two mathematically sound techniques of spectral graph learning and gradient descent optimization, and we show that their solutions converge to a global optimum solution—a desired property that cannot be guaranteed by sampling approaches. Through experimental analysis on a number of real world datasets, we demonstrate the performance of our framework against state-of-the-art algorithms, not only based on prediction accuracy but also in terms of domain relevance of the discovered substructures.

- **Subnetwork Mining with Spatial and Temporal Smoothness:** In many real-world applications, data is represented in the form of networks with structures and attributes changing over time. The dynamic changes not only happen at nodes/edges, forming local subnetwork processes, but also eventually influence global states of networks. The need to understand what these local network processes are, how they evolve and consequently govern the progression of global network states has become increasingly important. In this work, we explore these questions and develop a novel algorithm for mining a succinct set of subnetworks that are predictive and evolve along with the progression of global network states. Our algorithm is designed in the framework of logistic regression that fits a model for multi-states of network samples. Its objective function considers both the spatial network topology and temporal smooth transition between adjacent global network states, and we show that its global optimum solution can be achieved via steepest descent. Extensive experimental analysis on both synthetic and real world datasets demonstrates the effectiveness of our algorithm against competing methods, not only in the prediction accuracy but also in terms of domain relevance of the discovered subnetworks.

- **Discrepancy-aware Network Regularization:** Network regularization is an effective tool for incorporating structural prior knowledge to learn coherent models over networks, and has yielded provably accurate estimates in applications ranging from spatial economics to neuroimaging studies. Recently, there has been an increasing interest in extending network regularization to the spatio-temporal case to accommodate the evolution of networks. However, in both static and spatio-temporal cases, missing or corrupted edge weights can compromise the ability of network regularization to discover desired solutions. To address these gaps, we propose a novel approach—*discrepancy-aware network regularization* (DANR)—that is robust to inadequate regularizations and effectively captures model evolution and structural changes over spatio-temporal networks. We develop a distributed and scalable algorithm based on alternating direction method of multipliers (ADMM) to solve the proposed problem with guaranteed convergence to global optimum solutions. Experimental results on both synthetic and real-world networks demonstrate that our approach achieves improved performance on various tasks, and enables interpretation of model changes in evolving networks.

- **Reconstructing Coupled Networks Under Domain Constraints:** Modeling the behavior of coupled networks is challenging due to their intricate dynamics. For example in neuroscience, it is of critical importance to understand the relationship between the functional neural processes and anatomical connectivities. Modern neuroimaging techniques allow us to separately measure functional connectivity through fMRI imaging and the underlying white matter wiring through diffusion imaging. Previous studies have shown that structural edges in brain networks improve the inference of functional edges and vice versa. In this work, we investigate the idea of coupled networks through an optimization framework by focusing on interactions between structural edges and functional edges of brain networks. We consider both types of edges as observed instances of random variables that represent different underlying network processes. The proposed framework does not depend on Gaussian assumptions and achieves a more robust performance on general data compared with existing approaches. To incorporate existing domain knowledge into such studies, we propose a novel formulation to place hard network constraints on the noise term while estimating interactions. This not only leads to a cleaner way of applying network constraints but also provides a more scalable solution when network connectivity is sparse. We validate our method on multishell diffusion and task-evoked fMRI datasets from the Human Connectome Project, leading to both important insights on structural backbones that support various types of task activities as well as general solutions to the study of coupled networks.

# Chapter 2

# Regularized Spectral Learning for Global-state Network Data

## 2.1 Introduction

With the increasing advances in hardware and software technologies for data collection and management, practitioners in data mining are now confronted with more challenges from the collected datasets: the data are no longer as simple as objects with flattened representation but now embedded with relationships among variables describing the objects. This sort of data is often referred to as *network* or *graph* data. In the literature, there are a large number of techniques developed to mine useful patterns from network databases, ranging from frequent (sub)networks mining [37], network classification/clustering [38, 39] to anomaly detection [40]. Often, even for the same data mining task, we may need different algorithms to be developed depending on whether the networks are *directed* or *indirected*, or whether the data resides at nodes, edges or both of them [37].

In this work, the focus is on a specific class of interesting networks in which we have a series of network instances that share a common structure but may have different dynamic values on local nodes and/or edges. In addition, each network instance is associated with a global state indicating the occurrence of an event. Such a class of *global-state* network data can be used to model a number of real-world applications ranging from opinion evolution in social networks [41], regulatory networks in biology [42] to brain networks in neuroscience [43]. For example, we possess the same set of genes (nodes) embedded in regulatory networks. Yet, research in systems biology shows that the gene expression levels (node values) may vary across individuals and for some specific genes, their over-expressions may impact those in the neighbors through the regulatory network. These local effects may jointly encode a logical function that determines the occurrence of a disease [42, 44]. In analyzing these types of network data, a natural question to be asked is how one can learn a function that can determine the global-state values of the networks based on the dynamic values captured at their local nodes along with the network topology? More specifically, is it possible to identify a small succinct set of influential discriminative subnetworks whose local-node values have the maximum impact on the global states and thus uncover the complex relationships between local entities and the global-state network properties? In searching for an answer, obviously, a naive approach would enumerate all possible subnetworks and seek those who have the most discriminative potential. Nonetheless, as the number of subnetworks is *exponentially* proportional to the numbers of nodes and edges, this approach generally is analytically intractable and might not be feasible for large scale networks. A more practical approach is to perform heuristic sampling from the space of subnetworks. Though greatly reducing the number of subnetworks to be visited, the sampling approaches might still suffer from suboptimal solutions and might further lose explanation capability due to the large number of generating subnetworks.

In this work, we propose a novel algorithm for mining a set of concise subnetworks whose local-state node values discriminate networks of different global-state values. Unlike the existing techniques that directly search through the exponential space of subnetworks, our proposed method is fundamentally different by investigating the discriminative subnetworks in a low dimensional transformed subspace. Toward this goal, we construct on top of the network database three meta-graphs to learn the network neighboring relationships. The first meta-graph is built to capture the network topology sharing across network instances which serves as the network constraint in our subspace learning function. The two subsequent meta-graphs essentially capture the relationships between neighboring networks, especially those located close to the potential discriminative boundary. In this setting, our algorithm aims to discover a unique low dimensional subspace to which: i) networks sharing similar global state values are mapped close to each other while those having different global values are mapped far apart; ii) the common network topology is smoothly preserved through constraints on the learning process. In this way, our algorithm helps to attack two challenging issues at the same time. It first avoids searching through the original space of exponential number of subnetworks by learning a single subspace via the optimization of a single dual-objective function. Second, our network topology constraint not only matches properly with our subspace learning function, its quadratic form naturally imposes the $L_2$-norm shrinkage over the connecting nodes, resulting in an effective selection of relevant and dominated nodes for the subnetworks embedded in the induced subspace. Additionally, the principal technical contributions of our work is the formulation of our learning objective function that is mathematically founded on spectral learning and its advantages therefore not only ensure the stability but also the global optimum of the uncovered solutions.

In summary, we claim the following contributions: (i) *Novelty:* We formulate the problem of mining discriminative subnetworks by transformed subspace learning—an approach that is fundamentally different from most existing techniques that address the problem in the original high-dimensional network space. (ii) *Flexibility:* We propose a novel dual-objective function along with constraints to ensure learning of a single subspace in which different global state networks are well discriminated while smoothly retaining their common topology. (iii) *Optimality:* We develop a mathematically sound solution to solve the constrained optimization problem and show that the optimal solution can be achieved using matrix eigen-decomposition. (iv) *Practical relevance:* We evaluate the performance of the proposed technique on both synthetic and real world datasets and demonstrate its appealing performance against related techniques in the literature.

## 2.2 Preliminaries and Problem Setting

In this section, we first introduce some preliminaries related to network data with global state values and then give the definition of our problem on mining discriminative subgraphs to distinguish global state networks.

*(Network data instance)* Given $V_i = \{v_1, v_2, \ldots, v_{n_i}\}$ as a set of nodes and $E_i \subseteq V_i \times V_i$ as a set of edges, each connecting two nodes $(v_p, v_q)$ if they are known to relate or influence each other, we define a network instance (or snapshot) $N_i$ as a quadruple $N_i = (V_i, E_i, L_i, S_i)$ in which $L_i$ is a function operating on the local states of nodes $L_i : V_i \to \mathbb{R}$ and $S_i$ encodes the global network state of $N_i$.

In this work, we consider $N_i$ as an indirected network and values at its local nodes are numerical (both continuous and binary) while its global state is a discrete value. Since each $N_i$ is associated with $S_i$ as its state property, $N_i$ is often referred to as a *global-state* network. For example, in the gene expression data, each $N_i$ corresponds to a subject and a local state indicates the gene expression level at node $v_p \in V_i$ whereas the global state encodes the presence or absence of the disease, i.e., $S_i \in \{presence, absence\}$. Likewise in a dynamic social network, a value at each node $v_p$ may encode the political standpoint of an individual whereas the global state indicates the overall political viewpoint of the entire community at some specific time (snapshot). Both local and global states may change across different network snapshots.

For network instances/snapshots with different structures, we may use the null value to denote the state of a missing node and consequently, an edge in a network instance is valid only if it connects two non-null nodes. Now, let us consider a database consisting $m$ network instances $\mathbb{N} = \{N_1, N_2, \ldots, N_m\}$, we further define the following network over these network instances:

*(Generalized network - first meta-graph)*

We define the generalized network $N$ as a triple $N = (V, E, K)$ where $V = V_1 \cup V_2 \ldots \cup V_m$ and if $\exists (v_p, v_q) \in E_i$, such an edge also exists in $E$. For a valid edge $E(p, q) \in E$, we associate a weight $K(p, q)$ as the fraction of network instances having edge $E(p, q)$ in their topology structure,i.e., $K(p, q) = m^{-1} \times \sum_i E_i(p, q)$ with $E_i(p, q) = 1$ if there exists an edge between $v_p$, $v_q$ in network $N_i$. As such, $K(p, q)$ is naturally normalized between $(0, 1]$. The value of 1 means the corresponding edge exists in all $N_i$'s while a value close to 0 shows that the edge only exists in a small fraction of network data.

It should be noted here that while we have no edge values at each individual network $N_i$, we have non-zero value associated with each existing edge $E(p, q)$ in the generalized network $N$. Indeed, $K(p, q)$ reflects how frequently there is an edge between $v_p$ and $v_q$ or equivalently, how strongly is the mutual influence between two entities $v_p$ and $v_q$ across all networks. As $N$ is defined based on all network instances, we also view $N$ as our first meta-graph with $V$ being its vertices and $K$ capturing its graph topology generalized from the network topology of all network instances. We are now ready to define our problem as follows.

*Problem Definition*

Given a database of network data instances/snapshots $\mathbb{N} = \{N_1, N_2, \ldots, N_m\}$, we aim to learn an optimal and succinct set of subnetworks with respect to the topology structure generalized in the first meta-graph that well discriminate network instances with different global state values.

## 2.3 Spectral Solution for Constrained Dual-Objective Formulation

### 2.3.1 Meta-Graphs over Network Instances

As mentioned in the above sections, searching for optimal subnetworks in the fully high dimensional original network space is always challenging and potentially intractable. We adopt an indirect yet more viable approach by transforming the original space into a low dimensional space of which networks with different global-states are well distinguished while concurrently retaining the generalized network topology captured by our first meta-graph. Toward this goal, we develop two neighboring *meta-graphs* based on both the local state values and global state values.

We denote these two meta-graphs respectively by $G^+$ and $G^-$. Their vertices correspond to the network instances while a link connecting two vertices represents the neighboring relationship between two corresponding network instances. For the meta-graph $G^+$, we denote $\mathbf{A}^+$ as its affinity matrix that captures the similarity of neighboring networks having the same global state values. Likewise, we denote $\mathbf{A}^-$ as the affinity matrix for meta-graph $G^-$ that captures the similarity of neighboring networks yet having different global network states. As such, $\mathbf{A}^+$ and $\mathbf{A}^-$ respectively encode the weights on the vertex-links of two corresponding graphs $G^+$ and $G^-$. In computing values for these affinity matrices, with each given network instance $N_i$, we find its $k$ nearest neighboring networks based on the local state values and divide them into two sets, those sharing similar global state values and those having different global states. More specifically, let $k\text{NN}(N_i)$ be the neighboring set of $N_i$, then elements of $\mathbf{A}^+$ and $\mathbf{A}^-$ are computed as: $\mathbf{A}^+_{ij} = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|}$ if $S_i = S_j$ and $N_j \in k\text{NN}(N_i)$ or $N_i \in k\text{NN}(N_j)$, otherwise we set $\mathbf{A}^+_{ij} = 0$. And $\mathbf{A}^-_{ij} = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|}$ if $S_i \neq S_j$ and $N_j \in k\text{NN}(N_i)$ or $N_i \in k\text{NN}(N_j)$, otherwise $\mathbf{A}^-_{ij} = 0$.

In these equations, we have denoted the boldface letters $\mathbf{v}_i$ and $\mathbf{v}_j$ as the vectors encoding the dynamic local states of $N_i$'s and $N_j$'s nodes, and have used the cosine distance to define the similarity between two network instances. It is worth mentioning that, though existing other measures for network data [45], our using of cosine distance is motivated by the observation that we can view each node as a single feature and thus the network data can be essentially considered as a special case of very high dimensional data. As such, the symmetric and nonnegative cosine measure can be effectively used though obviously the other ones [45] can also be directly applied here.

It is also important to give the intuition behind our above computation. First, notice that both $\mathbf{A}^+$ and $\mathbf{A}^-$ are the affinity matrices having the same size of $m \times m$ since we calculate for every network instance. Second, while $\mathbf{A}^+$ captures the similarity of network instances sharing the same global states and neighboring to each other, $\mathbf{A}^-$ encodes the similarity of different global state networks yet also neighboring to each other. Such networks are likely to locate close to the discriminative boundary function and thus they play essential roles in our subsequent learning function. Third, both $\mathbf{A}^+$ and $\mathbf{A}^-$ are sparse and symmetric matrices since only $k$ neighbors are involved in computing for each network and if $N_j$ is neighboring to $N_i$, we also consider the inverse relation, i.e., $N_i$ is neighboring to $N_j$. Moreover, $\mathbf{A}^-$ is generally sparser compared to $\mathbf{A}^+$ as the immediate observation from the second remark.

### 2.3.2 Constrained Dual-Objective Function

Let us recall that $\mathbf{v}_i$ is the vector encoding the node states of the corresponding network $N_i$ and let us denote the transformation function that maps $\mathbf{v}_i$ into our novel target subspace by $f(\mathbf{v}_i)$. We first formulate the two objective functions as follows:

$$\arg\min_f \sum_{i=1}^{m}\sum_{j=1}^{m}(f(\mathbf{v}_i) - f(\mathbf{v}_j))^2\mathbf{A}_{ij}^{+} \tag{2.1}$$

$$\arg\max_f \sum_{i=1}^{m}\sum_{j=1}^{m}(f(\mathbf{v}_i) - f(\mathbf{v}_j))^2\mathbf{A}_{ij}^{-} \tag{2.2}$$

To gain more insights into these setting objectives, let us take a look at the first Eq.(2.1). If two network instances $N_i$ and $N_j$ have similar local states in the original space (i.e., $\mathbf{A}_{ij}^{+}$ is large), this first objective function will be penalized if the respective points $f(\mathbf{v}_i)$ and $f(\mathbf{v}_j)$ are mapped far part in the transformed space. As such, minimizing this cost function is equivalent to maximizing the similarity amongst instances having the same global network states in the reduced dimensional subspace. On the other hand, looking at Eq.(2.2) can tell us that the function will incur a high penalty (proportional to $\mathbf{A}_{ij}^{-}$) if two networks having different global states are mapped close in the induced subspace. Thus, maximizing this function is equivalent to minimizing the similarity among neighboring networks having different global states in the novel reduced subspace. As mentioned earlier, such networks tend to locate close to the discriminative boundary function and hence, maximizing the second objective function leads to the maximal margin among clusters of different global-state networks.

Having the mapping function $f(.)$ to be optimized above, it is crucial to ask which is an appropriate form for it. Either a linear or non-linear function can be selected as long as it effectively optimizes two objectives concurrently. Nonetheless, keeping in mind that our ultimate goal is to derive a set of succinct discriminative subnetworks along with their *explicit* nodes. Optimizing a non-linear function is generally not only more complex but importantly may lose the capability in explaining how the new features have been derived (since they will be the *non-linear* combinations of the original nodes). We therefore would prefer $f(.)$ as in the form of a linear combination function and following this, $f(.)$ can be represented explicitly as a transformation matrix $U_{n\times d}$ that linearly combines $n$ nodes into $d$ novel features ($d \ll n$) of the induced subspace. For the sake of discussion, we elaborate here for the projection onto 1-dimensional subspace (i.e., $d = 1$). The solution for the general case $d > 1$ will be straightforward once we obtain the solution for this base case. Given this simplification and with little algebra, we recast our first objective function as follows:

$$\begin{aligned}\arg\min_{\mathbf{u}} \sum_{i=1}^{m}\sum_{j=1}^{m} \|\mathbf{u}^T\mathbf{v}_i - \mathbf{u}^T\mathbf{v}_j\|^2 \mathbf{A}^+_{ij} &= \sum_{i=1}^{m}\sum_{j=1}^{m} tr\left(\mathbf{u}^T(\mathbf{v}_i-\mathbf{v}_j)(\mathbf{v}_i-\mathbf{v}_j)^T\mathbf{u}\right)\mathbf{A}^+_{ij} \\ &= tr\left(\sum_{i=1}^{m}\sum_{j=1}^{m}\left(\mathbf{u}^T(\mathbf{v}_i-\mathbf{v}_j)\mathbf{A}^+_{ij}(\mathbf{v}_i-\mathbf{v}_j)^T\right)\mathbf{u}\right) \\ &= 2tr\left(\mathbf{u}^T\mathbf{V}\mathbf{D}^+\mathbf{V}^T\mathbf{u}\right) - 2tr\left(\mathbf{u}^T\mathbf{V}\mathbf{A}^+\mathbf{V}^T\mathbf{u}\right) = 2tr\left(\mathbf{u}^T\mathbf{V}\mathbf{L}^+\mathbf{V}^T\mathbf{u}\right) \qquad (2.3)\end{aligned}$$

in which we have used $tr(.)$ to denote the trace of a matrix and $\mathbf{V}$ as the matrix whose column $i$th accommodates the dynamic local states of network instance $N_i$ (i.e., $\mathbf{v}_i$), forming its size of $n \times m$. Also, $\mathbf{D}$ is the diagonal matrix whose $\mathbf{D}^+_{ii} = \sum_j \mathbf{A}^+_{ij}$ and we have defined $\mathbf{L}^+ = \mathbf{D}^+ - \mathbf{A}^+$, which can be shown to be the Laplacian matrix [46]. For the second objective function in Eq.(2.2), we can repeat the same computation which yields to the following form:

$$\begin{aligned}\arg\max_{\mathbf{u}} \sum_{i=1}^{m}\sum_{j=1}^{m} \|\mathbf{u}^T\mathbf{v}_i - \mathbf{u}^T\mathbf{v}_j\|^2 \mathbf{A}^-_{ij} \\ &= 2tr\left(\mathbf{u}^T\mathbf{V}\mathbf{D}^-\mathbf{V}^T\mathbf{u}\right) - 2tr\left(\mathbf{u}^T\mathbf{V}\mathbf{A}^-\mathbf{V}^T\mathbf{u}\right) \\ &= 2tr\left(\mathbf{u}^T\mathbf{V}\mathbf{L}^-\mathbf{V}^T\mathbf{u}\right) \end{aligned} \tag{2.4}$$

where again $\mathbf{D}^-$ is the diagonal matrix with $\mathbf{D}^-_{ii} = \sum_j \mathbf{A}^-_{ij}$ and we have defined $\mathbf{L}^- = \mathbf{D}^- - \mathbf{A}^-$.

Notice that while the above formulations aim at discriminating different global state networks in the low dimensional subspace, it has not yet taken into consideration the generalized network structure captured by our first meta-graph. As described previously, the mutual interactions among nodes are also important in determining the global network states. Also according to Definition 5, the larger the value placing on the link between nodes $v_p$ and $v_q$, the more likely they are being involved in the same process. Therefore, we would expect our mapping vector $\mathbf{u}$ not only separating well different global state networks but also ensuring its smoothness property w.r.t. the generalized network topology characterized by the first meta-graph $N$.

Toward the above objective, we formulate the network topology as a constraint in our learning objective function, and in order to be consistent with the approach based on spectral graph analysis, we encode the topology captured in $N$ by an $n \times n$ constraint matrix $\mathbf{C}$ whose elements are defined by:

$$\mathbf{C}_{pq} = \mathbf{C}_{qp} = \begin{cases} \sum_q K(p,q) & \text{if } v_p \equiv v_q \\ -K(p,q) & \text{if } v_p \text{ and } v_q \text{ are connected} \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

It is easy to show that, by this definition, $\mathbf{C}$ is also the Laplacian matrix and its quadratic form, taking **u** as the vector, is always non-negative:

$$\begin{aligned}\mathbf{u}^T\mathbf{C}\mathbf{u} &= \sum_{p=1}^{n} u_p^2 \sum_{q=1}^{n} K(p,q) - \sum_{p=1}^{n}\sum_{q=1}^{n} u_p u_q K(p,q) \\ &= \frac{1}{2}\sum_{p=1}^{n}\sum_{q=1}^{n} K(p,q)(u_p - u_q)^2 \geq 0 \qquad (2.6)\end{aligned}$$

in which $u_p$, $u_q$ are components of vector **u**. It is possible to observe that if $K(p,q)$ is large, indicating nodes $v_p$ and $v_q$ are strongly interacted in large portion of the network instances, the coefficients of $u_p$ and $u_q$ should be similar (i.e., smooth) in order to minimize this equation. From the network-structure perspective, we would say that if $v_p$ is known as a node affecting the global network state, its selection in the transformed space will increase the possibility of being selected of its nearby connected node $v_q$ if $K(p,q)$ is large, leading to the formation of discriminative subnetworks in the induced subspace. Therefore, in combination with the dual-objective function formulated above, we finally claim our constrained optimization problem as follows (the constants can be omitted due to optimization):

$$\begin{aligned}\mathbf{u}^* =& \arg\max_{\mathbf{u}} \left\{ tr\left(\mathbf{u}^\mathbf{T}\mathbf{V}(\mathbf{L}^- - \mathbf{L}^+)\mathbf{V}^T\mathbf{u}\right)\right\} \\ & \text{subject to } \ \mathbf{u}^T\mathbf{C}\mathbf{u} \leq t \\ & \qquad \text{and } \ \mathbf{u}^T\mathbf{V}\mathbf{D}^+\mathbf{V}^T\mathbf{u} = 1 \qquad (2.7)\end{aligned}$$

The first network topology constraint aims to retain the smoothness property of **u** whereas the second constraint aims to remove its freedom, meaning that we need **u**'s direction rather than its magnitude. The network topology constraint is beneficial in two ways. First as presented above, it offers a convenient and natural way to incorporate the network topology into our space transformation learning process. Second, as being formulated in the vector quadratic form, it essentially imposes the features/nodes selection through the coefficients of $\mathbf{u}$ by shrinking those of irrelevant nodes toward zero while crediting large values to those of relevant nodes. Indeed, this quadratic $L_2$-norm is a kind of regularization which is often referred to as the ridge shrinkage in statistics for regression [47]. The parameter $t$ is used to control the amount of shrinkage. The smaller the value of $t$, the larger the amount of shrinkage.

### 2.3.3 Solving the Function

In order to solve our dual objective function associated with constraints, we resort the Lagrange multipliers method and following this, Eq. (2.7) can be rephrased as follows:

$$\mathcal{L}(\mathbf{u},\lambda) = \mathbf{u}^T \left(\mathbf{V}\widetilde{\mathbf{L}}\mathbf{V}^T - \alpha\mathbf{C}\right)\mathbf{u} - \lambda\left(\mathbf{u^T V D V^T u} - 1\right) \tag{2.8}$$

of which, to simplify notations, we have denoted $\widetilde{\mathbf{L}} = \mathbf{L}^- - \mathbf{L}^+$, $\mathbf{D} = \mathbf{D}^+$ and $\alpha$ is used in replacement for $t$ as there is a one-to-one correspondence between them [47]. Taking the derivative of $\mathcal{L}(\mathbf{u},\lambda)$ with respect to vector $\mathbf{u}$ yields:

$$\frac{\partial\mathcal{L}(\mathbf{u},\lambda)}{\partial\mathbf{u}} = 2\left(\mathbf{V}\widetilde{\mathbf{L}}\mathbf{V}^T - \alpha\mathbf{C}\right)\mathbf{u} - 2\lambda\mathbf{V D V^T u} \tag{2.9}$$

And equating it to zero leads to the generalized eigenvalue problem:

$$\left(\mathbf{V}\widetilde{\mathbf{L}}\mathbf{V}^T - \alpha\mathbf{C}\right)\mathbf{u} = \lambda\mathbf{V D V^T u} \tag{2.10}$$

It is noticed that $\mathbf{V}$ is a singular matrix and its rank is at most $\min(n, m)$, making the combined matrix on the right hand side not directly invertible. We therefore decompose $\mathbf{V}\mathbf{D}^{1/2}$ into $\mathbf{P}\boldsymbol{\Sigma}\mathbf{Q}^T$, where columns in $\mathbf{P}$ and $\mathbf{Q}$ are respectively called the left and right (orthonormal) singular vector of $\mathbf{VD}^{1/2}$ while $\mathbf{\Sigma}$ stores its singular values. Note that this decomposition is always possible since $\mathbf{D}$ is a non-negative diagonal matrix of node degrees. Additionally, both $\mathbf{P}$ and $\mathbf{Q}$ can be represented in the square matrices while $\mathbf{\Sigma}$ a rectangular one of $n \times m$ size according to the most general decomposition form in [48]. Following this, the combined matrix on the right hand size can be rewritten as:

$$\mathbf{VDV}^T = \mathbf{P\Sigma}^2\mathbf{P}^T \tag{2.11}$$

And in order to get a stable solution, we keep the top ranked singular values in $\mathbf{\Sigma}$ such as their summation explains for no less than 95% of the total singular values[1]. Let us denote $\mathbf{B}^* = \mathbf{P\Sigma}^{-2}\mathbf{P}^T$ as the inversion of the right hand side and before showing our optimal solution, we need the following proposition:

**Proposition 1.** *Let* $\mathbf{P}$ *be the matrix of left singular vectors of* $\mathbf{VD^{1/2}}$ *defined above, then its row vectors are also orthogonal, i.e.,* $\mathbf{PP}^T = \mathbf{I}$

*Proof:* Let $\mathbf{a}$ be an arbitrary vector, we need to show $\mathbf{PP}^T\mathbf{a} = \mathbf{a}$. Due to the orthogonal property of left singular vectors, it is true that $\mathbf{P}^T\mathbf{P} = \mathbf{I}$. The inversion of $\mathbf{P}$ therefore is equal to $\mathbf{P}^T$ and given arbitrary vector $\mathbf{a}$, there is a uniquely determined vector $\mathbf{b}$ such that $\mathbf{Pb} = \mathbf{a}$. Consequently,

$$\mathbf{PP}^T\mathbf{a} = \mathbf{PP^TPb} = \mathbf{Pb} = \mathbf{a}$$

It follows that $\mathbf{PP}^T = \mathbf{I}$ since $\mathbf{a}$ is an arbitrary vector.

**Proposition 2.** *Given* $\mathbf{B} = \mathbf{P\Sigma}^2\mathbf{P}^T$*, we have* $\mathbf{BB}^* = \mathbf{I}$

*Proof:* The proof of this proposition is straightforward given Proposition 1.

[1]Note that since $(\mathbf{VD}^{1/2})(\mathbf{VD}^{1/2})^T$ is Hermitian and positive semidefinite, the diagonal entries in $\mathbf{\Sigma}$ are always real and nonnegative.

Now, for simplicity, let us denote $\mathbf{A}$ for the combined matrix $(\mathbf{V}\widetilde{\mathbf{L}}\mathbf{V}^T - \alpha\mathbf{C})$, then it is straightforward to see that $\mathbf{u}$ turns out to be the eigenvector of the equation:

$$\mathbf{B}^*\mathbf{A} = \lambda\mathbf{u} \tag{2.12}$$

with the maximum value is given by the following proposition.

**Proposition 3.** *Given matrix* $\mathbf{A} = \mathbf{V}\widetilde{\mathbf{L}}\mathbf{V}^T - \alpha\mathbf{C}$ *and* $\mathbf{B} = \mathbf{V}\mathbf{D}\mathbf{V}^T$ *defined above, the maximum value of* $\mathbf{u}^T\mathbf{A}\mathbf{u}$ *subjected to* $\mathbf{u}^T\mathbf{B}\mathbf{u} = 1$ *is the largest eigenvalue of* $\mathbf{B}^*\mathbf{A}$.

*Proof:* Due to Proposition 2, it is straightforward to see that:

$$\mathbf{u}^T\mathbf{A}\mathbf{u} = \mathbf{u}^T\mathbf{B}\mathbf{B}^*\mathbf{A}\mathbf{u}$$

On the other hand, $\mathbf{u}^T\mathbf{B}\mathbf{B}^*\mathbf{A}\mathbf{u} = \mathbf{u}^T\mathbf{B}\lambda\mathbf{u}$ by equation Eq. (2.12) and further taking into account our second constraint, it follows that:

$$\max_{\mathbf{u}:\mathbf{u}^T\mathbf{B}\mathbf{u}=1}\{\mathbf{u}^T\mathbf{A}\mathbf{u}\} = \max\{\lambda\}$$

From this proposition, it is safe to say that $\mathbf{u}^* = \mathbf{u}_1$ as the first eigenvector of $\mathbf{B}^*\mathbf{A}$ corresponding to its largest eigenvalue $\lambda_1$ is our optimal solution. Since eigenvectors and eigenvalues go in pair, the second optimal solution is the second eigenvector $\mathbf{u}_2$ corresponding to the second largest eigenvalue $\lambda_2$ and so on. Consequently, in the general case, if $d$ is the number of unique global network states, our optimal transformed space is the one spanned by the top $d$ eigenvectors. In the next section, we present a method to select optimal features/nodes along with the subnetworks formed by these nodes.

### 2.3.4 Subnetwork Selection

In essence, our top $d$ eigenvectors play the role of space transformation which projects network data from the original high dimensional space into the induced subspace of $d$ dimensions. Their coefficients essentially reflect how the original nodes (features) have been combined or more specifically, the degree of node's importance in contributing to the subspace that optimally discriminates network instances. Following the approaches adopted in [49, 50] with $c$ as the user parameter, we select top $c$ entries in each $\{\mathbf{u}_i\}_{i=1}^d$ corresponding to the selective nodes. Nonetheless, it is possible that there will be more than $c$ nodes selected by combining from $d$ eigenvectors. Therefore, in practice, we may use a simple approach by first selecting the largest absolute entries across $d$ eigenvectors:

$$\mathbf{v} = \{v_1, \dots, v_n\} \text{ where } v_p = \max_i |u_{i,p}| \tag{2.13}$$

where $u_{i,p}$ is the $p$-th entry of eigenvector $\mathbf{u}_i$, and then selecting nodes according to the top $c$ ranking entries in $\mathbf{v}$. The subnetworks forming from these nodes can be straightforwardly obtained by matching to the nodes in our generalized network $N$ defined in Definition 5, along with their connecting edges stored in $E$. These subnetworks can be visualized which offers the user an intuitive way to examine the results.

### 2.3.5 Computational Complexity

We name our algorithm `SNL`, an acronym stands for SubNetwork spectral Learning. Its computation complexity is analyzed as follows. We first need to compute edges' weights according to Definition 5 to build our first meta-graph which takes $O(n^2m)$ since there are at most $n(n-1)/2$ edges in the generalized network $N$. Second, in building the two subsequent meta-graphs, the cosine distance between any two network instances is computed which amounts to $O(n^2m)$ or $O(mn\log n)$ in case the multidimensional binary search tree is used [51]. Also, since the size of matrix $\mathbf{VD}^{1/2}$ is $m \times n$, its singular value decomposition takes $O(mn\log n)$ with the Lanczos technique [46]. Likewise, the eigen-decomposition of the matrix $\mathbf{B}^*\mathbf{A}$ takes $O(n^2\log n)$ since its size is $n \times n$. Therefore, in combination, the overall complexity is at most $O(n^2m + n^2\log n)$ assuming that the number of nodes is larger than the number of network instances.

## 2.4 Experiments on Synthetic and Protein Protein Interaction (PPI) Networks

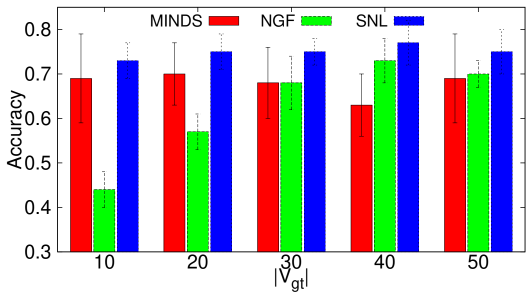
### 2.4.1 Datasets and Experimental Setup

We compare the performance of `SNL` against `MINDS` [44] which is among the first approaches formally addressing the global-state network classification problem by a sub-network sampling. Another algorithm for comparison is the Network Guided Forests (`NGF`) [52] designed specifically for protein protein interaction (PPI) networks. We use both synthetic and real world datasets for experimentation. Since global states are available in all datasets, we compare average accuracy in 10-fold cross validation for synthetic data, and 5-fold cross validation for real data (due to smaller numbers of network instances). For `SNL`, the cross validation is further used to select its optimal $\alpha$ parameter (shortly discussed below). Unless otherwise indicated, we set $k = 10$ and use the linear-SVM to perform training and testing in the transformed space (keeping top 50 nodes) in `SNL`. We set `MINDS`' parameters as follows: 10000 sampling iterations, 0.8 discriminative potential threshold and $K = 200$ as recommended in the original paper [44]. The Gini index is used for the tree building in `NGF` and we set its improvement threshold $\epsilon = 0.02$ [52].

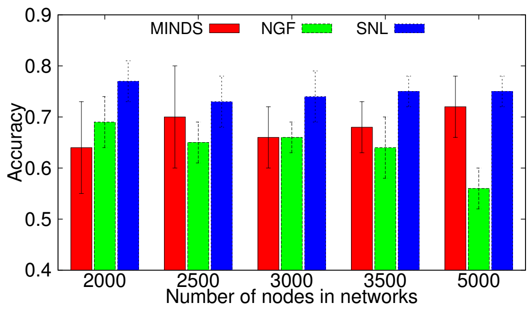
### 2.4.2 Results on Synthetic Datasets

We use synthetic data to evaluate the performance of our technique in training robust classifiers and selecting relevant subnetworks. We generate scale-free backbone networks by preferential attachment of a predefined size adding 20 edges for each new node. The probabilities of backbone edges are sampled from a truncated Gaussian distributions: $N(0.9, 0.1)$ for edges among *ground truth nodes* (pre-selected nodes of high-correlation with the network state) and $N(0.7, 0.1)$ for the rest of the edges. The weighted backbone serves as our generalized template to generate network instances by independently sampling the existence of every edge based on its probability. The global states are binary $S_i \in \{0, 1\}$ with balanced distribution. We further add noise to both global and local states of ground truth nodes, respectively with levels of 10% and 30%.

**Varying $|V_{gt}|$:** In the first set of experiments, we aim to test whether the performance of all algorithms is affected by the number of ground truth nodes. To this end, we generate 5 datasets by fixing $m = 1000$ instances, $n = 3000$ nodes and vary the ground truth nodes $|V_{gt}|$ from 10 to 50. In Figure 2.1a, we report the average accuracy (and standard deviation) of all algorithms in 10-fold cross validation. As one may observe, `SNL` performs stably regardless of the change in the ground truth sizes. Compared to the other techniques, its classification is always consistently higher across all cases. The `MINDS` technique also performs well on this experimental setting yet the `NGF` seems to be sensitive to the small ground truth sizes. For small $|V_{gt}|$, the sampling strategy based on density areas employed in `NGF` has little chance to select the ground truth nodes, making its accuracy close to a random technique. When more ground truth nodes are introduced, `NGF` has higher possibility to sample high-utility nodes and in the last two datasets, its performance is on par with that of `MINDS`. Nonetheless, its accuracy only peaks at 73% in the best case which is lower than 77% in `SNL` (last column).



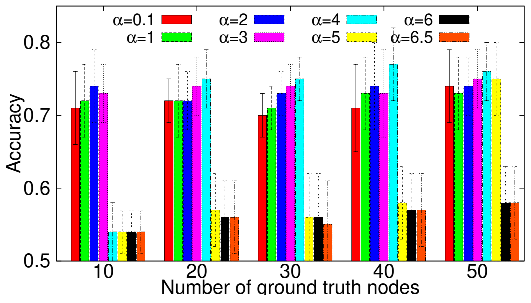
(a) Accuracy of all algorithms by varying ground truth subnetworks' nodes.



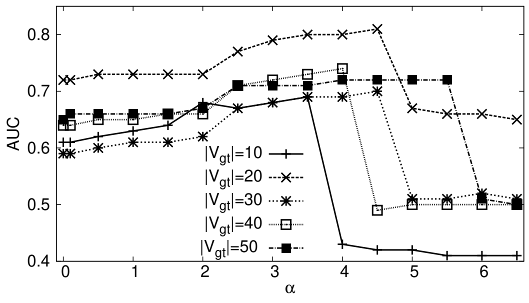
(b) Accuracy of all algorithms by varying network size.

Figure 2.1: Accuracy performance of SNL on synthetic datasets with varying groundtruth nodes and network size, against baseline methods MINDS and NGF.

**Varying network size:** In the second set of experiments, we evaluate the performance of all algorithms by varying the network sizes. Specifically, we fix $m = 3000$ network instances, $|V_{gt}| = 50$ ground truth nodes and generate 5 datasets having the network size varied from 2000 to 5000 nodes. The classification performance along with the standard deviation is reported in Figure 2.1b. It is possible to see that the performance traits are similar to those in our first set of experiments. SNL's classification accuracy remains high while that of NGF decreases with the increase of network size. This again can be explained by the extension of the searching subnetwork space, leading to the lower likelihood of both NGF and MINDS in identifying relevant subnetworks with potentially discriminative nodes. The slightly better performance of MINDS (compared to NGF) is due to its accuracy thresholding in selecting candidate substructures. The set of MINDS' selected trees are thus qualitatively better. Nonetheless, as compared to SNL, our subspace learning approach show more competitive results. Moreover, since the low-dimensional subspace learnt in SNL is unique and linearly combined from the most discriminative nodes, its performance also shows more stable, indicated by the small standard deviation across all cases.

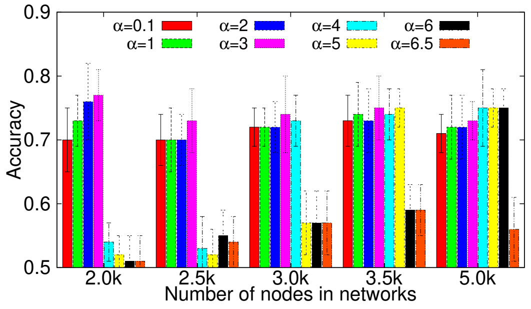


(a) Network effect on accuracy for different numbers of ground truth nodes.

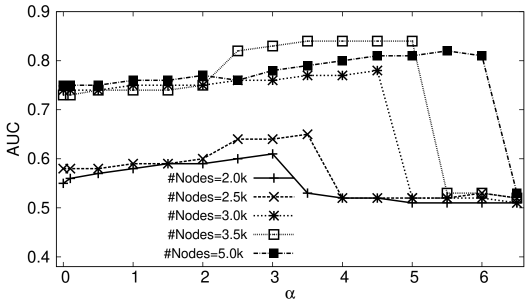


(b) AUC performance for different numbers of ground truth nodes.

Figure 2.2: Impact of numbers of ground truth nodes.



(a) Network effect on accuracy for different network sizes.



(b) AUC performance for different network sizes.

Figure 2.3: Impact of network sizes.

**Effect of network topology:** In order to provide more insights into the performance of SNL, we further test the network effect. As presented in Section 2.3, $\alpha$ is the parameter controlling the influence of the network information on the subspace learning process. The higher the $\alpha$, the more preference putting on the heavily connected nodes. We report in Figures 2.2a,2.3a the accuracy of SNL by varying $\alpha$ from 0.1 to 6.5 and in Figures 2.2b,2.3b its ability in discovering the ground truth nodes. For the latter case, we validate the performance through the usage of area under the ROC curve (AUC) [47].

As expected, incorporating the network structure in the subspace learning process improves both classification rate and the AUC in uncovering the ground truth nodes. The plots in Figures 2.2a,2.3a show that the accuracy initially improves for increasing influence of the network ($\alpha \leq 5$) and then decreases as the network component becomes prevalently dominant ($\alpha > 5$). This is because for large $\alpha$, SNL tends to incorporate irrelevant nodes solely based on their strong connections to the neighbors (yet their local values might not help classifying global state values). Another notable observation is that, in larger instances or ground truth feature sets, the optimal $\alpha$ tends to increase as well. Moreover, the values of $\alpha$ that maximize classification accuracy also result in optimal AUC in identifying the ground truth nodes (Fig. 2.2b,2.3b). These experiments clearly show the helpful information provided by the network topology in uncovering the groundtruth features. Also, the performance of NGF and MINDS has been excluded in this set of experiments to save space and we will shortly discuss their AUC performance with a real-world dataset.

### 2.4.3 Real-world Datasets

We use 4 real-world datasets to evaluate the performance of SNL and its competing methods. The features in all datasets correspond to micro-array expression measurements of genes; the topology structures relating features correspond to gene interaction networks; and the global network states correspond to phenotypic traits of the subjects/instances. The statistics of our datasets are listed in Table 2.1. Two of our real-world datasets, breast cancer and embryonic development, were also used for experimentation in the original NGF method [52]. Our other datasets come from a study on maize properties [53] and a human liver metastasis study [54] combined with a functional network [55]. The network samples are used as provided in the original studies, except for maize where we down-sample one of the classes to balance the global state distribution.

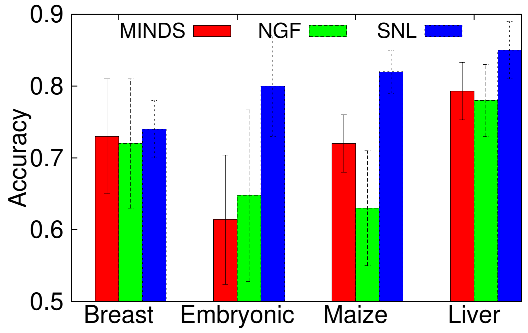
**Classification performance:** The comparison of classification accuracy for all techniques and datasets is presented in Figure 2.4a. We report the average accuracy and

Table 2.1: Real-world dataset statistics and sources

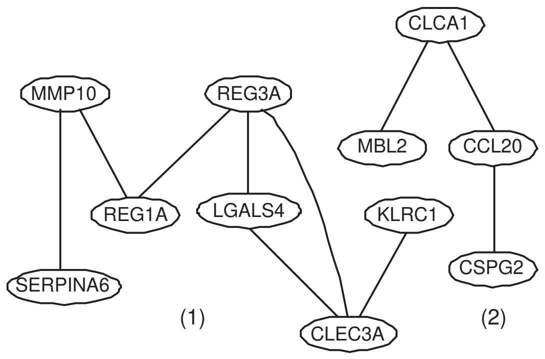
| **Datasets** | **Genes** | **Edges** | **Instances** | **Global State** |
|---|---|---|---|---|
| **Breast cancer** | 11203 | 57235 | 295 | cancer/non-cancer |
| **Embryonic development** | 1321 | 5227 | 35 | developmental tissue layer |
| **Maize** | 8574 | 298510 | 344 | high/low oil production |
| **Liver metastasis** | 7383 | 251916 | 123 | disease/non-disease |

standard deviation from the 5-fold stratified cross validation. All techniques perform competitively on the breast cancer data, achieving more than 70% of classification accuracy on average. The accuracy of `SNL` dominates significantly that of the sampling techniques on the embryonic and maize datasets (at least 15% and 10% improvement respectively) and less so in the liver dataset. The separation is highest in the datasets of small number of instances and big number of feature nodes – the settings in which `SNL` is particularly effective. Beyond average performance improvement, `SNL`'s accuracy is also more stable across all folds as it considers the global network structure when learning a subspace for classification, while the alternatives perform sampling in the exponential space of substructures.

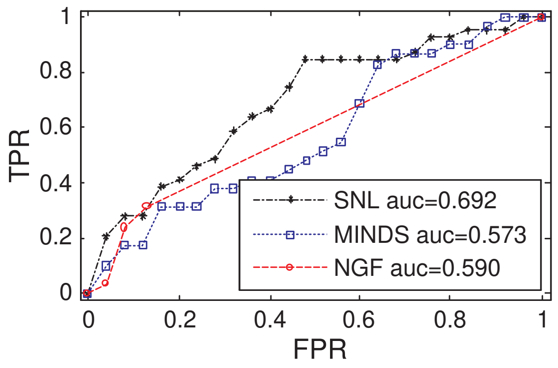
**Subnetwork discovery:** Unlike the synthetic datasets where we can control the ground truth network features, it is generally much harder to obtain ground truth subnetworks for real world datasets. However, as an attempt to look deeper into the results, we choose the Liver metastasis and further investigate the meaningful subnetworks generated by the `SNL`. For this dataset, out of top 50 nodes of highest coefficient values (ref. Section 2.3.4), about one third of the nodes are connected into four subnetworks. We depict in Figure 3.6 the two largest ones which respectively contain 7 and 4 connected gene nodes. Among these selected subnetworks, the genes `REG1A` and `REG3A` are particularly interesting since they are in agreement with the ones found in [1] which was shown to be involved in the



(a) Classification performance of all algorithms on four real world datasets



(b) Subnetworks identified by the SNL related to the liver metastasis



(c) ROC performance over the liver metastasis ($x$-axis is false positive rate, $y$-axis is true positive rate)

Figure 2.4: Performance of SNL on protein-protein interaction (PPI) networks against baseline methods NGF and MINDS.

liver metastasis cancer. As a comparison against `MINDS` and `NGF`, we notice that both methods generate multiple binary-trees where each node has only a *single* parent. Moreover, while `SNL` can provide a natural rank of important genes based on their coefficients (from the learnt subspace), it is less trivial to define important genes from `NGF` and `MINDS` as they both generate thousands of trees. For the purpose of measuring biological relevance of obtained genes, we define a ranking for these competing techniques based on the frequency of genes appeared in the generated trees. For comparison, we select 46 metastasis-specific genes identified in [1] to serve as a ground truth set (39 intersect with our network and expression data) and plot the ROC performance of all algorithms in Figure 2.4c. Note that, this is only a partial ground truth set, since identifying all genes associated with this disease is a subject of ongoing research [1]. It is observed that the ranking produced by `SNL` includes more ground truth genes than those of `NGF` and `MINDS` at increasing false-positive rates. The higher true positive rates of `SNL` makes it a better method for identifying new genes associated with the phenotype of interest. In practice, this is an important feature of the algorithm since validating even a single gene related to cancer is both time-wise and financially costly. As shown in Figure 2.4c, while the ROC performance of `NGF` and `MINDS` are only at 0.59 and 0.57 AUC, that value of `SNL` is 0.69 which clearly demonstrates large gap of better performance.

## 2.5 Related Work

Mining discriminative subspaces from global-state networks is a novel and challenging problem. Two lines of work close to this problem are network classification and mining evolving subgraphs from dynamic network data. In the network classification case, most representative algorithms are `LEAP` [56], `graphSig` [57], `GAIA` [58] and `COM` [59] which generally assume a database consisting of positive and negative networks that need to be classified. These approaches, though diverse in terms of their underlying algorithms, all aim at extracting a set significant subnetworks that are *more frequent* in one class of positive networks and *less frequent* in the negative class. Different from the above problems, we aim to mine subnetworks which are represented in all network instances; yet the node values along with the network structures can discriminate the global states of the networks. Another line of related research focuses on mining dynamic evolving subnetworks [10, 60, 61]. The problem in this case is to obtain subnetworks over time that evolve significantly (outliers) from other network locations. This setting therefore do not model the problem developed in this work since the dynamic network snapshots neither contain global-state values nor can remove their temporal property.

Several studies in systems biology have indicated the critical role of the network structure in identifying protein modules related to clinical outcomes, for both regression [62, 63, 42] and classification [52, 44]. In the classification setting which is related to our study, the `NGF` [52] is an ensemble approach that builds a forest of trees jointly voting for the class of a network instance. Resided at the `NGF`'s core is the `CART` (classification and Regression tree) technique and in order to build a decision tree within the PPI network, `NGF` starts with a root node and progressively includes connected nodes as long as the improvement in class separation (measured by Gini index) is no smaller than a given threshold. The study in [44] is the first one to formally introduce the problem of subnetwork mining in global-state networks and further propose the `MINDS` algorithm to solve it. Similar to `NGF`, `MINDS` adopts network-constraint decision trees and is also an ensemble classifier. Nonetheless, it increases the quality of decision trees by developing a novel concept of editing map over the space of potential subnetworks and exploits Monte Carlo Markov Chain sampling over this novel data structure to seek decision trees with maximum classification potential. Unlike the frequency-based and sampling classification discussed above, our approach is fundamentally different as it searches for the most discriminative subnetworks in a single low dimensional subspace through the spectral learning technique, which generally leads to more stable and high-accuracy performance.

## 2.6 Conclusion

We proposed a novel algorithm named `SNL` to address the challenging problem of uncovering the relationship between local state values residing on nodes and the global network events. While most existing studies address this problem by sampling the exponential subnetworks space, we adopt an efficient and effective subspace transformation approach. Specifically, we define three meta-graphs to capture the essential neighboring relationships among network instances and devise a spectral graph theory algorithm to learn an optimal subspace in which networks with different global-states are well separated while the common structure across samples is smoothly respected to enable subnetwork discovery. Through experimental analysis on synthetic data and real-world datasets, we demonstrated its appealing performance in both classification accuracy and the real-world relevance of the discovered discriminative subnetwork features.

# Chapter 3

# Learning Predictive Substructures for Network Data

## 3.1 Introduction

Network analysis plays a key role in understanding complex data. One important task in network analysis is learning local substructures that have impact on the global network properties. Compared to feature selection in high dimensional data, the analysis in a network context is more challenging due to the inherent interactions among features associated with nodes. Consider an example of Alzheimer's disease in neuroscience [43, 64] where the task is to find markers that can predict the disease states. Due to the natural relationship between neural activity and neural connectivity, the loss of neurons/synapses in one brain region usually impacts the cognitive functions of other brain regions that are *physically* and *functionally* connected [65]. Hence, it is not sufficient to simply identify isolated neuronal populations (brain regions) whose activation is abnormal. Instead, it is crucial to consider the underlying network communication among neuronal populations that leads to different states of the disease [64].

Another example comes from biology, where studies have shown that biological outcomes (i.e. global properties) are determined by modules of proteins that interconnect within the context of protein-protein interaction (PPI) networks [52]. Such networks are essential in understanding genetic and regulatory diseases such as cancers and neurological disorders. Given the expression levels of genes for multiple patients, the task in this setting is to identify predictive substructures associated with the disease, while respecting the underlying interactions among implicated genes.

The success of machine learning and data mining algorithms depends on the representation of the underlying data. This is particularly true for network data where irrelevant nodes and edges may limit the understanding and hinder the discovery of local processes that impact and govern the global behavior of the network. Understanding what these local subnetwork processes are, how they vary and what makes them different across network populations is critically important for a variety of domain applications. Network data (associated with network nodes) can naively be treated as an instance of high-dimensional data and common feature extraction methods such as PCA/SVD [66, 67] can be adopted for its analysis. While such methods can optimize the prediction quality, the models they learn often lack domain relevance since they do not incorporate the underlying network interactions, and thus may fail to discover actual processes behind the observed data. A better suited approach to identifying discriminative substructures is to directly sample the space of subnetworks and find ones that lead to high prediction of the global network states [52, 44]. While such approaches emphasize locality of the selected substructures, they need to explore an exponential number of connected subnetworks. Additionally, it is hard to ensure a unique and stable result across different runs due to the inherent drawback of sampling. Both stability and compactness of the predictive substructures are keys to unearth the complex relationships between local node values and global network properties.

In this work, we propose a novel framework to learn a small set of subnetwork structures that are predictive of the global network properties. In contrast to existing approaches that either ignore the network topology or sample over the exponential space of subnetworks, we formulate the predictive subnetworks learning problem as explicit node selection subject to network-constrained regularization. Our framework is not only able to discover a succinct set of predictive subnetwork structures but also, in combination with a simple classifier such as the linear SVM, can accurately predict the global network behavior within the space spanned by these substructures. Our learning task comprises of two steps. In the first step, we learn an optimal embedding subspace in which networks of different global states are maximally separated while those of the same state are clustered together. Each dimension of this low dimensional subspace essentially reflects a different aspect in separating networks labeled by different global states. In the second step, we perform substructures discovery through measuring the nodes' importance, regularized by their underlying local connectivity, along each intrinsic dimension of the embedding subspace. Specifically, we adopt L1-norm to explicitly remove irrelevant nodes that have little or no impact on the global network behavior, and L2-norm to enforce connectivity among selected nodes. We solve the first step of subspace learning via matrix eigen-decomposition by taking the graph embedding approach, while for the second step, we employ the gradient descent technique to minimize fitting errors. The proposed framework possesses many appealing properties: (1) it makes no assumptions about the statistical distribution of network data; (2) it has a solid mathematical background from spectral graph learning and convex gradient optimization; (3) it is guaranteed to achieve the optimum solution in each step. Our experimental analysis using four real world datasets demonstrates the superior performance of the proposed algorithm not only in terms of prediction accuracy against state-of-the-art algorithms, but also in discovering predictive network substructures with domain relevance. To the best of our knowledge, the proposed framework is the first that combines both subspace learning and network structures discovery with explicit node-selection, providing better understanding of the complex relationships between local node values and global network behavior.

## 3.2 Problem Setting and Preliminaries

### 3.2.1 Common Notation and Definitions

**Definition 1.** (Network sample) *A* network sample *is a triple $S_i = (\mathcal{V}_i, E_i, \mathcal{F})$, where $\mathcal{V}_i = \{v_1, v_2, \ldots, v_{m_i}\}$ is a set of nodes, $E_i \subseteq \mathcal{V}_i \times \mathcal{V}_i$ is a set of undirected edges, and $\mathcal{F}$ is a function labeling each node with a real number.*

Let $\mathcal{DS} = \{S_1, S_2, \ldots, S_n\}$ be a network dataset that consists of $n$ network samples. Each network sample $S_i \in \mathcal{DS}$ is associated with a discrete *global* state or label $\ell_i$. Unlike chemical compounds or XML web data where network samples may have considerably different network topologies [57, 56], we focus on a family of networks whose topologies are relatively stable across network samples. Following this, we define an aggregate network $S$ generalizing all samples $S_i$'s, as follows:

**Definition 2.** (Aggregate network) *Let $S = (\mathcal{V}, E, W)$ be a network summarizing the aggregate structure of all $S_i \in \mathcal{DS}$, where $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \ldots \cup \mathcal{V}_n$, $E \subseteq \mathcal{V} \times \mathcal{V}$ and $E_i \subseteq E$ $\forall E_i$. Each edge $E(p,q) \in E$ is associated with a positive weight $W(p,q)$ defined as the fraction of network samples containing that edge in their structure, i.e., $W(p,q) = n^{-1} \times \sum_i E_i(p,q)$ with $E_i(p,q) = 1$ if $v_p$ connects $v_q$ in $S_i$. Since all $S_i$'s are undirected networks, $W \in \mathbb{R}^{m \times m}$ defined for $S$ is a symmetric matrix, with $m$ as the total number of nodes in $\mathcal{V}$.*

The setting defined above is general enough to accommodate a number of practical network applications including social networks, human brain networks or protein-protein interactions. For instance, $S_i$ can model a snapshot captured at $i$-th timestamp of a social network, where $v_p$ corresponds to a user whose local state presumably encodes her political viewpoint, while the global state $\ell_i$ indicates the overall political viewpoint of the entire community (at the $i$-th timestamp). Snapshots captured at different times vary slowly in terms of network structure. Likewise, $S_i$ can also be used to encode a human brain network where the local value at an edge reflects the statistical correlation between a pair of brain regions and a global state $\ell_i$ encodes whether the subject is healthy or diseased. Different subjects may differ in their local node values but possess similar brain network structures [65].

### 3.2.2 Relationships among Network Samples

In many practical applications, the discriminative structure behind the observed network data can be captured by only a small number of latent regularities (dimensions) since many irrelevant nodes/features may have little or no impact on the global network labels. Learning an informative set of predictive subnetworks that influence the global states depends on how the relationships among network samples are captured and represented. Toward this goal, we construct two graphs encoding the (dis)similarity among network samples. Our optimization framework employs these relationship graphs to learn a low dimensional subspace that captures both the neighboring and discriminative structures among network samples. The construction of the graphs of network samples follows a similar construction from [68]. We next summarize the main steps of this construction and introduce the necessary notation.

Let $\mathcal{G}^+$ be defined as the first graph in which each vertex represents a network sample $S_i \in \mathcal{DS}$. A link is placed between two vertices if the corresponding network samples $S_i$ and $S_j$ have the same global state, and $S_i$ is among the $k$ nearest neighbors ($kNN$) of $S_j$ or vice versa. Each link is further associated with a non-negative value $\mathbf{K}^+_{ij}$ quantifying the degree of similarity between $S_i$ and $S_j$. Since the network samples have a similar structure, only their node values are used to compute their neighborhood similarity. Furthermore, as each $S_i$ is a subnetwork of $S$ according to Def.5, we use an $m$-dimensional vector $\mathbf{v}_i$ (recall $m$ is the total nodes in $S$) to store its local node values, with a null value being used to denote the value of a missing node. This makes any two network samples comparable in term of their local node values. As such, we adopt the cosine distance (though more complex measures could be used [69, 50]) to compute $\mathbf{K}^{+}_{ij}$ as follows:

$$\mathbf{K}^{+}_{ij} = \begin{cases} \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|}, & \text{if } \ell_i = \ell_j, S_j \in kNN(S_i) \text{ or } S_i \in kNN(S_j) \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

Likewise, $\mathcal{G}^{-}$ is defined as a second graph with vertices also representing network samples and a link connecting two vertices, say $S_i$ and $S_j$, if $S_i \in kNN(S_j)$ or $S_j \in kNN(S_i)$, yet their global states must be different. Likewise, a non-negative value $\mathbf{K}^{-}_{ij}$ is associated with a link to reflect how similar $S_i$ and $S_j$ are. Its computation is analogous to Eq.(3.1) except one difference: $\ell_i \neq \ell_j$.

It is important to note that while the development in [68] builds graphs among network instances in a similar fashion, its ultimate goal is classification of the global network states, while our goal here is feature selection in the form of subnetworks. As such, our feature selection framework can be combined with any classifier for global network states within subnetworks as features, including the classifier developed in [68]. Another way to view our current approach is as reducing the dimensions (features) in a controlled manner (i.e., enforcing a desired number of preserved features), while being aware of the network structure. Thus, it enables a more scalable and accurate training of classifiers, due to the reduced size of the network instances.

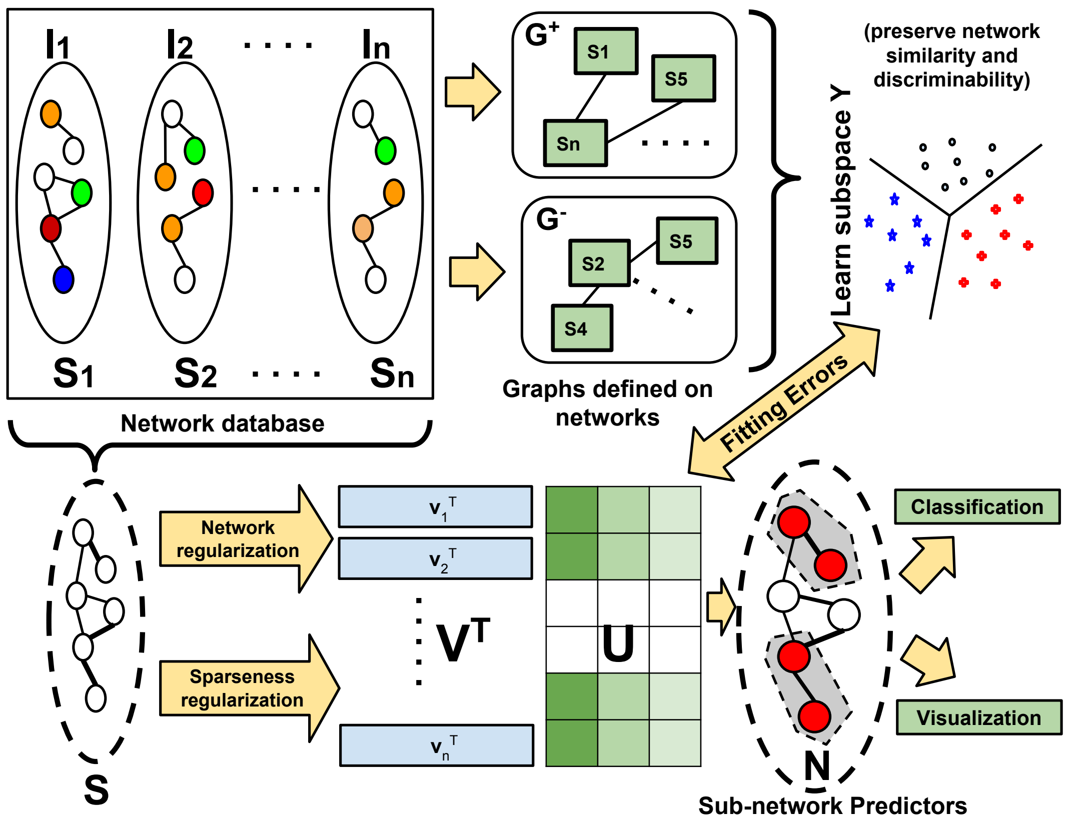


Figure 3.1: Overview of our framework in learning subnetwork structures for global network state prediction. An optimal subspace **Y** separating different network states is learnt first, then relevant subnetworks are mined through approximating **Y** with network and sparseness regularizations.

## 3.3 Learning Discriminative Subspace and Subnetworks

Given a set of network samples with global states, our goal is to uncover a succinct set of subnetwork structures whose local node values have the highest impact on global network states. Mining the optimal set of such predictive subnetworks is a non-trivial task since the number of possible subnetworks grows exponentially with the network size. As mentioned earlier, sampling is obviously one possible approach to overcome this challenge. However, without an effective indexing or pruning strategy, one might have to sample an impractically large number of times to ensure good quality substructures. Alternatively, methods operating on vector data and ignoring the network structures address only the goal of prediction accuracy while lacking domain relevance.

Our proposed solution differs from the above two extremes by adopting the framework summarized in Fig.3.1 (with notation is summarized in Table 3.1). Specifically, given $\mathcal{DS}$, we build two graphs $\mathcal{G}^+$ and $\mathcal{G}^-$. These two graphs serve our first objective of learning a low dimensional subspace $\mathbf{Y}$ in which network samples with different global states are well-distinguished while concurrently retaining the local similarities among network samples of the same global states. After obtaining $\mathbf{Y}$, we measure the importance of each node along each intrinsic dimension. This objective is achieved by minimizing the fitting errors between $\mathbf{Y}$ and the combination on the networks' local node values $\mathbf{V}$ by using a matrix $\mathbf{U}$.

| | |
|---|---|
| $S_i$, $\ell_i$ | a network sample and its global state/label |
| $S$, $\mathbf{C}_{m\times m}$ | the aggregate network (Def. 5) and its Laplacian |
| $n$, $m$ | #network samples and #nodes in $S$ |
| $\mathcal{G}^+$, $\mathcal{G}^-$ | similarity graphs among network samples |
| $\mathbf{K}^+$, $\mathbf{K}^-$ | affinity matrices of $\mathcal{G}^+$ and $\mathcal{G}^-$ |
| $\mathbf{V}_{m\times n}$ | node local states (features) for all samples $S_i$'s |
| $\mathbf{Y}_{n\times d}$ | local states embedding in a low-dimensional subspace |
| $\mathbf{U}_{m\times d}$ | output node-selection matrix |
| $\lambda_1, \lambda_2$ | regularization parameters for sparsity and network impact |
| $c$ | #nodes in selected substructures (controlled by $\lambda_1$) |

Table 3.1: Summary of notations used in the work.

Such an error minimizing process is further complemented by L1-norm to remove irrelevant nodes, and L2-norm to enforce proximity of selected nodes in the general network structure $S$ (Def.5). The predictive substructures are then selected based on $\mathbf{U}$'s sparse coefficients. The obtained substructures can be used for visualization and classification of new instances.

## 3.3.1 Subspace Learning

Given the two (dis)similarity graphs (defined in Sec.3.2.2), we construct our first objective function for learning a subspace in which the similarity relations among network samples of the same states are optimally preserved. This objective is formulated as follows:

$$\begin{aligned}\text{minimize} &\sum\nolimits_i \sum\nolimits_j \|\mathbf{y}_i - \mathbf{y}_j\|_2^2 \mathbf{K}_{ij}^+ \\ &= \sum\nolimits_i \sum\nolimits_j (\|\mathbf{y}_i\|_2^2 + \|\mathbf{y}_j\|_2^2 - 2\mathbf{y}_i^T \mathbf{y}_j) \mathbf{K}_{ij}^+ \\ &= 2\mathbf{Y^T}(\mathbf{D}^+ - \mathbf{K}^+)\mathbf{Y} = \mathbf{2tr}(\mathbf{Y^T L^+ Y}) \end{aligned} \tag{3.2}$$

in which $\mathbf{L}^+ = \mathbf{D}^+ - \mathbf{K}^+$ is the Laplacian matrix of $\mathcal{G}^+$; $\mathbf{D}^+$ is a diagonal matrix whose entries are summations over $\mathbf{K}^+$'s rows, i.e.,$\mathbf{D}_{ii}^+ = \sum_j \mathbf{K}_{ij}^+$, and $\mathbf{y}_i$, $\mathbf{y}_j$ (as $\mathbf{Y}$'s columns) are the maps of network samples $S_i$ and $S_j$ in our low dimensional subspace.

According to Eq.(3.2), similar network instances $S_i$ and $S_j$ (i.e. high $\mathbf{K}_{ij}^+$) mapped to distant points $\mathbf{y}_i$ and $\mathbf{y}_j$ in the resulting subspace would incur a high cost. Minimizing this objective function is thus equivalent to optimally preserving the local similarities among network samples having the same global state.

Analogously, we construct the second objective function to maximize the separation of instances of different global states:

$$\begin{aligned}\text{maximize} &\sum\nolimits_i \sum\nolimits_j \|\mathbf{y}_i - \mathbf{y}_j\|_2^2 \mathbf{K}_{ij}^- \\ &= 2\mathbf{Y}^{\mathbf{T}}(\mathbf{D}^- - \mathbf{K}^-)\mathbf{Y} = \mathbf{2tr}(\mathbf{Y}^{\mathbf{T}}\mathbf{L}^-\mathbf{Y}),\end{aligned} \tag{3.3}$$

where $\mathbf{L}^- = \mathbf{D}^- - \mathbf{K}^-$ is the Laplacian of $\mathcal{G}^-$, $\mathbf{D}^-$ is a diagonal matrix, $\mathbf{D}_{ii}^- = \sum_j \mathbf{K}_{ij}^-$, and the embedding $\mathbf{Y}$ is similarly defined as in Eq.(3.2). Unlike Eq.(3.2), here we aim to minimize the similarity between network samples $S_i$ and $S_j$ with different global states in the induced subspace. Such network instances might reside close to the classification boundary and hence, by minimizing their similarity, we improve their separability in the induced subspace. Combining the two objectives, we obtain:

$$\begin{aligned}\mathbf{Y} = &\arg\max_{\mathbf{Y}} \left\{\mathbf{tr}\left(\mathbf{Y}^{\mathbf{T}}(\mathbf{L}^- - \beta\mathbf{L}^+)\mathbf{Y}\right)\right\} \\ &\text{subject to} \quad \mathbf{Y}^{\mathbf{T}}\mathbf{D}^+\mathbf{Y} = \mathbf{I}\end{aligned} \tag{3.4}$$

where $\beta \in (0,1)$ is a trade-off parameter between the two objectives $\mathbf{L}^-$ and $\mathbf{L}^+$. It can be tuned via cross validation and based on empirical evaluation we find that $\beta \in [0.2, 0.4]$ often results in good performance. Additionally, the constraint $\mathbf{Y}^{\mathbf{T}}\mathbf{D}^+\mathbf{Y} = \mathbf{I}$ is used to ensure the uniqueness of the obtained subspace $\mathbf{Y}$. In solving this constrained trace optimization, we resort to the *Largrange multipliers* method [67]. Let us simplify the notation $\mathbf{L} = \mathbf{L}^- - \beta\mathbf{L}^+$ and $\mathbf{D}$ for $\mathbf{D}^+$. Our trace optimization in Eq.(3.4) can be recast as:

$$\mathcal{L}(\mathbf{Y}, \mathbf{\Lambda}) = \mathbf{Y}^{\mathbf{T}}\mathbf{L}\mathbf{Y} - \mathbf{\Lambda}\left(\mathbf{Y}^{\mathbf{T}}\mathbf{D}\mathbf{Y} - \mathbf{I}\right) \tag{3.5}$$

where $\mathbf{\Lambda}$ is a diagonal matrix with diagonal entries corresponding to the Lagrange multipliers. Taking the derivative of $\mathcal{L}$ with respect to $\mathbf{Y}$ and equating it to zero yields:

$$\mathbf{L}\mathbf{Y} = \mathbf{\Lambda}\mathbf{D}\mathbf{Y} \tag{3.6}$$

which gives us the optimal $\mathbf{Y}$ as the leading eigenvectors (corresponding to the largest eigenvalues) of the matrix $\mathbf{D^{-1}L}$. We choose the number of selected eigenvectors equal to $d$—the number of unique global network states, since a subspace with $d$ dimensions is generally sufficient to capture the discriminative structure among $d$ different classes.

## 3.3.2 Learning Sparse Subnetwork Structures

In the low-dimensional subspace $\mathbf{Y}$ (obtained according to Eq.(3.6)), each row represents the embedding of a network sample and each column reflects a different aspect in distinguishing global network states. We next proceed to quantifying the importance of each node in each dimension of $\mathbf{Y}$, or equivalently, the contribution of each node in differentiating global network states.

Let $\mathbf{V}$ be the matrix storing local node states for all network instances (see Sec.3.2.2) and $\mathbf{U}$ be a node-selection matrix that we aim to learn. Our objective is to minimize the fitting errors $\|\mathbf{Y} - \mathbf{V}^T\mathbf{U}\|_F^2$, while using the L2-norm to impose the network structure, and the L1-norm to remove irrelevant nodes.

In essence, each column $\mathbf{u} \in \mathbf{U}$ contains $m$ coefficients for combining $m$ $\mathbf{V}$'s rows in approximating one dimension of subspace $\mathbf{Y}$. Thus, by sparsifying and regularizing $\mathbf{u}$'s coefficients subject to the network structure, we learn the degree of importance of every node in discriminating global network states. Specifically, we formulate the network-constrained regularization via the Laplacian matrix $\mathbf{C}$ as follows:

$$\mathbf{C}_{pq} = \mathbf{C}_{qp} = \begin{cases} \sum_q W(p,q) & \text{if } v_p \equiv v_q \\ -W(p,q) & \text{if } v_p,\ v_q \text{ are connected} \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

The quadratic form of $\mathbf{C}$ is always non-negative, i.e., $\mathbf{u}^T\mathbf{C}\mathbf{u} = \frac{1}{2}\sum_{p=1}^{m}\sum_{q=1}^{m} W(p,q)(u_p-$

$u_q)^2 \geq 0$ (where $u_p$ and $u_q$ are entries or components of $\mathbf{u}$. Recall from Def.5 that a large $W(p,q)$ indicates nodes $v_p$ and $v_q$ influence each other strongly, their corresponding coefficients, $u_p$ and $u_q$, should be similar in order to minimize this quadratic equation. It is thus intuitive to say that, if node $v_p$ impacts the global network state, its selection in $\mathbf{U}$ will increase the utility of also selecting its neighbor $v_q$, if the weight $W(p,q)$ on the edge connecting the two nodes is high. This network-regularized selection leads to connected subnetwork structures in our solution.

Another important regularization we impose on learning the node-selection matrix $\mathbf{U}$ is to explicitly exclude those nodes that have little or no impact on global state values. These nodes are either irrelevant or redundant for the classification of global network states and excluding them makes the final selected substructures concise and easy to interpret. One of the most effective ways to accomplish this task is to impose the L1-norm [67] on each vector $\mathbf{u}$. By its nature, L1-norm will shrink some of entries in the node-selection vector $\mathbf{u}$ to zero if a sufficiently large weight is placed on this sparsity regularization. In combination with the network regularization, our overall regularized subnetwork selection objective is formulated as follows:

$$\arg\min_{\mathbf{U}} \|\mathbf{Y} - \mathbf{V}^{\mathbf{T}}\mathbf{U}\|_F^2 + \lambda_2 \sum_{\mathbf{u}\in\mathbf{U}} \mathbf{u}^{\mathbf{T}}\mathbf{C}\mathbf{u} + \lambda_1 \sum_{\mathbf{u}\in\mathbf{U}} |\mathbf{u}|, \tag{3.8}$$

where $|\mathbf{u}| = \sum_{t=1}^{m} |u_t|$. The $\lambda_1$ and $\lambda_2$ are the trade-off factors controlling respectively the shrinkage imposed on $\mathbf{u}$'s and the aggregate network topology (their impact will be demonstrated in Sec.3.4.5). A larger value of $\lambda_1$ will result in a smaller number of selected nodes while a larger value of $\lambda_2$ emphasizes the network structure.

In minimizing this objective function, note that Eq.(3.8) is not strictly convex due to the absolute term $\sum_{\mathbf{u}\in\mathbf{U}} |\mathbf{u}|$. We thus optimize it through the gradient descent approach in which each $\mathbf{u}$ is optimized individually. We rewrite the function as follows:

$$\begin{aligned}\mathcal{R}(\mathbf{u}) &= \|\mathbf{y} - \mathbf{V^T u}\|_2^2 + \lambda_2 \mathbf{u^T C u} + \lambda_1 |\mathbf{u}| \\ &= \sum\nolimits_i (y_i - \mathbf{v_i^T u})^2 + \lambda_2 \mathbf{u^T C u} + \lambda_1 \sum\nolimits_t |u_t| \end{aligned} \tag{3.9}$$

The derivative of the first term of $\mathcal{R}(\mathbf{u})$, denoted $\mathcal{R}_1(\mathbf{u})$, w.r.t. a coordinate $u_t$ is:

$$\begin{aligned}\frac{\partial \mathcal{R}_1(\mathbf{u})}{\partial u_t} &= 2 \sum\nolimits_i (y_i - u_t v_{it} - \bar{\mathbf{u}}^T \bar{\mathbf{v}}_i)(-v_{it}) \\ &= 2 \sum\nolimits_i v_{it}^2 u_t - 2 \sum\nolimits_i (v_{it})(y_i - \bar{\mathbf{u}}^T \bar{\mathbf{v}}_i) \end{aligned} \tag{3.10}$$

where $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}_i$ denote the respective vectors excluding their $t$-th entry. Likewise, taking the derivative of the second term of $\mathcal{R}(\mathbf{u})$, denoted by $\mathcal{R}_2(\mathbf{u})$, w.r.t. $u_t$ leads to:

$$\begin{aligned}\frac{\partial \mathcal{R}_2(\mathbf{u})}{\partial u_t} &= \frac{\partial}{\partial u_t} \lambda_2 \sum\nolimits_p \sum\nolimits_q u_p u_q \mathbf{C}_{pq} \\ &= \lambda_2 (\sum_{p \neq t} \mathbf{C}_{pt} u_p + \sum_{q \neq t} \mathbf{C}_{qt} u_q + 2\mathbf{C}_{tt} u_t) \\ &= \lambda_2 (\sum\nolimits_p \mathbf{C}_{pt} u_p + \sum\nolimits_q \mathbf{C}_{qt} u_q) = 2\lambda_2 \mathbf{u}^T \mathbf{C}_{\star t} \\ &= 2\lambda_2 \mathbf{C}_{tt} u_t + 2\lambda_2 \bar{\mathbf{u}}^T \bar{\mathbf{C}}_{\star t} \end{aligned} \tag{3.11}$$

where $\mathbf{C}_{\star t}$ denotes the $t$-th column of matrix $\mathbf{C}$ and again $\bar{\mathbf{C}}_{\star t}$ is the respective vector excluding its $t$-th component.

Note that our last term in Eq.(3.9) is not a smooth function; so its derivative w.r.t. $u_t$ is calculated as follows:

$$\frac{\partial \mathcal{R}_3(\mathbf{u})}{\partial u_t} = \frac{\partial \lambda_1 |\mathbf{u}|}{\partial u_t} = \begin{cases} \{-\lambda_1\} & \text{if } u_t < 0 \\ \{+\lambda_1\} & \text{if } u_t > 0 \\ [-\lambda_1, +\lambda_1] & \text{if } u_t = 0 \end{cases} \tag{3.12}$$

By combining the three derivatives, we get:

$$\frac{\partial \mathcal{R}(\mathbf{u})}{\partial u_t} = \begin{cases} \{a_1 u_j - a_2 - \lambda_1\} & \text{if } u_t < 0 \\ \{a_1 u_j - a_2 + \lambda_1\} & \text{if } u_t > 0 \\ [-a_2 - \lambda_1, -a_2 + \lambda_1] & \text{if } u_t = 0 \end{cases} \tag{3.13}$$

where, for simplicity, we have introduced $a_1 = 2\sum_i v_{it}^2 + 2\lambda_2 \mathbf{C}_{tt}$, which is always positive, and $a_2 = 2\sum_i (v_{it})(y_i - \bar{\mathbf{u}}^T \bar{\mathbf{v}}_i) - 2\lambda_2 \bar{\mathbf{u}}^T \bar{\mathbf{C}}_{\star t}$. Setting this equations to zero and solving for $u_t$ finally yields:

$$u_t = \begin{cases} (a_2 + \lambda_1)/a_1 & \text{if } a_2 < -\lambda_1 \\ (a_2 - \lambda_1)/a_1 & \text{if } a_2 > +\lambda_1 \\ 0 & \text{if } a_2 \in [-\lambda_1; +\lambda_1] \end{cases} \tag{3.14}$$

By the gradient descent optimization [67], we repeatedly update $\mathbf{u}$'s components until it converges. In what follows the convergence of our method is proven.

**Proposition 4.** *Let* $\mathbf{u}$ *be a* $m \times 1$ *vector, and* $\mathcal{R}_1(\mathbf{u}) = \|\mathbf{y} - \mathbf{V^T u}\|_2^2$ *be a scalar function of* $n$ *variables with second order derivatives defined on a convex domain* $\mathcal{D}$. *Then its second derivative is positive semi-definite and* $\mathcal{R}_1(\mathbf{u})$ *is convex.*

Proof: Given $\mathcal{R}_1(\mathbf{u}) = \|\mathbf{y} - \mathbf{V^T u}\|_2^2$ where $\mathbf{y}$ is a column in $\mathbf{Y}$, it is to see that the second derivative $\partial^2 \mathcal{R}_1(\mathbf{u})/\partial \mathbf{u}^2 = \mathbf{VV^T}$. Its quadratic form $\mathbf{b^T V V^T b}$ is non-negative given any $m \times 1$ vector $\mathbf{b}$ and thus $\partial^2 \mathcal{R}_1(\mathbf{u})/\partial \mathbf{u}^2$ is positive semi-definite. Using Taylor approximation [70] of $\mathcal{R}_1(\mathbf{u})$ near $\mathbf{u}$, with a $\gamma \in (0,1)$ and $\mathbf{u}, \mathbf{u}+\mathbf{h} \in \mathcal{D}$, we can write:

$$\mathcal{R}_1(\mathbf{u}+\mathbf{h}) = \mathcal{R}_1(\mathbf{u}) + \frac{\partial \mathcal{R}_1(\mathbf{u})}{\partial \mathbf{u}}^T \mathbf{h} + \frac{1}{2}\mathbf{h}^T \frac{\partial^2 \mathcal{R}_1(\mathbf{u}+\gamma\mathbf{h})}{\partial \mathbf{u}^2}\mathbf{h}$$

As shown above, the second derivative of $\mathcal{R}_1(\mathbf{u})$ is positive semi-definite, hence, the last term in this equation is non-negative, resulting in the following inequality:

$$\mathcal{R}_1(\mathbf{u}+\mathbf{h}) \geq \mathcal{R}_1(\mathbf{u}) + \frac{\partial \mathcal{R}_1(\mathbf{u})}{\partial \mathbf{u}}^T \mathbf{h} \tag{3.15}$$

Now, given any $\mathbf{u}'$ and $\mathbf{u}''$ in $\mathcal{D}$ and let $\zeta \in (0,1)$ be a scalar, we have $\mathbf{u} = \zeta\mathbf{u}' + (1-\zeta)\mathbf{u}'' \in \mathcal{D}$ due to the convexity of $\mathcal{D}$. By Eq.(3.15), it follows that:

$$\begin{cases} \mathcal{R}_1(\mathbf{u}') & \geq \mathcal{R}_1(\mathbf{u}) + \frac{\partial \mathcal{R}_1(\mathbf{u})}{\partial \mathbf{u}}^T (\mathbf{u}' - \mathbf{u}) \\ \mathcal{R}_1(\mathbf{u}'') & \geq \mathcal{R}_1(\mathbf{u}) + \frac{\partial \mathcal{R}_1(\mathbf{u})}{\partial \mathbf{u}}^T (\mathbf{u}'' - \mathbf{u}) \end{cases} \tag{3.16}$$

Multiplying the first row in Eq.(3.16) by $\zeta$, the second row by $(1-\zeta)$ and summing up yields:

$$\begin{aligned} &\zeta\mathcal{R}_1(\mathbf{u}') + (1-\zeta)\mathcal{R}_1(\mathbf{u}'') \\ &\quad \geq \mathcal{R}_1(\mathbf{u}) + \frac{\partial \mathcal{R}_1(\mathbf{u})}{\partial \mathbf{u}}^T \left(\zeta\mathbf{u}' + (1-\zeta)\mathbf{u}'' - \mathbf{u}\right) = \mathcal{R}_1(\mathbf{u}) \end{aligned}$$

which shows the convexity of $\mathcal{R}_1(\mathbf{u})$. □

**Proposition 5.** *Let* $\mathbf{u}$ *be* $m \times 1$ *vector and let* $\mathcal{R}_2(\mathbf{u}) = \lambda_2 \mathbf{u}^{\mathbf{T}}\mathbf{C}\mathbf{u}$ *be a scalar function of* $n$ *variables with second order derivatives defined on a convex domain* $\mathcal{D}$*. Then* $\mathcal{R}_2(\mathbf{u})$ *is a convex function.*

Proof: With similar arguments as those in Proposition 1, it is straightforward to prove $\mathcal{R}_2(\mathbf{u})$ is a convex function given that $\mathbf{C}$ is a positive semi-definite matrix as defined in Eq(3.7). □

**Proposition 6.** *Given* $\mathcal{R}(\mathbf{u}) = \mathcal{R}_1(\mathbf{u}) + \lambda_2\mathcal{R}_2(\mathbf{u}) + \lambda_1\mathcal{R}_3(\mathbf{u})$ *where* $\lambda_1, \lambda_2$ *are non-negative, our optimization function* $\mathcal{R}(\mathbf{u})$ *formulated in Eq.*(3.9) *is convex.*

Proof: For any $\mathbf{u}', \mathbf{u}'' \in \mathcal{D}$, $\zeta \in (0,1)$, where $\mathbf{u} = \zeta\mathbf{u}' + (1-\zeta)\mathbf{u}''$, the function $\mathcal{R}_3(\mathbf{u})$

is also convex since:

$$\begin{aligned}\zeta\mathcal{R}_3(\mathbf{u}') + (1-\zeta)\mathcal{R}_3(\mathbf{u}'') &= \zeta|\mathbf{u}'| + (1-\zeta)|\mathbf{u}''| \\ &= |\zeta\mathbf{u}'| + |(1-\zeta)\mathbf{u}''| \geq |\zeta\mathbf{u}' + (1-\zeta)\mathbf{u}''| = \mathcal{R}_3(\mathbf{u})\end{aligned}$$

in which we have used the triangle inequality in the second row. Based on (i) Proposition 4, 5, (ii) the fact that the summation of convex functions over a convex domain $\mathcal{D}$ is also convex [70] and (iii) given that both $\lambda_1, \lambda_2 \geq 0$, it follows that $\mathcal{R}(\mathbf{u})$ is a convex function. □

**Corollary 1.** *Proposition 6 guarantees the convergence of our gradient descent method and the solution it finds is the unique optimum.*

### 3.3.3 Subnetwork Structures

Due to the L1-regularization, the number of non-zero coefficients in each $\mathbf{u} \in \mathbf{U}$ is usually small according to the setting of $\lambda_1$. Let us further index $\{\mathbf{u}_i\}_{i=1}^d$ for column vectors in $\mathbf{U}$ and assume that, for a specific $\lambda_1$ value, $c$ is the number of non-zero coefficients from each vector $\mathbf{u}_i$. In a general case, these coefficients might be slightly different across $\{\mathbf{u}_i\}_{i=1}^d$. Thus, we first rank the largest absolute non-zero entries across $\{\mathbf{u}_i\}_{i=1}^d$ and aggregate them into a single final vector $\mathbf{u}$ as follows:

$$\mathbf{u} = (u_1, \ldots, u_m)^T \text{ with } u_p = \max_i |\mathbf{u}_{i,p}| \tag{3.17}$$

where $\mathbf{u}_{i,p}$ is the $p$-th entry of eigenvector $\mathbf{u}_i$. We then select nodes according to the top $c$ ranking entries in $\mathbf{u}$. The subnetworks induced by these nodes are obtained by matching with the nodes of $S$ defined in Def.5 along with connecting edges encoded in $E$.

### 3.3.4 Algorithm Complexity

We name our framework DIPS—DIscovering Predictive subnetwork Structures, and analyze its complexity in what follows. We first build the two similarity graphs $\mathcal{G}^+$ and $\mathcal{G}^-$ which takes $O(mn^2)$ in the worst case and can reduce to $O(mn \log n)$ with the use of a kd-Tree structure [51]. Both $\mathbf{D}$ and $\mathbf{L}$ in Eq.(3.6) are of size $n \times n$ and the eigen-decomposition for our first step takes $O(n^2 \log n)$ with the Lanczos technique. In the second step, only $a_2$ is impacted by the change of $\mathbf{u}$'s and its computation takes $O(m)$ assuming that the sizes of $\mathbf{u}$, $\mathbf{v}_i$ and $\bar{\mathbf{C}}_{\star j}$ are all $m \times 1$, while the time for $\mathbf{u}$'s updates in Eq.(3.14) can be considered as a constant. We need to update every component of $\mathbf{u}$, the computation thus amounts to $O(dMm^2)$ assuming $M$ is the maximal number of iterations until convergence of all $d$ vectors $\mathbf{u}$'s. The overall complexity is thus $O(mn \log n + n^2 \log n + dMm^2)$.

### 3.3.5 Discussion

It is worthwhile to mention that both the sparseness and network-constrained regularizations can be directly integrated into our first step formulated in Eq.(3.4). Nevertheless, that combination will result in a harder non-convex optimization problem, rendering standard convex optimization theory not applicable. In order to address this difficulty, we can either apply the minorization-maximization algorithm as developed in [71, 72] or introduce another variable to convert the problem into the bi-convex optimization [73]. However, in both cases, it is not certain whether stable optimum solutions can be achieved due to the original non-convexity optimization. We leave these issues for future studies.

## 3.4 Experiments on Various Types of Real-World Networks

### 3.4.1 Methodology

We compare the performance of DIPS against the following representative methods: (1) MINDS [44]—among the first subnetwork sampling approaches for global-state network classification, and (2) NGF [52]—a random forest based approach incorporating the structure of PPI networks. Additionally, we also compare DIPS against techniques that do not incorporate a network topology: (3) DIPSw/o, a variant of DIPS without network constraints, (4) SVM by first performing SVD for dimensionality reduction (retaining 95% of the singular values). For each of our datasets, we evaluate the performance of all algorithms via 5-fold stratified cross validation (CV). The parameters of DIPS are chosen based on the estimated prediction accuracy within every 4 training folds and tested on the left-out fold, following the protocol in [74, 75]. Unless otherwise specified, its parameters are tuned within the ranges: $k \in [30-80]$ with a step of 10; $\lambda_1 \in [0.15-0.005]$ and $\lambda_2 \in [0.1-1000]$ (in log-scale). For MINDS, the minimum discriminative potential for each network constrained decision tree is set to 0.8 and $K = 200$ [44]. For NGF, we use the Gini index for tree building and the threshold to stop growing a tree $\epsilon = 0.02$ [52]. In both MINDS and NGF we sample/grow $10,000$ tree substructures.

### 3.4.2 Image Network Dataset

Since most network datasets (presented next) lack ground truth for the best features, we conduct an experiment on image data, namely the CMUFace [76], in order to evaluate the relevance of uncovered predictive subnetworks *via visualization*. Though images do not originally involve explicit network structures, studying them as graphs has been deemed advantageous [77]. In particular, graphs enable the discovery of local image properties, especially in image denoising since many pixels may be missing or tampered. Following [77], we first down-sample the number of pixels to 50% and construct a common network topology relying on the remaining pixels. Within each image, a pixel corresponds to a node and has edges connected to the 5 nearest pixels. The value associated with an edge is inversely proportional to its length, and a value associated with a node is the grey level of the corresponding pixel. For the sake of visualization, we work with all straight pose images where open eyes and sunglasses are selected as the global network states, resulting in 156 network samples, each containing $1,920$ nodes and $11,172$ edges. The subnetwork structures corresponding to the "eye" areas are therefore the ground truth features.

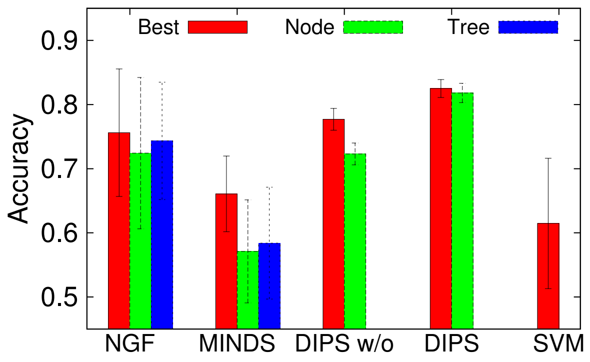


Figure 3.2: Prediction in image networks. Red bars show the best accuracies; Green bars show accuracies with substructures having 80 nodes; Blue bars show accuracies with 80 trees used as features in NGF and MINDS.

**Prediction Performance:** In Fig.3.2, we report the prediction accuracy of all algorithms averaged from 5-fold CV for 2 cases: (i) the best performance (red bars) not constrained by the number of selected nodes, and (ii) the best performance when the node number (i.e., value of $c$ in Sec.3.3.3) in the predictive subnetworks is 80 (equal to the number of ground truth nodes). Since NGF and MINDS are ensemble methods, we further report when their selected trees are counted as the number of features (blue bars).

As seen from Fig.3.2, both DIPS and DIPSw/o usually outperform the sampling methods and SVM. Only DIPSw/o is inferior to NGF when the selected nodes are constrained to 80. In fact, DIPSw/o achieves the best accuracy when its substructures contains only 30 nodes while DIPS achieves the best accuracy of 82.1% when selecting 71 nodes to form subnetworks. Both NGF and MINDS need a much higher number of nodes, respectively 355 and 202, to achieve their best accuracy. When the number of nodes is limited to 80, NGF achieves only 71.9% of accuracy, which is considerably lower than DIPS's 81.1%.

| Methods | 10 | 30 | 50 | 70 | 90 |
|---|---|---|---|---|---|
| NGF-node | 0.58(.10) | 0.66(.10) | 0.68(.06) | 0.72(.08) | 0.72(.09) |
| NGF-tree | 0.68(.07) | 0.72(.10) | 0.74(.09) | 0.72(.10) | 0.74(.10) |
| MINDS-node | 0.56(.05) | 0.57(.10) | 0.58(.08) | 0.57(.10) | 0.58(.09) |
| MINDS-tree | 0.58(.07) | 0.59(.10) | 0.58(.10) | 0.57(.07) | 0.59(.07) |
| DIPSw/o | 0.76(.02) | 0.77(.02) | 0.75(.02) | 0.72(.02) | 0.71(.02) |
| DIPS | 0.76(.02) | 0.79(.02) | 0.82(.02) | 0.82(.01) | 0.81(.01) |

Table 3.2: Prediction accuracy on image networks for varying number of features in predictive substructures between 10 and 90 (std. dev. in brackets)

Table 3.2 provides more insights into the performance of all methods for varying the number of selected nodes (trees in NGF- and MINDS-tree) from 10 to 90. The classification rates of NGF and MINDS show similar trends of achieving higher accuracies for larger numbers of selected nodes/trees, while generally stabilizing when more than 70 nodes/trees are selected. DIPSw/o outperforms DIPS for a small number of selected nodes but its classification rate quickly deteriorates as the number of selected nodes increases. As seen from the last row of Table 3.2, DIPS consistently maintains the high accuracy rate and its performance only drops when the number of nodes exceeds the size 80 of the ground truth nodes.

**Substructure Visualization:** We also explore the subnetworks selected by each algorithm. In Fig.3.3, we plot the selected substructures aggregated from 5-fold CV for all algorithms by limiting the selected nodes to 80. Though several subnetworks are found

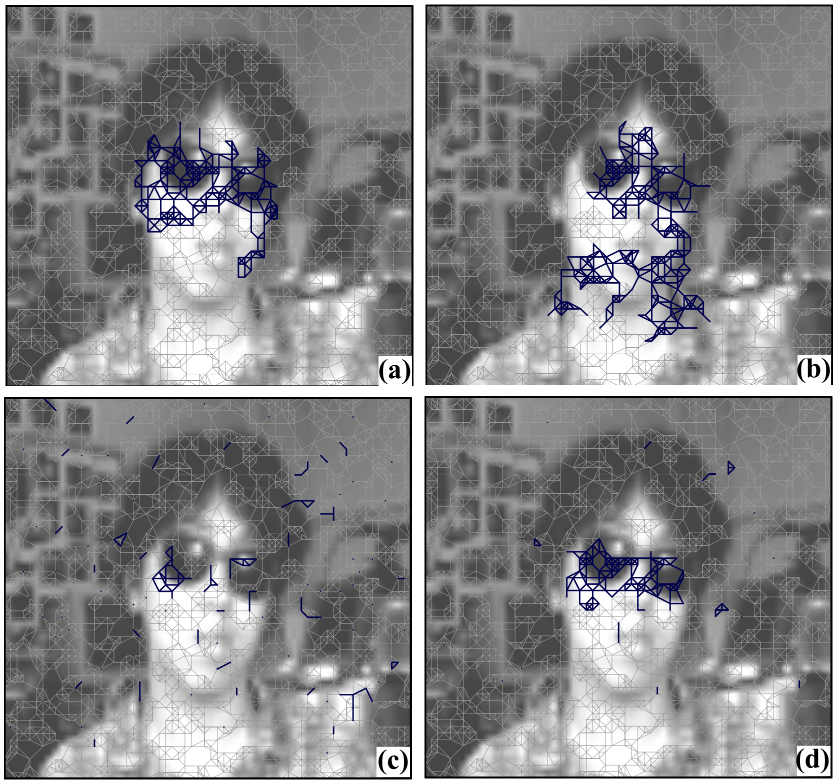


Figure 3.3: Subnetworks selected by each method in image data. Overall network is shown in grey while selected subnetworks are shown in blue (a background image is selected from sunglass class). (a) NGF (b) MINDS (c) DIPSw/o (d) DIPS.

in the area of the "eyes", NGF shows many irrelevant substructures selected across the space. Most subnetworks found by MINDS are well connected and less fragmented, yet some irrelevant regions are included. On the other hand, DIPSw/o (Fig.3.3c) discovers a fragment of ground truth nodes but many irrelevant isolated nodes are also seen throughout the entire network. This is justified by the lack of network regularization in this model. Overall, none of the methods discover better features than DIPS as visualized in Fig.3.3d. DIPS's substructures are consistent across all folds and highly overlapping with the ground truth region.

### 3.4.3 Brain Network Dataset

The second dataset we examine is collected from the ADNI project[1] focusing on the Alzheimer's disease. It consists of resting state fMRI scans captured from normal control (NC) and Alzheimer's disease (AD) subjects. For each fMRI scan, we employ the FSL toolbox [2] to extract sequences of responses from 112 anatomical volumes of interest (AVOI), each representing a brain region. We focus on the resulting functional brain network [43] by building an edge-dual network: a node in the edge-dual network is defined if the temporal correlation between the two underlying brain regions is $\geq 0.5$. The node value is the amount of correlation degree, and an edge connects two nodes if they share one of the underlying brain regions. This setting ensures that any predictive substructure involves at least two brain regions, which is consistent with the task of analyzing *functional* brain connectivity [65, 43]. After pre-processing, the dataset consists of 173 network samples, each having on average of $2,948$ nodes and $191,750$ edges.

**Prediction Performance:** We plot in Fig.3.4(a) the best prediction accuracy (red bars) of all algorithms, and the best prediction accuracy in which the predictive substructures involve 100 nodes (blue), and 100 trees (green) for NGF- and MINDS-tree variants. Performance with smaller settings of selected nodes is reported in Table 3.3. As observed, while NGF and MINDS are designed to find trees with low training errors, they both do not perform noticeably better than the baseline SVM. In the case of NGF, a root node selection in a densely connected area has limited impact. Both NGF and MINDS suffer from an exponential number of possible subnetworks within the densely connected brain graphs. In contrast, DIPS avoids this exponential complexity through subspace learning and network-constrained regularization. Examining the detailed performance in Table 3.3 provides further evidence of the accuracy gap between the two approaches.

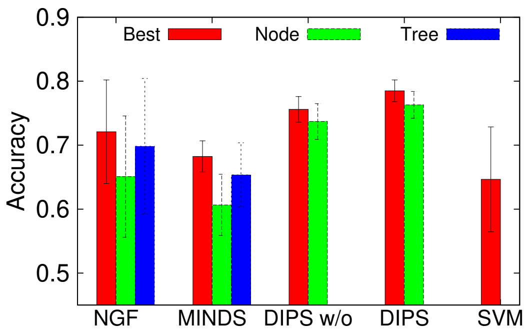
[1]http://www.adni-info.org/

While DIPS achieves 76% with 90 nodes in its predictive substructures, NGF and MINDS can only get between 65% and 68% in their best performance.

| Methods | 10 | 30 | 50 | 70 | 90 | 100 |
|---|---|---|---|---|---|---|
| NGF-node | 0.57(.08) | 0.64(.03) | 0.65(.02) | 0.66(.01) | 0.65(.01) | 0.65(.01) |
| NGF-tree | 0.64(.02) | 0.65(.01) | 0.65(.01) | 0.65(.02) | 0.65(.03) | 0.60(.03) |
| MINDS-node | 0.58(.08) | 0.61(.05) | 0.64(.02) | 0.61(.05) | 0.61(.03) | 0.61(.03) |
| MINDS-tree | 0.64(.02) | 0.61(.03) | 0.65(.01) | 0.66(.02) | 0.68(.03) | 0.67(.02) |
| DIPSw/o | 0.63(.02) | 0.71(.02) | 0.74(.03) | 0.76(.03) | 0.76(.03) | 0.74(.03) |
| DIPS | 0.63(.03) | 0.68(.03) | 0.71(.02) | 0.75(.02) | 0.76(.02) | 0.75(.04) |

Table 3.3: Accuracy rates on brain networks with varying numbers of selected features (std. dev. in brackets)

**Predictive Substructures:** Unlike image data where ground truth substructures can be marked via visualization, defining predictive features for Alzheimer's disease remains an ongoing challenge [64]. Nevertheless, one way to evaluate the quality of uncovered substructures from each technique is through their consistency in cross validation. Substructures that are consistently found across training folds are likely the disease-related biomarkers and they should be the first candidates for further investigation.



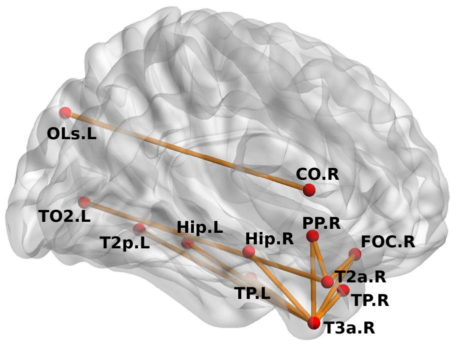


Figure 3.4: Performance on brain networks. (a) Prediction accuracy of all methods (b) Common predictive substructures found by DIPS.

We report the ratio between the intersection and union of substructures selected across training folds of each algorithm. Specifically, we observe that both sampling techniques show less overlapping substructures for all sizes of selected nodes varied from 10 to 100, with NGF starts reporting substructures of limited overlap (ratio of 0.017) only when its tree ensembles contain at least 70 unique nodes. DIPS consistently shows overlap ratios between 0.034 and 0.055 for the number of predictive nodes between 30 and 100. These results reveal that our method selects more stable predictive substructures across all training sets. Looking into the obtained structures, we observe that DIPS uncovers 11 nodes, out of 70 nodes, in common across all folds and these nodes form two large subnetworks as depicted in Fig.3.4(b). By further investigation, we observe that these substructures largely reside in the temporal lobe (`TP.L, TP.R, T2a.R, T2p.L, T3a.R`) and the hippocampus (`Hip.L/Hip.R`), which are the two important brain regions strongly impacted by Alzheimer's disease [78, 79].

### 3.4.4 Gene Network Datasets

The last two datasets we use contain microarray co-expression data from studies on embryonic development [52] and liver metastasis in human [1], combined with the corresponding functional PPI network [55]. The former dataset contains $1,321$ genes, $5,227$ edges and 35 subjects divided into two global labels based on the developmental tissue layers [52], while the latter comprises $7,383$ genes, $251,916$ edges from 123 subjects labeled as disease/non-disease states.

**Prediction Performance:** Fig.3.5(a) compares the prediction accuracy in both datasets. Due to space constraints, we only report the best performance with fine-tuned parameters for each technique. Other than SVM, all techniques perform competitively on the embryonic dataset, achieving more than 70% of prediction accuracy averaged from all folds. MINDS performs better than NGF on these two microarray datasets, and also better than DIPSw/o on the embryonic one. Nevertheless, the performance of DIPS is

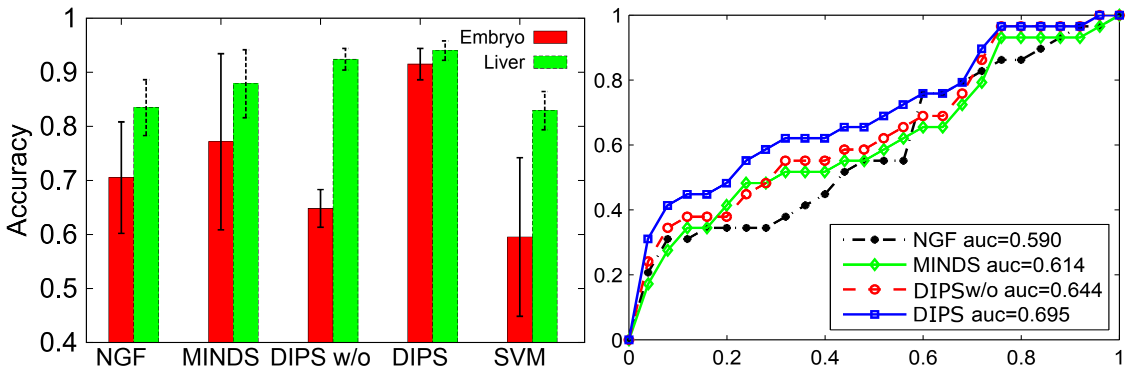


Figure 3.5: (a) Prediction accuracy in embryonic development and liver metastasis datasets combined with PPI networks. (b) ROC performance w.r.t. ground truth genes provided in [1] for liver metastasis.

by far the best, dominating both the baseline SVM and the sampling methods, with a significant accuracy advantage on the liver metastasis dataset, which has a larger number of nodes. Moreover, DIPS's relatively small standard deviation in accuracy in both network datasets demonstrates its stability in predicting global network states across all folds.

**Predictive Substructures:** As in the case of brain networks, there are no exact ground truths for microarray data. Nevertheless, we choose the more challenging liver metastasis dataset, along 39 cancer-related genes identified in [1] to serve as a ground truth set, and further investigate the overlap of subnetworks selected by DIPS in relation to the ground truth. At its best classification rate, DIPS's substructures involve 65 nodes from each training set and cumulatively 194 nodes from all training sets. Out of these, two subnetworks (shown in Fig.3.6) are consistently observed across all folds. The genes `MMP1`, `TIMP1`, `MMP2` and `TNFSF11` (shaded in Fig.3.6) are particularly interesting since they are in agreement with the findings in [1] that identify these genes as the main targets related to the liver metastasis.

As a comparison against other techniques, we evaluate the selected subnetworks in

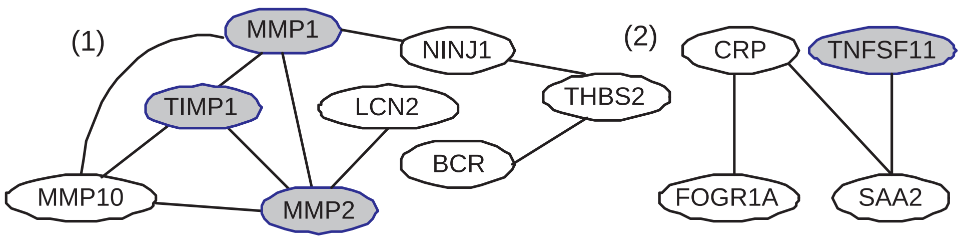
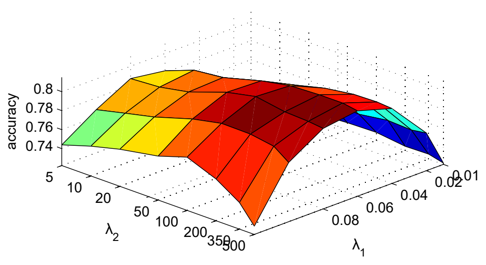


Figure 3.6: Typical predictive subnetworks uncovered by DIPS from liver metastasis. Shaded nodes correspond to domain-relevant genes.

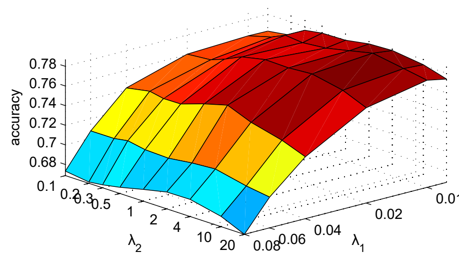
terms of ROC performance in identifying genes in the ground truth. Fig.3.5(b) plots the ROC curves of competing methods when they obtain their best prediction rate. For node ranking in DIPS and DIPSw/o, we rely on Eq.(3.17) while in NGF and MINDS, we rank nodes based on their frequency of occurrence in sampled trees. The ROC curves demonstrate that DIPS selects more ground truth nodes than NGF and MINDS for small false positive rates. As this rate increases, the behavior of all methods become quite similar since none of them are able to discover all ground truth genes. However, the higher rate of true positive makes DIPS a better candidate for identifying new target genes associated with the disease. This property is very important in practice since validating a new target gene is costly and a method with better ROC performance can provide better candidate genes to test experimentally.

### 3.4.5 Impact of regularization factors

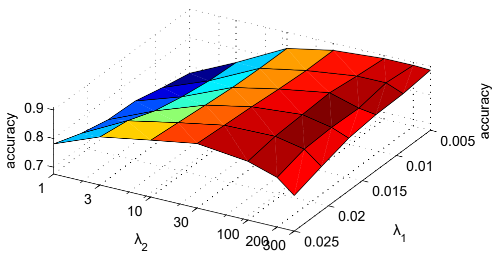
In order to provide more insights into the performance of DIPS, we conduct a series of experiments to examine the impact of the two regularization factors on the prediction accuracy. Recall that $\lambda_1$ controls the number of selected nodes, while $\lambda_2$ enforces the network connectivity within selected nodes. In Fig.3.7, we show the relationships between the two factors and the prediction accuracy on the 4 datasets. From the plots, a trend is observed that as $\lambda_1$ decreases, more nodes are selected, and the prediction accuracy keeps increasing. However, when $\lambda_1$ is too small, the prediction rate deteriorates in all datasets, especially on the image data. This can be explained by the fact that a very small $\lambda_1$ leads to an overwhelming number of selected substructures, which can potentially contain many irrelevant nodes and lead to over-fitting. A similar trend is also observed by varying $\lambda_2$. A small $\lambda_2$ leads to a low accuracy as the connectivity information is almost disregarded, while a large $\lambda_2$ favors tightly connected subnetworks whose local node values may not strongly influence the global network properties. The most important observation from the four plots is that all surfaces have convex-shapes which suggests that the optimal model can be selected at the peak point at which it has a small number of selected nodes to favor a simple model, while the network connectivity in the predictive substructures is maximized.



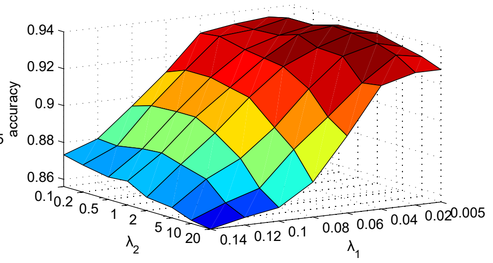
(a) Imaging networks



(b) Brain networks



(c) Embryonic develpoment



(d) Liver metastasis

Figure 3.7: Impact of regularization factors $\lambda_1$ and $\lambda_2$ on the prediction accuracy of DIPS in all datasets.

### 3.4.6 Scalability

We also evaluate the scalability of our proposed algorithms. Our implementation employs the Lanczos method [80] to compute the leading eigenvectors. To provide an idea about typical running times: DIPS takes 9s and 140s to complete a round of cross validation on the image and brain networks, whereas for the embryonic and liver cancer data, it takes 182s and 17s respectively. These computations are twice as fast as DIPSw/o since the network regularization term allows DIPS to converge much faster. NGF and MINDS take similar time on the image network data at 147s and 165s, and embryonic at 109s and 110s, respectively. However, both methods require much more time on the denser networks of liver and brain. In particular, they both take more than 1900s on the brain networks, due to training and examining a huge number of decision trees. DIPS thus shows much better scalability against the subnetwork sampling methods due to its efficient and effective subspace learning approach with network regularization.

## 3.5 Related Work

Recent research in bioinformatics and systems biology has demonstrated the untility of network structure for both classification [52, 44] and regression [62, 63] tasks. NGF [52] is one representative classification approach based on a random forest classifier that also incorporates biological constraints encoded by a protein-protein interaction (PPI) network. The random forest is iteratively built over features within a connected subgraph of the PPI.

Another recent classification scheme, MINDS [44], builds decision trees constrained by the network topology and employs Markov Chain Monte Carlo to sample the space of connected structures. Both MINDS and NGF are sampling ensemble methods that classify network instances based on majority voting. In contrast, our work avoids searching in the exponential space of subnetworks by learning a discriminative subspace and selecting predictive feature substructures within that subspace. In addition, our approach directly optimizes the compactness (feature selection) and discriminative power (accuracy) of features.

Our work also fundamentally differs from other subspace learning approaches [68, 66] that either focus on classification in a subspace instead of feature selection [68], or do not consider the network structure to obtain a subspace [66]. Our method selects a controlled-size subset of the features in a network-aware manner, which can then be used for classification or other types of analysis.

Graph classification based on frequent/discriminative substructures has also been considered for chemical and program call graph databases [57, 56]. Nodes in this setting are labeled by discrete labels (e.g. chemical element or function name) and the task is to find substructures that are *frequent* in one class of database graphs and *infrequent* in another.

The original database graphs are represented and classified as binary feature vectors encoding the inclusion/exclusion of discriminative subgraphs.

Our setting is different from frequent substructure classification as we work with continuous labels on nodes whose values are directly used for classification of the global states. The learnt substructures by our method are structurally contained in all network samples and selected feature node values collectively discriminate between global network states.

Dynamic network mining approaches also aim to discover subnetworks of interest in multiple discrete snapshots of an evolving network or in a multi-layer network [10, 81, 61, 82]. The goal in this line of work, however, is not classification of global network states, but the discovery of abnormal substructures that persist in time or across network layers.

## 3.6 Conclusions

We proposed DIPS, a novel framework for mining predictive substructures in global-state networks. Unlike existing techniques, DIPS avoids the exponential complexity of subnetwork sampling through a two-step scheme: (i) subspace learning, and (ii) predictive substructures selection with network regularization. We utilize two mathematically appealing approaches of spectral graph learning and gradient descent optimization, and we show that DIPS converges to a single optimum solution—a desired property that is hard to achieve by sampling approaches. Through experimental analysis over four real-world datasets, we demonstrated the advantageous performance of DIPS not only based on classification rate but also in terms of domain relevance of the discovered predictive subnetworks.

# Chapter 4

# Subnetwork Mining with Spatial and Temporal Smoothness

## 4.1 Introduction

Network data arises in a number of application domains ranging from neuroscience, biology, geography, to social sciences [4, 3]. Accordingly, network analysis has emerged as a major paradigm for exploring the complex processes behind the observed network data. Compared to high dimensional data, the analysis over the network data is more challenging due to the nature of inter-dependence among the data entities. Moreover, most network data are not static but keep changing over time. These changes not only happen at the local interactions within the network samples but also eventually influence the global behaviors of the network instances. Consider brain networks associated with the Alzheimer's disease (AD) as an example [27]. The death of neurons and synapses can impact the cognitive performance of some brain regions. At an early stage, the disease can be as mild as making the patient difficult in remembering some recent events; however, as the disease progresses with more impacted brain regions due to the neuronal connectivity, the decline of overall cognitive function can be experienced, including confusion, difficulty speaking, and eventually leading to fatality. Detecting brain subnetwork markers that highly predict and evolve along with the progression of the disease is thus an important task since it not only helps to characterize the disease but further provides the means to plan the right treatments at the right stage of the disease.

Analyzing network structural data has been widely studied in the literature with existing work focusing on community extraction [4, 83], frequent subgraph mining [84, 85], outlier detection [3, 86], and graph classification [87, 88]. Although these studies have substantially promoted our understanding, they tend to be explored in a simple setting of a single network (e.g., community/cluster discovery), or extended to multiple-network settings (e.g., frequent subgraph mining), yet without fully investigating the essential relationship between local network processes and global network behaviors. In this work, we broaden the research scope to a more complex setting in which we deal with multiple network samples, each of which is associated with a global label that reflects the current state of the entire network (e.g., disease stage, climate condition). Both local network interaction and global network state can evolve over time. The task is to uncover a small set of local network processes that have maximum impact on the global network states such that their evolution can be used to predict the evolution of global network states.

Certainly, one may argue that it is possible to view each network sample as a collection of edges and apply typical feature selection techniques like mutual information [89], statistical t-test [90], or perform PCA/SVD to generate new features prior to the analysis. While such an approach significantly reduces the number of features to be analyzed, the obtained models often lack domain relevance since the nature of interaction among network entities is completely ignored. A more feasible approach is to apply an effective graph classification method [87] to categorize network samples with different global states. Though this approach can be feasible for the goal of prediction, its newly generated features (typically in the form of frequent subgraphs [87, 85]) lack temporal smoothness, and thus are less successful in interpreting and explaining the intrinsic network processes governing global network properties.

We present a novel algorithm to discover a succinct set of informative subnetworks that are highly predictive w.r.t. the development of the global network states. Due to the smooth transition of global states, these local subnetwork processes do not change abruptly from state to state, but rather develop smoothly along with the progression of the network states. Our algorithm is developed in the framework of logistic regression that fits the model for multi-states of network samples. Each global state of networks is characterized by a parametric vector whose coefficients are learnt with regularization on both the network topology (i.e., spatial smoothness) and the transition between any two adjacent global network states (i.e., temporal smoothness). In order to ensure that only the most predictive subnetworks will be learnt, we further exploit the sparsity-inducing $L1$-norm imposed on each parametric vector to remove edges that have little or no impact on the progression of global network states. The proposed formulation is challenging to solve due to the introduction of non-smooth $L1$-norm term. We, however, will show that the developed function is convex and our solution based on the steepest descent is practically efficient. Our contribution can be summarized as follows: (i) We propose and motivate a novel problem to study the impact of local subnetwork processes on global network properties that both evolve over time. (ii) We propose an objective function to learn local subnetwork processes that are highly predictive for the development of global network states. (iii) We theoretically prove that the formulated function is convex (though not strictly) and develop a novel gradient descent algorithm to optimize for a global optimum solution. (iv) We extensively evaluate and demonstrate the appealing performance of the proposed technique on both synthetic and real-world applications.

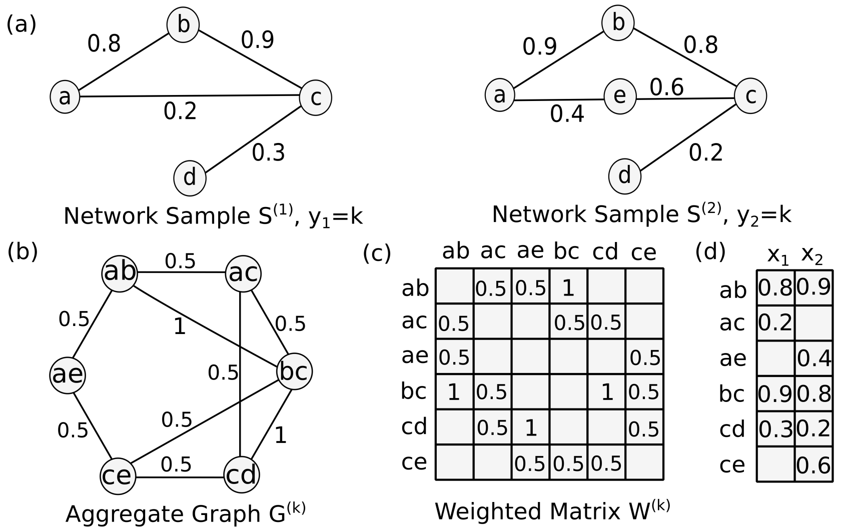
## 4.2 Preliminaries

**Definition 3.** (Network sample) *Let $S^{(i)} = (\mathcal{N}, \mathcal{E}_i)$ be a network sample, where $\mathcal{N}$ is a set of pre-defined nodes, $\mathcal{E}_i \subseteq \mathcal{N} \times \mathcal{N}$ is a set of undirected edges. There is a labeling function $\mathcal{F}$ operating on local edges' values that maps each edge to a real number $\mathcal{F} : \mathcal{E}^{(i)}_{uv} \rightarrow \mathbb{R}$.*

We denote $\mathcal{DS} = \{S^{(1)}, S^{(2)}, \ldots, S^{(n)}\}$ as the database that consists of $n$ network samples. Each network $S^{(i)}$ is further associated with a label $y_i$, indicating its global network state. Each $y_i$ receives a discrete value in $\{1, 2, \ldots, K\}$ and this order reflects the temporal evolution over global network states. It is important to note that indexing $(i)$'s are *not* networks' timestamps. Instead, we consider temporal development based on values of global network states. For example, $S^{(i)}$'s can be snapshots captured from a social or traffic network in *multiple* days, and $y_i$'s reflect whether such snapshots are in the morning, afternoon, or evening hours. Likewise, $S^{(i)}$'s can be brain networks scanned from *multiple* subjects, and associated $y_i$'s label their disease-stages advanced from mild to moderate to severe condition. The temporal development is thus at the *group* level, instead of a network individual. For each global state, we define an aggregate graph that generalizes network topology of all network samples having that network state. Prior to that, let us define the adjacency of two edges in a network sample as follows.

**Definition 4.** (Edge Adjacency) *Let $S^{(i)} = (\mathcal{N}, \mathcal{E}_i)$ be a network sample characterized by Def.3, we define a pair $\{\mathcal{E}^{(i)}_{uv}, \mathcal{E}^{(i)}_{st}\} \in \mathcal{E}^{(i)}$ as adjacent edges if they have one node in common; otherwise, we call $\{\mathcal{E}^{(i)}_{uv}, \mathcal{E}^{(i)}_{st}\}$ non-adjacent or distant edges of $S^{(i)}$.*

**Definition 5.** (Aggregate Graph for $k$-th state) *Let $\mathcal{DS}_k = \{S^{(i)} \in \mathcal{DS} | y_i = k\}$, we define $\mathcal{G}^{(k)} = (\mathcal{V}^{(k)}, \mathcal{L}^{(k)}, \mathcal{W}^{(k)})$ as an aggregate graph that captures edge adjacency of all networks in $\mathcal{DS}_k$, where $\mathcal{V}^{(k)} = \{v_1, \ldots, v_m\}$ is the vertex set with each $v_p$ corresponds to an edge found in at least one $S^{(i)} \in \mathcal{DS}_k$; $\mathcal{L}^{(k)} \subseteq \mathcal{V}^{(k)} \times \mathcal{V}^{(k)}$ is the set of links. A link $\mathcal{L}^{(k)}_{pq}$ connects* *vertices $v_p$ and $v_q$ if the two corresponding edges are adjacent in at least one $S^{(i)} \in \mathcal{DS}_k$ according to Def.4. Each link $\mathcal{L}_{pq}^{(k)}$ is associated with a non-negative value $\mathcal{W}_{pq}^{(k)}$ encoding the fraction of $S^{(i)}$'s in $\mathcal{DS}_k$ having that edge-adjacency in their network structures.*


(a)
0.8
0.9
0.2
0.3
a
b
c
d
Network Sample $S^{(1)}$, $y_1$=k
0.9
0.8
0.4
0.6
0.2
a
b
e
c
d
Network Sample $S^{(2)}$, $y_2$=k
(b)
ab
ac
ae
bc
ce
cd
0.5
0.5
1
0.5
0.5
0.5
0.5
1
0.5
Aggregate Graph $G^{(k)}$
(c)
Weighted Matrix $W^{(k)}$
(d)

| | ab | ac | ae | bc | cd | ce |
|---|---|---|---|---|---|---|
| ab | | 0.5 | 0.5 | 1 | | |
| ac | 0.5 | | | 0.5 | 0.5 | |
| ae | 0.5 | | | | | 0.5 |
| bc | 1 | 0.5 | | | 1 | 0.5 |
| cd | | 0.5 | 1 | | | 0.5 |
| ce | | | 0.5 | 0.5 | 0.5 | |

| | $x_1$ | $x_2$ |
|---|---|---|
| ab | 0.8 | 0.9 |
| ac | 0.2 | |
| ae | | 0.4 |
| bc | 0.9 | 0.8 |
| cd | 0.3 | 0.2 |
| ce | | 0.6 |

Figure 4.1: (a) network samples; (b) aggregate graph; (c) weighted matrix; (d) vectors encoding edge values.

Fig.4.1 shows a simple example to illustrate the concepts presented in the three definitions above. Two network samples having the same $k$-th state are shown in Fig.4.1(a) while their aggregate graph $\mathcal{G}^{(k)}$ is depicted in Fig.4.1(b). A value of 0.8 associated with edge $\{ab\}$ in $S^{(1)}$ encodes the degree to which nodes $a$ and $b$ are related in $S^{(1)}$, whilst the value of 1 associated with link $\{ab, bc\}$ in $\mathcal{G}^{(k)}$ indicates that two edges $\{ab\}$ and $\{bc\}$ are found adjacent in both $S^{(1)}$ and $S^{(2)}$. In the following, we refer to aggregate graphs shortly as *graphs*, and associate the terms "vertex" and "link" specifically with such graphs. Fig.4.1(c) shows $\mathcal{W}^{(k)}$ as a matrix representing graph $\mathcal{G}^{(k)}$.

## 4.3 Multinomial Logistic Regression

For the class of network applications addressed in this study, though network samples may have considerably different values associated with local edges, their network topologies are generally stable across samples (e.g., human brain networks, snapshots from a sensor or social network, etc.). We therefore utilize a high dimensional vector $x_i = [x_{i1}, \ldots, x_{im}]^T \in \mathcal{R}^m$ to store the local edge values of a network sample $S^{(i)}$ (examples of $x_i$'s are illustrated in Fig.4.1(d)). Under the framework of logistic regression, we directly model the posterior probability of a $k$-th state imposed on an input sample $x_i$ as follows:

$$P(y_i = k|x_i, \Theta, b) = \frac{\exp(\theta_k^T x_i - b_k)}{\sum_{j=1}^{K} \exp(\theta_j^T x_i - b_j)} \tag{4.1}$$

where $b_k$ and $\theta_k$ are respectively the bias term and the weight vector that characterize the corresponding class $k$. Across all $K$ models, scalars $b_k$'s form the bias vector $b$ while vectors $\theta_k$'s form the matrix $\Theta$. They both are the parameters to be estimated and with the maximum likelihood optimization, their coefficients can be fitted by minimizing the negative log likelihood over $n$ network samples:

$$NLL(\Theta, b) = -\log \prod_{i=1}^{n} P(y_i = k|x_i, \Theta, b) \tag{4.2}$$

## 4.4 Spatial and Temporal Smoothness

*Spatial Smoothness:* Network topology plays a key role in determining how nodes communicate. In order to discover subnetworks related to global network states, the connectivity patterns within network samples have to be taken into account. Toward this goal, within each $k$-th class of network samples, we aim to regularize the parametric vector $\theta_k$ subject to the topology captured by $\mathcal{G}^{(k)}$, through using its Laplacian matrix $C^{(k)}$ defined by:

$$C^{(k)}_{pq} = \begin{cases} 1 - \mathcal{W}^{(k)}_{pq}/d_p & \text{if } v_p = v_q \text{ and } d_p \neq 0 \\ -\mathcal{W}^{(k)}_{pq}/\sqrt{d_p d_q} & \text{if } v_p \text{ and } v_q \text{ are connected in } \mathcal{G}^{(k)} \\ 0 & \text{otherwise} \end{cases}$$

in which $d_p$ is the vertex degree of $v_p$. Thus, the negative log likelihood is incorporated with the spatial network constraint term as follows:

$$F(\Theta, b) = -\log \prod_{i=1}^{n} P(y_i = k | x_i, \Theta, b) + \frac{\lambda_1}{2} \sum_k \theta_k^T C^{(k)} \theta_k \tag{4.3}$$

with $\lambda_1 \geq 0$, and we can further expand:

$$\theta_k^T C^{(k)} \theta_k = \sum_{v_p} \sum_{v_q} \left( \frac{\theta_k(p)}{\sqrt{d_p}} - \frac{\theta_k(q)}{\sqrt{d_q}} \right)^2 \mathcal{W}^{(k)}_{pq} \tag{4.4}$$

From this equation, it is clearly seen that if $\mathcal{W}^{(k)}_{pq}$ is large, indicating two vertices $v_p$ and $v_q$ in graph $\mathcal{G}^{(k)}$ strongly interact in a large portion of the network samples, the coefficients at $p$-th and $q$-th entries of vector $\theta_k$ should be similar, i.e., smooth, in order to minimize this equation. That means the selection of either $v_p$ or $v_q$ will encourage the selection of the other one due to the large $\mathcal{W}^{(k)}_{pq}$, leading to the formation of subnetworks selected in the final results. This network term is thus considered as the spatial smoothness since it minimizes the difference between connected vertices in the aggregate graph.

*Temporal Smoothness:* Given the smooth progression on the observable global network states, it is particularly important to capture the temporal changes in the network connectivity patterns developed along with this process. For any two consecutive global states, one would expect that the local subnetwork processes influencing the development of global network states will evolve in a smooth way rather than in an abrupt manner. Following our objective function developed above, we thus further develop another regularization term that penalizes large deviations between any two parametric vectors $\theta_k$ and $\theta_{k+1}$, which respectively account for the two consecutive global network states. This gives rise to the following formulation:

$$F(\Theta, b) = -\log \prod_{i=1}^{n} P(y_i = k | x_i, \Theta, b) + \frac{\lambda_1}{2} \sum_k \theta_k^T C^{(k)} \theta_k + \frac{\lambda_2}{2} \sum_{k=1}^{K-1} \|\theta_{k+1} - \theta_k\|_2^2 \quad (4.5)$$

where $\lambda_2 \geq 0$ is a parameter regularizing the impact of temporal smoothness. It is worth noting that the $L_2$ norm on the parametric vectors encourages coefficients at a $p$-th feature across different states to be grouped together, which is essential for tracking the temporal changes of various local network processes along with the evolution of the global network states. Moreover, though each $\theta_k$ is learnt for a global state and their coefficients can be different in scale, such parametric vectors are usually very sparse with majority of entries are zeros (presented in the next section) in order to uncover a small set of predictive subnetworks. The temporal smoothness thus can be interpreted as penalizing the dissimilarity in sparseness across network samples with adjacent global network states, ensuring the smooth changes in the succinct set of uncovered predictive subnetworks.

## 4.5 Optimization with Sparse Model

In solving the objective optimization function formulated in Eq.(4.5), we resort to the steepest gradient descent method since there is no closed form solution for Eq.(4.5). For convenience, network state $y_i$ is re-formatted as "1-of-$K$" encoding vector $\mathbf{y}_i =$

$[\mathbf{y}_i^1, \ldots, \mathbf{y}_i^K]^T$ such that only $\mathbf{y}_i^k = 1$ and all other entries are 0 if the original scalar $y_i = k$ (e.g., if $K = 3$ and $y_i = 2$, then $\mathbf{y}_i = [0, 1, 0]$). Following this, the conditional likelihood of $y_i$ given $x_i$ can be written as:

$$P(y_i = k|x_i, \Theta, b) = \prod_{k=1}^{K} P(\mathbf{y}_i^k|x_i, \Theta, b)^{\mathbf{y}_i^k} \tag{4.6}$$

For brevity, we denote $P(k|x_i)^{\mathbf{y}_i^k}$ for $P(\mathbf{y}_i^k|x_i, \Theta, b)^{\mathbf{y}_i^k}$ by omitting parameter set $\{\Theta, b\}$. And its log form can be expanded by: $\log P(k|x_i)^{\mathbf{y}_i^k} = \sum_k \mathbf{y}_i^k(\theta_k^T x_i + b_k) - \log \sum_j exp(\theta_j^T x_i + b_j)$. In this form, it is straightforward to show that function $F(\Theta, b)$ in Eq.(4.5) is strictly convex and the optimal solution for parameters $\theta_k$'s and $b_k$'s can be achieved through iterative updates with their first derivatives given by:

$$\begin{aligned} F'(\theta_k) = \frac{\partial F}{\partial \theta_k} &= \sum_i (P(k|x_i) - \mathbf{y}_i^k)x_i + \lambda_1 C^{(k)}\theta_k \qquad (4.7)\\ &- \lambda_2(\theta_{k-1} - \theta_k) + \lambda_2(\theta_k - \theta_{k+1}) \quad \text{for } \theta_k's \\ F'(b_k) = \frac{\partial}{\partial b_k} &= \sum_i (P(k|x_i) - \mathbf{y}_i^k) \qquad \text{for } b_k's. \qquad (4.8)\end{aligned}$$

However, such a solution does not set any coefficient of $\theta_k$'s to zero and thus, no predictive subnetworks are selected which lacks crucial point of interpretation. We extend our solution to the more challenging optimization by further imposing the sparseness on $\theta_k$'s through constraining their L1-norm within a constant value $t > 0$:

$$\begin{aligned} f(\Theta, b) = &- \log \prod_{i=1}^{n} \prod_{k=1}^{K} P(k|x_i)^{\mathbf{y}_i^k} + \frac{\lambda_1}{2} \sum_k \theta_k^T C^{(k)} \theta_k \\ &+ \frac{\lambda_2}{2} \sum_{k=1}^{K-1} \|\theta_{k+1} - \theta_k\|_2^2 \qquad \text{subject to } \sum_k |\theta_k| \le t \end{aligned} \tag{4.9}$$

The advantage of L1-norm constraint [74] is that it causes many coefficients to be exactly zero when $t$ is set sufficiently small. Together with our network regularization terms, this constraint naturally performs subnetwork discovery by keeping only the most relevant substructures that are predictive to global network states. Eq.(4.9) can be equivalently formulated in the Lagrangian form as follows:

$$\begin{aligned} f(\Theta, b) = & -\log \prod_{i=1}^{n} \prod_{k=1}^{K} P(k|x_i)^{\mathbf{y}_i^k} + \frac{\lambda_1}{2} \sum_k \theta_k^T C^{(k)} \theta_k \\ & + \frac{\lambda_2}{2} \sum_{k=1}^{K-1} \|\theta_{k+1} - \theta_k\|_2^2 + \beta \sum_{k=1}^{K} |\theta_k| \end{aligned} \tag{4.10}$$

with $\beta > 0$ playing the role of $t$, but a larger value of $\beta$ forces more coefficients in $\theta_k$'s equal to 0. Unlike linear regression problem, our optimization function $f(\Theta, b)$ here is no longer strictly convex and obviously, there exists no closed form expression for it. Before showing Eq.(4.10) is minimizable via the approach of steepest descent, let us denote $F(\theta_k)$ for a single network state by:

$$\begin{aligned} F(\theta_k) = & -\log \prod_i P(k|x_i)^{\mathbf{y}_i^k} \theta_k + \frac{\lambda_1}{2} \theta_k^T C^{(k)} \theta_k \\ & + \frac{\lambda_2}{2} \left( \|\theta_{k+1} - \theta_k\|_2^2 + \|\theta_k - \theta_{k-1}\|_2^2 \right) \end{aligned} \tag{4.11}$$

For simplification, we have included the bias term $b_k$ as the first entry of the parametric vector $\theta_k$ and thus it is skipped in the notation of $F(\theta_k)$. Note that this bias coefficient is *excluded* from both the network regularization and the model smoothness terms. In optimizing each single parametric vector $\theta_k$, Eq.(4.10) can be simply written as:

$$f(\theta_k) = F(\theta_k) + \beta |\theta_k| \tag{4.12}$$

We have the following proposition:

**Proposition 7.** *Given the spatial smoothness and temporal smoothness defined in Eq.*(4.3)*&*(4.5)*, function $F(\theta_k)$ defined Eq.*(4.11) *is convex.*

Proof: The proof of this proposition is straightforward by showing that its Hessian matrix is positive definite. More specifically, continuing with the first derivative shown in Eq.(4.7), it can be shown that the second derivative of $F(\theta_k)$ with respect to $\theta_k$ is:

$$\begin{aligned} F''(\theta_k) &= \sum_i x_i x_i^T (P(k|x_i) - P(k|x_i)^2) + \lambda_1 C^{(k)} + \lambda_2 I \\ &= X P^{(k)} X^T + \lambda_1 \left( C^{(k)} + \frac{\lambda_2}{\lambda_1} I \right) \end{aligned} \tag{4.13}$$

where $P^{(k)}$ is the diagonal matrix with entries $(P(k|x_i) - P(k|x_i)^2)$'s, $X$ is the matrix with $x_i$'s as its columns and $I$ is the identity matrix. It is obvious that, given any non-zero vector $u \in \mathcal{R}^m$, the quadratic form $u^T F''(\theta_k) u > 0$ since: (i) $(P(k|x_i) - P(k|x_i)^2) \geq 0$ as $0 \leq P(k|x_i) \leq 1$; (ii) $C^{(k)}$ is positive definite according to Eq.(4.4); (iii) and both $\lambda_1$ and $\lambda_2$ are non-negative numbers by our setting. □

**Proposition 8.** *Given $f(\theta_k) = F(\theta_k) + \beta|\theta_k|$ where $\beta$ is non-negative, our optimization function $f(\theta_k)$ is convex.*

Proof: Let $h(\theta_k) = \beta|\theta_k|$ and for any $\theta_k^{(1)}, \theta_k^{(2)}$ defined in a convex domain, $\zeta \in (0, 1)$, and $\theta_k = \zeta\theta_k^{(1)} + (1 - \zeta)\theta_k^{(2)}$, then $h(\theta_k)$ is also a convex function since:

$$\begin{aligned} \zeta h(\theta_k^{(1)}) + (1-\zeta) h(\theta_k^{(2)}) &= \beta\zeta|\theta_k^{(1)}| + \beta(1-\zeta)|\theta_k^{(2)}| \\ = \beta|\zeta\theta_k^{(1)}| + \beta|(1-\zeta)\theta_k^{(2)}| &\geq \beta|\zeta\theta_k^{(1)} + (1-\zeta)\theta_k^{(2)}| = h(\theta_k) \end{aligned}$$

in which the triangle inequality has been used in the second row and given $\beta \geq 0$. In combination with Proposition 7, and note that the summation of convex functions defined in the convex domain is also convex [70], it follows that $f(\theta_k)$ is a convex function. □

Proposition 8 is important as it ensures that our steepest descent algorithm will converge. In the spirit of gradient descent, our algorithm keeps updating parametric vector $\theta_k$ by $\theta_k + \delta_k$ as long as the overall function is being reduced. Let us specify:

$$f(\theta_k) = F(\theta_k) + \beta|\theta_k| \qquad \text{and} \tag{4.14}$$

$$f(\theta_k + \delta_k) = F(\theta_k + \delta_k) + \beta|\theta_k + \delta_k| \tag{4.15}$$

Then the algorithm finds $\delta_k$ such that $f(\theta_k) - f(\theta_k + \delta_k)$ is maximized. That means the next step of moving $\theta_k$ will lead to the steepest descent in reducing $f(\theta_k)$. This maximization is equivalent to minimization of $f(\theta_k + \delta_k) - f(\theta_k)$:

$$\begin{aligned}\underset{\delta_k}{\text{argmin}}\ g(\delta_k) &= f(\theta_k + \delta_k) - f(\theta_k) \\ &= F(\theta_k + \delta_k) - F(\theta_k) + \beta(|\theta_k + \delta_k| - |\theta_k|) \end{aligned} \tag{4.16}$$

Further formulating terms on the right hand size, the 2nd order Taylor expansion can be exploited to get:

$$\begin{aligned} F(\theta_k + \delta_k) &= F(\theta_k) + \delta_k^T F'(\theta_k) + \frac{1}{2}\delta_k^T F''(\theta_k)\delta_k \text{ or} \\ F(\theta_k + \delta_k) - F(\theta_k) &= \delta_k^T F'(\theta_k) + \frac{1}{2}\delta_k^T F''(\theta_k)\delta_k \end{aligned} \tag{4.17}$$

where $F'(\theta_k)$ and $F''(\theta_k)$ are respectively the gradient vector (Eq.(4.7)), and the Hessian (Eq.(4.13)). Then, Eq.(4.16) can be rewritten as:

$$\underset{\delta_k}{\text{argmin}}\, g(\delta_k) = F'(\theta_k)^T\delta_k + \frac{1}{2}\delta_k^T F''(\theta_k)\delta_k + \beta\left(|\theta_k{+}\delta_k|{-}|\theta_k|\right)$$

However, optimizing this function is still challenging since it is not smooth due to the two absolute terms. Therefore, in order for minimization, the subderivative of the function is required. For simplicity, we provide the calculation w.r.t. each coefficient of

$\delta_k$. In this case, let $\delta_{kj}$ be the $j$-th entry of $\delta_k$, then the function can be written by:

$$g(\delta_{kj}) = F'_j(\theta_k)\times\delta_{kj} + \frac{1}{2}F''_{jj}(\theta_k)\times\delta^2_{kj} + \beta(|\theta_{kj}+\delta_{kj}|-|\theta_{kj}|)$$

where $F'_j(\theta_k)$ is the $j$-th entry of the gradient vector $F'(\theta_k)$, and $F''_{jj}(\theta_k)$ is the $j$-th entry in the diagonal of the Hessian matrix $F''(\theta_k)$. Consequently, we separate the subderivative w.r.t. $\delta_{kj}$ into the following cases:

$$\frac{\partial g(\delta_{kj})}{\partial\delta_{kj}} = \begin{cases} F'_j(\theta_k) + F''_{jj}(\theta_k)\delta_{kj} + \beta & \text{if } \delta_{kj} > -\theta_{kj} \\ F'_j(\theta_k) + F''_{jj}(\theta_k)\delta_{kj} - \beta & \text{if } \delta_{kj} < -\theta_{kj} \end{cases}$$

Note that the derivative of the absolute term is not defined when $\theta_{kj} - \delta_{kj} = 0$, or in this case we have $\delta_{kj} = -\theta_{kj}$.

Now we know that $\delta_{kj}$ is optimal if the minimum-norm subgradient at $\delta_{kj}$ is equal to zero. Thus, combining with the setting of the subderiviate to zero yields:

$$\delta_{kj} = \begin{cases} -\frac{F'_j(\theta_k)+\beta}{F''_{jj}(\theta_k)} & \text{if } F'_j(\theta_k) < F''_{jj}(\theta_k)\theta_{kj} - \beta \\ -\frac{F'_j(\theta_k)-\beta}{F''_{jj}(\theta_k)} & \text{if } F'_j(\theta_k) > F''_{jj}(\theta_k)\theta_{kj} + \beta \\ -\theta_{kj} & \text{otherwise} \end{cases} \tag{4.18}$$

For the bias term $b_k$, which we previously included as the top entry of $\theta_k$, its corresponding deviation is $\delta_{k0} = -F'_j(\theta_k)/F''_{jj}(\theta_k)$ since $b_k$ is excluded from both spatial and temporal regularization. Given $\delta_k$'s computed above, our algorithm iteratively updates $\theta_k$'s until there is no reduction on the overall objective function in Eq.(4.10). The final smoothly developed subnetworks are found by matching the optimal $\theta_k$'s with the graph topology of $\mathcal{G}^{(k)}$'s defined in Def. 5.

**Algorithm Complexity:** We name our algorithm `SLR` (Subnetwork Learning with Reg-

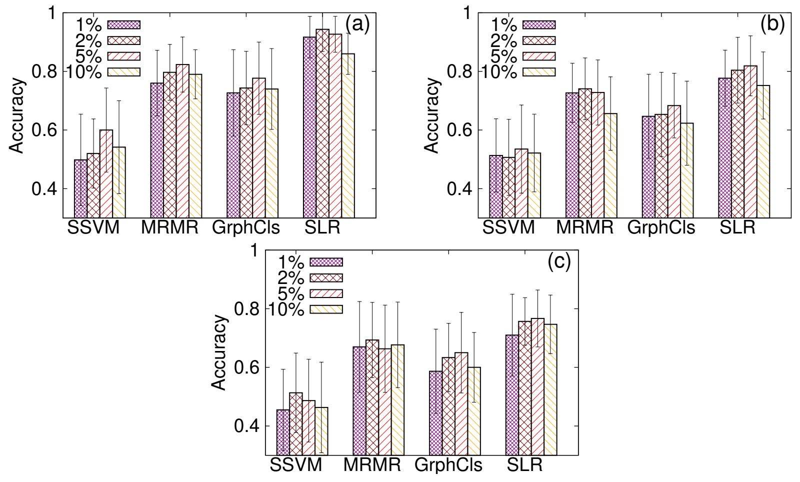


Figure 4.2: Prediction accuracy of all techniques on synthetic datasets. Plots (a)-(c) correspond to CNR=$[1.5, 1.0, 0.5]$ respectively of all algorithms with GndSNet=$[1\%, 2\%, 5\%, 10\%]$ (std.dev. is shown on top of each bar).

ularization) and analyze its complexity as follows. First, in terms of network topology, each network $S^{(i)}$ can be viewed as a subgraph of $\mathcal{G}^{(k)}$. Without loss of generality, we assume $m$ as the maximum number of $\mathcal{G}^{(k)}$'s vertices. The computation of the log likelihood term therefore takes $O(Knm)$ time while that of the spatial smoothness takes $O(Km^2)$ (see Eq.(4.10)). The calculation of temporal smoothness takes $O(Km)$, whereas the 1st and 2nd derivatives of $F(\Theta, b)$ take $O(Knm)$ and $O(Km^2)$ respectively. These quantities are computed at each iteration of the gradient descent and thus the overall computation is $O(J(Km^2 + nm))$ with $J$ as the number of iterations. Compared to $m$, both $K$ and $J$ are very small in practice. More importantly, the computation is not always quadratic in $m$ since the subnetworks' size greatly reduces after each iteration due to the sparseness imposed on $\theta_k$'s.

## 4.6 Experiments on Synthetic and ADNI Datasets

We compare `SLR` against the following techniques: (1) The well-known `MRMR` framework [89] that views network samples as collections of edges and performs edge selection based on mutual information; (2) `SSVM` by first applying SVD for dimensionality reduction (retaining 95% of the singular values) followed by SVM; (3) A recent typical graph classification method [85], named `GrphCls`, that works in a supervised setting and without side view information. We present two sets of experiments. First, to understand the strengths and limitations of our method, we utilize synthetic datasets that allow us to perform a number of controlled experiments. Second, we test all algorithms on the human functional brain networks to evaluate their practicability. Performance of every algorithm is evaluated via 5-fold stratified cross validation, in which their optimal parameters are chosen based on the estimated prediction accuracy within every 4 training folds and tested on the left-out fold according to [74]. Unless otherwise specified, $\lambda_1$ and $\lambda_2$ in `SLR` are selected from the ranges: $[0.001 - 50]$ with logscale step while with `GrphCls` [85], its *min-sup* is chosen from the range $[0.2 - 0.5]$ with step of 0.05. For `MRMR`, we use the forward edge selection scheme.

### 4.6.1 Data with known ground truth

Following the approach described in [91, 92], the synthetic datasets are generated by summary statistics associated with edges and adding ground truth signals at predefined subnetwork regions. The distribution of edges' values within the ground truth regions are generated with spatially contiguous correlations and conform to the background network structure. We use the contrast-to-noise (CNR) [91, 92] to control the difference between ground truth and non-ground truth edges. Three batches of datasets are generated with CNR=$[1.5, 1.0, 0.5]$. Within each CNR setting, we further vary the percentage of edges (forming ground truth subnetworks) GndSNet=$[1\%, 2\%, 5\%, 10\%]$ of the total network edges, and the size of a ground truth subnetwork is varied between 3 to 15 edges each. In total, 12 datasets have been generated and their network samples are labeled to three global states. In simulation the evolving local network processes, ground truths for network samples labeled by global state 1 are firstly created, then the ground truths for networks with global state 2 are generated with the varying overlapping $[60\% - 90\%]$ with the ones in state 1. In a similar way, we generate the ground truths for network samples with global state 3 based on the ones in state 2. For each individual dataset, we generate 3K network samples evenly distributed into three global network states. Their network structures form approximately 6K vertices and 125K links within each graph $\mathcal{G}^{(k)}$ in Def.5.

**Prediction Accuracy:** Fig.4.2(a)-(c) shows the experimental results of `SLR` and the other three competing methods in predicting the global network states. Each plot in the figure corresponds to a CNR setting, and each bar corresponds to a setting of GndSNet. The accuracy values are obtained by averaging the prediction rate from cross validation. As one observes, `SLR` performs stably over variation in both CNR and GndSNet.

Its prediction performance is better than `MRMR` since the network information is fully explored, while is superior to the network-based `GrphCls` since the temporal smoothness further narrows down the searching space to a small set of relevant and stable predictive subnetworks. Among all techniques, `SSVM` is less successful and the possible reason is that when irrelevant edges are prevalent in the data, aggregating all edges to form a lower dimensional subspace might not lead to a satisfactory performance.

**Ground truth subnetwork discovery:** We compute ROC curves based on the set of edges retrieved from the selected substructures w.r.t. the ground truth subnetworks. This is plotted in Fig.4.3(a)-(c) for all algorithms except `SSVM` since its new features combine information from all edges. The results are reported for GndSNet=5% as performance at

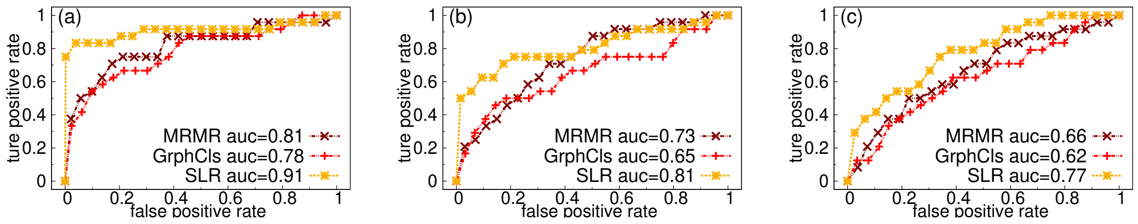


Figure 4.3: ROC performance of the algorithms in uncovering the ground truth subnetworks. Plots (a)-(c) correspond to 3 levels of CNR=$[1.5, 1.0, 0.5]$ with GndSNet=5% (similar trends were observed for other settings of GndSNet).

other settings shows similar trends. For edge ranking in our `SLR` method, we rely on the absolute values of the parametric vectors, whereas for `GrphCls`, we rank edges based on their aggregated frequency in the significant subgraphs [85]. In `MRMR`, the ranking is based on the order in which edges are incrementally selected. The ROC curves shown in all three plots demonstrate that `SLR` uncovers more relevant subnetworks than `GrphCls` and `MRMR` for small positive rates. As this rate increases, the behaviors of three techniques become similar since none of them can fully uncover all ground truths. However, the higher ROC performance clearly makes `SLR` a better candidate in identifying local subnetworks influencing global network properties, which is practically important since validating the relevance of a subnetwork is often costly in real applications.

**Impact of spatial and temporal network regularization:** To provide more insights into the performance of our algorithm, we further report a series of experiments in examining the impact of $\lambda_1$ and $\lambda_2$. The intrinsic relationship between these two factors and `SLR`'s prediction rate is plotted in Fig.4.4. In Fig.4.4(a-c), we show the impact of $\lambda_1$ on prediction rate when fixing $\lambda_2$ at different settings. An important trend can be seen that either setting $\lambda_1$ too small or large do not lead to a high prediction rate. A small $\lambda_1$ causes isolated edges to be selected since the network topology is disregarded, while a large $\lambda_1$ can overly favor strongly connected substructures yet not related to the global network states. A similar trend is also seen in Figure 4.4(d-f) where we fix $\lambda_1$

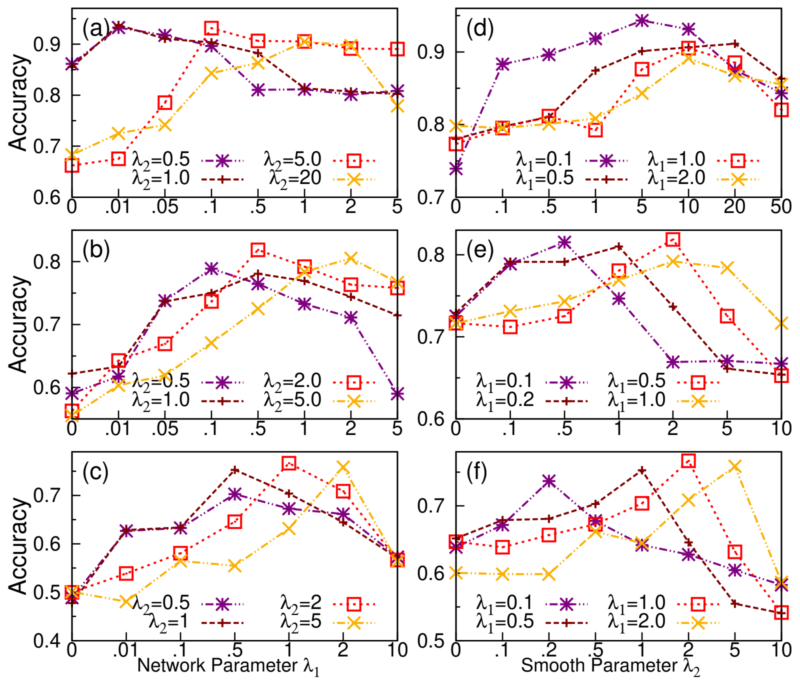


Figure 4.4: Impact of $\lambda_1$ and $\lambda_2$ on prediction accuracy. Plots in three rows correspond to CNR=$[1.5, 1.0, 0.5]$ respectively with GndSNet=5%. Plots (a)-(c) show the impact of $\lambda_1$ while plots (d)-(f) show the impact of $\lambda_2$.

and vary $\lambda_2$. A small setting of $\lambda_2$ causes different subnetworks across different global network states, while its large value may force them to be too similar across states, even for non-ground truth substructures, both leading to a low prediction accuracy.

### 4.6.2 Real world dataset

We choose the important application of analyzing human brain networks associated with the Alzheimer's Disease. The analysis of brain data has recently attracted much attention from the data mining community with convincing results demonstrated in [85, 93, 94, 95]. However, unlike most previous studies which focus on a small number of subjects and especially not for the *temporal* development of the disease, we analyze a large scale cohort of 180 subjects obtained from http://www.adni-info.org/, and evenly distributed into three global states: normal control (NC), mild cognitive impairment

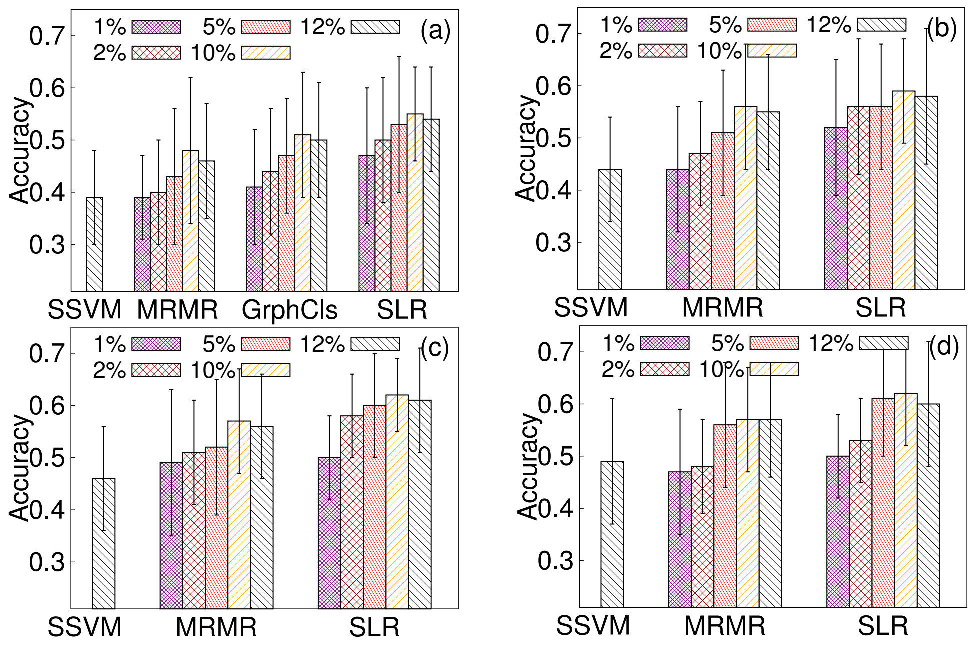


Figure 4.5: Prediction accuracy on human brain networks. Plots (a)-(d) correspond to datasets `C8`, `C4`, `C2` and `C0` respectively, with the selected subnetworks varied from 1%, 2%, 5%, 10% to 12% of the total edges. `GrphCls` does not handle well densely connected datasets, while `SSVM` is shown with a single column due to using aggregated features.

(MCI) and Alzheimer's disease (AD). FSL toolbox [2] is used to convert an fMRI scan to a functional brain network, comprising of 112 brain regions. A value associated with an edge (connecting two brain regions) is evaluated by the correlation between their blood oxygen level-dependent time series. Since there is no gold standard for choosing a proper threshold for functional correlation, we follow the general approach in [96, 65] by selecting four thresholds ranging from $[0.8, 0.4, 0.2, 0]$ to remove weak correlations, resulting in four network datasets respectively denoted by `C8`, `C4`, `C2` and `C0`, with numbers of vertices/links varying from 776/15535 to 5515/49284.

**Prediction Performance:** We evaluate all algorithms on network state prediction by comparing their performance on five settings of selected subnetworks between 1%, 2%, 5% 10% and 12% of total unique edges. Fig.4.5 reveals that all algorithms yield better prediction rates for higher numbers of selected substructures. However, a level larger than

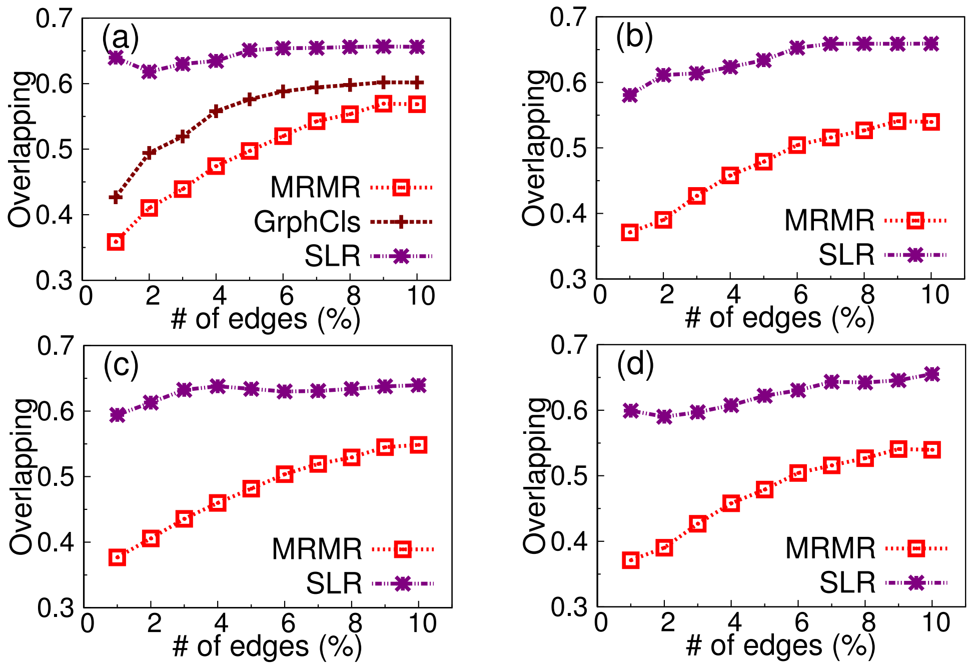


Figure 4.6: Overlapping percentage of selected edges across all training data. Plots (a)-(d) correspond to four network datasets C8, C4, C2 and C0.

10% does not lead to better prediction accuracy, due to the prevalence of noisy edges. Among all examined techniques, SSVM shows the lowest performance at the prediction rate of 39% in C8, a result that is only marginally better than the random guess of 33%. GrphCls performs much better than SSVM but only works on the sparse network data C8. For other datasets with denser network samples, GrphCls takes much longer time handle the large amount of possible substructures. The performance of SLR is by far the best, dominating both mutual information based MRMR and the frequent subgraph-based GrphCls, with a large accuracy advantage on both sparsely and densely connected network data. Looking deeper, we also see that for the same level of prediction rate, subnetworks uncovered by SLR is usually more succinct compared to other competing methods. For example, on the densely connected dataset C2, it achieves 60% of prediction accuracy with 5% of total edges combined from its substructures, as compared to the second-best technique MRMR, which requires double the number of edges for a prediction rate of 57%.

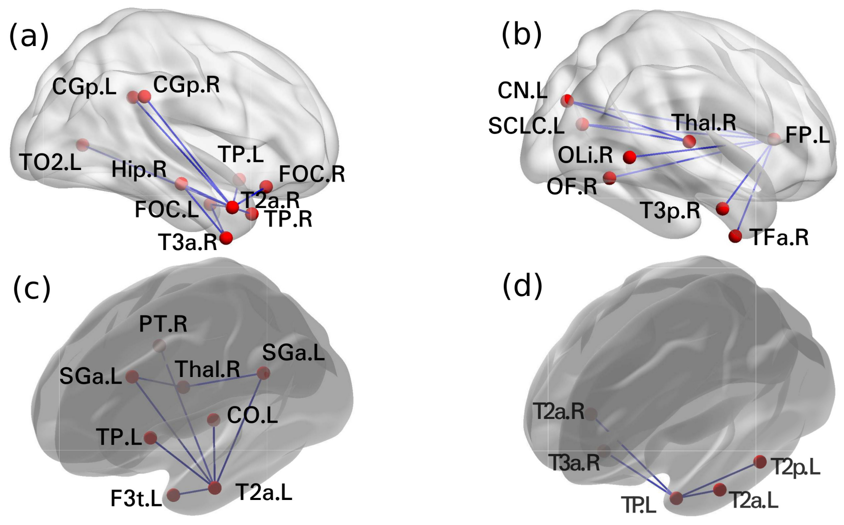


Figure 4.7: Four typical subnetwork markers (visualized with node hubs) consistently selected by SLR. Nodes are brain regions abbreviated in the HO atlas standard [2]. (a) T2a.R: Right Middle Temporal Gyrus, anterior division; (b) T2a.L: Left Middle Temporal Gyrus, anterior division; (c) FP.L: Left Frontal Pole; (d) TP.L: Left Temporal Pole.

**Subnetwork discovery:** We evaluate the quality of uncovered substructures from each technique through their consistency in cross validation. Subnetworks that are consistently selected across training folds are likely the disease-related biomarkers and they should be the first candidate for further investigation. Fig.4.6 provides overlapping percentage (y-axis) of discovered subnetworks across all training folds for the four datasets. As observed, although the overlap tends to increase as the number of selected edges increases, none of the competing methods consistently produce results as good as SLR, which implies that SLR selects the most stable and consistent predictive substructures across all training folds.

To validate that our proposed method finds meaningful subnetworks, we further investigate its selected subnetworks. Fig.4.7 displays the top four subnetworks consistently discovered from all training folds. For easier visualization, we plot each subnetwork with a core node that has the highest node degree (like community hub [4]). It is found that the core nodes like `T2a.R`, `T2a.L` and `TP.L` indeed reside within the temporal lope—the brain region strongly impacted by Alzheimer's disease as reported in [97, 98].

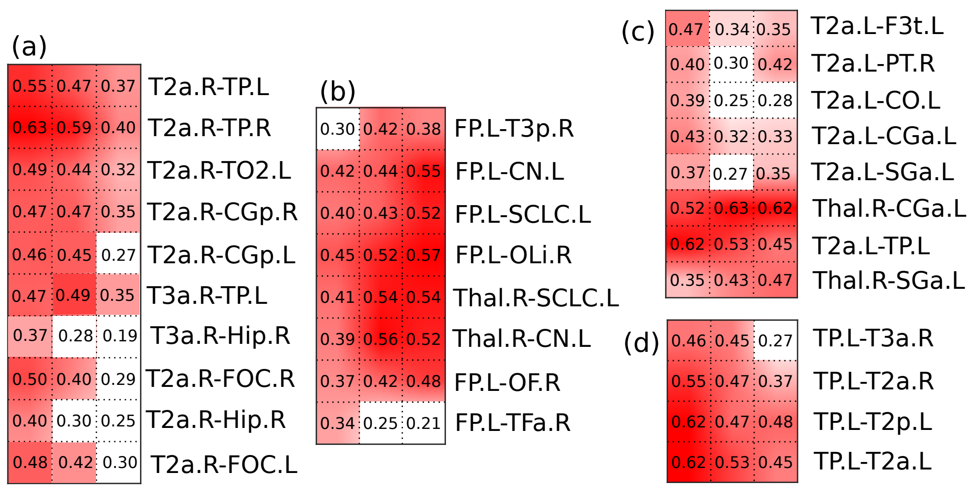


Figure 4.8: Smoothly changing subnetworks. Columns from left to right in each plot correspond to NC, MCI and AD states. Dark background colors show selected edges whereas entries' values show functional correlations averaged from network samples within the corresponding group.

Fig.4.8 further provides visualization on how these subnetworks have changed smoothly across the global network states. Three columns in each plot correspond to NC, MCI and AD states. As seen, functional correlations deteriorate noticeably from NC to AD groups in most cases, especially those in the `T2a.R` subnetwork (Fig. 4.8(a)) and the `TP.L` subnetwork (Fig. 4.8(d)). This phenomenon can be explained by the fact that, once `T2a.R` and `TP.L` regions are damaged by the disease, their cognitive performance is significantly reduced which further impacts other functionally connected regions. However, we also observe that in some circumstances, the functional correlation increases from NC to AD, as between `FP.L` and `SCLC.L, OLi.R` in Fig. 4.8(b), or between `Thal.R` and `CGa.L, SGa.L` in Fig. 4.8(c). This increased functional connectivity might support the "compensatory recruitment hypothesis" [98].

## 4.7 Related Work

Analyzing network structural data has been widely studied in the literature with most existing work focusing on community detection [4], frequent subgraph mining [84], outlier detection [3], and graph classification [87, 58]. Close to our study is the line of work on graph classification. Though diverse in terms of underlying approaches, most algorithms [58, 56] generally assume a database consisting of "positive" and "negative" graphs, and aim at extracting a set significant subgraphs that are frequently *present* in one class but *absent* in the other, which subsequently are used as new (binary) features to train classifiers. These approaches, recently being extended to semi-supervised setting [99], uncertain graphs [100], or multiple side-view [85] which we have adapted for our empirical comparison. Another related work is the one developed in [93] that directly addresses the progressive data but in the *non-network* context. Moreover, its view on the subject behaviors' progression differs from ours in which it assumes the smooth changes in predictive variables appear in every model while minimizing the models' difference learnt at various time points. In contrast, our study explores a single model in which the changes in local network processes can lead to the changes in global network states.

Another line of related studies are from feature selection where one can view each network sample as a collection of edges and use statistical analysis t-test [101, 95] or mutual information [102, 89] to select edges that lead to the statistical difference among various global network states. The advantage is that these studies can make an individual edge or region-based analysis [101, 102], instead of analyzing entire networks as a whole. However, they lack the capability of analyzing both intra and inter-connectivities among different network regions at the same time. Our work is also related to dynamic network mining that aims to discover subnetworks of interest in multiple discrete snapshots of an evolving network [81] or in a multi-layer network [82]. The goal in this line of work, however, is not predicting global network states, but the discovery of abnormal substructures that persist in time or across network layers.

## 4.8 Conclusions

In this work, we address an important problem of mining a succinct set local subnetworks that are predictive for the progression of global network states. We develop `SLR` as a novel algorithm that fits a model for multi-states of network samples subject to two important constraints: (i) spatial smoothness imposed on the network topology to ensure the well-connected substructures; (ii) temporal smoothness to discover predictive subnetworks evolving along with the progression of global network states. `SLR` further imposes the sparsity-inducing $L1$-norm to explicitly remove edges that have little or no impact on the progression of global network states and we show that the overall optimization function is convex. Extensive experiments on both synthetic datasets and the emerging brain networks demonstrate the appealing performance of our algorithm not only in terms of prediction accuracy but also in the consistency of the discovered subnetworks with existing literature, providing a better understanding of the intrinsic relationship between the evolution of local network processes and the progression of global network behaviours.

# Chapter 5

# Discrepancy-aware Network Regularization

## 5.1 Introduction

Network regularization is a fundamental approach to encode and incorporate general relationships among variables, which are represented as nodes and linked together by weighted edges that describe their local proximity. A significant amount of effort has been devoted to developing successful regularizations [16, 103, 104, 18, 105] that take advantage of prior knowledge on network structures to enhance estimation performance for a variety of application settings, including image denoising [106], genomic data analysis [13] and neuroimaging-based classification [107].

We consider the general setting of network regularization, now proposed as a convex optimization problem defined on an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with node set $\mathcal{V}$ and edge set $\mathcal{E}$:

$$\min \sum_{i \in \mathcal{V}} f_i(\boldsymbol{x}_i) + \lambda \sum_{(j,k) \in \mathcal{E}} \omega_{jk} \cdot g(\boldsymbol{x}_j, \boldsymbol{x}_k), \tag{5.1}$$

where on each node $i$, we intend to learn a local model $\boldsymbol{x}_i \in \mathbb{R}^d$, which in addition to minimizing a predefined *convex* loss function $f_i : \mathbb{R}^d \to \mathbb{R}$, carries certain relations to its neighbors, defined by the function $g : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$.

In this work, we focus on the particular setting where $g(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$, also known as a *sum-of-norms* (SON) regularizer. It assumes that the network is composed of multiple clusters, suggesting that all nodes within a cluster share the same consensus model ($\boldsymbol{x}_i = \cdots = \boldsymbol{x}_j$). As the observed data on each node may be sparse, the SON regularizer allows nodes to "borrow" observations from their neighbors to improve their own models, as well as to determine the network cluster to which they belong. The SON objective was first introduced in [15] for convex clustering problems and used off-the-shelf sequential convex solvers. Chi & Lange [108] adapt SON clustering to incorporate similarity weights with an arbitrary norm and solve the problem by both alternating direction method of multipliers (ADMM) or alternating minimization algorithm (AMA) in a parallel manner. More recently, Hallac et al. [18] point out that weighted SON (named as Network Lasso) allows for simultaneous clustering and optimization on graphs, and is highly suitable for a broad class of large-scale network problems.

In network regularization, a static weight assigned to an edge determines how strictly the difference between models on the corresponding nodes is being penalized, relative to the other node pairs in the network. However, unlike few scenarios in which network information is explicitly given, edge weights are usually unavailable or even infeasible to obtain in most real-world networks (e.g., gene regulatory networks [109]). Moreover, due to possible measurement errors and inaccurate prior knowledge, the assigned edge weights don't necessarily align with the underlying clustering structure of networks as shown in [110, 111]. As a result, heavily relying on the edge weights in determining how much penalty should be applied to neighboring models may contaminate the discovered solutions.

Similar intuition applies when we extend to the temporal setting. Models on nodes of temporal networks usually change at the level of groups over time. However, some groups exhibit different evolution patterns than others [112]. Moreover, the grouping structures themselves may evolve with time [113]. As static regularization cannot capture such temporal evolution, a direct solution is to induce a time-varying local consensus by employing the SON objective on both spatial and temporal directions, but we face an more drastic problem: how to assign weights between snapshots.

Consequently, a framework that is robust to missing or corrupted edge weights in network-based regularization and adapts to spatio-temporal settings is desired. Note that without the assigned network weights, the setting in Eq. (5.1) degrades to simply applying isotropic regularization for all the edges. In this work, we propose a generic formulation, called the *discrepancy-aware network regularization* (DANR), which deploys a suitable amount of anisotropic network regularization in both spatial and temporal aspects. DANR infers models at each node per timestamp and can learn evolution of models and transitions of network structures over time. We develop an ADMM-based algorithm that adopts an efficient and distributed iterative scheme to solve problems formulated by the DANR, and show that the proposed solution obtains guaranteed convergence towards global optimal solutions. By applying to both synthetic and real-world datasets, we demonstrate the effectiveness of the proposed approach on various network problems.

## 5.2 Modified Regularization for Spatial Networks

### 5.2.1 Discrepancy-aware Network Regularization

Since existing network-based regularizers rely on known weights on edges, the corresponding solutions get misled when the edge weights are erroneous or unknown. Directly learning the unknown edge weights by using the same regularizer as Eq. (5.1) would lead to an optimization problem:

$$\min_{\boldsymbol{x},\boldsymbol{\omega}} \quad \sum_{i\in\mathcal{V}} f_i(\boldsymbol{x}_i) + \lambda \sum_{(j,k)\in\mathcal{E}} \omega_{jk} \left\|\boldsymbol{x}_j - \boldsymbol{x}_k\right\|_2 \tag{5.2}$$

which yields the trivial all-zero solution of $\boldsymbol{\omega}$ and thus becomes unsatisfactory. Instead of imposing more sophisticated model-based or problem-dependent regularization as suggested in [114], we consider an alternative formulation that explicitly accounts for discrepancies between models on adjacent nodes:

$$\min_{\boldsymbol{x},\boldsymbol{\alpha}} \quad \sum_{i\in\mathcal{V}} f_i(\boldsymbol{x}_i) + \lambda \cdot \mathcal{R}_S(\boldsymbol{x},\boldsymbol{\alpha}) \tag{5.3}$$

$$\mathcal{R}_S(\boldsymbol{x},\boldsymbol{\alpha}) = \mu \sum_{(j,k)\in\mathcal{E}} \omega_{jk} \left\|\boldsymbol{x}_j + \boldsymbol{\alpha}_{jk} - \boldsymbol{x}_k\right\|_2 + (1-\mu)\|\boldsymbol{\alpha}\|_{1,p} \tag{5.4}$$

where we denote $\|\boldsymbol{\alpha}\|_{1,p} = \sum_{(j,k)\in\mathcal{E}} \|\boldsymbol{\alpha}_{jk}\|_p$ as the sum of $p$-norms of all $\boldsymbol{\alpha}_{jk}$'s. We set a penalty parameter $\lambda \geq 0$ to control the overall strength of network regularization, and a portion parameter $0 < \mu < 1$ to control the emphasis between the two terms in $\mathcal{R}_S(\boldsymbol{x},\boldsymbol{\alpha})$. We name this formulation in Eq. (5.3) the *discrepancy-aware network regularization* (DANR).

In the first term of $\mathcal{R}_S$, we define a *discrepancy-buffering (DB) variable* $\boldsymbol{\alpha}_{jk} \in \mathbb{R}^d$ to denote the preserved discrepancy between local model parameters $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. More specifically, when an edge weight $\omega_{jk}$ is given but potentially imprecise or otherwise corrupted, $\boldsymbol{\alpha}_{jk}$ would compensate for abnormally large differences between models $\boldsymbol{x}_j$ and $\boldsymbol{x}_k$ to reduce the magnitude of $\omega_{jk}$-weighted edge penalty term in $\mathcal{R}_S$. The DB variable provides additional flexibility to its associated models, allowing them to stay adequately close to their own local solutions w.r.t. minimizing local loss functions, and avoid over-penalized consensus. When all $\omega_{jk}$'s are not given, $\boldsymbol{\alpha}_{jk}$'s enable solving Eq. 5.3 under anisotropic regularizations even with homogeneous weights.

The second term of $\mathcal{R}_S$ is the $\ell_{1,p}$-regularizer (with $1 < p < \infty$) [115] that ensures sparsity at the group level. Note that we regard each variable vector $\boldsymbol{\alpha}_{jk}$ as a group ($|\mathcal{E}|$ groups in total). The first key intuition here is that the regularization on $\boldsymbol{\alpha}$ helps to exclude trivial solutions, that is $\boldsymbol{\alpha}_{jk} = \boldsymbol{x}_k - \boldsymbol{x}_j$. Second, it allows us to identify a succinct set of non-zero vectors $\boldsymbol{\alpha}_{jk}$'s, compensating for possible intrinsic discrepancies between two adjacent nodes.

To sum up, discrepancy-buffering variable $\boldsymbol{\alpha}_{jk}$'s are designed to elaborately adjust network regularization strength on all edges, and thus reduce negative effects of the unsquared norm regularizer. We will show in later sections that this modified formulation remains convex and tractable via parallel optimization algorithms on large-scale networks.

### 5.2.2 Distributed ADMM-based Solution

In this section, we propose an ADMM-based algorithm for solving the ST-DANR problem in Eq. (5.16) and present the convergence and complexity of the proposed algorithm. The algorithm can be easily adapted to the spatial-only DANR problem in Eq. (5.3) by omitting temporal-related updates.

The ADMM method was originally derived in [116] and has been reformulated in many contexts including optimal control and image processing [117]. The method can be considered as combining augmented Lagrangian methods and the method of multipliers [19, 118]. It aims to solve optimization problems with two-block separable convex objectives in the following form of

$$\min \ f(\boldsymbol{x}) + g(\boldsymbol{z}) \qquad s.t.\ A\boldsymbol{x} + B\boldsymbol{z} = \boldsymbol{c} \tag{5.5}$$

Observe that our proposed objective in Eq. (5.16) has a separable convex objective function: it can be reorganized into two-block separable convex objectives as in Eq. (5.5), for which ADMM methods guarantee convergence to global optimal solutions [119, 120, 119].

More precisely, to fit our problem into the ADMM framework, we first define $\boldsymbol{x}=\{\boldsymbol{x}_j\}^{j\in V}$ for model parameter vectors on the given undirected network $\mathcal{G}$, and define consensus variables $\boldsymbol{u} = [\{\boldsymbol{u}_{jk}, \boldsymbol{u}_{kj}\}^{(j,k)\in\mathcal{E}}]$ as copies of $\boldsymbol{x}$ in the spatial penalty term $\mathcal{R}_S(\boldsymbol{x}, \boldsymbol{\alpha})$. Then we rewrite Eq. (5.16) as follows:

$$\begin{aligned}\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{\alpha}} \quad & \sum_{j\in\mathcal{V}} f_j(\boldsymbol{x}_j) + \lambda_1(1-\mu_1)\|\boldsymbol{\alpha}\|_{1,p} \\ & + \lambda_1\mu_1 \sum_{(j,k)\in\mathcal{E}} \omega_{jk} \left\|\boldsymbol{u}_{jk} + \boldsymbol{\alpha}_{jk} - \boldsymbol{u}_{kj}\right\|_2 \\ \text{s.t.} \quad & [\boldsymbol{u}_{jk}, \boldsymbol{u}_{kj}] \quad = [\boldsymbol{x}_j, \boldsymbol{x}_k], \quad (j,k)\in\mathcal{E}\end{aligned} \tag{5.6}$$

in which the first term corresponds to the $f$ block in Eq. (5.5), and the remaining terms correspond to the $g$ block. The equality constraints on Eq. (5.16) are used to force consensus between variables $\boldsymbol{x}$ and $\boldsymbol{u}$. Next, we derive the augmented Lagrangian of Eq. (A.1):

$$\begin{aligned}\mathcal{L}_{\rho_1}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{\alpha},\boldsymbol{\delta}) = & \sum_{j\in V} f_j(\boldsymbol{x}_j) + \lambda_1(1-\mu_1)\|\boldsymbol{\alpha}\|_{1,p} + \sum_{(j,k)\in E}\Big(\lambda_1\mu_1\omega_{j,k}\|\boldsymbol{u}_{jk} + \boldsymbol{\alpha}_{jk} - \boldsymbol{u}_{kj}\|_2 \\ & + \frac{\rho_1}{2}\|\boldsymbol{x}_j - \boldsymbol{u}_{jk} + \boldsymbol{\delta}^u_{jk}\|_2^2 - \frac{\rho_1}{2}\|\boldsymbol{\delta}^u_{jk}\|_2^2 + \frac{\rho_1}{2}\|\boldsymbol{x}_k - \boldsymbol{u}_{kj} + \boldsymbol{\delta}^u_{kj}\|_2^2 - \frac{\rho_1}{2}\|\boldsymbol{\delta}^u_{kj}\|_2^2\Big)\end{aligned} \tag{5.7}$$

where $\boldsymbol{\delta} = [\{\boldsymbol{\delta}^u_{jk}, \boldsymbol{\delta}^u_{kj}\}^{(j,k)\in\mathcal{E}}]$ are scaled dual variables for each equality constraint on the elements of $\boldsymbol{u}$. The parameters $\rho_1 > 0$ penalize the violation of equality constraints in the spatial and temporal domain [121] respectively. The iterative scheme of ADMM under the above setting can be written as follows, with $l$ denoting the iteration index:

$$\left.\begin{aligned} &\boldsymbol{x}^{(l+1)} = \underset{\boldsymbol{x}}{\operatorname{argmin}}\ \mathcal{L}_{\rho_1}(\boldsymbol{x}, (\boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{\delta})^{(l)}) \\ &(\boldsymbol{u}, \boldsymbol{\alpha})^{(l+1)} = \underset{\boldsymbol{u},\boldsymbol{\alpha}}{\operatorname{argmin}}\ \mathcal{L}_{\rho_1}(\boldsymbol{x}^{(l+1)}, \boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{\delta}^{(l)}) \\ &\boldsymbol{\delta}^{(l+1)} = \boldsymbol{\delta}^{(l)} + (\tilde{\boldsymbol{x}}^{(l+1)} - \boldsymbol{u}^{(l+1)}) \end{aligned}\right\} \tag{5.8}$$

where $\tilde{\boldsymbol{x}}$=$[\{\boldsymbol{x}_j, \boldsymbol{x}_k\}^{(j,k)\in\mathcal{E}}]$ is composed of replicated elements in $\boldsymbol{x}$, and thus has a one-to-one correspondence with elements in $\boldsymbol{u}$.

Because the augmented Lagrangian in Eq. (5.7) has a separable structure as well, we can further split the optimization above over each univariate element in $\boldsymbol{x}$ and $\boldsymbol{z}$. Next, we provide details for each ADMM update step.

**$\boldsymbol{x}$-Update.** In the first update step of our ADMM updating scheme, we can decompose the problem of minimizing $\boldsymbol{x}$ into separately minimizing $\boldsymbol{x}_j$ for each node $j$:

$$\boldsymbol{x}_j^{(l+1)} = \underset{\boldsymbol{x}_j}{\operatorname{argmin}} \left( f_j(\boldsymbol{x}_j) + \frac{\rho_1}{2} \sum_{k\in N_j} \|\boldsymbol{x}_j - \boldsymbol{u}_{jk}^{(l)} + \boldsymbol{\delta}_{jk}^{u(l)}\|_2^2 \right)$$

As above equation suggests, $\boldsymbol{x}_j$ on node $j$ first receives local information from the corresponding consensus variables belonging to all of its neighbors $k \in N_j$, then updates its value to minimize the loss function and remain close to neighbouring consensus variables. Since all remaining regularizations are quadratic, the $\boldsymbol{x}$-update problem can be efficiently solved whenever the loss function $f_j$ has certain properties, such as strong convexity.

**$(\boldsymbol{u}, \boldsymbol{\alpha})$-Update.** We can further decompose the $(\boldsymbol{z}, \boldsymbol{\alpha})$-update step in Eq. (A.2) into subproblems on each edge (and its related variables $\boldsymbol{u}_{jk}, \boldsymbol{u}_{kj}, \boldsymbol{\alpha}_{jk}$) as follows, which can be solved in parallel:

$$(\boldsymbol{u}_{jk}^{(l+1)}, \boldsymbol{u}_{kj}^{(l+1)}, \boldsymbol{\alpha}_{jk}^{(l+1)}) = \text{argmin } \Big(\lambda_1(1-\mu_1)\|\boldsymbol{\alpha}_{jk}\|_p + \lambda_1\mu_1\omega_{jk}\|\boldsymbol{u}_{jk} + \boldsymbol{\alpha}_{jk} - \boldsymbol{u}_{kj}\|_2 + \frac{\rho_1}{2}\|\boldsymbol{x}_j^{(l+1)} - \boldsymbol{u}_{jk} + \boldsymbol{\delta}_{jk}^{u(l)}\|_2^2 + \frac{\rho_1}{2}\|\boldsymbol{x}_k^{(l+1)} - \boldsymbol{u}_{kj} + \boldsymbol{\delta}_{kj}^{u(l)}\|_2^2\Big) \qquad (5.9)$$

Notice that the sum of convex functions which are defined on different sets of variables preserves the convexity. To minimize the objective with $\boldsymbol{u}_{jk}, \boldsymbol{u}_{kj}$ and $\boldsymbol{\alpha}_{jk}$ simultaneously, the convexity of Eq. (5.2.2) motivates us to adopt an alternating descent algorithm, which minimizes each component iteratively with respect to $(\boldsymbol{u}_{jk}, \boldsymbol{u}_{kj})$ and $\boldsymbol{\alpha}_{jk}$ while holding the other fixed. In detail, we solve Eq. (5.10) and Eq. (5.11) iteratively until convergence is achieved:

$$(\boldsymbol{u}_{jk}^{(l'+1)}, \boldsymbol{u}_{kj}^{(l'+1)}) = \text{argmin}\Big(\lambda_1\mu_1\omega_{jk}\|\boldsymbol{u}_{jk} + \boldsymbol{\alpha}_{jk}^{(l')} - \boldsymbol{u}_{kj}\|_2 + \frac{\rho_1}{2}\|\boldsymbol{x}_j^{(l+1)} - \boldsymbol{u}_{jk} + \boldsymbol{\delta}_{jk}^{u(l)}\|_2^2 + \frac{\rho_1}{2}\|\boldsymbol{x}_k^{(l+1)} - \boldsymbol{u}_{kj} + \boldsymbol{\delta}_{kj}^{u(l)}\|_2^2\Big) \qquad (5.10)$$

$$\boldsymbol{\alpha}_{jk}^{(l'+1)} = \text{argmin}\Big((1-\mu_1)\|\boldsymbol{\alpha}_{jk}\|_p + \mu_1\omega_{jk}\|\boldsymbol{u}_{jk}^{(l'+1)} + \boldsymbol{\alpha}_{jk} - \boldsymbol{u}_{kj}^{(l'+1)}\|_2\Big) \qquad (5.11)$$

where $l'$ denoting the iteration index of alternating descent. Further, we can utilize the analytic solution to speed up the calculation of $\boldsymbol{u}_{jk}$ and $\boldsymbol{u}_{kj}$:

**Proposition 9.** *Problem (5.10) has a closed-form solution:*

$$\left.\begin{aligned} \boldsymbol{u}_{jk}^{(l'+1)} &= (1-\theta)\boldsymbol{a} + \theta\boldsymbol{b} - \theta\boldsymbol{\alpha}_{jk}^{(l')} \\ \boldsymbol{u}_{kj}^{(l'+1)} &= \theta\boldsymbol{a} + (1-\theta)\boldsymbol{b} + \theta\boldsymbol{\alpha}_{jk}^{(l')} \end{aligned}\right\}$$

*where we denote* $\boldsymbol{a} = \boldsymbol{x}_j^{(l'+1)} + \boldsymbol{\delta}_{jk}^{u\ (l')}$, $\boldsymbol{b} = \boldsymbol{x}_k^{(l'+1)} + \boldsymbol{\delta}_{kj}^{u\ (l')}$, $c = \lambda_1(1-\mu_1)\omega_{jk}$, *and* $\theta = c/(\rho_1\|\boldsymbol{a} - \boldsymbol{b} + \boldsymbol{\alpha}_{jk}^{(l')}\|_2)$ *for simplification.*

Proof: We present the analytic solution for updating $(\boldsymbol{u}_{jk}, \boldsymbol{u}_{kj})$, introduced in Eq.

(2.9). For simplicity, we omit iteration index $l'$, and denote $\boldsymbol{a} = \boldsymbol{x}_j + \boldsymbol{\delta}^u_{jk}$, $\boldsymbol{b} = \boldsymbol{x}_k + \boldsymbol{\delta}^u_{kj}$ and $c = \lambda_1(1-\mu_1)\omega_{jk}$.

$$\min \quad g(\boldsymbol{u}_{jk}, \boldsymbol{u}_{kj}) = c \left\| \boldsymbol{u}_{jk} + \boldsymbol{\alpha}_{jk} - \boldsymbol{u}_{kj} \right\|_2 + \frac{\rho_1}{2} \left( \| \boldsymbol{a} - \boldsymbol{u}_{jk} \|_2^2 + \| \boldsymbol{b} - \boldsymbol{u}_{kj} \|_2^2 \right) \tag{5.12}$$

We discuss the problem in the following two cases:

Case (1): When the objective function in Eq. (5.12) is differentiable under the condition of $\|\boldsymbol{u}_{jk} + \boldsymbol{\alpha}_{jk} - \boldsymbol{u}_{kj}\|_2 \neq 0$, taking the sufficient condition of optimality as $\nabla g = 0$, we have:

$$\begin{aligned} c\frac{\boldsymbol{u}_{jk} + \boldsymbol{\alpha}_{jk} - \boldsymbol{u}_{kj}}{\left\|\boldsymbol{u}_{jk} + \boldsymbol{\alpha}_{jk} - \boldsymbol{u}_{kj}\right\|_2} - \rho_1(\boldsymbol{a} - \boldsymbol{u}_{jk}) = 0 \\ -c\frac{\boldsymbol{u}_{jk} + \boldsymbol{\alpha}_{jk} - \boldsymbol{u}_{kj}}{\left\|\boldsymbol{u}_{jk} + \boldsymbol{\alpha}_{jk} - \boldsymbol{u}_{kj}\right\|_2} - \rho_1(\boldsymbol{b} - \boldsymbol{u}_{kj}) = 0 \end{aligned}$$

and we can solve above linear system equations to obtain:

$$\begin{aligned} \boldsymbol{u}_{jk} &= \frac{(c + \tau\rho_1)\boldsymbol{a} + c\boldsymbol{b} - c\boldsymbol{\alpha}_{jk}}{2c + \tau\rho_1} \\ \boldsymbol{u}_{kj} &= \frac{c\boldsymbol{a} + (c + \tau\rho_1)\boldsymbol{b} + c\boldsymbol{\alpha}_{jk}}{2c + \tau\rho_1} \end{aligned}$$

where we define $\tau \coloneqq \|\boldsymbol{u}_{jk} + \boldsymbol{\alpha}_{jk} - \boldsymbol{u}_{kj}\|_2$ which still depends on $\boldsymbol{u}_{jk}$ and $\boldsymbol{u}_{kj}$. Thus we plug the expressions of $\boldsymbol{u}_{jk}$ and $\boldsymbol{u}_{kj}$ back into the definition of $\tau$, and achieve the expression of $\tau = \|\boldsymbol{a} - \boldsymbol{b} + \boldsymbol{\alpha}_{jk}\|_2 - 2c/\rho_1$ which is fully comprised of fixed variables. Hence, we have the solution as below:

$$\begin{aligned} \boldsymbol{u}^*_{jk} &= (1-\theta)\boldsymbol{a} + \theta\boldsymbol{b} - \theta\boldsymbol{\alpha}_{jk} \\ \boldsymbol{u}^*_{kj} &= \theta\boldsymbol{a} + (1-\theta)\boldsymbol{b} + \theta\boldsymbol{\alpha}_{jk} \end{aligned} \tag{5.13}$$

where $\theta \coloneqq c/(\rho_1 \|\boldsymbol{a} - \boldsymbol{b} + \boldsymbol{\alpha}_{jk}\|_2) \geq 0$.

Case (2): When the objective function in (5.12) is not differentiable, we need to solve the quadratic problem $\min \|\boldsymbol{a} - \boldsymbol{u}_{jk}\|_2^2 + \|\boldsymbol{b} - \boldsymbol{u}_{kj}\|_2^2$ with the constraint $\|\boldsymbol{u}_{jk} + \boldsymbol{\alpha}_{jk} - \boldsymbol{u}_{kj}\|_2 = 0$. It yields $\boldsymbol{u}_{jk}^* = (\boldsymbol{a} + \boldsymbol{b} - \boldsymbol{\alpha}_{jk})/2$, $\boldsymbol{u}_{kj}^* = (\boldsymbol{a} + \boldsymbol{b} + \boldsymbol{\alpha}_{kj})/2$, which is in the same form of Eq. (5.13) if $\theta = 1/2$.

To determine which of above differentiable conditions is satisfied, we compare resulting objective values in two cases. Denote $F_1^*$ and $F_2^*$ as the optimal objective value in above two cases, we compute their difference as:

$$\frac{F_1^* - F_2^*}{\|\boldsymbol{a} - \boldsymbol{b} + \boldsymbol{\alpha}_{jk}\|_2} = (\theta^2 - \frac{1}{4})\frac{c}{\theta} + \frac{c\tau\rho_1}{2c + \tau\rho_1}$$

From the definition of $\tau$ and $\theta$, we can derive $\tau\rho_1 = c(1/\theta - 2)$. Combining above:

$$\frac{F_1^* - F_2^*}{c\|\boldsymbol{a} - \boldsymbol{b} + \boldsymbol{\alpha}_{jk}\|_2} = -\frac{(\theta - \frac{1}{2})^2}{\theta}$$

which shows that $F_1^* \leq F_2^*$ unless $\theta = 1/2$. Therefore, we can have a unified solution as Eq. (5.13). □

**$\boldsymbol{\delta}$-Update.** In the last step of our ADMM updating scheme, we have fully independent update rules for each scaled dual variable $\boldsymbol{\delta}_{jk}^u$ as follows:

$$\boldsymbol{\delta}_{jk}^{u\,(l+1)} = \boldsymbol{\delta}_{jk}^{u\,(l)} + (\boldsymbol{x}_j^{(l+1)} - \boldsymbol{u}_{jk}^{(l+1)}) \tag{5.14}$$

Finally, we present the pseudo-code of our DANR approach as **Algorithm 1**.

**Stopping Criterion and Global Convergence.** For our ADMM iterative scheme in Eq. (A.2), we use the norm of primal residual $\boldsymbol{r}^{(l)} = \tilde{\boldsymbol{x}}^{(l)} - \boldsymbol{u}^{(l)}$ and dual residual $\boldsymbol{s}^{(l)} = \rho_1(\boldsymbol{u}^{(l)} - \boldsymbol{u}^{(l+1)})$ as the termination measure. The optimality condition [19] of ADMM shows that if both residuals are small then the objective suboptimality must be small,

```
Algorithm 1: DANR
  Input: loss functions $\{f_j\}_{j\in\mathcal{V}}$, parameter $\lambda_1, \mu_1$.
  Output: $\{\boldsymbol{x}_j\}_{j\in\mathcal{V}}$, $\{\boldsymbol{\alpha}_{jk}\}_{(j,k)\in\mathcal{E}}$.
  while $\|\boldsymbol{r}^{(l)}\|_2 > \epsilon_1$, or $\|\boldsymbol{s}^{(l)}\|_2 > \epsilon_2$. do
      set $l = l + 1$;
      for $j \in \mathcal{V}$ do
          update $x_j^{(l+1)}$ ;
      while $\|\Delta\boldsymbol{u}^{(l')}\|_2 > \epsilon'$, or $\|\Delta\boldsymbol{\alpha}^{(l')}\|_2 > \epsilon'$ do
          set $l' = l' + 1$;
          for $(j,k) \in \mathcal{E}$ do
              update $u_{jk}^{(l'+1)}$, $u_{kj}^{(l'+1)}$ via Lemma 2.1;
              update $\alpha_{jk}^{(l'+1)}$ as Eq. (2.10);
      for $(j,k) \in \mathcal{E}$ do
          update $\delta_{jk}^{u\,(l+1)}$ as Eq. (2.11);
```

Figure 5.1: Pseudo-code of proposed DANR approach.

and thus suggests $\|\boldsymbol{r}^{(l)}\|_2 \leq \epsilon_1 \wedge \|\boldsymbol{s}^{(l)}\|_2 \leq \epsilon_2$ as a reasonable stopping criterion. Convex subproblems in $\boldsymbol{x}$-update and $(\boldsymbol{u}, \boldsymbol{\alpha})$-update need iterative methods to solve as well. The stopping criterion for these subproblems is naturally to keep iteration differences below thresholds, *i.e.* in $(\boldsymbol{u}, \boldsymbol{\alpha})$-update, we require $\|\Delta\boldsymbol{u}^{(l')}\|_2 \leq \epsilon'$ and $\|\Delta\boldsymbol{\alpha}^{(l')}\|_2 \leq \epsilon'$.

**Proposition 10.** *Our ADMM approach to solve the DANR problem is guaranteed to converge to the global optimum.*

*Proof:* Based on rigorous analysis in [19] and [119], we know that the convergence to the global optimum is guaranteed for ADMM algorithms on problems of the following form:

$$\min \ h(\boldsymbol{x}_0) + g(\boldsymbol{z}_0) \qquad s.t. \ A\boldsymbol{x}_0 + B\boldsymbol{z}_0 = \boldsymbol{c}_0 \tag{5.15}$$

where both $h$ and $g$ need to be convex functions and constraints are linear. Note that our problem in Eq.(2.2) is equivalent to Eq.(2.5). To satisfy conditions of convergence guarantee, we need to fit Eq.(2.5) to the form in Eq.(5.15).

We let $\boldsymbol{x}_0 = \boldsymbol{x}$ and let $\boldsymbol{z}_0 = [\boldsymbol{u}, \boldsymbol{\alpha}]$, and further define $h(\boldsymbol{x}_0) = \sum_{j\in\mathcal{V}} f_j(\boldsymbol{x}_j)$ and $g(\boldsymbol{z}_0) = \lambda_1(1-\mu_1)\|\boldsymbol{\alpha}\|_{1,p} + \lambda_1\mu_1 \sum_{(j,k)\in\mathcal{E}} \omega_{jk}\|\boldsymbol{u}_{jk} + \boldsymbol{\alpha}_{jk} - \boldsymbol{u}_{kj}\|_2$. Next, $h(\boldsymbol{x}_0)$ is convex because each cmponent $f_j(\boldsymbol{x}_j)$ is required to be convex in our problem setting. $g(\boldsymbol{z}_0)$ is also convex in terms of the joint variable $[\boldsymbol{u}, \boldsymbol{\alpha}]$, by the definition of convexity and the triangle inequality. Lastly, the constraint $\{[\boldsymbol{u}_{jk}, \boldsymbol{u}_{kj}] = [\boldsymbol{x}_j, \boldsymbol{x}_k], \forall(j,k)\in\mathcal{E}\}$ is also linear in terms of entries in $\boldsymbol{u}$ and $\boldsymbol{x}$. Summing up, we can see that Eq.(2.5) fits the form of Eq.(5.15). Then by theorems in [19] and [119], Our ADMM approach to solve the problem Eq.(2.2) is guaranteed to converge to the global optimum. $\square$

**Computational Complexity.** Let $N_c$ denote the number of iterations that ADMM takes to achieve an approximate solution $\hat{\boldsymbol{x}}$ with an accuracy of $\epsilon_x$. Based on the convergence analysis in [122], the time complexity scales as $O(1/\epsilon_x)$, which in our problem mostly depends on the properties of cost functions $f_{jt}$'s. Assume that all convex subproblems in $x$-update and $(u,\alpha)$-update are solved by general first-order gradient descent methods that take $N_x$ and $N_\alpha$ iterations to converge respectively, the overall complexity of the algorithm is therefore $O\big(N_c\big(N_x|\mathcal{V}| + N_\alpha|\mathcal{E}| + |\mathcal{E}|\big)\big)$.

## 5.3 Extension to Spatio-temporal Setting

The aforementioned shortcomings with pre-defined edge weights accentuate when we consider temporal networks since acquiring explicit temporal edge weights is usually not feasible. Thus, discrepancy-buffering variables are also crucial for the temporal setting.

Consider a temporal undirected network $\mathcal{G}$ consisting of $M$ sequential network snapshots $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_M\}$, where each network snapshot $\mathcal{G}_t$ contains a node set $\mathcal{V}_t$ and an edge set $\mathcal{E}_t$. We denote $\omega^t_{j,k}$ as the weight of spatial edge $(j_t, k_t)$ between nodes $j_t$ and $k_t$ within snapshot $\mathcal{G}_t$, and $\omega^{t,t+1}_j$ for the weight of temporal edge $(j_t, j_{t+1})$ that links node $j_t$ with $j_{t+1}$ across snapshots $\mathcal{G}_t$ and $\mathcal{G}_{t+1}$ (shown in Fig.5.2). On each node $j_t$, a *convex* loss function $f_{j,t} : \mathbb{R}^d \to \mathbb{R}$ is given to measure the fitness of a local model parameterized by $\boldsymbol{x}_{j,t} \in \mathbb{R}^d$. With sparse observations for all snapshots, a straightforward task is to find jointly optimal models for every node and every timestamp under the regularization for both network topology and temporal evolution. We can propose an extension of the DANR formulation to spatio-temporal networks, referred as ST-DANR:

$$\min_{\boldsymbol{x},\boldsymbol{\alpha},\boldsymbol{\beta}} \quad \sum_{j\in\mathcal{V}} \sum_{t=1}^{M} f_{j,t}(\boldsymbol{x}_{j,t}) + \lambda_1 \cdot \mathcal{R}_S(\boldsymbol{x},\boldsymbol{\alpha}) + \lambda_2 \cdot \mathcal{R}_T(\boldsymbol{x},\boldsymbol{\beta}) \tag{5.16}$$

where we define the discrepancy-aware regularizers as:

$$\begin{aligned}
\mathcal{R}_S(\boldsymbol{x},\boldsymbol{\alpha}) &= \mu_1 \sum_{t=1}^{M} \sum_{(j_t,k_t)\in\mathcal{E}_t} \big(\omega_{jk}^{t} \, \|\boldsymbol{x}_{j,t} + \boldsymbol{\alpha}_{jk,t} - \boldsymbol{x}_{k,t}\|_2\big) \\
&\qquad + (1-\mu_1)\|\boldsymbol{\alpha}\|_{1,p} \qquad \text{(spatial penalty term)} \qquad (5.17) \\
\mathcal{R}_T(\boldsymbol{x},\boldsymbol{\beta}) &= \mu_2 \sum_{j\in\mathcal{V}} \sum_{t=1}^{M-1} \big(\omega_{j}^{t,t+1} \|\boldsymbol{x}_{j,t} + \boldsymbol{\beta}_{j,t} - \boldsymbol{x}_{j,t+1}\|_2\big) \\
&\qquad + (1-\mu_2)\|\boldsymbol{\beta}\|_{1,p} \quad \text{(temporal penalty term)} \qquad (5.18)
\end{aligned}$$

As we are interested in the heterogeneous evolution of nodal models across time, the chosen unsquared norms in spatial and temporal regularization terms $\mathcal{R}_S(\boldsymbol{x},\boldsymbol{\alpha})$ and $\mathcal{R}_T(\boldsymbol{x},\boldsymbol{\beta})$ enforce piecewise consensus in both spatial and temporal aspects, which indicates abrupt changes of regional models or sudden transitions in network structures at particular timestamps, and also implies valid persistent models in the remaining segments of time [123]. Beside the spatial DB variables $\boldsymbol{\alpha}_{jk,t}$ in $\mathcal{R}_S(\boldsymbol{x},\boldsymbol{\alpha})$, we define another set of temporal DB variables $\boldsymbol{\beta}_{j,t} \in \mathbb{R}^d$ in $\mathcal{R}_T(\boldsymbol{x},\boldsymbol{\beta})$ to compensate for inadequate regularizations caused by temporal fluctuations. For example, when modeling housing prices in a city, $\boldsymbol{\beta}_{j,t}$ would tolerate anomalous short-term fluctuations in prices, while large values of $\boldsymbol{\alpha}_{jk,t}$ might be found near boundaries of disparate neighborhoods.

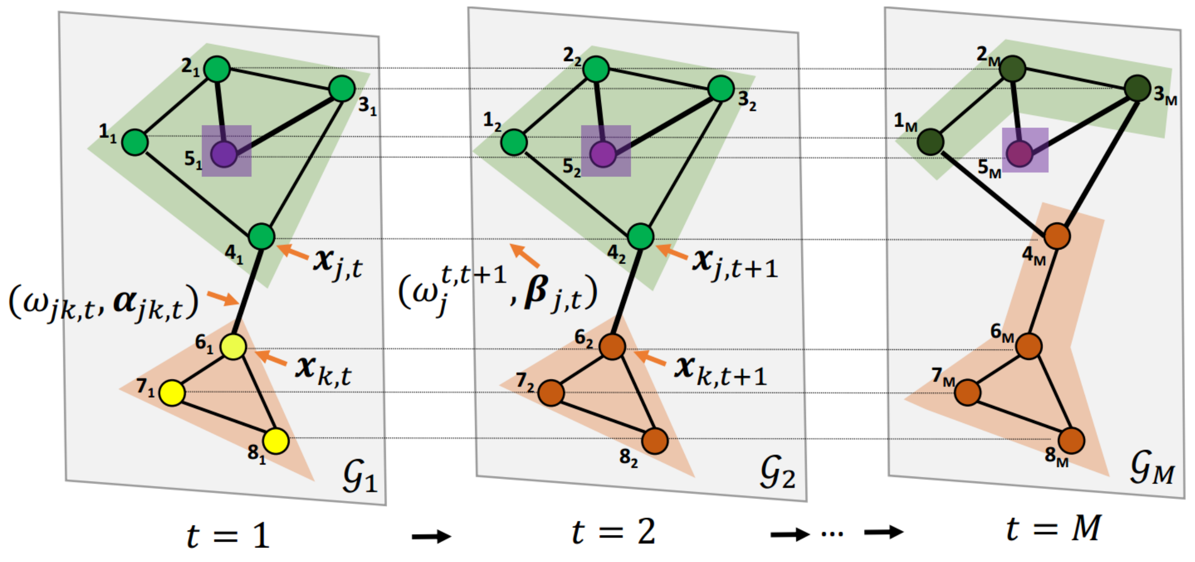


Figure 5.2: Overview of our problem setting in temporal networks.

**Adaptation to Streaming Data.** In many applications, observed network snapshots arrive in a streaming fashion. Instead of recomputing all models on past snapshots whenever a new snapshot is observed, incorporating existing models to facilitate the new incoming snapshot is more efficient. Assume that we have learned models $\{\hat{\boldsymbol{x}}_{j,t}\}_{j\in\mathcal{V}}$ for the $\tau$-th snapshot, we then read observations of the next $m$ snapshots indexed by $\tau+1,\cdots,\tau+m$ and attempt to learn models for them. To this intent, we deploy our ST-DANR formulation in Eq. (5.16) on a limited time interval $t=\tau,\cdots,\tau+m$, and then integrate established model at $t=\tau$ by fixing $\{\boldsymbol{x}_{j,\tau}\}_{j\in\mathcal{V}} = \{\hat{\boldsymbol{x}}_{j,\tau}\}_{j\in\mathcal{V}}$ and solving the remaining variables $\{\boldsymbol{x}_{j,t}\}_{j\in\mathcal{V}}^{t=\tau+1,\cdots,\tau+m}$ in the corresponding problem. Note that in our setting, temporal messages are only transferable through consecutive snapshots, meaning that fixing the $\tau$-th snapshot is the same as fixing all past snapshots.

Lastly, we point out that distributed ADMM-based solution for DANR (§2.2) can be adapted to ST-DANR, for either batch formulation (Eq. (5.16)) or streaming-case formulation. Details in Appendix.

**Temporal Evolutionary Patterns.** In this work, we consider the unsquared edge penalty term $\|\boldsymbol{x}_{j,t} + \boldsymbol{\beta}_{j,t} - \boldsymbol{x}_{j,t+1}\|_2$ in $\mathcal{R}_T$ since it enforces temporal reconstruction, indicating sharp transitions or change points at particular timestamps. Similar to the spatial case, there could be multiple available alternatives to enforce different types of evolutionary patterns [123]. For example, replacing the unsquared norm with squared norm form $\|\boldsymbol{x}_{j,t} + \boldsymbol{\beta}_{j,t} - \boldsymbol{x}_{j,t+1}\|_2^2$ will yield smoothly varying models along consecutive snapshots, and has been well studied in [124, 125, 126, 127].

## 5.4 Experiments on Synthetic, Housing Rental Price and LAGOS Datasets

In this section, we present an experimental analysis of the proposed method (DANR) and evaluate its performance on classification and regression tasks. First, we employ a synthetic dataset (§5.4.1) and two housing price datasets (BAY and SAC in §5.4.2) to demonstrate the set of scenarios and applications that are particularly good fit for DANR. In addition to an improvement on the estimation task, DANR learns fine-grained neighborhood boundaries and reveals interesting insights into the network's heterogeneous structure. Lastly, in order to validate the efficacy of ST-DANR in the temporal setting, we conduct experiments with the geospatial and temporal database of Northeastern US lakes (§5.4.3), from which we aim to estimate the water quality of lakes over 10 years.

**Baseline Algorithms.** In spatial network scenarios (§4.1 & §4.2), we compare DANR against three baselines: network lasso (NL) [18], robust multi-task feature learning (rMTFL) [128], factorized multi-task learning (FORMULA) [129]. In spatial-temporal network scenarios (§4.3), we compare ST-DANR with two widely-applied temporal regularizers in the literature, which are detailed in §4.3.

**Parameters Setting.** For both NL and DANR, we tune $\lambda$ parameters from $10^{-3}$

to $10^2$ where $\lambda^{n+1} = 1.3\lambda^n$. Concerning the DANR, for each value of $\lambda$, we further tune $\mu$ parameters from 0.3 to 1, where $\mu^{n+1} = \mu^n + 0.02$. Lastly, we set $p = 3$. We follow the same strategy for both spatial (DANR) and spatio-temporal (ST-DANR) variants of the proposed method. For all penalty parameters in rMTFL and FORMULA, we tune them in the same way as $\lambda$ in DANR. In addition, we vary the number of factorized base models in FORMULA from 1 to 50 to achieve its best performance. For each dataset, we standardize all features and response variables to zero mean and unit variance.

### 5.4.1 Node Classification with Limited Data

Following previous work [18], we first experiment with a synthetic network in which each node attempts to solve its own support vector machine (SVM) classifier, but only has a limited amount of data to learn from. We further split the nodes into clusters where each cluster has an underlying model, i.e., nodes in the same cluster share the same underlying model. Our aim is to learn accurate models for each node by leveraging their network connections and limited observations. Note that the neighbors with different underlying models provide misleading information to each other, which potentially harms the overall performance. Therefore, we later investigate the robustness of the DANR with a varying ratio of such malicious edges.

**Synthetic Network Generation.** We create a network of 100 nodes, split into 5 ground-truth communities $C^1$, $C^2$, $\cdots$, $C^5$. Each of these communities consists of 20 nodes. We denote the community of a node $i$ with $C_i$. Next, we form the network connections by adding edges between nodes based on the following criterion: The probability of adding an edge between any node pair $(i, j)$ is 0.5 if $C_i = C_j$, i.e., the edge connects nodes from the same community (*intra-community edge*). Otherwise, we set the probability as 0.02 if $C_i \neq C_j$, i.e., the edge connects nodes from different communities (*inter-community edge*). The resulting synthetic network $G_0(\mathcal{V}, \mathcal{E})$ has 100 nodes and 574 edges, with 82% of them being intra-community edges. Each ground-truth community $C^k$ has an underlying classifier model $\boldsymbol{X}^k \in R^{10}$, drawn independently from a normal distribution of zero mean and unit variance, where $k \in [1, \cdots, 5]$. Next, we generate 5 random training example pairs $(\boldsymbol{w}, y)$ per node, where $\boldsymbol{w} \in R^{10}$ denotes an observation and $y \in \{-1, 1\}$ denotes the ground-truth value to estimate. The ground-truth value $y$ for each observation $\boldsymbol{w}$ is then computed using the underlying classifier of the community that a node belongs to:

$$y = \text{sign}(\boldsymbol{w}^T \boldsymbol{X}^k + \eta) \tag{5.19}$$

where $\boldsymbol{X}^k \in R^{10}$ represents the corresponding weight of the classifier w.r.t. observations, while $\eta$ is the noise. All observations and noises are drawn independently from a normal distribution for each data point. Note that 5 observations are insufficient to accurately estimate a model $\boldsymbol{x}_i \in R^{10}$. Hence for this setting, the network connections play an important role as nodes can "borrow" training examples from its neighbors to improve their own models.

**Optimization Problem.** The objective function $f_i$ on each node is formulated by the soft margin SVM objective, where node $i$ estimates its model (i.e. separating hyperplane) $\boldsymbol{x}_i \in R^{10}$ using its five training examples $(\boldsymbol{w}, y)$ as follows:

$$\min \Big(\frac{1}{2}\boldsymbol{x}_i^T \boldsymbol{x}_i + C \sum_{l=1}^{5} \xi_l\Big) \quad \text{s.t. } y^{(l)}(\boldsymbol{x}_i^T \boldsymbol{w}^{(l)}) \geqslant (1 - \xi_l),\ \xi_l \geqslant 0$$

where $\xi_l$'s are slack variables that accounts for non-separable data. The constant $C$ controls the trade-off between minimizing the training error and maximizing the margin. We set $C$ to 0.75, which we empirically find to perform well.

**Test Results.** In following subsections, we first thoroughly inspect DANR and its prototype method NL on synthetic network $G_0$, in order to show the benefits of introducing discrepancy-buffering variable $\boldsymbol{\alpha}$. Later in §4.1.2, we evaluate the performance and robustness of DANR and all three baseline methods under more noisy scenarios. To assess the performance, prediction accuracies are computed over a separate set of 1000 test pairs (10 per node). The test pairs are again randomly generated using Eq. (5.19).

**Effect of Discrepancy-Buffering Variables.**

We compare DANR with NL and report the prediction accuracy with varying $\lambda$ values in Figure 5.3. Recall that the proposed method introduces the $\mu$ parameter which is coupled with the $\lambda$ parameter. Therefore in order to have an adequate comparison between DANR and NL, we modify our $\lambda$ parameter to be $\tilde{\lambda} = \lambda/\mu$. Doing so ensures that both methods apply the same amount of penalty to their network regularization terms (See Eq. 5.2 and 5.3). For each value of $\lambda$, we vary $\mu$ from 0.3 to 1 and report the highest accuracy achieved.

We first briefly mention the desired behaviors of the NL method and later accentuate its drawbacks. As shown by Figure 5.3, for small $\lambda$ values, the problem reduces to solving the local optimization problem where each node estimates its model solely based on its limited observations. This achieves 61.3% accuracy on the test set. Furthermore, as $\lambda$ increases, the NL penalty enforces nodes to form clusters, where nodes in the same cluster share the same model. This behavior firmly improves the test performance with prediction accuracy rising to 75.3%. Right after the peak performance is reached however, a slight increase in $\lambda$ causes a sudden drop in the performance and the problem reduces to solving the global optimization problem over the entire network. Therefore when $\lambda > \lambda_{critical}$, the problem converges to a common model for all the nodes in the network , i.e., $\boldsymbol{x}_i = \boldsymbol{x}_j$ for $\forall i, j \in \mathcal{V}$. This further drops the prediction accuracy to 56.4% on the test set.

The observed rapid drop in performance implies that the NL method is highly sensitive to the $\lambda$ parameter. There is only a narrow window of $\lambda$ values that result in a proper clustering of the network. As a result, one needs to excessively tune the $\lambda$ parameter around $\lambda_{critical}$ to find its optimal value. In addition, such clustering performance is also highly affected by the volume of "malicious" (inter-cluster) edges in the network, as shown in §5.4.1.

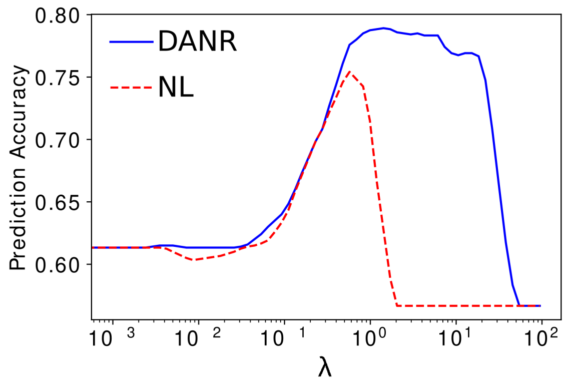


Figure 5.3: Prediction accuracy comparison with varying $\lambda$ values.

From Figure 5.3, we can observe that the DANR approach exhibits improved performance thanks to its discrepancy-buffering variables, $\alpha$, which allow our regularization term to better exploit the good edges while providing additional tolerance towards bad edges. As a result, DANR achieves 79.1% accuracy on the test set, outperforming the best baseline method by 3.8%. Another important observation is that DANR provides a much wider window for selecting near-optimal $\lambda$ than the baseline approach. That being said, we now analyze how the $\mu$ parameter couples with the $\lambda$ parameter, and further present its effect on DANR's performance.

Figure 5.4 displays the prediction accuracy vs. $\mu$, with a fixed $\lambda = 10^0$ and $10^1$. Intuitively, as $\lambda$ increases, the optimal value of $\mu$ also increases. The reason behind is that as $\lambda$ increases, the more non-zero discrepancy-buffering variables (parameterized by

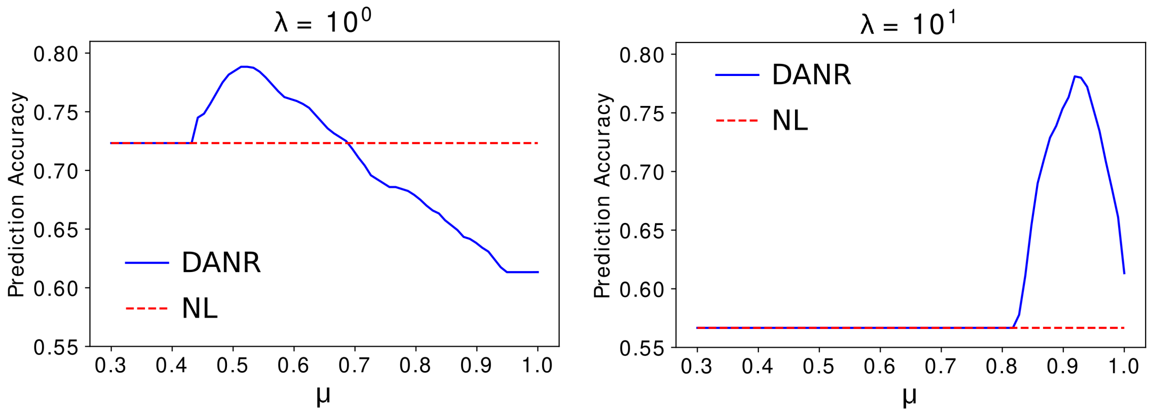


Figure 5.4: Effect of $\mu$ on the prediction accuracy, with a fixed $\lambda$.

$(1 - \mu)$) are needed to persist the accurate clustering formation since high values of $\lambda$ forces global consensus over the network.

**Robustness to Malicious Edges.**

Note that the ratio of inter-community edges in the earlier synthetic network $G_0$ is 18%. Here we use term *noise* for the ratio of such malicious edges in the network. Taking into account that the amount of noise is critical in learning accurate models for nodes in a network, we now investigate the efficacy of DANR w.r.t. varying noise. First, we remove all inter-community edges and later iteratively add malicious edges to the network. Then we solve the same problem with the noise varying from 0 to 0.6, and report prediction accuracies of all methods in Figure 5.5.

As shown, the performance of DANR, NL and FORMULA drops in the absence of noise. Meanwhile the pure multitask method rMTFL maintains the same low accuracy, since it doesn't utilize the network information. However, as we increase the noise, DANR starts to outperform NL and FORMULA, where the gap between the models' performances becomes larger at a higher ratio of malicious edges. This confirms that compared to the baseline methods, the DANR exhibits more robust performance in noisy

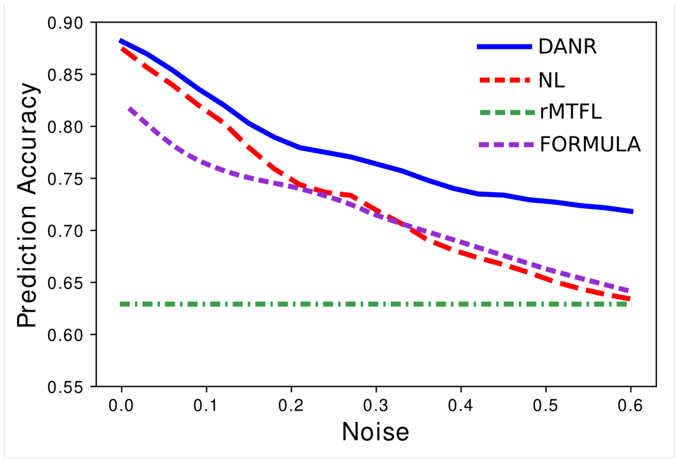


Figure 5.5: Prediction accuracy comparison with varying noise.

settings thanks to its discrepancy-buffering variables.

### Running Time and Scalability

| # nodes | DANR | NL | rMTFL | FORMULA |
|---|---:|---:|---:|---:|
| 100 | 4.18 s | 1.41 s | 3.32 s | 7.38 s |
| 500 | 15.10 s | 5.75 s | 7.88 s | 24.62 s |
| 1000 | 48.85 s | 12.36 s | 53.91 s | 62.68 s |
| 5000 | 146.78 s | 37.99 s | 221.52 s | 446.21 s |
| 10000 | 484.68 s | 123.81 s | - | - |

Table 5.1: Running time on synthetic datasets when we vary the number of nodes in the network. '-' means the method cannot finish within limited time period.

To test the scalability of our method, we create synthetic dataset in the same way as in §4.1, but varying the number of nodes in the graph from 100 to 10000. Moreover, we tune the probability of adding edges within and between clusters so that the degree of each node is maintained as 20 for all graph sizes. We compare the running time of all methods in the table above. The rMTFL and FORMULA cannot give a result when the graph has 10000 nodes. DANR and NL are both distributed methods, and therefore their running times increase almost linearly with the increasing number of nodes. Comparing with

NL, DANR spends a longer time to get more robust results, which suggests a trade-off between complexity and robustness.

### 5.4.2 Spatial: Housing Rental Price Estimation

In this section, our goal is to jointly (i) estimate the rental prices of houses by leveraging their geological location and the set of features; (ii) discover boundaries between neighborhoods. The intuition is that the houses in the same neighborhood often tend to have similar pricing models. However, such neighborhoods can have complex underlying structures (as described later in this section), which imposes additional challenges in learning accurate models.

**Dataset.** We experiment with two largely populated areas in Northern California: the Greater Sacramento Area (SAC) and the Bay Area (BAY). The anonymized data is provided by the property management software company *Appfolio Inc*, from which we further sample houses with at least one signed rental agreement during the year 2017. The resulting dataset covers 3849 houses in SAC and 1498 houses in BAY. Concerning the houses that have more than one rental agreement signed during 2017, we average the rental prices listed in all the agreements. Each house holds the information about its location (latitude/longitude), number of bedrooms, number of bathrooms, square footage, and the rental price. We regard these areas (SAC and BAY) as two separate datasets for the remainder of this section. We also randomly split 20 % of the houses in each dataset for testing.

**Network Construction.** After excluding test houses from both datasets, we construct two networks (one for each dataset) based on the houses' locations. An undirected edge exists between node $i$ and $j$, if at least one of them is in the set of 10 nearest neighbors of the other. (Note that node $j$ being one of the 10 nearest neighbors of $i$ doesn't necessarily imply that node $i$ is also in the set of nearest neighbors of $j$.) In the resulting networks, the average degree of a node is 12.16 for SAC and 12.04 for BAY. Additionally, we construct two versions of these networks, *weighted* and *unweighted*. While the weighted network has edge weights inversely proportional to the distances between houses, the unweighted network ignores the proximity between houses, and thus weights on all the edges are 1.

**Optimization Problem.** The model at each house estimates its rental price by solving a linear regression problem. More specifically, at each node $i$ we learn a 4-dimensional model $\boldsymbol{x}_i = [a_i, b_i, c_i, d_i]$. $\boldsymbol{x}$ simply represents the coefficients of each feature, which later is used to estimate the rental price $p_i$ as follows:

$$\begin{aligned}\overline{p_i} = a_i \cdot (\#bedrooms) + b_i \cdot (\#bathrooms)\\ +c_i \cdot (squarefootage) + d_i,\end{aligned}$$

where $d_i$ is the bias term. The training objective is

$$\min_{\boldsymbol{x},\boldsymbol{\alpha}} \sum_{i\in\mathcal{V}} \|\overline{p_i} - p_i\|_2^2 + c \cdot \|\boldsymbol{x}_i\|_2^2 + \lambda \cdot \mathcal{R}_S(\boldsymbol{x}, \boldsymbol{\alpha})$$

where $\mathcal{R}_S$ represents the proposed network regularization term (see Eq. 5.3), while $c$ is the regularization weight to prevent over-fitting.

**Test Results.** Once training converges, we predict the rental prices on the test set. To do that, we connect each house in the test set to its 10 nearest neighbors in the training set. We then infer the new model $\boldsymbol{x}_j$ by taking the average of the models on its neighbors: $\boldsymbol{x}_j = (1/|N(j)|)\sum_{k\in N(j)} \boldsymbol{x}_k$. The model $\boldsymbol{x}_j$ is further used to estimate the rental price of the corresponding house. Alternatively, one can also infer $\boldsymbol{x}_j$ by solving $\min_{\boldsymbol{x}_j} \sum_{k\in N(j)} \|\boldsymbol{x}_j - \boldsymbol{x}_k^*\|_2$, while keeping the models on neighbors fixed. However, we empirically find that simply averaging the neighbors' models performs better for both methods in this particular setting. We compute Mean Squared Error (MSE) over test nodes to evaluate the performance.

| Method | BAY | SAC |
|---|---|---|
| Local estimation ($\lambda = 0$) | 0.5984 | 0.6250 |
| Global estimation ($\lambda > \lambda_{critical}$) | 0.4951 | 0.5403 |
| rMTFL | 0.4774 | 0.4115 |
| FORMULA (unweighted) | 0.4446 | 0.3503 |
| FORMULA (weighted) | 0.4181 | 0.3379 |
| Network Lasso (unweighted) | 0.4392 | 0.3273 |
| Network Lasso (weighted) | 0.4173 | 0.3022 |
| DANR (unweighted) | **0.4106** | **0.2978** |

Table 5.2: MSE for housing rental price prediction on test set.

Table 5.2 displays the test results of eight different settings on both datasets. As shown, local & global estimations and rMTFL method produce high errors for both datasets. We further apply FORMULA and Network Lasso (NL) methods to both weighted and unweighted versions of the networks, while the proposed method is only applied to the unweighted version. Notice that the network weights are irrelevant for the local & global estimation settings and the rMTFL method. Intuitively, both FORMULA and NL performs better on the weighted setting compared to the unweighted setting for both datasets. As shown, the rental price estimation errors achieved by the NL are 0.4173 (weighted) vs. 0.4392 (unweighted) for BAY and 0.3022 (weighted) vs. 0.3273 (unweighted) for SAC. These results suggest that the pre-defined weights on these networks help to learn more accurate models on houses.

However, we further argue that although such pre-defined weights imprsssove the overall clustering performance, they don't account for more complex clustering scenarios, e.g., two nearby houses falling into different school districts or some houses having higher values compared to their neighbors due to geography, e.g., having a view of the city. That being said, DANR outperforms all the other baselines and achieves the smallest errors for both datasets; 0.4106 for BAY and 0.2978 for SAC. Especially, the DANR (unweighted) even outperforms the weighted version of the baseline approaches by a notable margin. This reveals that the DANR indeed accounts for such heterogeneities in data and provides enhanced clustering of the network.

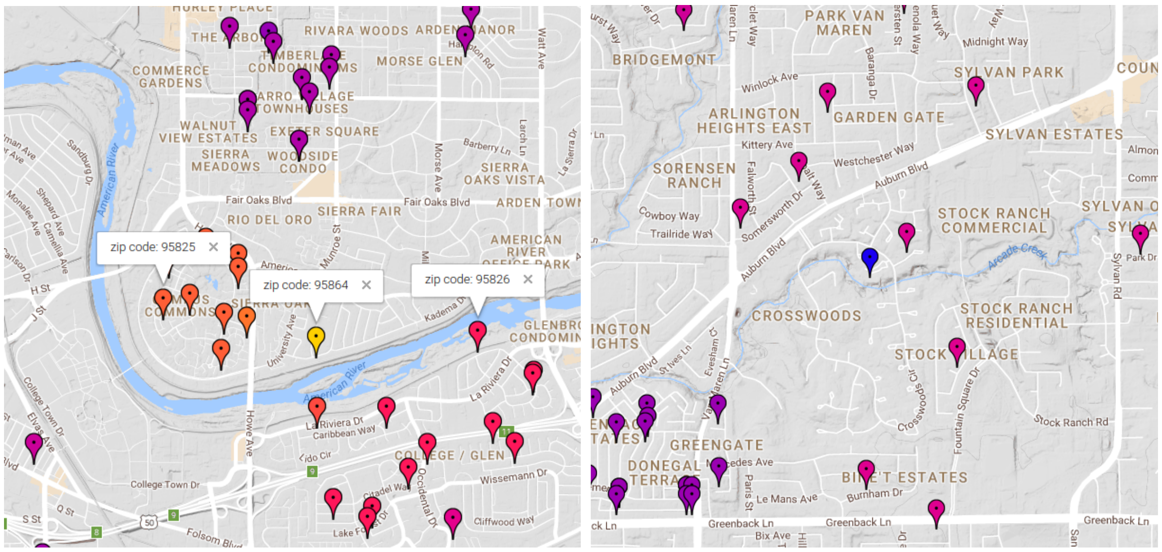


Figure 5.6: Examples of complex neighborhood structures captured by the DANR. In left, the house shown in yellow resides at the border of three different area codes. In right, the creek side house (colored in blue) differs from all of its neighbors, possibly due to its appealing location.

Figure 5.6 shows examples of two complex scenarios that are captured by the DANR from the SAC network. We use a color code to represent the clusters in the network, where the same colored houses $(i, j)$ are in consensus on their models $(\boldsymbol{x}_i = \boldsymbol{x}_j)$. In the left subfigure, the house shown in yellow uses a different model than all of its neighbors. Interestingly, it resides at the border of three different area codes. The area code for this house is 95864, while the area code on its west is 95825 and 95826 on its south-east. Additionally, we observe similar heterogeneous behaviors in some houses that are near creeks, lakes, and rivers. As an example, Figure 5.6 (right) displays a creek side house (colored in blue), which again uses a different underlying model from its neighbors.

### 5.4.3 Spatio-Temporal: Water Quality Estimation of Northeastern US Lakes

We now evaluate our method in spatio-temporal setting, where the aim is to dynamically estimate the water quality of Northeastern US lakes over years. We follow the same procedure in §5.4.2, but have an additional temporal regularization term in our objective that allows nodes to obtain signals from past snapshots of the network, along with their neighbors in the current snapshot.

**Dataset and Network.** We experiment with the geospatial and temporal dataset of lakes in 17 states in the Northeastern United States, called LAGOS [130]. The dataset holds extensive information about the physical characteristics, various nutrient concentration measurements (water quality), ecological context (surrounding land use/cover, climate etc.), and location of lakes; from which we select the following available set of features: {*max depth, surface area, elevation, annual mean temperature, % of agricultural land, % of urban land, % of forest land, % of wetland*}. The water quality metric to estimate is the summerly mean *chlorophyll concentration* of the lakes. We represent the feature vector with $\boldsymbol{w} \in \mathbb{R}^8$ and the water quality score with $y \in \mathbb{R}$.

During our experiments, we consider a 10-year period between 2000 and 2010. Due to sparsity in the data, we allow 2 years range between the two consecutive snapshots of the network. This results in total of 1039 lakes with all the aforementioned features and the water quality measurements available for each of the selected years. After randomly splitting 20% of the lakes for testing, we build our network by using the latitude/longitude information of lakes, where each lake (node) is connected to 10 nearest lakes.

**Optimization Problem.** After constructing the network, we now aim to *dynamically* estimate the water quality ($y_{i,t}$) of lakes by using their feature vectors ($\boldsymbol{w}_{i,t}$). More formally, for each year $t \in [2000, 2002, \cdots, 2010]$, we learn a model $\boldsymbol{x}_{i,t} \in \mathbb{R}^8$ per node by solving the following spatio-temporal problem in a *streaming* fashion: (4.16)

$$\min_{\boldsymbol{x},\boldsymbol{\alpha},\boldsymbol{\beta}} \sum_{i\in\mathcal{V}} f(\boldsymbol{x}_{i,t}, \boldsymbol{w}_{i,t}, y_{i,t}) + \lambda_1 \cdot \mathcal{R}_S^t(\boldsymbol{x}, \boldsymbol{\alpha}) + \lambda_2 \cdot \mathcal{R}_T^t(\boldsymbol{x}, \boldsymbol{\beta})$$

where $f$ is the linear regression objective and $\mathcal{R}_S$ is the DANR term applied on the spatial edges. $\mathcal{R}_T$ is the temporal regularization term, for which we consider two formulations as described next.

**Test results.** For each year, we first solve the spatial problem (Eq. (4.16) without the temporal term $\mathcal{R}_T$) and report the Mean Squared Errors in Table 5.3. Note that the test nodes are again inferred by averaging the neighbors' models in the current snapshot. Later, in order to successfully assess the improvements gained by the temporal discrepancy-buffering variables ($\boldsymbol{\beta}$), we solve the above spatio-temporal problem with three different temporal regularizers:

**DANR+T-SON:** DANR with temporal sum-of-norms regularizer where

$$\mathcal{R}_T^t(\boldsymbol{x}, \cdot) = \sum_{i\in\mathcal{V}} \|\boldsymbol{x}_{i,t} - \hat{\boldsymbol{x}}_{i,t-1}\|_2$$

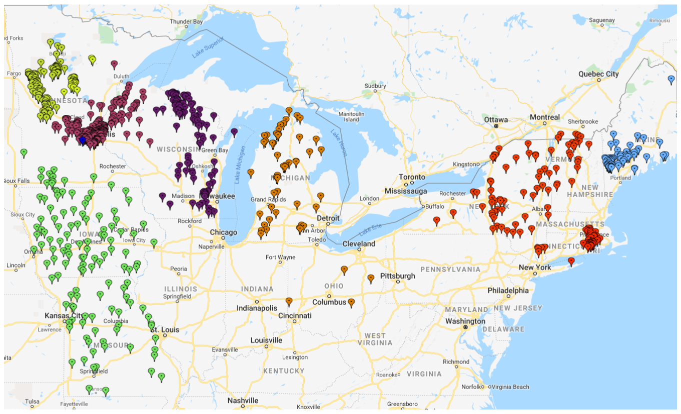
**DANR+T-SOS:** DANR with temporal sum-of-squares regularizer where

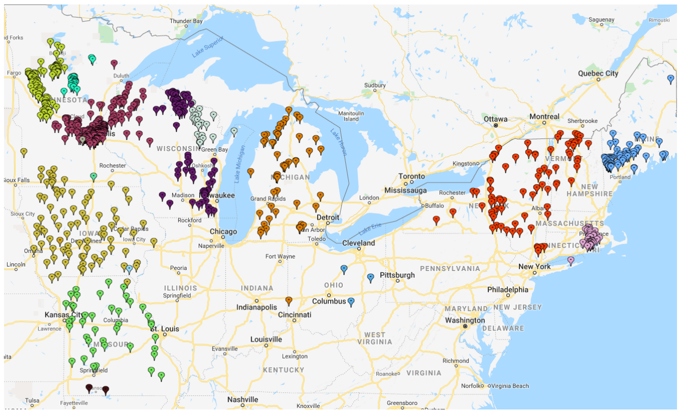
$$\mathcal{R}_T^t(\boldsymbol{x}, \cdot) = \sum_{i\in\mathcal{V}} \|\boldsymbol{x}_{i,t} - \hat{\boldsymbol{x}}_{i,t-1}\|_2^2$$

**ST-DANR:** Spatio-temporal discrepancy-aware network regularizer where

$$\mathcal{R}_T^t(\boldsymbol{x}, \boldsymbol{\beta}) = \mu_2 \cdot \sum_{i\in\mathcal{V}} \|\boldsymbol{x}_{i,t} - \hat{\boldsymbol{x}}_{i,t-1} + \boldsymbol{\beta}_{i,t}\|_2 + (1-\mu_2) \cdot \|\boldsymbol{\beta}\|_{1,p}$$

DANR+T-SON and DANR+T-SOS approaches simply apply two widely adopted temporal regularizers on the temporal edges (the sum-of-norms regularizer [131] and


(a) Year 2000


(b) Year 2002

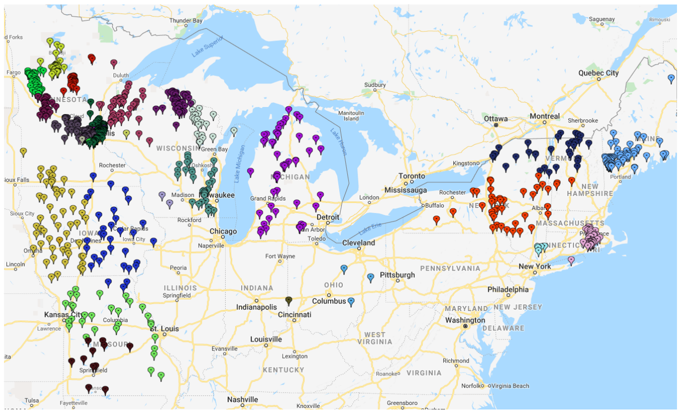

(c) Year 2004

Figure 5.7: Evolution of clustering captured by the ST-DANR over years.

sum-of-squares reqularizer [124, 127, 125] respectively), while ST-DANR includes the discrepancy-buffering variables on both spatial and temporal edges. Recall that we solve the Eq. (4.16) in a streaming fashion for simplicity, i.e., each snapshot of the network solves the problem while holding the models learned on the previous snapshot (if available) fixed. Potentially, the performance can further be improved by allowing updates on the previous snapshots.

| Method | 2000 | 2002 | 2004 | 2006 | 2008 | 2010 |
|---|---|---|---|---|---|---|
| DANR | 0.8479 | 0.9033 | 0.6384 | 0.6722 | 0.4556 | 0.4113 |
| + T-SON | N/A | 0.8308 | 0.5744 | 0.6061 | 0.4109 | 0.3517 |
| + T-SOS | N/A | 0.8045 | 0.5618 | 0.6045 | 0.3978 | 0.3476 |
| ST-DANR | N/A | 0.8037 | 0.5604 | 0.5866 | 0.3844 | 0.3311 |

Table 5.3: MSE for water quality estimation over years.

As we can see from the Table 5.3, leveraging the temporal connections between the two consecutive snapshots significantly improves the performance. The DANR+T-SON outperforms the DANR by 8%-14%, while DANR+TSOS outperforms the DANR by 10%-15%. Moreover, the ST-DANR consistently outperforms both baselines for all the years. This confirms that the proposed formulation accounts for the heterogeneous nature of the temporal networks where some group of nodes exhibits different evolution patterns than the others.

Figure 5.7 further displays the models (color-coded) learned on three consecutive snapshots, corresponding to years 2000, 2002 and 2004. In 2000, the formed clusters don't go much beyond the boundaries of the states, resulting in coarse clustering of the network. Yet some states such as Missouri and Iowa are grouped into one cluster (shown in green). This suggests that accurately estimating the chlorophyll concentration of lakes based on the selected set of features is indeed challenging. Specifically, the estimation problem depends on several other external factors that potentially affect the volume of plants and algae in lakes; cultural eutrophication, nutrient inputs from human activities to name a few [130]. However, as it can be seen from the models learned in 2002 and 2004, the clustering performance improves once we allow temporal regularization to synchronize models between two consecutive snapshots, which is analogous with the reduction in errors over years as reported in Table 5.3. Overall, we observe a split of clusters over time, e.g., in Wisconsin, Minnesota, Iowa and New England. Additionally, a dark brown cluster in south Missouri begins to appear in 2002 and further spreads north in 2004. This indicates that by leveraging the temporal connections, the ST-DANR learns improved models and clustering while allowing for heterogeneity in group level evolutions of nodes.

## 5.5 Related Work

Among existing network-based regularization approaches, there are two major types of edge objectives that are most commonly used: the square-norm objective and the unsquared-norm objective, encouraging different styles of expected solutions. The squared-norm edge objective (i.e., $\sum_{jk} \omega_{jk} \|\boldsymbol{x}_j - \boldsymbol{x}_k\|_2^2$), well-known as *graph Laplacian regularization* [132], assumes underlying models on nodes are smoothly varying as one traverses edges in the network, and accordingly enforces similar but not identical models on linked nodes. Due to the merits of simple computations and good performance, graph Laplacian regularizer has gained popularity in solving problems, such as image deblurring and denoising [106], genomic data analysis [13], semi-supervised regression [133], non-negative matrix factorization [134] and semi-definite programming [135]. However, the square-norm objective usually induces very dense models. The unsquared-norm objective, known as *sum-of-norms* regularizer [15, 131], bears some similarity to *scale-valued fused lasso* signal approximator [16] that exists in signal processing applications, for example, *total-variation* (TV) regularizer for image denoising, and *graph trend filtering* (GTF) regularizer for nonparametric regression. TV regularizer [136] is developed to penalize both the horizontal and vertical differences between neighboring pixels, which can be thought of as a special scalar-valued network lasso on a 2D grid network. GTF [105] generalizes the successful idea of trend filtering to graphs, by directly penalizing higher order differences across nodes. All regularizers above require correct edge weights so that network structures can be properly used to improve joint estimation. In contrast, our proposed approach allows ambiguous input of network edge weights by introducing the discrepancy-buffering variables. Also, as a variation of SON regularization, our formulation enjoys small model complexity as local consensus is imposed across large-scale networks.

## 5.6 Conclusions

We propose discrepancy-aware network regularization (DANR) approach for spatio-temporal networks. By introducing discrepancy-buffering variables, the DANR automatically compensates for inaccurate prior knowledge encoded in edge weights, and enables modeling heterogeneous temporal patterns of network clusters. We develop a scalable algorithm based on alternating direction method of multipliers (ADMM) to solve the proposed problem, which enjoys analytic solutions for decomposed subproblems and employs a distributed update scheme on nodes and edges. Our experimental results show that DANR successfully captures structural changes over spatio-temporal networks and yields enhanced models by providing robustness towards missing or inaccurate edge weights.

# Chapter 6

# Reconstructing Coupled Networks Under Domain Constraints

## 6.1 Introduction

Recently, there has been an effort to move research from the investigation of single networks to the more realistic scenario of multiple coupled networks. In this work, we consider the case of pairs of networks, $(\mathcal{G}_1, \mathcal{G}_2)$, that are categorized into different modalities over a population. Such coupled network systems can be found in infrastructures of modern society (energy-communication), financial systems (ownership-trade), or even human brains (anatomical substrate-cortical activation). Our goal is to reconstruct one network from information on the other, and during such a process obtaining a concise interpretation of how one network affects the other.

To achieve the above goal, we consider an edge-by-edge formulation. We treat one set of edges in $\mathcal{G}_1$ as predictors and the other set of edges in $\mathcal{G}_2$ as response variables in a multivariate linear regression model. Past research for the above problem relies on the restrictive Gaussian assumption, which simplifies the problem but is difficult to justify, especially in the domain of brain architectures. Adopting Gaussian assumption on non-Gaussian data can significantly prevent the detection of conditional dependencies and may lead to incorrectly inferred relationships among variables.

The learning of relationships between two different modalities can be difficult without sufficient data. As a result, in sparser data settings, the ability to specify constraints based on domain knowledge can be beneficial. For example, in the case of brain data, functional edges have mainly local influences, and structural edges are more responsible for long-distance influences [137, 138]. We want preferences encoded in domain knowledge to guide the selection of partial correlations of unexplained noise terms in the constructed model.

Based on the above motivations, we propose a flexible and efficient framework CC-MRCE ($\underline{C}$onvex-set $\underline{C}$onstrained $\underline{M}$ultivariate $\underline{R}$egression with $\underline{C}$ovariance $\underline{E}$stimation) that simultaneously learns both regression coefficients between two coupled networks and the correlation structure of noise terms. In a departure from existing methods, our framework encodes domain knowledge as a set of convex constraints and adopts a pseudolikelihood-based neighborhood-selection objective in partial correlation estimation, which has been shown to be more robust to non-Gaussian data. Because of the CC-MRCE objective's bi-convex nature, we alternately solve a regression sub-problem and a constrained partial correlation sub-problem until convergence. The latter sub-problem requires feasible solutions under given domain constraints that we render tractable via a modified two-stage proximal gradient descent method.

We illustrate the use of our method in the context of the human brain. Brain data presents one of the greatest technical challenges in analysis and modeling due to a network-based characterization [139, 140], non-Gaussian nature of data, high dimensionality, a small number of samples, and the need to incorporate domain knowledge. We apply the proposed framework on the Human Connectome Project (HCP) dataset [141], where two coupled networks are constructed from fMRI scans (representing cortical activation) and diffusion scans (representing the anatomical substrate). We successfully predict a brain functional network from the given structural network; our method outperforms previous state-of-art methods, and our obtained models are easier to interpret. We investigate the structure-function coupling for seven different tasks. Our findings agree with the nature of fMRI task and brain region functions in existing literature, thus validating our model's ability to discover meaningful couplings.

Our main contributions are as follows: (1) We propose a regularized multiple regression approach that adapts to non-Gaussian data. (2) We incorporate prior domain knowledge to model estimation by formulating constraints into an optimization problem. (3) We develop a fast method based on nested FISTA for solving the proposed optimization problem. (4) We show the effectiveness of our model on HCP brain data using quantitative comparisons with existing approaches as well as a qualitative analysis.

## 6.2 Constrained Multiple-Output Regression Formulation

In this section, we first introduce existing works on multiple regression under Gaussian assumptions and then motivate our approach under non-Gaussian settings and domain constraints.

### 6.2.1 Multiple-output Regression Problem

Let $\mathcal{D}$ be an $n$-subject sample set in which all subjects share the same coupling $(\mathcal{G}_1, \mathcal{G}_2)$ but have different edge values. For subject $i$ in $\mathcal{D}$, let $\boldsymbol{x}^{(i)} = (x_1^{(i)}, \cdots, x_p^{(i)})$ be $p$-dimensional inputs that represent edge values in the first modality network $\mathcal{G}_1$, and

$\boldsymbol{y}^{(i)} = (y_1^{(i)}, \cdots, y_p^{(i)})$ be $p$-dimensional outputs[1] that stand for edge values in the second modality network $\mathcal{G}_2$. We assume that the inputs $\boldsymbol{x}_i$ and outputs $\boldsymbol{y}_i$ are correlated through a multivariate linear regression model:

$$\boldsymbol{y}^{(i)} = \boldsymbol{x}^{(i)}\boldsymbol{B} + \boldsymbol{\epsilon}^{(i)}, \quad \text{for } i = 1, ..., n \tag{6.1}$$

where $\boldsymbol{B}$ is the $p \times p$ regression coefficient matrix and its element $\beta_{jk}$ is the regression coefficient that measures the cross-modality impact of edge $x_j$ to edge $y_k$, and $\boldsymbol{\epsilon}^{(i)}$ is the noise vector of subject $i$. The model can be expressed in the matrix form:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{B} + \boldsymbol{E} \tag{6.2}$$

where row $i$ of $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ and $\boldsymbol{Y} \in \mathbb{R}^{n\times p}$ are the structural and functional edge vectors $\boldsymbol{x}^{(i)}$ and $\boldsymbol{y}^{(i)}$ of subject $i$.

A straightforward approach to estimating $\boldsymbol{B}$ is to solve $p$ separate regression problems, assuming noise terms are independent and uncorrelated. Recently, advanced methods have been proposed to exploit the correlation in noise terms to improve the modeling. They accomplish the goal by introducing an assumption that noise terms $\boldsymbol{\epsilon}^{(1)}$,...,$\boldsymbol{\epsilon}^{(n)}$ are all *i.i.d.* Gaussian $\mathcal{N}(\boldsymbol{0}, \boldsymbol{\Omega}^{-1})$ and then simultaneously estimating regression coefficients $\boldsymbol{B}$ and inverse covariance matrix $\boldsymbol{\Omega}$ of the noise terms. Two popular methods along this direction are MRCE [142] and CGGM [143, 144, 145]. The MRCE method considers the conditional distribution $\boldsymbol{Y}|\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{X}\boldsymbol{B}, \boldsymbol{\Omega}^{-1})$ and estimates both $\boldsymbol{B}$ and $\boldsymbol{\Omega}$ by alternately minimizing the negative conditional Gaussian likelihood, with the $\ell_1$ lasso penalty applied on the entries of $\boldsymbol{B}$ and $\boldsymbol{\Omega}$. The other method, CGGM, further assumes that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are jointly Gaussian. Under such formulation, the conditional distribution of

[1]In general, models do not require the same dimensions for inputs and outputs. We use the equality setting only for simplicity.

$\boldsymbol{Y}|\boldsymbol{X}$ is given by $\mathcal{N}(-\boldsymbol{X}\boldsymbol{\Omega}_{XY}\boldsymbol{\Omega}^{-1}, \boldsymbol{\Omega}^{-1})$, which reparameterizes the regression coefficient $\boldsymbol{B}$ as $-\boldsymbol{\Omega}_{XY}\boldsymbol{\Omega}^{-1}$. Compared with MRCE, the objective of CGGM is based on the negative conditional Gaussian likelihood as well, but is jointly convex for $\boldsymbol{\Omega}_{XY}$ and $\boldsymbol{\Omega}$, and therefore more friendly to computation.

### 6.2.2 Relaxing Gaussian Assumptions

Although MRCE and CGGM have received significant attention in solving multi-output regression problems, one drawback of these two approaches is the Gaussian assumption, especially for applications to brain data [146, 147, 148]. Recall that MRCE assumes the Gaussian noise, and CGGM further assumes joint Gaussian distribution over both inputs and outputs. We tested whether the HCP structural and functional data is Gaussian with a significance level of 0.05. The test rejects the Gaussian null hypothesis for 97.5% of structural edges and 36.3% of functional ones. Since our sample size is small, false negatives are more likely to occur [149], namely failing to reject the Gaussian hypothesis when the underlying data is non-Gaussian. Therefore, the proportion of non-Gaussian data in brain networks is expected to be even higher. Thus, relying on Gaussian assumptions is likely to affect the constructed models negatively.

To avoid a Gaussian assumption, we propose a pseudolikelihood approach for learning multi-output regression models by optimizing the following objective function:

$$\min_{\{\boldsymbol{B}_k\},\{\omega_{jk}\}} \Bigg[ -n\sum_{j=1}^{p} \log \omega_{jj} + \frac{1}{2}\sum_{j=1}^{p}\sum_{i=1}^{n} \Bigg( \omega_{jj}(\boldsymbol{y}_j^{(i)} - \boldsymbol{x}^{(i)}\boldsymbol{B}_j) + \sum_{k\neq j} \omega_{jk}(\boldsymbol{y}_k^{(i)} - \boldsymbol{x}^{(i)}\boldsymbol{B}_k) \Bigg)^2 + \lambda_1 \sum_{j<k} |\omega_{jk}| + \lambda_2 \sum_{j<k} |\beta_{jk}| \Bigg]$$

or in a neat matrix notion:

$$\begin{aligned}\min_{\boldsymbol{B},\boldsymbol{\Omega}} \quad & -n\log|\boldsymbol{\Omega}_D| + \frac{1}{2}\mathrm{tr}\left((\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{B})^T(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{B})\boldsymbol{\Omega}^2\right) \\ & + \lambda_1\|\boldsymbol{B}\|_1 + \lambda_2\|\boldsymbol{\Omega}_X\|_1 \end{aligned} \tag{6.3}$$

where $\boldsymbol{\Omega} = \{\omega_{jk}\}$ denotes the inverse covariance matrix, $\boldsymbol{B} = \{\beta_{jk}\}$ denotes the coefficient matrix, and $\boldsymbol{\Omega}_D$ and $\boldsymbol{\Omega}_X$ denote the diagonal and off-diagonal parts of $\boldsymbol{\Omega}$. The proposed objective can be considered as a reparameterization of the Gaussian likelihood with $\boldsymbol{\Omega}^2$ and an approximation to the log-determinant term. It has been proven that under mild singularity conditions, such reparameterization can guarantee estimation consistency for distributions with sub-Gaussian tails [150, 151].

Next, we develop an optimization algorithm to minimize the objective. The objective function itself is not jointly convex for both variables $\boldsymbol{B}$ and $\boldsymbol{X}$, but remains convex with respect to each of them while keeping the other fixed. Therefore, we adopt the alternating minimization idea. In the $t$-th iteration, we first fix $\boldsymbol{B}$ as the estimated $\hat{\boldsymbol{B}}^{(t-1)}$ from the previous $(t-1)$-th iteration, and calculate the empirical covariance matrix $\boldsymbol{S}$ of noise terms:

$$\boldsymbol{S}^{(t-1)} = \frac{1}{n}(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{B}^{(t-1)})^T(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{B}^{(t-1)}) \tag{6.4}$$

Next, we estimate the inverse covariance matrix: $\boldsymbol{\Omega}^{(t)}$ with the given $\boldsymbol{S}^{(t-1)}$ as a constant:

$$\boldsymbol{\Omega}^{(t)} = \arg\min_{\boldsymbol{\Omega}} -\log|\boldsymbol{\Omega}_{\boldsymbol{D}}| + \frac{1}{2}\mathrm{tr}\left(\boldsymbol{S}^{(t-1)}\boldsymbol{\Omega}^2\right) + \lambda_2\|\boldsymbol{\Omega}_X\|_1 \tag{6.5}$$

Observe that the above subproblem follows CONCORD's original form, which is more robust to heavy-tailed data [151, 152] than the conventional Gaussian likelihood approach and can be efficiently solved using proximal gradient methods with a convergence rate of $O(1/t^2)$ [153]. Lastly, we keep $\boldsymbol{\Omega}$ fixed at ${\boldsymbol{\Omega}^{(t)}}^2$ and optimize the regression coefficients $\boldsymbol{B}$:

$$\boldsymbol{B}^{(t)} = \arg\min_{\boldsymbol{B}} \ \frac{1}{2}\text{tr}\left((\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{B})^T(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{B}){\boldsymbol{\Omega}^{(t)}}^2\right) + \lambda_1\|\boldsymbol{B}\|_1 \tag{6.6}$$

Note that subproblem (6.6) is convex when ${\boldsymbol{\Omega}^{(t)}}^2$ is positive semi-definite. We present the above regression-based approach as CONCORD-MRCE in **Algorithm 1**.

```
Algorithm 1: CONCORD-MRCE
  Input: penalty parameter $\lambda_1$ and $\lambda_2$
  Initialize $t = 0$, $\hat{\boldsymbol{B}}^{(0)} = \mathbf{0}$ and $\hat{\boldsymbol{\Omega}}^{(0)} = \hat{\boldsymbol{\Omega}}(\hat{\boldsymbol{B}}^{(0)})$.
  while not converged do
      step 1: Compute $\hat{\boldsymbol{S}}^{(t-1)} = \hat{\boldsymbol{S}}(\hat{\boldsymbol{B}}^{(t-1)})$ as Eq.(6.4);
      step 2: Update $\hat{\boldsymbol{\Omega}}^{(t)} = \hat{\boldsymbol{\Omega}}(\hat{\boldsymbol{S}}^{(t-1)})$ as Eq. (6.5) by calling CONCORD($\hat{\boldsymbol{S}}^{(t-1)}$) ;
      step 3: Update $\hat{\boldsymbol{B}}^{(t)} = \hat{\boldsymbol{B}}(\hat{\boldsymbol{\Omega}}^{(t)})$ as Eq.(6.6);
  return $\boldsymbol{B}$ and $\boldsymbol{\Omega}$
```

### 6.2.3 Imposing Domain Constraints

Due to limited sample size of real-world datasets and their high dimensionality, incorporating accurate domain constraints can reduce the search space and avoid over-fitting.

Under a linear mapping assumption, the partial correlation of response variables arises only from correlations in the noise terms. Therefore, $\boldsymbol{\Omega}$ not only represents the inverse covariance of noise terms, but also equals the conditional inverse covariance of $\boldsymbol{Y}|\boldsymbol{X}$. The nonzero entries of $\boldsymbol{\Omega}$ encode direct relationships among the target modality outputs $\boldsymbol{Y}$ that cannot be explained by weighted inputs $\boldsymbol{X}\boldsymbol{B}$ of the source modality. It will be beneficial if the zero-vs-nonzero structure of $\boldsymbol{\Omega}$ is partially given by domain experts and used as hard constraints in model estimation.

More formally, let $M$ be a binary matrix that has the same dimensions as $\boldsymbol{\Omega}$. We can define a convex matrix set $\mathbb{S}_M$, containing all matrices that share the same set of zero entries with $M$. We can then improve the previous regression-based approach to estimate $\mathbf{\Omega}$ under the domain constraint that takes the form of $\mathbf{\Omega} \in \mathbb{S}_M$, written in an equivalent unconstrained convex form:

$$\begin{aligned}\hat{\Omega} = \arg\min_{\boldsymbol{B},\mathbf{\Omega}} & -\frac{n}{2}\log|\mathbf{\Omega}_D^2| + \frac{1}{2}tr((\boldsymbol{Y}-\boldsymbol{XB})^T(\boldsymbol{Y}-\boldsymbol{BX})\mathbf{\Omega}^2) \\ & + \lambda_1\|\boldsymbol{B}\|_1 + \lambda_2\|\mathbf{\Omega}_X\|_1 + \mathbb{I}\{\mathbf{\Omega} \in \mathbb{S}_M\} \qquad (6.7)\end{aligned}$$

where $\mathbb{I}\{\mathbf{\Omega} \in \mathbb{S}_M\}$ is an indicator function. This formulation can be extended to $\mathbf{\Omega} \in C$ whenever $C$ is a closed convex set of positive definite matrices.

## 6.3 Alternating Minimization Solution

In this section, we show how to adapt the previous solution when the inverse covariance $\mathbf{\Omega}$ is constrained during estimation. Notice that the ideas of **Algorithm 1** can be mostly used to solve Eq (6.7), except for the $\mathbf{\Omega}$-update step, which is now affected by the added constraints. The new $\mathbf{\Omega}$-update step needs to solve the following sub-problem:

$$\begin{aligned}\mathbf{\Omega}^{(t)} = \arg\min_{\mathbf{\Omega}} & -\log|\mathbf{\Omega}_D^2| + \operatorname{tr}\left(\boldsymbol{S}^{(t-1)}\mathbf{\Omega}^2\right) \\ & + \lambda_2\|\mathbf{\Omega}_X\|_1 + \mathbb{I}\{\mathbf{\Omega} \in \mathbb{S}_M\} \qquad (6.8)\end{aligned}$$

We follow the FISTA (F̲ast I̲terative S̲oft-T̲hresholding A̲lgorithm [154]) approach that is used in CONCORD [153]. This method utilizes an accelerated gradient algorithm using soft-thresholding as its proximal operator for the $L_1$ norm and achieves a fast $O(1/t^2)$ convergence rate. Previous work [153] has also applied FISTA for partial correlation estimation and proved its efficiency. To adapt our constrained problem into the FISTA framework, we split our objective function (6.8) into a smooth part and a non-smooth part:

$$h_1(\mathbf{\Omega}) = -\log|\mathbf{\Omega}_D^2| + tr(\boldsymbol{S}\mathbf{\Omega}^2)$$
$$h_2(\mathbf{\Omega}) = \lambda_2\|\mathbf{\Omega}_X\|_1 + \mathbb{I}\{\mathbf{\Omega} \in \mathbb{S}_M\}$$

For any symmetric matrix $\mathbf{\Omega}$, the gradient of the smooth function can be easily calculated as: $\nabla h_1(\mathbf{\Omega}) = -2\mathbf{\Omega}_D^{-1} + 2\mathbf{\Omega}\boldsymbol{S}$. With this formulation, we now adapt the FISTA iterative scheme to solve our network-constrained problem (6.8):

$$\alpha_{t+1} = (1 + \sqrt{1 + 4\alpha_t^2})/2 \tag{6.9}$$
$$\mathbf{\Theta}^{(t+1)} = \mathbf{\Omega}^{(t)} + \frac{\alpha_t - 1}{\alpha_{t+1}}(\mathbf{\Omega}^{(t)} - \mathbf{\Omega}^{(t-1)}) \tag{6.10}$$
$$\mathbf{\Omega}^{(t+1)} = \text{prox}_{\gamma h_2}[\mathbf{\Theta}^{(t+1)} - (n\tau_t/2)\nabla h_1(\mathbf{\Theta}^{(t+1)})] \tag{6.11}$$

where $\tau_t$ is the step length and $t$ denotes the iteration number. $\gamma$ is a trade-off parameter that controls the extent to which the proximal operator maps points towards the minimum of $h_2(\mathbf{\Omega})$, with larger values of $\gamma$ associated with larger movement near the minimum.

In these iterative steps, $\mathbf{\Theta}_{t+1}$ is an expected position, updated purely by momentum. Within each loop, the algorithm first takes a gradient step of the estimated future position (Eq.6.10) and then applies the proximal mapping of a closed convex function $h_2(\mathbf{\Omega})$.

In contrast to the standard FISTA approach, the composite function $h_2(\mathbf{\Omega})$ consists of a sparsity penalty and a network-constrained indicator function. More specifically, we can write down the explicit form of Eq. (6.11) according to the proximal operator definition:

$$\begin{aligned}
\hat{\boldsymbol{\Omega}} \; &= \hat{\boldsymbol{\Omega}}_X + \boldsymbol{A}_D \\
\hat{\boldsymbol{\Omega}}_X &= \operatorname{prox}_{\gamma h_2}(\boldsymbol{A}_X) \\
&= \arg\min_{\boldsymbol{\Omega}_X \in C} \frac{1}{2\gamma}\|\boldsymbol{\Omega}_X - \boldsymbol{A}_X\|_F^2 + \lambda_2 \|\boldsymbol{\Omega}_X\|_1 \qquad (6.12)
\end{aligned}$$

where $\boldsymbol{A} = \boldsymbol{\Theta}^{(t+1)} - (n\tau_t/2)\nabla h(\boldsymbol{\Theta}^{(t+1)})$. Instead of directly solving the original problem (6.12), we consider its dual problem as follows. Let matrix $\boldsymbol{H}$ be the dual variable of matrix $\boldsymbol{\Omega}$. We have:

$$\begin{aligned}
&\min_{\boldsymbol{\Omega}_X \in C} \; (\frac{1}{2\gamma}\|\boldsymbol{\Omega}_X - \boldsymbol{A}_X\|_F^2 + \lambda_2 \max_{\|\boldsymbol{H}_X\|_\infty \le 1} \operatorname{vec}(\boldsymbol{H}_X)^T \operatorname{vec}(\boldsymbol{\Omega}_X)) \\
= &\max_{\|\boldsymbol{H}_X\|_\infty \le 1} \min_{\boldsymbol{\Omega}_X \in C} \frac{1}{2\gamma}\Big(\|\boldsymbol{\Omega}_X - (\boldsymbol{A}_X - \gamma\lambda_2 \boldsymbol{H}_X)\|_F^2 \\
&\qquad - \|\boldsymbol{A}_X - \gamma\lambda_2 \boldsymbol{H}_X\|_F^2 + \|\boldsymbol{A}_X\|_2^2\Big) \qquad (6.13)
\end{aligned}$$

where $\boldsymbol{A}_D$ and $\boldsymbol{A}_X$ denote the diagonal and off-diagonal part of $\boldsymbol{A}$. Since the initial objective function above is convex in $\boldsymbol{\Omega}_X$ and concave in $\boldsymbol{H}_X$, we exchange the order of the minimum and maximum operator in which the inner minimization problem has an obvious solution through orthogonal projection theorem [155], written as

$$\boldsymbol{\Omega}_X = \mathbb{P}_C\,(\boldsymbol{A}_X - \gamma\lambda_2 \boldsymbol{H}_X) \qquad (6.14)$$

where $\mathbb{P}_C$ is defined as an projection operator: $\mathbb{P}_C(\boldsymbol{\Gamma}) = \operatorname{argmin}_{\boldsymbol{R} \in C} \|\boldsymbol{R} - \boldsymbol{\Gamma}\|_F^2$ and its orthogonal projection operator $\mathbb{P}_{C^\perp}$ is defined as $I - \mathbb{P}_C$. In the special case that $C = \mathbb{S}_M$, projection $\mathbb{P}_C(\boldsymbol{\Gamma})$ is equivalent to removing invalid nonzero entries of the input matrix $\boldsymbol{\Gamma}$.

Inserting the optimal $\boldsymbol{\Omega}_X$ back into objective (6.13), we now obtain the final dual form of the problem (6.12):

$$\hat{\boldsymbol{H}}_X = \underset{\|\boldsymbol{H}_X\|_\infty \le 1}{\arg\min} \|\boldsymbol{A}_X - \gamma\lambda_2 \boldsymbol{H}_X\|_2^2 - \|\mathbb{P}_{C^\perp}(\boldsymbol{A}_X - \gamma\lambda_2 \boldsymbol{H}_X)\|_2^2 \tag{6.15}$$

where any solution $\hat{\boldsymbol{H}}_X$ to the dual problem corresponds to a primal solution through Eq.(6.14). Since the dual objective is continuously differentiable and constraints on $l_\infty$-norm are convex, we can again efficiently solve it with additional inner FISTA iterations, which is to minimize an equivalent composite objective $\min g_1(\boldsymbol{H}_X) + g_2(\boldsymbol{H}_X)$, where $g_1(\boldsymbol{H}_X)$ is smooth and $g_2(\boldsymbol{H}_X)$ is non-smooth:

$$g_1(\boldsymbol{H}_X) = \|\boldsymbol{A}_X - \gamma\lambda_2 \boldsymbol{H}_X\|_2^2 - \|\mathbb{P}_{C^\perp}(\boldsymbol{A}_X - \gamma\lambda_2 \boldsymbol{H}_X)\|_2^2$$
$$g_2(\boldsymbol{H}_X) = I_{\{\|\boldsymbol{H}_X\|_\infty \le 1\}}(\boldsymbol{H}_X).$$

To adapt FISTA to the problem (6.15), the only thing left is to obtain the gradient of smooth function $g_1(\boldsymbol{H}_X)$, for which we need the lemma below:

**Lemma 1.** *[156] If $g$ is a closed proper convex function, and for any positive $t$, define a proximal operator $g_t(\boldsymbol{x}) := \inf_{\boldsymbol{u}} \left[g(\boldsymbol{u}) + \frac{1}{2t}\|\boldsymbol{u} - \boldsymbol{x}\|^2\right]$, then its infimum is attained at the unique point* $\mathrm{prox}_t(g)(\boldsymbol{x})$*. Further, $g_t$ is continuously differentiable on $\mathbb{E}$ with a $1/t$-Lipschitz gradient given by $\nabla g_t(\boldsymbol{x}) = (\boldsymbol{x} - \mathrm{prox}_t(g)(\boldsymbol{x}))/t$.*

According to Lemma 1, we simply plug $g(\boldsymbol{u}) = \delta(\boldsymbol{u} \in C)$, and obtain that $\mathrm{prox}_t(g)(\boldsymbol{x}) = \mathbb{P}_C(\boldsymbol{x})$. Therefore the gradient $\nabla g_1$ is calculated as:

$$\nabla g_1(\boldsymbol{H}_X) = -2\gamma\lambda_2 \cdot \mathbb{P}_C(\boldsymbol{A}_X - \gamma\lambda_2 \boldsymbol{H}_X). \tag{6.16}$$

and the proximal mapping of function $g_2(\boldsymbol{\Omega})$ becomes a projection of $\boldsymbol{H}_X$ into the $L_\infty$-

ball:

$$\left(\text{prox}_{g_2}(\boldsymbol{H})\right)_{ij} = sign(\boldsymbol{H}_X)\min\{|\boldsymbol{H}_X|, \mathbf{1}_X\} \tag{6.17}$$

This completes the modified $\boldsymbol{\Omega}$-update step in the constrained setting (referred to as Constrained-CONCORD). Its corresponding pseudocode is presented in **Algorithm 3**. The overall framework of simultaneously estimating $\boldsymbol{\Omega}$ and $\boldsymbol{B}$ is presented in **Algorithm 2**, which we name as CC-MRCE.

```
Algorithm 2: CC-MRCE
  Input: penalty parameter $\lambda_1$ and $\lambda_2$, convex constraint set $C$.
  Initialize $\hat{\boldsymbol{B}}^{(0)} = \mathbf{0}$ and $\hat{\boldsymbol{\Omega}}^{(0)} = \hat{\boldsymbol{\Omega}}(\hat{\boldsymbol{B}}^{(0)})$.
  while not converged do
      step 1: Compute $\hat{\boldsymbol{S}}^{(t-1)} = \hat{\boldsymbol{S}}(\hat{\boldsymbol{B}}^{(t-1)})$ as Eq.(6.4);
      step 2: Update $\hat{\boldsymbol{\Omega}}^{(t)} = \hat{\boldsymbol{\Omega}}(\hat{\boldsymbol{S}}^{(t-1)})$ by calling
      Constrained-CONCORD($\hat{\boldsymbol{S}}^{(t-1)}$,$E$,$\lambda_1$);
      step 3: Update $\hat{\boldsymbol{B}}^{(t)} = \hat{\boldsymbol{B}}(\hat{\boldsymbol{\Omega}}^{(t)})$ as Eq.(6.6);
  return $\boldsymbol{B}$ and $\boldsymbol{\Omega}$
```

In every step of the while loop, CC-MRCE calls sub-function Constrained-CONCORD to estimate the partial correlation matrix $\boldsymbol{\Omega}^{(t)}$ under hard constraints, i.e. the solution $\boldsymbol{\Omega}^{(t)}$ shares the same zero and nonzero pattern as matrix $E$ (shown in **Algorithm 3**). Such constraints can be replaced by any set of convex constraints on $\boldsymbol{\Omega}^{(t)}$. Note that superscript $t'$ and $t$ in **Algorithm 3** are iteration counters of inner and outer stages in the Constrained-CONCORD algorithm, respectively.

**Lemma 2.** *Let $L(g_1)$ be the Lipschitze constant of the gradient of objective function $g_1(\boldsymbol{H}_X)$, then $L(g_1) \leq 2\lambda_2^2\gamma^2$.*

Although we use line-search to pick a proper step length $\kappa_{t'}$ in the inner-loop of **Algorithm 3**, it can be replaced with a constant step length $\kappa_{t'} = 2\lambda_2^2\gamma^2$ according to the above lemma.

**Algorithm 3:** Constrained-CONCORD

```
Input: sample covariance matrix $\boldsymbol{S}$, sparsity pattern $E$, penalty parameter $\lambda_2$
Output: partial correlation matrix $\boldsymbol{\Omega}$
set $\boldsymbol{\Theta}^{(1)} = \boldsymbol{\Omega}^{(0)} \in \mathbb{S}^p_M$, $\alpha_1 = 1$, $\tau_{(0,0)} \le 1$, $c < 1$
while not converged do
    $\boldsymbol{G}^{(t)} = \nabla h_1(\boldsymbol{\Theta}^{(t)})$
    search largest $\tau_t \in \{c^j \tau_{(t,0)}\}_{j=0,1,\cdots}$
        $\boldsymbol{A}^{(t)} = \boldsymbol{\Theta}^{(t)} - (n\tau_t/2)\boldsymbol{G}^{(t)}$
        set $\tilde{\boldsymbol{\Theta}}^{(1)} = \boldsymbol{H}^{(0)} \in \mathbb{S}_{d\times d}$, $\tilde{\alpha}_1 = 1$, $\kappa_{(0,0)} \le 1$, $\tilde{c} < 1$
        while not converged do
            $\tilde{\boldsymbol{G}}_X^{(t')} = \nabla g_1(\boldsymbol{H}_X^{(t')})$
            search largest $\kappa_{t'} \in \{\tilde{c}^{j'} \kappa_{(t',0)}\}_{j'=0,1,\cdots}$
                $\boldsymbol{H}_X^{(t')} = \mathrm{prox}_{g_2}\left(\tilde{\boldsymbol{\Theta}}_X^{(t')} - \kappa_{t'}\tilde{\boldsymbol{G}}_X^{(t')}\right)$
                check backtrack line search criterion
            $\tilde{\alpha}_{t'+1} = (1 + \sqrt{1 + 4\tilde{\alpha}_{t'}^2})/2$
            $\tilde{\boldsymbol{\Theta}}^{(t'+1)} = \boldsymbol{H}_X^{(t')} + \frac{\alpha_{t'}-1}{\alpha_{t'+1}}\left(\boldsymbol{H}_X^{(t')} - \boldsymbol{H}_X^{(t'-1)}\right)$
            compute $\kappa_{(t'+1,0)}$
        $\boldsymbol{\Omega}^{(t)} = \mathbb{P}_C({\boldsymbol{A}_X}^{(t)} - \gamma\lambda_2 {\boldsymbol{H}_X}^{(t)}) + \boldsymbol{A}_D^{(t)}$
        check backtrack line search criterion
    $\alpha_{t+1} = (1 + \sqrt{1 + 4\alpha_t^2})/2$
    $\boldsymbol{\Theta}^{(t+1)} = \boldsymbol{\Omega}^{(t)} + \frac{\alpha_t - 1}{\alpha_{t+1}}(\boldsymbol{\Omega}^{(t)} - \boldsymbol{\Omega}^{(t-1)})$
    compute $\tau_{(t+1,0)}$
```

## 6.4 Experiments on Synthetic and HCP Datasets

We present two sets of experiments. First, to understand our method's strengths and limitations, we utilize simulated datasets that allow us to inspect both reconstruction performance and model selection performance. We compare the proposed method CC-MRCE with other baselines when the underlying data distribution does not follow the Gaussian assumption. We show that both the non-Gaussian assumption and the network constraints contribute to the improvement of performance. Second, we conduct experiments on the Human Connectome Project (HCP) data [141]. Our model offers a quantitative advantage over baseline methods for predicting functional networks from structural ones; at the same time, our results agree with existing neuroscience literature.

### 6.4.1 Application to Simulated Data

**Data Generation.** Using a similar approach to existing works [150, 151, 142], we generate our simulated dataset by first synthesizing two key model parameter matrices $\boldsymbol{\Omega}_0$ and $\boldsymbol{B}_0$, and then construct input, output, and noise terms (*i.e.* $\boldsymbol{x}^{(i)}$, $\boldsymbol{y}^{(i)}$ and $\boldsymbol{\epsilon}^{(i)}$'s).

Note that every positive-definite matrix has a Cholesky decomposition that takes the form of $\boldsymbol{L}\boldsymbol{L}^T$, where $\boldsymbol{L}$ is a lower triangular matrix $\boldsymbol{L}$, and if $\boldsymbol{L}$ is sparse enough then $\boldsymbol{L}\boldsymbol{L}^T$ is sparse as well. Therefore, we first sample a sparse lower triangular $\boldsymbol{L}$ with real and positive diagonal entries, and then generate our inverse covariance matrix with $\boldsymbol{\Omega}_0 = \boldsymbol{L}\boldsymbol{L}^T$. The generated $p \times p$ positive definite matrix $\boldsymbol{\Omega}_0$ has 10% nonzeros entries and a condition number of 4.3. To demonstrate the robustness of proposed method CC-MRCE on non-Gaussian data, we sample the noise terms $\{\boldsymbol{\epsilon}^{(i)}\}_{i=1}^n$ according to a multivariate $t$-distribution with zero mean and covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{\Omega}_0^{-1}$.

Next, we generate a sparse coefficient matrix $\boldsymbol{B}_0$ using the matrix element-wise product trick, $\boldsymbol{B}_0 = \boldsymbol{W} \circ \boldsymbol{K} \circ \boldsymbol{Q}$. In this construction approach, $\boldsymbol{W}$ has entries with independent draws from standard normal distribution $\mathcal{N}(0,1)$. Each entry in $\boldsymbol{K}$ is drawn independently from a Bernoulli distribution that takes value 1 with probability $s_1$. $\boldsymbol{Q}$ has rows that are either all one or all zero, which are determined by independent Bernoulli draws with success probability $s_2$. Generating the sparse $\boldsymbol{B}_0$ in this manner, we not only control its sparsity level, but also forcibly make $(1 - s_2)p$ predictors to be irrelevant for $p$ responses, and guarantee that each relevant predictor is associated with $s_1 p$ response variables.

In the following experiments, the probabilities $s_1$ and $s_2$ are chosen to be 0.15 and 0.8, the sample size $n$ is fixed at 50, and the input and output dimensions $p$ and $q$ are both set to 20. The input $\boldsymbol{x}^{(i)}$'s are sampled from a multivariate normal distribution $\boldsymbol{\mathcal{N}}(0, \boldsymbol{\Sigma_X})$ where $(\boldsymbol{\Sigma_X})_{jk} = 0.5^{|j-k|}$, following previous works [157, 158]. The output $\boldsymbol{y}^{(i)}$'s are calculated as the linear model assumption in Eq.(6.1). We replicate the above process for independently generating a validation dataset of the same sample size. All penalty parameters are selected simultaneously and tuned according to the validation error.

**Methods.** In this heavy-tailed setting, we compare the performance of CC-MRCE to CGGM and MRCE, which are developed under Gaussian settings. The CGGM implementation that we used in this experiment is provided by [159] and has been optimized for large-scale problems and limited memory. The MRCE implementation is provided by [142]. Both CGGM and MRCE do not adapt to the network constraints we impose here.

In order to inspect the effectiveness of network constraints, we apply multiple variations of constraint sets to the proposed CC-MRCE method, designated as CC-MRCE (unconstrained), CC-MRCE (SNR: 2.0), CC-MRCE (SNR: 1.0) and CC-MRCE (perfect). *Network constraints* in each variation are defined as follows. For CC-MRCE (perfect), we choose $\mathbb{S}_E = \{\boldsymbol{\Omega} : \boldsymbol{\Omega}(j,k) = 0 \text{ if } \boldsymbol{\Omega}_0(j,k) = 0\}$, which forces selected nonzeros in solution $\boldsymbol{\Omega}$ to completely fall into ground-truth nonzeros. For CC-MRCE (SNR: 2.0) and CC-MRCE (SNR: 1.0), we loosen the feasible set by adding $50\%\|\boldsymbol{\Omega}_0\|_0$ and $100\%\|\boldsymbol{\Omega}_0\|_0$ spurious nonzero positions, which are randomly sampled from positions of zero entries in $\boldsymbol{\Omega}_0$. For CC-MRCE (unconstrained), we remove all constraints so that the method only relies on regularized multi-regression.

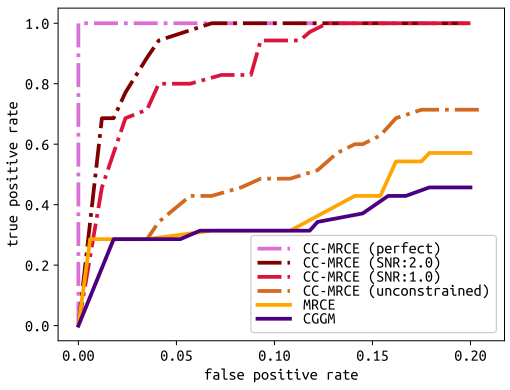
**Performance Evaluation.** We evaluate the reconstruction performance of models using the conventional MSE error (in percentage). Correlation coefficients between predicted and ground-truth outputs are also provided along with their corresponding p-values. In addition, we use the relative area under the curve (AUC) of receiver operating characteristic (ROC) curves [160, 161], with regards to $\boldsymbol{\Omega}$ and $\boldsymbol{B}$, as key measures to compare model selection performance of all these methods. The AUC of a perfect ROC curve, which would be 1, indicates an ideal recovery of gound truth zero-vs-nonzero structure in $\mathbf{\Omega}$ (or $\boldsymbol{B}$). However, models with large false-positive rates (FPR) are barely meaningful in real scenarios. To focus on the initial portion of ROC curves, we control the FPRs simultaneously to be smaller than 0.2 for both $\mathbf{\Omega}$ and $\boldsymbol{B}$ estimation. Thus, the maximum AUC value that a model can reach is just 0.2. For ease of comparison, we provide relative AUC values, divided by 0.2 to normalize to 1. For each method, we run the algorithm with at least 25 appropriate parameter pairs $(\lambda_1, \lambda_2)$ to get its ROC curve. Recall that all methods in this section are required to estimate 800 parameters given $n = 50$ samples.

**Test Results.** Table 6.1 displays the test results of six different settings in all measures mentioned above. As can be seen, all four CC-MRCE variations (with different network constraints) obtain significantly smaller reconstruction MSE percentages, higher correlation coefficients, and smaller $p$-values. Note that CGGM behaves the worst in all measures since it is deeply rooted in the Gaussian setting and is consequently misled by these assumptions. We also see that the performance of CC-MRCE gradually improves when the applied network constraints are more informative.

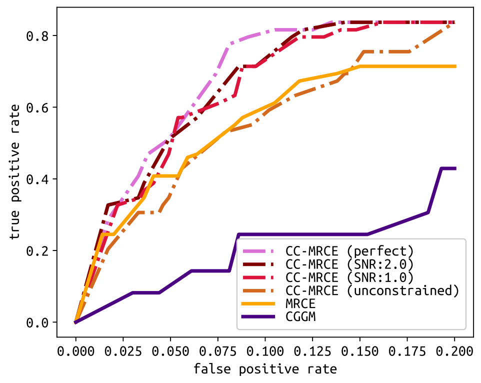
| Model | MSE (100%) | Pearson's $r$-score | min p-value | max p-value | AUC w.r.t. $\mathbf{\Omega}$ | AUC w.r.t. $\boldsymbol{B}$ |
|---|---|---|---|---|---|---|
| CC-MRCE (unconstrained) | 50.88 | 0.71 | 5.14E-09 | 0.10 | 0.52 | 0.53 |
| CC-MRCE (SNR:1.0) | 43.24 | 0.76 | 2.26E-11 | 0.08 | 0.85 | 0.64 |
| CC-MRCE (SNR:2.0) | 43.18 | 0.76 | 2.10E-11 | 0.07 | 0.92 | 0.66 |
| CC-MRCE (perfect) | 42.98 | 0.76 | 2.69E-11 | 0.06 | 1.00 | 0.67 |
| MRCE | 60.22 | 0.65 | 8.68E-09 | 0.17 | 0.37 | 0.54 |
| CGGM | 76.21 | 0.55 | 2.14E-07 | 0.96 | 0.33 | 0.19 |

Table 6.1: Reconstruction performance and model selection performance of models on simulated dataset. CC-MRCE variations uniformly outperform MRCE and CGGM. CC-MRCE variations with more strict constraints perform better.

We also plot ROC curves for $\boldsymbol{\Omega}$ and $\boldsymbol{B}$ estimation in Figure 6.1a and 6.1b, respectively. It is clear that CC-MRCE performs better than MRCE and CGGM, across different choices of network constraints. For $\boldsymbol{\Omega}$ estimation, as expected, CC-MRCE with more strict constraints has steeper curves, suggesting that it recovers mostly correct partial relationships between variables with very few spurious connections, and therefore achieves higher AUC scores. CC-MRCE (perfect) behaves perfectly for $\boldsymbol{\Omega}$-ROC, by its definition. For the estimation of matrix $\boldsymbol{B}$, a similar phenomena demonstrates that network constraints improve the learning of regression coefficients and lead to a better reconstruction performance. Without imposing network constraints, unconstrained formulation of CC-MRCE is likely to generate a biased estimate of $\hat{\boldsymbol{\Omega}}$ on small datasets and can not recover ground truth features in $\boldsymbol{B}$.



(a) ROC curve for $\boldsymbol{\Omega}$ estimation



(b) ROC curve for $\boldsymbol{B}$ estimation

Figure 6.1: Benefit from domain constraints, CC-MRCE obtains better ROC curves when uncovering nonzero entries of $\boldsymbol{B}$ and $\boldsymbol{\Omega}$.

### 6.4.2 Application to Human Connectome Data

**Problem Formulation.** Many works in literature report on coupling between brain structural connectivities (SCs) and functional connectivities (FCs) for both resting state

[162, 32, 163, 164] and task-evoked states [32, 165, 166, 167]. One such recent work [35] maps SC to resting-state FC by aligning both the eigenvalues and eigenvectors of a subject's SC and FC matrices, and evaluates the mapping by reconstructing resting FCs from SCs.

Inspired by the domain observation [138] that two functional connections sharing the same node are more likely to form a meaningful pathway or functional activation pattern, we constrain the partial correlations between non neighboring edges to be zero, i.e. $\boldsymbol{\Omega}_{jk} = 0$ if $e_j$ and $e_k$ are not incident edges in the fMRI-constructed network.

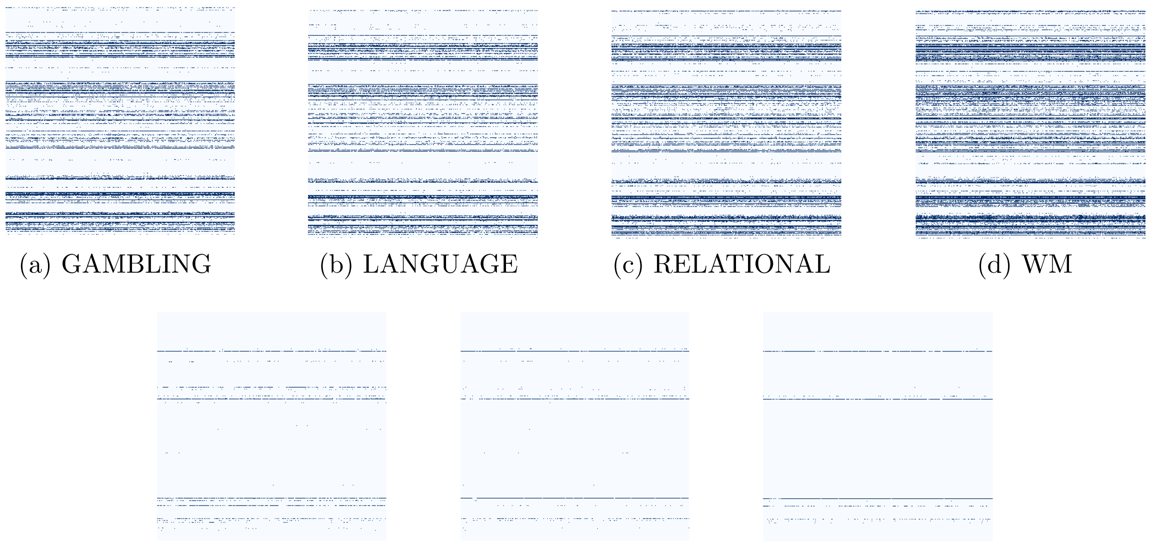
We conducted experiments by reconstructing task FCs from SCs of 51 subjects with their fMRI data from HCP dataset under seven task states. We used the parcellation scheme in [162] with a spatial scale of 33, resulting in 83 brain regions and 3403 possible edges. Columns in predictor matrix, Equation (6.2), $\boldsymbol{X} \in \mathbb{R}^{51\times3403}$ represent SCs, whose entries are numbers of white matter streamlines intersecting pairs of brain regions, and columns in response matrix $\boldsymbol{Y} \in \mathbb{R}^{51\times3403}$ represent FCs, whose entries are functional correlations between cortical activities of brain regions. $\boldsymbol{B}$ denotes the mapping (or coupling) between SCs and FCs, and $\boldsymbol{E}$ denotes the part of FCs that cannot be explained by SCs. We also normalized SC values to (0, 1], a range comparable to FC. We ran a 10-fold cross-validation of our model for each task, splitting data in a 9-1 train-validation ratio (46-5 split for the 51 subjects). We selected the optimal hyperparameters $\lambda_1$ and $\lambda_2$ by a $5 \times 5$ grid search in the log-scale between $10^{-1.6}$ to $10^{-0.4}$, keeping the models with smallest Mean Squared Error (MSE) percentage on the validation sets, averaged across 10 folds. In our case, both $\lambda_1$ and $\lambda_2$ have optimal values around 0.1 across tasks. Aside from MSE, we also tested the Pearson correlation coefficient between predicted FCs and ground truth FCs, (referred to as Pearson's $r$-score, listed in Table 6.2) and minimum, maximum $p$-values. The reconstruction MSE percentage is below 1% for the training data and around 8% for the validation data. Strong and significant positive correlations are shown for both training ($r$-score around 0.6 to 0.8) and validation ($r$-score around 0.5) data. These results indicate our model's effectiveness in FC reconstruction by exploiting cross-modal coupling between SC-FC and domain prior knowledge on FC-FC relationships.

**Performance Evaluation.** Regarding the ability of finding accurate mappings between SCs and FCs, we compared our model with the optimization approach CGGM, a deep learning approach Variational AutoEncoder (VAE) [168], and the Spectral Mapping method [35]. For CGGM, the time required to run a 10-fold cross-validation with one set of hyperparameters on a single task ranges from one day to a week. So we ran three sets of hyperparameters on EMOTION and LANGUAGE, selected the best parameter pair, and fixed it for the rest of the five tasks. In particular, we chose penalty terms on $\boldsymbol{B}$ and $\boldsymbol{E}$ to be both 0.01. For VAE, both encoder and decoder consist of two fully connected layers, with latent variable dimensions being two. We use MSE as the training loss. In our experiments, increasing the number of layers or latent dimensions of VAE did not improve the final performance. For the Spectral Mapping method, we follow the setup of the original paper, setting maximum path length $k$ to seven. The 10-fold cross validation results of CGGM, VAE and Spectral Mapping method are reported in Table 6.2.

The assumption of Gaussian noise weakens CGGM's performance on the HCP dataset: it has unstable and large average MSE percentages across tasks, and the correlations between predictions and ground truth are very small and even negative, which are also not statistically significant with regards to $p$-values. On the other hand, VAE models have stable MSE percentage (around 80%) and average Pearson's $r$-scores with small standard deviations across all tasks. The correlations of VAE models are weak (all around 0.08), yet statistically significant with $p$-values constantly smaller than 0.0025 for all tasks. This shows VAE learns a slightly meaningful mapping, but with such a small sample size, deep learning models are unlikely to perform well. Lastly, the Spectral Mapping

| Tasks | | Reconstruction MSE percentage (100%) | Pearson's $r$-score |
|---|---|---|---|
| EMOTION | CGGM | 89.78 ± 21.95 | -0.0309 ± 0.0285 |
| | VAE | 81.57 ± 2.38 | 0.0777 ± 0.0070 |
| | Spectral Mapping | 61.54 ± 33.84 | 0.4228 ± 0.0349 |
| | Random $\boldsymbol{B}$ | 17.50 ± 1.84 | -0.0014 ± 0.0086 |
| | **CC-MRCE** | **8.84** ± 0.84 | **0.4575** ± 0.0402 |
| LANGUAGE | CGGM | 41.38 ± 7.10 | 0.0270 ± 0.0448 |
| | VAE | 72.68 ± 6.33 | 0.0815 ± 0.0081 |
| | Spectral Mapping | 54.23 ± 30.18 | 0.4764 ± 0.0383 |
| | Random $\boldsymbol{B}$ | 35.02 ± 58.49 | 0.0020 ± 0.0052 |
| | **CC-MRCE** | **7.95** ± 0.87 | **0.4988** ± 0.0205 |
| MOTOR | CGGM | 117.85 ± 27.32 | -0.0016 ± 0.0456 |
| | VAE | 77.30 ± 4.24 | 0.0782 ± 0.0110 |
| | Random $\boldsymbol{B}$ | 57.26 ± 29.63 | 0.4156 ± 0.0548 |
| | Random $\boldsymbol{B}$ | 21.02 ± 7.75 | 0.0023 ± 0.0090 |
| | **CC-MRCE** | **7.80** ± 0.66 | **0.4807** ± 0.0480 |
| GAMBLING | CGGM | 108.99 ± 32.83 | -0.0211 ± 0.0795 |
| | VAE | 79.14 ± 3.65 | 0.0804 ± 0.0071 |
| | Spectral Mapping | 54.73 ± 30.70 | 0.4781 ± 0.0380 |
| | Random $\boldsymbol{B}$ | 22.63 ± 13.15 | 0.0033 ± 0.0072 |
| | **CC-MRCE** | **7.86** ± 1.72 | **0.5014** ± 0.0301 |
| SOCIAL | CGGM | 112.82 ± 36.07 | -0.0064 ± 0.076 |
| | VAE | 79.42 ± 3.72 | 0.0772 ± 0.0088 |
| | Spectral Mapping | 47.89 ± 28.09 | 0.4912 ± 0.0353 |
| | Random $\boldsymbol{B}$ | 18.68 ± 3.74 | 0.0025 ± 0.0071 |
| | **CC-MRCE** | **6.81** ± 0.95 | **0.5578** ± 0.0404 |
| RELATIONAL | CGGM | 147.61 ± 49.24 | -0.0265 ± 0.0754 |
| | VAE | 81.11 ± 2.98 | 0.0706 ± 0.0081 |
| | Spectral Mapping | 54.86 ± 30.68 | 0.4758 ± 0.0412 |
| | Random $\boldsymbol{B}$ | 31.77 ± 18.30 | -0.0019 ± 0.012 |
| | **CC-MRCE** | **8.43** ± 1.09 | **0.4858** ± 0.0460 |
| WM (Working Memory) | CGGM | 81.46 ± 29.72 | -0.0447 ± 0.0566 |
| | VAE | 77.99 ± 2.89 | 0.0799 ± 0.0109 |
| | Spectral Mapping | 56.25 ± 32.85 | 0.4767 ± 0.0579 |
| | Random $\boldsymbol{B}$ | 103.96 ± 111.99 | -0.0041 ± 0.0089 |
| | **CC-MRCE** | **7.69** ± 1.76 | **0.4968** ± 0.0543 |

Table 6.2: Functional connectivity reconstruction performance of seven tasks with different models.



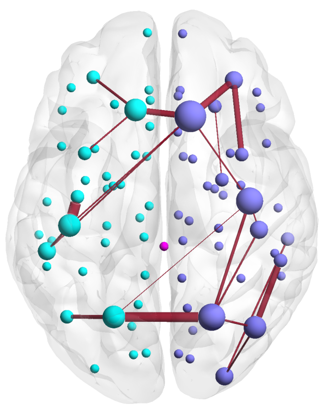
(a) GAMBLING (b) LANGUAGE (c) RELATIONAL (d) WM

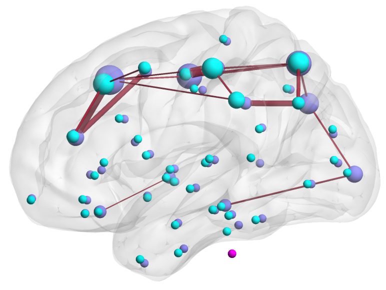
(e) EMOTION (f) MOTOR (g) SOCIAL

Figure 6.2: Visualizations of the $\boldsymbol{B}$ matrix for seven tasks. For each task, each fold in the 10-fold cross validation may lead to different models. Here, we only show those entries that are nonzero more than five times.

method is designed for maximizing the correlation between fMRI prediction and ground truth for brain data, so it performs well as for correlation, however the prediction values are off, resulting in high MSE percentages. In all, our regression-based model performs better in both correlation and value reconstruction, showing its superiority in prediction on non-Gaussian data with small sample sizes.

**Result Interpretation.** Apart from better reconstruction performance, our model also has the advantage of result interpretability. We can explore the SC-FC mapping through the resulting coefficient matrix $\boldsymbol{B}$s. Since our problem definition is FC = SC $\cdot$ $\boldsymbol{B} + \boldsymbol{E}$, the $i^{th}$ row in $\boldsymbol{B}$ corresponds to $i^{th}$ edge pair in the SC vector. In the following, we say an edge $i$ exists if row $i$ of $\boldsymbol{B}$ has nonzero entries. As the experiment is run under the 10-fold cross validation setting and each training partition may generate a different mapping $\boldsymbol{B}$, we consider an entry in the common $\boldsymbol{B}$ to be nonzero if it is nonzero more than five times in these 10 trials. The results are shown in Figure 6.2.


(a) Axial (from top)


(b) Sagittal (from left)

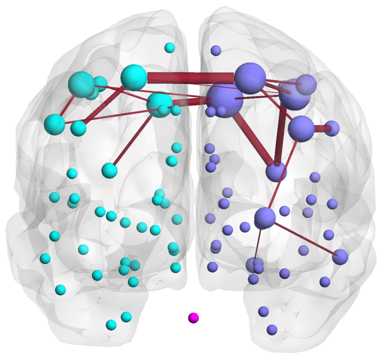
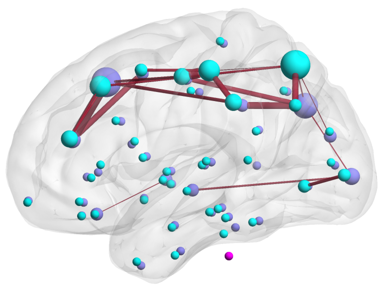
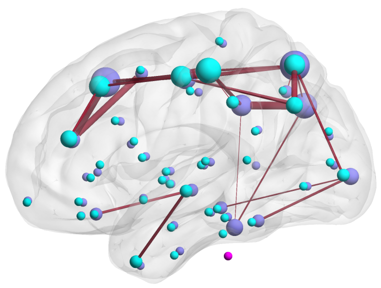
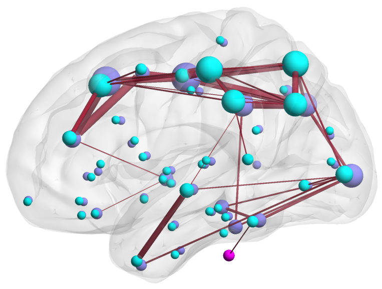

(c) Coronal (from back)

Figure 6.3: Visualization of the edges contributing to all seven tasks. Blue nodes are left hemisphere brain regions, and purple nodes are right ones. Node size denotes the degree and edge width denotes its importance, as in the mapping $\boldsymbol{B}$.

From the figure, we can see for every task, several rows in $\boldsymbol{B}$ have many more nonzero entries than the others. This indicates the existence of several significant structural edges being responsible for most of the functional activities. To test if this assumption is valid, we compared $\boldsymbol{B}$s from our model to randomly generated $\boldsymbol{B}$s *with the same levels of sparsity*. The resulting MSE percentages, although having large variances, often have smaller means than that of CGGM and VAE, implying the importance of sparsity level of the coupling. However, the resulting $r$-scores of predicted FCs using a random $\boldsymbol{B}$ is the lowest among all methods. Together with very large $p$-values, the results predicted by random coupling show no correlation between predictions and ground truth. This indicates that our models learned meaningful mapping information from SCs to FCs, and that *structured sparsity* of $\boldsymbol{B}$ is important for getting predictions besides the level of sparsity alone.


(a) LANGUAGE


(b) GAMBLING


(c) RELATIONAL

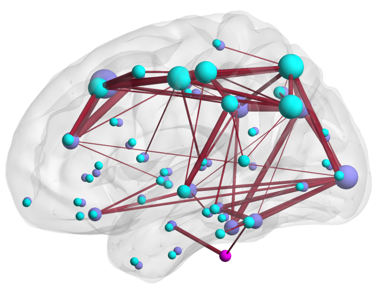

(d) WM

Figure 6.4: Task-specific visualizations for high-contributing structural edges. Assuming that the maximum number of nonzero entries of a row in $\boldsymbol{B}$ is $m$, we only show the edges correspond to rows containing more than $m/2$ nonzero entries.

We now analyze the $\boldsymbol{B}$ matrices for different cognitive tasks. During fMRI data acquisitions of all seven tasks, participants are presented with visual cues, either as images or videos, and they need to use motions such as pressing buttons to complete the tasks [169]. Interestingly, apart from the LANGUAGE task, the mappings learned by our model predicts the strongest contribution of left precentral and left postcentral connection, which is on the motor cortex responsible for right-side body movement. From this, we assume most participants use their right hands to conduct the required finger movements for these tasks. All mappings also contain edges in the occipital lobe, complying with the visual nature of these tasks. The visualization of common edges that

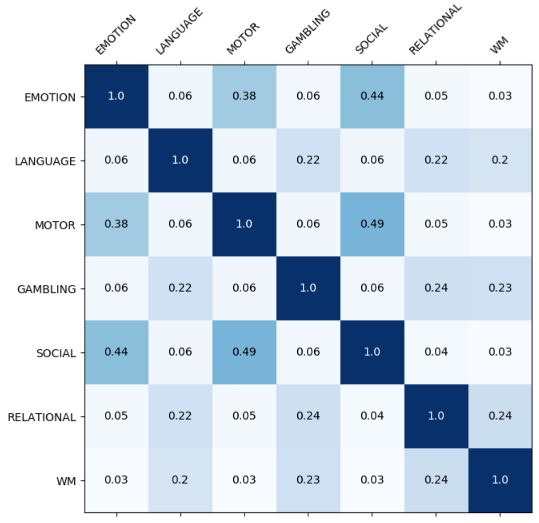
| EMOTION | LANGUAGE | MOTOR | GAMBLING |
|---|---|---|---|
| 5.34 | 25.58 | 3.60 | 35.36 |
| SOCIAL | RELATIONAL | WM | |
| 4.12 | 37.77 | 32.53 | |

Table 6.3: Overlap ratios (%) of predicted B (SCs-FCs mapping) across 10 folds for seven tasks.

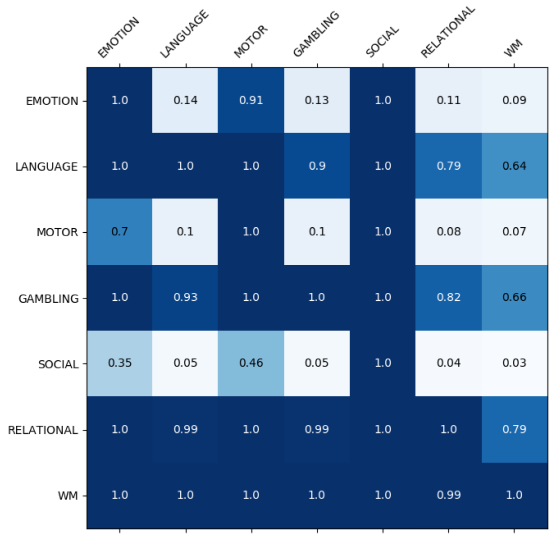
exist in all seven tasks is shown in Figure 6.3. This “backbone” roughly resembles the Default Mode Network [30, 165].

We then examined which structural edges contribute significantly to the functional activity under different tasks. For this, we plot the “high-contributing” edges in LANGUAGE, GAMBLING, RELATIONAL and WM tasks in Figure 6.4. An edge is considered as high-contributing if the number of nonzero entries of its corresponding row in $\boldsymbol{B}$ is more than half of the maximum number of nonzero entries of any row in $\boldsymbol{B}$. From Figure 6.4, we notice although a common backbone exists, structural connections in different brain regions are responsible for specific tasks (e.g. SCs in and around hippocampus area appear to be highly contributing to the WM FCs but not so for the LANGUAGE ones, which is consistent with the literature [170]). We also plot both entry-wise and edge-wise overlap ratio for the mapping of seven tasks in Figure 6.5.

Another interesting phenomenon in Figure 6.2 is that the number of nonzero entries of $\boldsymbol{B}$ for LANGUAGE, GAMBLING, RELATIONAL and WM are much larger than the other three tasks: EMOTION, MOTOR and SOCIAL, although the final model for each task have a similar level of prediction performance and similar hyperparameters. This is largely caused by the nature of non-overlapping $\boldsymbol{B}$s for EMOTION, MOTOR and SOCIAL tasks: their $\boldsymbol{B}$ overlap ratios are significantly smaller than the other four tasks as shown in Table 6.3. Here we define the overlap ratio as the number of nonzero entries in the common $\boldsymbol{B}$ (entry $ij$ being nonzero if it's nonzero more than five times) over the number of nonzero entries in $\boldsymbol{B}_\cup = \boldsymbol{B}_1 \cup \cdots \cup \boldsymbol{B}_{10}$ with $\boldsymbol{B}_k$ being the predicted



(a) Entry-wise overlap



(b) Edge-wise overlap

Figure 6.5: Entry-wise and edge-wise overlap ratio for the mapping of seven tasks. 5a considers entry-wise overlap of predicted $\boldsymbol{B}$ of different tasks. The value on position (task $i$, task $j$) is the entry-wise IoU (Intersection over Union) of task $i$'s $\boldsymbol{B}$ and task $j$'s $\boldsymbol{B}$, i.e. number of nonzero entries in $\boldsymbol{B}_i \cap \boldsymbol{B}_j$ over number of nonzero entries in $\boldsymbol{B}_i \cup \boldsymbol{B}_j$. 5b considers the the SC edges responsible for different tasks. An edge is considered to exist when its corresponding row in $\boldsymbol{B}$ has nonzero entries. The value in position (task $i$, task $j$) is the number of common SC edges of task $i$ and task $j$ over the number of SC edges of task $j$.

$\boldsymbol{B}$ using the $k^{th}$ split in 10-fold cross validation. This is also why we omit these three tasks for Figure 6.4, as the predicted $\boldsymbol{B}$s are not stable across the population and only a few common connections show significant contributions. We assume this results from group heterogeneity when carrying out these tasks. Further studies with heterogeneous models for the population will be useful to verify this assumption.

## 6.5 Conclusions

In this work, we proposed a regularized regression-based approach (CC-MRCE) for jointly learning linear models and partial correlations among variables under domain constraints. Motivated by the neuroscience application of predicting functional brain activities from structural connections, the CC-MRCE method discards the Gaussian assumption and incorporates domain constraints into model estimation. We further developed a fast algorithm based on nested FISTA to solve the optimization problem. With synthetic data analysis, we demonstrated that both domain constraints and assumption of non-Gaussian data contribute to the performance improvement of CC-MRCE. Our experimental results on Human Connectome Project data show that CC-MRCE outperforms existing methods on prediction tasks and uncovers couplings that agree with existing neuroscience literature.

# Chapter 7

# Conclusions

In this thesis, we have investigated how to leverage network structures, where a set of inter-dependent entities is encoded as nodes and interactions among entities are encoded as weighted edges. Multiple problems were discussed and their solutions were validated on real-world datasets arising from various applications, including cancer genomic study, neurological disease diagnosis, image feature extraction, house pricing and water quality estimation. We have implemented several computational approaches under specific scenarios of realistic applications, dealing with the necessity of incorporating prior knowledge, adjusting model assumptions and improving computational efficiency. The breadth and depth of the problems and techniques applied in this work has led as to the following observations:

- A subtle problem associated with neuroimaging datasets is high sensitivity to artifactual associations due to confounding effects. Shared confounding factors that feed into two variables can induce false associations and bias the estimate of true correlations. Confound effects are problematic due to their huge variety of potential imaging artifacts, including many factors that can affect both the imaging and non-imaging variables of interest: head motion, head size, changes in breathing rate and depth, and scanner hardware and software changes [171]. It is of significance to develop and apply sophisticated preprocessing methods for removing the confound effect from imaging data [172, 173].

- It is now a few datasets have started to embrace "big" imaging — with subject numbers in tens of thousands, or significant increases in the quantity of imaging and non-imaging data collected per subject. For example, the Human Connectome Project (HCP) dataset has completed imaging of over a thousand young adults, and utilizing vast improvements in the spatial and temporal resolutions with multiple modalities. However, most neuroimaging studies continue to have modest sample sizes (n $\approx$ 50 - 100) and modest amounts of data collected per subject, especially for research projects that aim at task-specific problems. Therefore for many datasets in this thesis, we prefer conventional learning methods rather than deep learning approaches that require large amount of training data.

- The small sample size and high dimensionality of real-world datasets motivate approaches for identifying and using prior or domain knowledge. In many applications, users have additional knowledge on the composition of entities that should high probability of belong to the same community or category. We would like such preferences to guide the selection of models or features. And network or graph structure is a natural format of encoding abundant knowledge. Therefore, in this thesis, we proposed several network-regularized methods for the incorporation of such domain knowledge into tasks such as graph classification. Similar approaches exist other problems as well. For example in LDA literature, aware that many types of knowledge can also be expressed with two primitives on word pairs, [174] propose the Dirichlet Forest prior to encode the domain knowledge associated with the set of Must-Links and Cannot-Links, replacing the Dirichlet prior over the topic-word multinomial distribution. Experiments show that these domain-knowledge-aware methods outperform original baselines in most cases.

- Computational efficiency has to be taken into account during method development in this thesis. As the nature of network or graph, the problem search space is usually exponentially growing with the number of nodes. Although sampling methods can mitigate the computational burden to some extent, we adopt more efficient approaches such as spectral methods and convex optimization formulations. It is also helpful if the problem can be decomposed into smaller one and be parallelly solved.

# Appendix A

# Extended DANR Formulation under Spatio-Temporal Setting

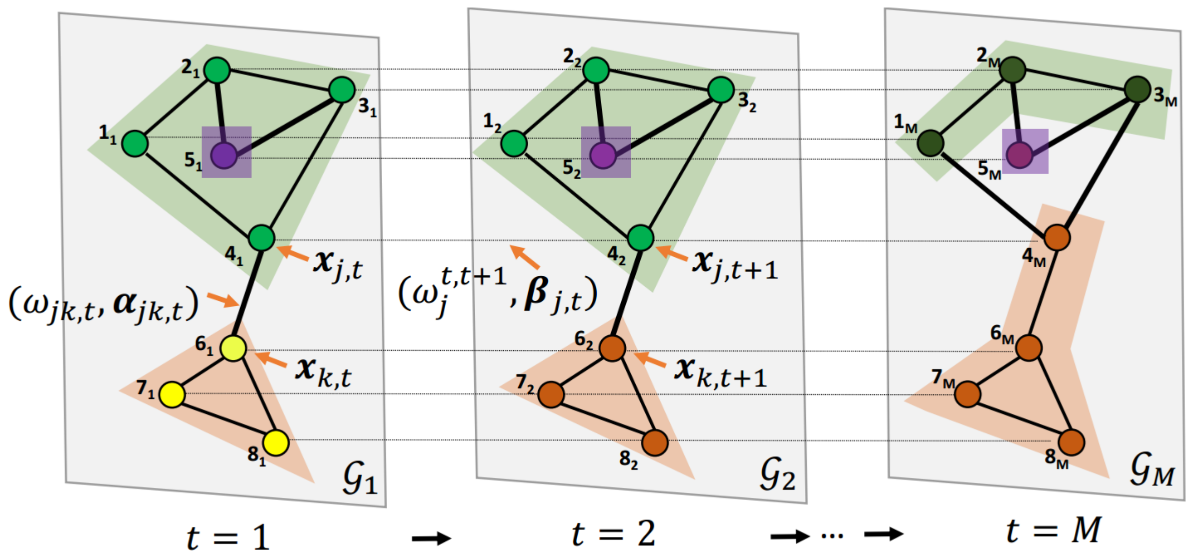


Figure A.1: Overview of our problem setting in temporal networks.

## A.1 Batch Formulation

Similar to the spatial only case in DANR method, we define $\boldsymbol{x}$=$\{\boldsymbol{x}_{j,t}\}_{t=1...M}^{j\in V}$ for model parameter vectors on temporal undirected network $\mathcal{G}$, and define consensus variables

$$\boldsymbol{z} = [\boldsymbol{u}, \boldsymbol{v}] = [\{\boldsymbol{u}_{jk,t,0}, \boldsymbol{u}_{kj,t,0}\}_{t=1...M}^{(j,k)\in\mathcal{E}_t}, \{\boldsymbol{v}_{j,t,1}\}_{t=1...M-1}^{j\in\mathcal{V}_t}, \{\boldsymbol{v}_{j,t,2}\}_{t=2...M}^{j\in\mathcal{V}_t}]$$

as copies of $\boldsymbol{x}$ in the corresponding spatial penalty term $\mathcal{R}_S(\boldsymbol{x}, \boldsymbol{\alpha})$ and temporal penalty term $\mathcal{R}_T(\boldsymbol{x}, \boldsymbol{\beta})$. Then we rewrite the problem as follows:

$$\begin{aligned}
\min_{\boldsymbol{x},\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}} \quad & \sum_{t=1}^{M}\sum_{j\in\mathcal{V}} f_{j,t}(\boldsymbol{x}_{j,t}) \\
& + \lambda_1\mu_1 \sum_{t=1}^{M}\sum_{(j,k)\in\mathcal{E}} \omega_{jk}^{t} \left\|\boldsymbol{u}_{jk,t} + \boldsymbol{\alpha}_{jk,t} - \boldsymbol{u}_{kj,t}\right\|_2 \\
& + \lambda_2\mu_2 \sum_{j\in\mathcal{V}}\sum_{t=1}^{M-1} \omega_{j}^{t,t+1} \left\|\boldsymbol{v}_{j,t,1} + \boldsymbol{\beta}_{j,t} - \boldsymbol{v}_{j,t+1,2}\right\|_2 \\
& + \lambda_1(1-\mu_1)\|\boldsymbol{\alpha}\|_{1,p} + \lambda_2(1-\mu_2)\|\boldsymbol{\beta}\|_{1,p} \qquad \text{(A.1)} \\
\text{s.t.} \quad & [\boldsymbol{u}_{jk,t}, \boldsymbol{u}_{kj,t}] \quad = [\boldsymbol{x}_{j,t}, \boldsymbol{x}_{k,t}], \quad t = 1, \dots, M \\
& [\boldsymbol{v}_{j,t,1}, \boldsymbol{v}_{j,t+1,2}] = [\boldsymbol{x}_{j,t}, \boldsymbol{x}_{j,t+1}], \quad t = 1, \dots, M-1
\end{aligned}$$

The iterative scheme of ADMM under the above setting is modified as follows, with $l$ denoting the iteration index:

$$\left.\begin{aligned}
&\boldsymbol{x}^{(l+1)} = \underset{\boldsymbol{x}}{\operatorname{argmin}}\ \mathcal{L}_{\rho_1,\rho_2}(\boldsymbol{x}, (\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta})^{(l)}) \\
&(\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{\beta})^{(l+1)} = \underset{\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}}{\operatorname{argmin}}\ \mathcal{L}_{\rho_1,\rho_2}(\boldsymbol{x}^{(l+1)}, \boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta}^{(l)}) \\
&\boldsymbol{\delta}^{(l+1)} = \boldsymbol{\delta}^{(l)} + (\tilde{\boldsymbol{x}}^{(l+1)} - \boldsymbol{z}^{(l+1)})
\end{aligned}\right\} \qquad \text{(A.2)}$$

where $\tilde{\boldsymbol{x}}$=$[\{\boldsymbol{x}_{j,t},\boldsymbol{x}_{k,t}\}_{t=1\ldots M}^{(j,k)\in\mathcal{E}_t},\{\boldsymbol{x}_{j,t}\}_{t=1\ldots M-1}^{j\in\mathcal{V}_t},\ \{\boldsymbol{x}_{j,t}\}_{t=2\ldots M}^{j\in\mathcal{V}_t}]$ is composed of replicated elements in $\boldsymbol{x}$.

**$\boldsymbol{x}$-Update**. Now we need to decompose the problem of minimizing $\boldsymbol{x}$ into separately minimizing $\boldsymbol{x}_{jt}$ for each node $j_t$ :

$$\boldsymbol{x}_{j,t}^{(l+1)} = \underset{\boldsymbol{x}_{j,t}}{\mathrm{argmin}}\ \Big(f_{j,t}(\boldsymbol{x}_{j,t}) + \frac{\rho_1}{2}\sum_{k\in N_{j,t}}\|\boldsymbol{x}_{j,t}-\boldsymbol{u}_{jk,t}^{(l)}+\boldsymbol{\delta}_{jk,t}^{u(l)}\|_2^2$$
$$+\frac{\rho_2}{2}\|\boldsymbol{x}_{j,t}-\boldsymbol{v}_{j,t,1}^{(l)}+\boldsymbol{\delta}_{j,t,1}^{v(l)}\|_2^2+\frac{\rho_2}{2}\|\boldsymbol{x}_{j,t}-\boldsymbol{v}_{j,t,2}^{(l)}+\boldsymbol{\delta}_{j,t,2}^{v(l)}\|_2^2\Big) \tag{A.3}$$

**$(\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta})$-Update**. The $(\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta})$-update step in Eq. (A.2) can be decoupled into separate updates for a spatial $(\boldsymbol{u},\boldsymbol{\alpha})$-update and a temporal $(\boldsymbol{v},\boldsymbol{\beta})$-update. For spatial $(\boldsymbol{u},\boldsymbol{\alpha})$-update, we have:

$$\begin{aligned}(\boldsymbol{u}_{jk,t}^{(l'+1)},\boldsymbol{u}_{kj,t}^{(l'+1)}) &= \mathrm{argmin}\Big(\lambda_1\mu_1\omega_{jk}^{t}\|\boldsymbol{u}_{jk,t}+\boldsymbol{\alpha}_{jk,t}^{(l')}-\boldsymbol{u}_{kj,t}\|_2\\ &\quad+\frac{\rho_1}{2}\|\boldsymbol{x}_{j,t}^{(l+1)}-\boldsymbol{u}_{jk,t}+\boldsymbol{\delta}_{jk,t}^{u(l)}\|_2^2\\ &\quad+\frac{\rho_1}{2}\|\boldsymbol{x}_{k,t}^{(l+1)}-\boldsymbol{u}_{kj,t}+\boldsymbol{\delta}_{kj,t}^{u(l)}\|_2^2\Big)\\ \boldsymbol{\alpha}_{jk,t}^{(l'+1)} &= \mathrm{argmin}\Big(\mu_1\omega_{jk}^{t}\|\boldsymbol{u}_{jk,t}^{(l'+1)}+\boldsymbol{\alpha}_{jk,t}-\boldsymbol{u}_{kj,t}^{(l'+1)}\|_2\\ &\quad+(1-\mu_1)\|\boldsymbol{\alpha}_{jk,t}\|_p\Big)\end{aligned}$$

We can follow the same path to solve the temporal $(\boldsymbol{v},\boldsymbol{\beta})$-update, which is omitted here.

**$\boldsymbol{\delta}$-Update**. In the last step of our ADMM updating scheme, we have fully independent update rules for each scaled dual variable $\boldsymbol{\delta}_{jk,t}^{u}$, $\boldsymbol{\delta}_{j,t,1}^{v}$ and $\boldsymbol{\delta}_{j,t,2}^{v}$ as follows:

$$\left.\begin{aligned}{\boldsymbol{\delta}_{jk,t}^{u}}^{(l+1)} &= {\boldsymbol{\delta}_{jk,t}^{u}}^{(l)}+(\boldsymbol{x}_{j,t}^{(l+1)}-\boldsymbol{u}_{jk,t}^{(l+1)})\\ {\boldsymbol{\delta}_{j,t,1}^{v}}^{(l+1)} &= {\boldsymbol{\delta}_{j,t,1}^{v}}^{(l)}+(\boldsymbol{x}_{j,t}^{(l+1)}-\boldsymbol{v}_{j,t,1}^{(l+1)})\\ {\boldsymbol{\delta}_{j,t+1,2}^{v}}^{(l+1)} &= {\boldsymbol{\delta}_{j,t+1,2}^{v}}^{(l)}+(\boldsymbol{x}_{j,t+1}^{(l+1)}-\boldsymbol{v}_{j,t+1,2}^{(l+1)})\end{aligned}\right\} \tag{A.4}$$

## A.2 Streaming Case Formulation

Following the same notions in batch formulation of ST-DANR, we formally present the streaming case formulation:

$$\begin{aligned}
\min_{\boldsymbol{x},\boldsymbol{\alpha},\boldsymbol{\beta}} \quad & \sum_{t=\tau+1}^{\tau+m} \sum_{j\in\mathcal{V}} f_{j,t}(\boldsymbol{x}_{j,t}) + \lambda_1(1-\mu_1)\|\boldsymbol{\alpha}\|_{1,p} + \lambda_2(1-\mu_2)\|\boldsymbol{\beta}\|_{1,p} \\
&+ \lambda_1\mu_1 \sum_{t=\tau+1}^{\tau+m} \sum_{(j,k)\in\mathcal{E}} \omega_{jk}^{t} \left\|\boldsymbol{x}_{jk,t} + \boldsymbol{\alpha}_{jk,t} - \boldsymbol{x}_{kj,t}\right\|_2 \\
&+ \lambda_2\mu_2 \sum_{j\in\mathcal{V}} \Big( \omega_j^{\tau,\tau+1} \left\|\hat{\boldsymbol{x}}_{j,\tau,1} + \boldsymbol{\beta}_{j,\tau} - \boldsymbol{x}_{j,\tau+1,2}\right\|_2 + \sum_{t=\tau+1}^{\tau+m-1} \omega_j^{t,t+1} \left\|\boldsymbol{x}_{j,t,1} + \boldsymbol{\beta}_{j,t} - \boldsymbol{x}_{j,t+1,2}\right\|_2 \Big)
\end{aligned} \tag{A.5}$$

where $\|\boldsymbol{\alpha}\|_{1,p} = \sum_{jk\in\mathcal{E}} \sum_{t=\tau+1}^{\tau+m} \|\boldsymbol{\alpha}_{jk,t}\|_p$ and $\|\boldsymbol{\beta}\|_{1,p} = \sum_{j\in\mathcal{V}} \sum_{t=\tau}^{\tau+m-1} \|\boldsymbol{\beta}_{j,t}\|_p$. We then fit the problem to classic two-block ADMM framework as we did for ST-DANR in batch formulation:

$$\begin{aligned}
\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{v},\boldsymbol{\alpha},\boldsymbol{\beta}} \quad & \sum_{t=\tau+1}^{\tau+m} \sum_{j\in\mathcal{V}} f_{j,t}(\boldsymbol{x}_{j,t}) + \lambda_1(1-\mu_1)\|\boldsymbol{\alpha}\|_{1,p} + \lambda_2(1-\mu_2)\|\boldsymbol{\beta}\|_{1,p} \\
&+ \lambda_1\mu_1 \sum_{t=\tau+1}^{\tau+m} \sum_{(j,k)\in\mathcal{E}} \omega_{jk}^{t} \left\|\boldsymbol{u}_{jk,t} + \boldsymbol{\alpha}_{jk,t} - \boldsymbol{u}_{kj,t}\right\|_2 \\
&+ \lambda_2\mu_2 \sum_{j\in\mathcal{V}} \Big( \omega_j^{\tau,\tau+1} \left\|\hat{\boldsymbol{x}}_{j,\tau} + \boldsymbol{\beta}_{j,\tau} - \boldsymbol{v}_{j,\tau+1,2}\right\|_2 \\
&\qquad + \sum_{t=\tau+1}^{\tau+m-1} \omega_j^{t,t+1} \left\|\boldsymbol{v}_{j,t,1} + \boldsymbol{\beta}_{j,t} - \boldsymbol{v}_{j,t+1,2}\right\|_2 \Big) \\
\text{s.t.} \quad & [\boldsymbol{u}_{jk,t}, \boldsymbol{u}_{kj,t}] = [\boldsymbol{x}_{j,t}, \boldsymbol{x}_{k,t}], \quad t = \tau+1, \ldots, \tau+m \\
& [\boldsymbol{v}_{j,t,1}, \boldsymbol{v}_{j,t+1,2}] = [\boldsymbol{x}_{j,t}, \boldsymbol{x}_{j,t+1}], \ t = \tau+1, \ldots, \tau+m-1
\end{aligned} \tag{A.6}$$

where $\hat{\boldsymbol{x}}_\tau$ is the estimated models at $\tau$-th snapshot. Note that the existence of $\boldsymbol{\beta}_{j,\tau}$ and $\boldsymbol{v}_{j,\tau+1,2}$ causes a difference between Eq. (A.6) and the batch case formula in Eq. (A.1), which leads to a modification of updates in $(\boldsymbol{v}, \boldsymbol{\beta})$-updates w.r.t. these two variables:

$$\begin{aligned}
(\boldsymbol{v}_{j,\tau+1,2}^{(l+1)}, \boldsymbol{\beta}_{j,\tau}^{(l+1)}) = \text{argmin } \Big( & \lambda_2 \mu_2 \omega_j^{\tau,\tau+1} \|\hat{\boldsymbol{x}}_{j,\tau} + \boldsymbol{\beta}_{j,\tau} - \boldsymbol{v}_{j,\tau+1,2}\|_2 \\
+ \frac{\rho_2}{2} \|\boldsymbol{x}_{j,\tau+1}^{(l+1)} - \boldsymbol{v}_{j,\tau+1,2} + \boldsymbol{\delta}_{j,\tau+1,2}^{v(l)}\|_2^2 + \lambda_2(1-\mu_2)\|\boldsymbol{\beta}_{j,\tau}\|_p \Big)
\end{aligned}$$

This subproblem can still be solved by alternative minimization with following iterative scheme:

$$\begin{aligned}
\boldsymbol{v}_{j,\tau+1,2}^{(l'+1)} &= \text{argmin } \Big( \lambda_2 \mu_2 \omega_j^{\tau,\tau+1} \Big\|\hat{\boldsymbol{x}}_{j,\tau} + \boldsymbol{\beta}_{j,\tau}^{(l')} - \boldsymbol{v}_{j,\tau+1,2}\Big\|_2 + \frac{\rho_2}{2} \|\boldsymbol{x}_{j,\tau+1}^{(l+1)} - \boldsymbol{v}_{j,\tau+1,2} + \boldsymbol{\delta}_{j,\tau+1,2}^{v(l)}\|_2^2 \Big) \\
\boldsymbol{\beta}_{j,\tau}^{(l'+1)} &= \text{argmin}\Big( \mu_2 \omega_j^{\tau,\tau+1} \Big\|\hat{\boldsymbol{x}}_{j,\tau} + \boldsymbol{\beta}_{j,\tau} - \boldsymbol{v}_{j,\tau+1,2}^{(l'+1)}\Big\|_2 + (1-\mu_2)\|\boldsymbol{\beta}_{j,\tau}\|_p \Big)
\end{aligned}$$

Each step above is a simple optimization problem. Besides, updating steps for all remaining variables remain the same as in solution for batch formulation.

# Appendix B

# Preprocessing of HCP Data

In our empirical experiments, all resting-state and task-evoked fMR images as well as diffusion MR images were processed by the minimal HCP preprocessing pipeline and converting the MR images into network representations. Well-known brain imaging toolboxes such as FSL, freesurfer, AFNI and ANTs are be used to preprocess.

The effect of intra-scan head motion was eliminated by applying a 6-DOF rigid body spatial transform to the reference image. Second, image geometric distortion due to magnetic field in homogeneities was removed through field map correction and boundary-based registration to multi-modal (T1, T2, and GFA weighted) structural images of each subject. Functional and diffusion images of all subjects were registered to a standard template space, within a spatial realignment step which assumes both affine transformation and nonlinear large deformation diffeomorphic metric mapping. After fitting to the template space, all voxel-wise time-series were frequency filtered in order to block irrelevant information for tasks of interest. Finally, a newly-developed multi-modal parcellation was applied to cortical gray matter and create 300-500 regions of interest (ROIs).

To extract a network from the functional MR images, regional time series from each region were averaged to estimate inter-regional connection strength (by Pearson's correlation, partial correlation or magnitude-squared coherence). The network's nodes correspond to the ROIs and an edge connecting two nodes was associated with a value corresponding to the degree of coherence computed from the two ROIs' time-series. A standardized array with metadata was used to store, analyze, and share data.

The diffusion data was also used to extract a structural brain network by first inferring in each voxel, using spherical convolution, a orientation distribution function (ODF) representing where white matter fibers are oriented in partial volumes of the voxels. A global tractography of structural connectivity can then be learned using a markov random field model which optimizes a connection energy which minimizes bending of the ODFs while encouraging connections between neighboring voxels [175]. This generation of structural streamlines was done per-subject in each subject's native space and then joined into a network defined over the same nodes as that of the fMRI data. Connections between two regions were attributed by geometric information describing the connection bundles going from one to the other, such as bundle density, multi-directional heterogeneity, or median diffusivity. Empirical experiments on HCP dataset have shown this MRF-based streamline generation approach works significantly better than graph-based efficiency approach in terms of structural network construction.

# Bibliography

[1] D. H. Ki *et. al.*, *Whole genome analysis for liver metastasis gene signatures in colorectal cancer*, *Int J Cancer* (2007).

[2] S. Smith, *Advances in func. and struc. MRI analysis and implementation as FSL*, *Neuroimage* (2004).

[3] L. Akoglu *et. al.*, *Graph based anomaly detection and description: a survey*, *DMKD* (2015).

[4] S.Harenberg *et. al.*, *Community detection in large-scale networks: a survey and empirical evaluation*, *Comp. Stat.* (2014).

[5] C. C. Aggarwal and H. Wang, *Managing and Mining Graph Data.* Springer Publishing Company, Incorporated, 1st ed., 2010.

[6] E. Bullmore and O. Sporns, *Complex brain networks: graph theoretical analysis of structural and functional systems*, *Nature Reviews Neuroscience* **10** (2009), no. 3 186–198.

[7] M. X. Hoang, X. Dang, X. Wu, Z. Yan, and A. K. Singh, *GPOP: scalable group-level popularity prediction for online content in social networks*, in *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pp. 725–733, 2017.

[8] B. Ribeiro, M. X. Hoang, and A. K. Singh, *Beyond models: Forecasting complex network processes directly from data*, in *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pp. 885–895, 2015.

[9] X. Dang, A. Silva, A. K. Singh, A. Swami, and P. Basu, *Outlier detection from network data with subnetwork interpretation*, in *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, pp. 847–852, 2016.

[10] M. Mongiovì, P. Bogdanov, and A. K. Singh, *Mining evolving network processes*, in *ICDM*, 2013.

[11] S. Ranu, M. Hoang, and A. Singh, *Mining discriminative subgraphs from global-state networks*, in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 509–517, ACM, 2013.

[12] J. Dutkowski and T. Ideker, *Protein networks as logic functions in development and cancer*, *PLoS Comput Biol* **7** (2011), no. 9 e1002180.

[13] C. Li and H. Li, *Network-constrained regularization and variable selection for analysis of genomic data*, *Bioinformatics* **24** (2008), no. 9 1175–1182.

[14] C. Li and H. Li, *Variable selection and regression analysis for graph-structured covariates with an application to genomics*, *The annals of applied statistics* **4** (2010), no. 3 1498.

[15] T. D. Hocking *et. al.*, *Clusterpath an algorithm for clustering using convex fusion penalties*, in *ICML*, 2011.

[16] R. Tibshirani *et. al.*, *Sparsity and smoothness via the fused lasso*, *Journal of the Royal Statistical Society* (2005), no. 1.

[17] S. Yang, J. Wang, W. Fan, X. Zhang, P. Wonka, and J. Ye, *An efficient admm algorithm for multidimensional anisotropic total variation regularization problems*, in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 641–649, ACM, 2013.

[18] D. Hallac *et. al.*, *Network lasso: Clustering and optimization in large graphs*, in *ACM SIGKDD*, 2015.

[19] S. Boyd *et. al.*, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, *Foundations and Trends® in Machine Learning* **3** (2011), no. 1.

[20] D. S. Bassett, N. F. Wymbs, M. A. Porter, P. J. Mucha, J. M. Carlson, and S. T. Grafton, *Dynamic reconfiguration of human brain networks during learning*, *Proceedings of the National Academy of Sciences* **108** (2011), no. 18 7641–7646.

[21] M. M. Chun, J. D. Golomb, and N. B. Turk-Browne, *A taxonomy of external and internal attention*, *Annual review of psychology* **62** (2011) 73–101.

[22] P. Hagmann, M. Kurant, X. Gigandet, P. Thiran, V. J. Wedeen, R. Meuli, and J.-P. Thiran, *Mapping human whole-brain structural networks with diffusion mri*, *PloS one* **2** (2007), no. 7 e597.

[23] N. T. Markov, M. Ercsey-Ravasz, D. C. Van Essen, K. Knoblauch, Z. Toroczkai, and H. Kennedy, *Cortical high-density counterstream architectures*, *Science* **342** (2013), no. 6158.

[24] H.-J. Park and K. Friston, *Structural and functional brain networks: from connections to cognition*, *Science* **342** (2013), no. 6158.

[25] H.-J. Park, M. Kubicki, C.-F. Westin, I.-F. Talos, A. Brun, S. Peiper, R. Kikinis, F. A. Jolesz, R. W. McCarley, and M. E. Shenton, *Method for combining information from white matter fiber tracking and gray matter parcellation*, *American journal of neuroradiology* **25** (2004), no. 8 1318–1324.

[26] C. Summerfield, M. Greene, T. Wager, T. Egner, J. Hirsch, and J. Mangels, *Neocortical connectivity during episodic memory formation*, *PLoS Biol* **4** (2006), no. 5 e128.

[27] *2015 Alzheimer's disease facts and figures*, *Alzheimer's & Dementia* **11** (2015), no. 3.

[28] C.-Y. Lo, P.-N. Wang, K.-H. Chou, J. Wang, Y. He, and C.-P. Lin, *Diffusion tensor tractography reveals abnormal topological organization in structural cortical networks in alzheimer's disease*, *Journal of Neuroscience* **30** (2010), no. 50 16876–16885.

[29] F. C. Morabito, M. Campolo, D. Labate, G. Morabito, L. Bonanno, A. Bramanti, S. De Salvo, A. Marra, and P. Bramanti, *A longitudinal eeg study of alzheimer's disease progression based on a complex network approach*, *International journal of neural systems* **25** (2015), no. 02 1550005.

[30] M. D. Greicius, K. Supekar, V. Menon, and R. F. Dougherty, *Resting-state functional connectivity reflects structural connectivity in the default mode network*, *Cerebral Cortex* **19** (2009), no. 1 72–78.

[31] D. S. Bassett and E. Bullmore, *Small-world brain networks*, *The neuroscientist* **12** (2006), no. 6 512–523.

[32] A. M. Hermundstad, D. S. Bassett, K. S. Brown, E. M. Aminoff, D. Clewett, S. Freeman, A. Frithsen, A. Johnson, C. M. Tipper, M. B. Miller, *et. al.*, *Structural foundations of resting-state and task-based functional connectivity in the human brain*, *Proceedings of the National Academy of Sciences* **110** (2013), no. 15 6169–6174.

[33] D. S. Bassett, J. A. Brown, V. Deshpande, J. M. Carlson, and S. T. Grafton, *Conserved and variable architecture of human white matter connectivity*, *Neuroimage* **54** (2011), no. 2 1262–1279.

[34] J. Goñi, M. P. Van Den Heuvel, A. Avena-Koenigsberger, N. V. De Mendizabal, R. F. Betzel, A. Griffa, P. Hagmann, B. Corominas-Murtra, J.-P. Thiran, and O. Sporns, *Resting-brain functional connectivity predicted by analytic measures of*

*network communication*, *Proceedings of the National Academy of Sciences* **111** (2014), no. 2 833–838.

[35] C. O. Becker, S. Pequito, G. J. Pappas, M. B. Miller, S. T. Grafton, D. S. Bassett, and V. M. Preciado, *Spectral mapping of brain functional connectivity from diffusion imaging*, *Scientific Reports* **8** (2018), no. 1 1–15.

[36] A. Venkataraman, Y. Rathi, M. Kubicki, C.-F. Westin, and P. Golland, *Joint modeling of anatomical and functional connectivity for population studies*, *IEEE transactions on medical imaging* **31** (2011), no. 2 164–182.

[37] C. Jiang, F. Coenen, and M. Zito, *A survey of frequent subgraph mining algorithms*, *Knowledge Eng. Review* **28** (2013), no. 1 75–105.

[38] C. C. Aggarwal and H. Wang, *A survey of clustering algorithms for graph data*, in *Managing and Mining Graph Data*, pp. 275–301. 2010.

[39] N. S. Ketkar, L. B. Holder, and D. J. Cook, *Empirical comparison of graph classification algorithms.*, in *ICDM*, pp. 259–266, 2009.

[40] L. Akoglu, M. McGlohon, and C. Faloutsos, *oddball: Spotting anomalies in weighted graphs*, in *PAKDD (2)*, pp. 410–421, 2010.

[41] D. Lee, O.-R. Jeong, and S.-g. Lee, *Opinion mining of customer feedback data on the web*, ICUIMC '08, pp. 230–235, ACM, 2008.

[42] C. Li and H. Li, *Network-constrained regularization and variable selection for analysis of genomic data*, *Bioinformatics* **24** (2008), no. 9 1175–1182.

[43] I. N. Davidson *et. al.*, *Network discovery via constrained tensor analysis of fmri data*, in *KDD*, 2013.

[44] S. Ranu, M. Hoang, and A. K. Singh, *Mining discriminative subgraphs from global-state networks*, in *KDD*, 2013.

[45] S. Soundarajan, T. Eliassi-Rad, and B. Gallagher, *Which network similarity method should you choose?*, in *Workshop on Information Networks at NYU*, 2013.

[46] G. H. Golub and C. F. V. Loan, *Matrix Computations.* The Johns Hopkins University Press, 3rd ed., 1996.

[47] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning. Data Mining, Inference, and Prediction.* 2001.

[48] A. K. Cline and I. S. Dhillon, *Computation of the Singular Value Decomposition.* Handbook of Linear Algebra, CRC Press, 2006.

[49] X. H. Dang, I. Assent, R. T. Ng, A. Zimek, and E. Schubert, *Discriminative features for identifying and interpreting outliers*, in *ICDE*, pp. 88–99, 2014.

[50] X. H. Dang *et. al.*, *Local outlier detection with interpretation*, in *PKDD*, pp. 304–320, 2013.

[51] J. L. Bentley, *Multidimensional binary search trees used for associative searching*, *Communications ACM* (1975).

[52] J. Dutkowski and T. Ideker, *Protein networks as logic functions in development and cancer*, *PLoS* (2011).

[53] L. Hui, P. Zhiyu, *et. al.*, *Genome-wide association study dissects the genetic architecture of oil biosynthesis in maize kernels*, *Nature Genetics* **45** (2013) 43–50.

[54] D. H. Ki, H.-C. Jeung, C. H. Park, S. H. Kang, G. Y. Lee, W. S. Lee, N. K. Kim, H. C. Chung, and S. Y. Rha, *Whole genome analysis for liver metastasis gene signatures in colorectal cancer*, *International journal of cancer* **121** (2007), no. 9.

[55] D. Ruth *et. al.*, *Genes2fans: connecting genes through functional association networks*, *BMC bioinformatics* (2012).

[56] X. Yan *et. al.*, *Mining significant graph patterns by leap search*, in *SIGMOD*, 2008.

[57] S. Ranu *et. al.*, *Graphsig: A scalable approach to mining significant subgraphs in large graph databases*, in *ICDE*, 2009.

[58] N. Jin *et. al.*, *GAIA: graph classification using evolutionary computation*, in *SIGMOD*, 2010.

[59] N. Jin, C. Young, and W. Wang, *Graph classification based on pattern co-occurrence*, in *CIKM*, 2009.

[60] P. Bogdanov, C. Faloutsos, M. Mongiovì, E. E. Papalexakis, R. Ranca, and A. K. Singh, *Netspot: Spotting significant anomalous regions on dynamic networks*, in *SDM*, 2013.

[61] P. Bogdanov, M. Mongiovì, and A. K. Singh, *Mining heavy subgraphs in time-evolving networks*, in *ICDM*, 2011.

[62] C. Li and H. Li, *Variable selection and regression analysis for graph-structured covariates with an application to genomics*, *The Annals of Applied Statistics* (2010).

[63] J. Noirel *et. al.*, *Identifying differentially expressed subnetworks with mmg*, *Bioinformatics* (2008).

[64] M. Weiner *et. al.*, *The Alzheimer's disease neuroimaging initiative: A review of papers published since its inception*, *Alzheimer's & Dementia* **9** (2013), no. 1.

[65] O. Sporns, *Networks analysis, complexity, and brain function*, *Complex.* **8** (Sept., 2002) 56–60.

[66] M. Wall, A. Rechtsteiner, and L. Rocha, *Singular value decomposition and principal component analysis*, *A Practical Approach to Microarray Data Analysis* (2003).

[67] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer-Verlag, 2006.

[68] X. H. Dang *et. al.*, *Discriminative subnetworks with regularized spectral learning for global-state network data*, in *ECML*, 2014.

[69] S. Soundarajan, T. Eliassi-Rad, and B. Gallagher, *A guide to selecting a network similarity method.*, in *SDM*, 2014.

[70] S. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge University Press, 2004.

[71] D. R. Hunter and K. Lange, *A tutorial on MM algorithms*, *The American Statistician* (2004).

[72] K. Lange, D. R. Hunter, and I. Yang, *Optimization transfer using surrogate objective functions*, *Journal of Computational and Graphical Statistics* (2000).

[73] J. Gorski, F. Pfeuffer, and K. Klamroth, *Biconvex sets and optimization with biconvex functions: a survey and extensions*, *Math. Meth. of OR* **66** (2007), no. 3 373–407.

[74] T. Hastie *et. al.*, *The Elements of Statistical Learning. Data Mining, Inference, and Prediction.* 2009.

[75] D. Krstajic *et. al.*, *Cross-validation pitfalls when selecting and assessing regression and classification models*, *J. Cheminformatics* (2014).

[76] K. Bache and M. Lichman, *UCI repository*, 2013.

[77] D. I. Shuman *et. al.*, *The emerging field of signal processing on graphs*, *IEEE Signal Process. Mag.* (2013).

[78] B. E. Leonard, *Alzheimer's disease and stroke: possible biochemical causes and treatment strategies.* John Wiley and Sons, 1992.

[79] A. R. Crossman and D. Neary, *Neuroanatomy. An illustrated colour text.* Churchill Livingstone Elsevier, 2010.

[80] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods.* SIAM, 1998.

[81] M. Kivelä, Arenas, *et. al.*, *Multilayer networks*, *Journal of Complex Networks* **2** (2014), no. 3 203–271.

[82] P. Holme and J. Saramäki, *Temporal networks*, *Physics Reports* **519** (2012), no. 3 97–125.

[83] A. Silva *et. al.*, *Graph wavelets via sparse cuts*, in *SIGKDD*, 2016.

[84] C. Jiang *et. al.*, *A survey of frequent subgraph mining algorithms*, *Knowledge Eng. Review* **28** (2013), no. 1.

[85] B. Cao *et. al.*, *Mining brain networks using multiple side views for neurological disorder identification*, in *ICDM*, 2015.

[86] A. K. S. P. B. Xuan Hong Dang, Arlei Silva and A. Swami, *Outlier detection from network data with subnetwork interpretation*, in *ICDM*, 2016.

[87] N. S. Ketkar *et. al.*, *Empirical comparison of graph classification algorithms*, in *CIDM*, 2009.

[88] X. H. Dang, A. K. Singh, P. Bogdanov, H. You, and B. Hsu, *Discriminative subnetworks with regularized spectral learning for global-state network data*, in *ECML*, 2014.

[89] H. Peng *et. al.*, *Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy*, *IEEE TPAMI* **27** (2005), no. 8.

[90] G. Isabelle and E. André, *An introduction to variable and feature selection*, *JMLR* (2003).

[91] F. Martino *et. al.*, *Combining multivariate voxel selection and support vector machines for mapping and classification of fMRI spatial patterns*, *NeuroImage* (2008).

[92] S. M. Smith *et. al.*, *Network modelling methods for fMRI*, *NeuroImage* **54** (2011), no. 2 875 – 891.

[93] J. Zhou *et. al.*, *A multi-task learning formulation for predicting disease progression*, in *SIGKDD*, 2011.

[94] A. K. S. Xuan Hong Dang, Hongyuan You and P. Bogdanov, *Learning predictive substructures with regularization for network data*, in *ICDM*, 2015.

[95] X. Kong and P. Yu, *Brain network analysis: a data mining perspective*, *SIGKDD Explorations* (2013).

[96] S. L. Simpson *et. al.*, *Analyzing complex functional brain networks: Fusing statistics and network science to understand the brain*, *Statistical Surveys* **7** (2013).

[97] K. Supekar *et. al.*, *Network analysis of intrinsic functional brain connectivity in alzheimer's disease.*, *PLoS* (2008).

[98] E. W. Lang *et. al.*, *Brain connectivity analysis: A short survey*, *Intell. Neuroscience* **2012** (Jan., 2012).

[99] X. Kong and P. Yu, *Semi-supervised feature selection for graph classification*, in *SIGKDD*, 2010.

[100] X. Kong *et. al.*, *Discriminative feature selection for uncertain graph classification*, in *SIAM-SDM*, 2013.

[101] J. Kim *et. al.*, *Comparison of statistical tests for group differences in brain networks*, *NeuroImage* (2014).

[102] E. L. Dennis *et. al.*, *Functional brain connectivity using fMRI in aging and Alzheimer's disease*, *Neuropsychol Rev* (2014).

[103] J. Sharpnack, A. Singh, and A. Rinaldo, *Sparsistency of the edge lasso over graphs*, in *AISTATS*, 2012.

[104] D. I. Shuman *et. al.*, *The emerging field of signal processing on graphs*, *IEEE Signal Processing Magazine* **30** (2013), no. 3.

[105] Y.-X. Wang, J. Sharpnack, A. J. Smola, and R. J. Tibshirani, *Trend filtering on graphs*, in *AISTATS*, 2015.

[106] J. Pang *et. al.*, *Graph laplacian regularization for image denoising*, *IEEE Transactions on Image Processing* (2017).

[107] B. Xin, Y. Kawahara, Y. Wang, and W. Gao, *Efficient generalized fused lasso and its application to the diagnosis of alzheimer's disease.*, in *AAAI*, 2014.

[108] E. C. Chi *et. al.*, *Splitting methods for convex clustering*, *Journal of Computational and Graphical Statistics* **24** (2015), no. 4.

[109] Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabási, *Controllability of complex networks*, *Nature* **473** (2011), no. 7346 167.

[110] L. Anselin, *Under the hood: Issues in the specification and interpretation of spatial regression models*, *Agricultural economics* **27** (2002), no. 3.

[111] R. Harris, J. Moffat, and V. Kravtsova, *In search of 'w'*, *Spatial Economic Analysis* **6** (2011), no. 3 249–270.

[112] M. Kim and J. Leskovec, *Nonparametric multi-group membership model for dynamic networks*, in *NIPS*, 2013.

[113] Q. Ho *et. al.*, *Evolving cluster mixed-membership blockmodel for time-evolving networks*, in *AISTATS*, 2011.

[114] H.-F. Yu *et. al.*, *Temporal regularized matrix factorization for high-dimensional time series prediction*, in *NIPS*, 2016.

[115] J. E. Vogt and V. Roth, *A complete analysis of the $\ell_{1,p}$ group-lasso*, in *ICML*, 2012.

[116] D. Gabay and B. Mercier, *A dual algorithm for the solution of nonlinear variational problems via finite element approximation*, *Computers & Mathematics with Applications* (1976), no. 1.

[117] Y. Peng *et. al.*, *Robust alignment by sparse and low-rank decomposition for linearly correlated images*, *PAMI* (2012).

[118] J. Nocedal *et. al.*, *Numerical Optimization.* 2006.

[119] J. Eckstein *et. al.*, *On the douglas—rachford splitting method and the proximal point algorithm for maximal monotone operators*, *Mathematical Programming* **55** (1992), no. 1.

[120] D. Gabay, *Chapter ix applications of the method of multipliers to variational inequalities*, *Studies in mathematics and its applications* **15** (1983).

[121] R. Nishihara *et. al.*, *A general analysis of the convergence of admm*, in *ICML*, pp. 343–352, 2015.

[122] B. He and X. Yuan, *On non-ergodic convergence rate of douglas–rachford alternating direction method of multipliers*, *Numerische Mathematik* **130** (2015), no. 3 567–577.

[123] D. Hallac, Y. Park, S. Boyd, and J. Leskovec, *Network inference via the time-varying graphical lasso*, 2017.

[124] Y. Chi *et. al.*, *Evolutionary spectral clustering by incorporating temporal smoothness*, in *ACM SIGKDD*, 2007.

[125] H. Wang, F. Nie, and H. Huang, *Low-rank tensor completion with spatio-temporal consistency*, in *AAAI*, 2014.

[126] S. Poddar *et. al.*, *Dynamic mri using smoothness regularization on manifolds (storm)*, *IEEE TMI* **35** (2016), no. 4.

[127] X.-H. Dang *et. al.*, *Subnetwork mining with spatial and temporal smoothness*, in *SDM*, 2017.

[128] P. Gong, J. Ye, and C. Zhang, *Robust multi-task feature learning*, in *ACM SIGKDD*, 2012.

[129] J. Xu *et. al.*, *Factorized multi-task learning for task discovery in personalized medical models*, in *SDM*, 2015.

[130] P. A. Soranno *et. al.*, *Lagos-ne*, *GigaScience* **6** (2017), no. 12.

[131] F. Lindsten *et. al.*, *Clustering using sum-of-norms regularization*, in *Statistical Signal Processing Workshop*, 2011.

[132] M. Belkin, P. Niyogi, and V. Sindhwani, *Manifold regularization: A geometric framework for learning from labeled and unlabeled examples*, *Journal of machine learning research* **7** (2006), no. Nov 2399–2434.

[133] M. Belkin *et. al.*, *Regularization and semi-supervised learning on large graphs*, in *COLT*, 2004.

[134] N. Guan, D. Tao, Z. Luo, and B. Yuan, *Manifold regularized discriminative nonnegative matrix factorization with fast gradient descent*, *IEEE Transactions on Image Processing* **20** (2011), no. 7 2030–2048.

[135] K. Q. Weinberger, F. Sha, Q. Zhu, and L. K. Saul, *Graph laplacian regularization for large-scale semidefinite programming*, in *NIPS*, 2007.

[136] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, *Physica D: nonlinear phenomena* **60** (1992), no. 1-4.

[137] M. Rubinov and O. Sporns, *Complex network measures of brain connectivity: uses and interpretations*, *NeuroImage* **52** (2010), no. 3 1059–1069.

[138] K. Batista-García-Ramó and C. I. Fernández-Verdecia, *What we know about the brain structure–function relationship*, *Behavioral Sciences* **8** (2018), no. 4 39.

[139] D. S. Bassett and O. Sporns, *Network neuroscience*, *Nature Neuroscience* **20** (2017), no. 3 353.

[140] D. S. Bassett, P. Zurn, and J. I. Gold, *On the nature and use of models in network neuroscience*, *Nature Reviews Neuroscience* **19** (2018), no. 9 566–578.

[141] D. C. Van Essen, S. M. Smith, D. M. Barch, T. E. Behrens, E. Yacoub, K. Ugurbil, W.-M. H. Consortium, *et. al.*, *The wu-minn human connectome project: an overview*, *NeuroImage* **80** (2013) 62–79.

[142] A. J. Rothman, E. Levina, and J. Zhu, *Sparse multivariate regression with covariance estimation*, *Journal of Computational and Graphical Statistics* **19** (2010), no. 4 947–962.

[143] K.-A. Sohn and S. Kim, *Joint estimation of structured sparsity and output structure in multiple-output regression via inverse-covariance regularization*, in *Artificial Intelligence and Statistics*, pp. 1081–1089, 2012.

[144] M. Wytock and Z. Kolter, *Sparse gaussian conditional random fields: Algorithms, theory, and application to energy forecasting*, in *International Conference on Machine Learning*, pp. 1265–1273, 2013.

[145] X.-T. Yuan and T. Zhang, *Partial gaussian graphical model estimation*, *IEEE Transactions on Information Theory* **60** (2014), no. 3 1673–1687.

[146] F. Freyer, K. Aquino, P. A. Robinson, P. Ritter, and M. Breakspear, *Bistability and non-gaussian fluctuations in spontaneous cortical activity*, *Journal of Neuroscience* **29** (2009), no. 26 8512–8524.

[147] J. Hlinka, M. Paluš, M. Vejmelka, D. Mantini, and M. Corbetta, *Functional connectivity in resting-state fmri: is linear correlation sufficient?*, *NeuroImage* **54** (2011), no. 3.

[148] A. Eklund, T. E. Nichols, and H. Knutsson, *Cluster failure: Why fmri inferences for spatial extent have inflated false-positive rates*, *Proceedings of the National Academy of Sciences* **113** (2016), no. 28 7900–7905.

[149] M. Columb and M. Atkinson, *Statistical analysis: sample size and power estimations*, *Bja Education* **16** (2016), no. 5.

[150] J. Peng, P. Wang, N. Zhou, and J. Zhu, *Partial correlation estimation by joint sparse regression models*, *Journal of the American Statistical Association* **104** (2009), no. 486 735–746.

[151] K. Khare, S.-Y. Oh, and B. Rajaratnam, *A convex pseudolikelihood framework for high dimensional partial correlation estimation with convergence guarantees*, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **77** (2015), no. 4 803–825.

[152] P. Koanantakool, A. Ali, A. Azad, A. Buluc, D. Morozov, L. Oliker, K. Yelick, and S.-Y. Oh, *Communication-avoiding optimization methods for distributed massive-scale sparse inverse covariance estimation*, in *International Conference on Artificial Intelligence and Statistics*, pp. 1376–1386, PMLR, 2018.

[153] S. Oh, O. Dalal, K. Khare, and B. Rajaratnam, *Optimization methods for sparse pseudo-likelihood graphical model selection*, in *Advances in Neural Information Processing Systems*, pp. 667–675, 2014.

[154] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, *SIAM Journal on Imaging Sciences* **2** (2009), no. 1 183–202.

[155] R. T. Rockafellar, *Convex analysis*, vol. 28. Princeton University Press, 1970.

[156] J.-J. Moreau, *Proximité et dualité dans un espace hilbertien*, *Bulletin de la Société mathématique de France* **93** (1965) 273–299.

[157] M. Yuan and Y. Lin, *Model selection and estimation in the gaussian graphical model*, *Biometrika* **94** (2007), no. 1 19–35.

[158] J. Peng, J. Zhu, A. Bergamaschi, W. Han, D.-Y. Noh, J. R. Pollack, and P. Wang, *Regularized multivariate regression for identifying master predictors with application to integrative genomics study of breast cancer*, *The Annals of Applied Statistics* **4** (2010), no. 1 53.

[159] C. McCarter and S. Kim, *Large-scale optimization algorithms for sparse conditional gaussian graphical models*, in *Artificial Intelligence and Statistics*, pp. 528–537, 2016.

[160] T. Fawcett, *An introduction to roc analysis*, *Pattern Recognition Letters* **27** (2006), no. 8 861–874.

[161] J. Friedman, T. Hastie, and R. Tibshirani, *Applications of the lasso and grouped lasso to the estimation of sparse graphical models*, tech. rep., Stanford University, 2010.

[162] P. Hagmann, L. Cammoun, X. Gigandet, R. Meuli, C. J. Honey, V. J. Wedeen, and O. Sporns, *Mapping the structural core of human cerebral cortex*, *PLoS Biology* (2008).

[163] C. Honey, O. Sporns, L. Cammoun, X. Gigandet, J.-P. Thiran, R. Meuli, and P. Hagmann, *Predicting human resting-state functional connectivity from structural connectivity*, *Proceedings of the National Academy of Sciences* **106** (2009), no. 6 2035–2040.

[164] C. J. Honey, R. Kötter, M. Breakspear, and O. Sporns, *Network structure of cerebral cortex shapes functional connectivity on multiple time scales*, *Proceedings of the National Academy of Sciences* **104** (2007), no. 24 10240–10245.

[165] M. E. Raichle, *The brain's default mode network*, *Annual Review of Neuroscience* **38** (2015) 433–447.

[166] M. W. Cole, D. S. Bassett, J. D. Power, T. S. Braver, and S. E. Petersen, *Intrinsic and task-evoked network architectures of the human brain*, *Neuron* **83** (2014), no. 1 238–251.

[167] E. N. Davison, K. J. Schlesinger, D. S. Bassett, M.-E. Lynall, M. B. Miller, S. T. Grafton, and J. M. Carlson, *Brain network adaptability across task states*, *PLoS Computational Biology* **11** (2015), no. 1.

[168] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, *ArXiv Preprint arXiv:1312.6114* (2013).

[169] D. M. Barch, G. C. Burgess, M. P. Harms, S. E. Petersen, B. L. Schlaggar, M. Corbetta, M. F. Glasser, S. Curtiss, S. Dixit, C. Feldt, *et. al.*, *Function in the human connectome: task-fmri and individual differences in behavior*, *NeuroImage* **80** (2013) 169–189.

[170] A. Baddeley, C. Jarrold, and F. Vargha-Khadem, *Working memory and the hippocampus*, *Journal of Cognitive Neuroscience* **23** (2011), no. 12 3855–3861.

[171] S. M. Smith and T. E. Nichols, *Statistical challenges in "big data" human neuroimaging*, *Neuron* **97** (2018), no. 2 263–268.

[172] G. Salimi-Khorshidi, G. Douaud, C. F. Beckmann, M. F. Glasser, L. Griffanti, and S. M. Smith, *Automatic denoising of functional mri data: combining independent component analysis and hierarchical fusion of classifiers*, *Neuroimage* **90** (2014) 449–468.

[173] J. L. Andersson and S. N. Sotiropoulos, *An integrated approach to correction for off-resonance effects and subject movement in diffusion mr imaging*, *Neuroimage* **125** (2016) 1063–1078.

[174] D. Andrzejewski, X. Zhu, and M. Craven, *Incorporating domain knowledge into topic modeling via dirichlet forest priors*, in *Proceedings of the 26th annual international conference on machine learning*, pp. 25–32, 2009.

[175] D. Christiaens, M. Reisert, T. Dhollander, S. Sunaert, P. Suetens, and F. Maes, *Global tractography of multi-shell diffusion-weighted imaging data using a multi-tissue model*, *NeuroImage* **123** (2015) 89–101.