On Time-varing Graphical Lasso for Functional Brain Network Connectivity Dynamics Inference

Hanlin Zhang[1], Ivor Cribben[1]
[1] University of Alberta, Edmonton, AB Canada

Modeling how brain connectivity networks evolve over time is a fundamental problem in neuroscience and holds the promise to reveal certain neuropsychiatric disorders. The estimation of time-varying networks for functional magnetic resonance imaging(fMRI) datasets is thus of increasing importance and interest. Graphical lasso(glasso) is widely used to capture pairwise structural dependencies for fMRI time series data. Many glasso variants like sliding-window glasso have been proposed and have achieved state-of-the-art results. However, they fail to uncover the underlying realistic dynamic structure of the resting-state brain connectivity network. Here, we introduce a joint glasso named time-varying glasso that captures the smoothly evolutionary patterns of resting-state fMRI time series. The generated static and dynamic networks are further used to classify patients and controls and we prove that classification results on dynamic networks outperforms that on static networks. We also develop an efficient implementation that achieves the scalable and distributed solution is in a divide-and-conquer manner. Experimental results show that the proposed classification algorithms outperform the widely-used baselines.

Index Terms—**Time series analysis; graphical models; functional connectivity; graphical lasso; fMRI**

I. Introduction

Recently, there has been a surge of interest in estimating functional brain connectivity networks based on fMRI time series. Gaussian graphical models(GGM) have been used to represent the structural dependencies between pairwise region of interests (ROIs).

Traditionally, the graphical lasso with a sparsity assumption [8] has been widely used to reveal underlying structures based on the fact that certain ROIs are restricted to communicate with the minority of others. In this way, the original graphical model problem is reduced into a sequence of regression problems.

For time-varying network estimation, a number of methods have been developed.

For example, [1], proposed a temporally smoothed l_1 regularized logistic regression method named TESLA is proposed as an approach that allows more flexibility in the smoothness of graph evolution, the sparsity of graph estimation, and for both sparsity within the regression function and smoothness across adjacent networks. For directed networks, the time-varying dynamic Bayesian networks (TV-DBN) has been proposed [25] for modeling the time-varying network structures underlying non-stationary biological time series. [24] proposed a kernel-reweighted logistic regression method (KELLER) for reverse-engineering the dynamic interactions between genes based on their time series of expression values. However, it is noteworthy that the above-mentioned method does not establish guarantees on smoothly-changing time-consistency under low-dimension conditions for time-varying GGM.

Machine learning models have been widely used in recent years for various problems [29]. To tackle the fMRI network classification problem, [20] proposes a framework based on Gaussian graphical models and an L1-norm regularized linear Support Vector Machines (SVM). Their method allows for pattern classification in both block-related and event-related fMRI data. [11] uses a sliding window technique to investigate time-varying brain connectivity for Alzheimer's disease classification. [19] introduces an end-to-end deep learning model for the classification of a neurological disorder from fMRI data. [17] proposes a convolutional neural network (CNN) architecture called connectome-convolutional neural network (CCNN) to distinguish between subject groups in a functional connectome classification problem [16] used node2vec [10] to embed the vertexes of graph in the node embedding space, and transform the brain network into images based on Magnetoencephalography (MEG) data.

Manuscript received December 1, 2012; revised August 26, 2015. Corresponding author: M. Shell (email: http://www.michaelshell.org/contact.html).

Despite their effectiveness, the previous paper focuses on problems that are different from our objective in this paper. Our research question is whether dynamic functional connectivity networks or a static network can perform better in the network classification problem. To compare their performance, we apply several methods to different fMRI datasets.

In this paper, our contributions are the following:

1) We introduce the time-varying glasso, a joint glasso with a Laplacian penalty and a glasso penalty, to better reflect the underlying evolutionary patterns of fMRI time series datasets.
2) We compare the estimation results of several commonly used network estimation methods with respect to classification accuracy.
3) We employ the Alternating Direction Method of Multipliers(ADMM) to achieve a scalable and distributed results of the proposed convex optimization problems.

II. Methods

We formalized our problem as a binary classification problem where the input is either a static or a dynamic network and the target is disease status(-1 for brain disorder, 1 for control). In particular, given fMRI time series of every subject, we aim to estimate the Gaussian graphical model in both a static and a dynamic manner using the graphical lasso and its variants.

We use three different models 1. static network model(estimated by the graphical lasso) 2. sliding window graphical lasso 3. time-varying graphical lasso. We apply these models to several fMRI datasets and compare their performance.

Here, we use the terms network and graph interchangeably.

A. Gaussian graphical model

Probability graphical models have proved an effective way to represent dependencies between random variables from real-world data, which is the foundation for further statistical inference and learning [15]. To represent pairwise dependencies among variables, the Gaussian graphical model (GGM), a Markov Network based on the precision matrix P, is commonly used. Consider a Gaussian distribution $p(X_1, X_2, \ldots, X_n) = \mathcal{N}(\mu; \Sigma)$, and let $\theta = \Sigma^{-1}$ be the inverse covariance matrix (precision matrix), where $\theta_{i,j} = 0$ if and only if $X_i \perp X_j \mid X_{-ij}$, i.e. $P(X_i, X_j \mid X_{-ij}) = P(X_i \mid X_{-ij})P(X_j \mid X_{-ij})$. Precision matrices encode partial correlations, which are more sophisticated dependence measures compared with a correlation network, and has been proved to offer high performance on network connectivity estimation with high-quality fMRI data [22]. The precision matrix can also be represented by an undirected graph. An undirected graph is determined accordingly for every subject, in which precision matrix is interpreted as the symmetric adjacency matrix A of graph $G := (V, E)$. Variables are treated as vertexes. To represent pair-wise interactions between variables, we use non-zero parameters to represent the existence of edges, while a missing edge between two nodes is equivalent to a zero entry in the precision matrix.

B. Graphical lasso

We first introduce graphical lasso for the static network. We assume that the precision matrix P is sparse for real data problems. It makes empirical sense to do this as brain ROIs are assumed to interact with only a small group of other ROIs. It also makes statistical sense as learning is feasible in high dimensions with a small sample. Based on the sparsity assumption, the graphical lasso (glasso) for sparse inverse covariance matrix estimation [8] on fMRI time-series data can be formulated as follows: Given N multivariate normal observations of dimension p, with mean μ and covariance Σ, the glasso aims to maximize the log-likelihood to uncover the dependency structure based on multivariate time series observations:

$$\max \quad \log\det\theta - \operatorname{Tr}(S\theta) - \lambda\|\theta\|_1 \tag{1}$$

where the precision matrix and empirical covariance matrix are denoted as $\theta = \Sigma^{-1}$ and S, respectively. θ must be symmetric positive-definite ($\mathbf{S}^p_{++}$). To estimate the precision matrix, the regression process is repeatedly applied for each node and the zero valued edges are removed. Zero parameters arise through the l_1 penalty. The penalty parameter λ regulates the sparsity level of precision matrices: larger λ yields a sparser precision matrix, while smaller values yield denser matrices.

C. Graphical lasso with sliding window technique

For the dynamic window technique we use and overlapping window. Specifically, assume that we have N subjects with T time points and p ROIs for each subject. Following [2], we estimate a precision matrix using the graphical lasso in equation 1 independently in every sliding window with length $d \in \mathbf{R}$ and step size $\delta \in \mathbf{R}$ to estimate dynamic networks $\Theta_i = (\theta_1, \ldots, \theta_n)$ for every subject x_i. The detailed experimental settings and parametric choosing are presented in the simulation section.

D. Time-varying Graphical lasso

Consider a sequence of multivariate observations in $\mathbf{R}^p$ sampled from a distribution $x \sim N(0, \Sigma(t))$. Time series are observed at every even time stamp $t_1, t_2, \ldots, t_T$. To construct structural dependencies, $\Theta = \{\theta_1, \theta_2, \ldots, \theta_T\}$ are estimated from the raw time series datasets, where $\theta_i = \Sigma_i^{-1}$ is the corresponding precision matrix.

Rather than treating networks as observable invariant entities, various penalties can be combined with the lasso penalty as a joint graphical lasso [7] to capture time consistency between every network snapshot [12]. For example, [1, 14] combine a lasso penalty and a fused penalty, thus enforcing both smoothness and sparsity. Perturbed-node joint graphical lasso can be used to detect single nodes relocations [18]. In [7], the group graphical lasso and fused graphical lasso are presented to cause global restructuring at a few timestamps. Following the naive assumption that each temporal snapshot of the brain network is completely independent and the generating process is time-invariant leads to high variance due to sample scarcity. In dynamic cases, instead of using the traditional sliding window technique which assumes networks to be independent of their neighborhoods, we utilize time-varying graphical lasso (TVGL) for estimation. Our key assumption is that the time-varying networks vary smoothly across time, therefore temporally adjacent networks are able to borrow information from each other.

Assuming a smoothly changing pattern holds, we utilize the Laplacian penalty function $\Phi(X) = \sum_{i,j} X_{i,j}^2$ (a sum-of-squares loss function that penalizes large deviations and encourages slight changes between every adjacent network[28]) with the alternating direction method of multipliers(ADMM) technique [3] to achieve a distributed and scalable solution. Estimating a series of time-evolving graphs using the joint graphical lasso leads to the following convex optimization problem:

$$\max \quad \log\det\theta_i - \operatorname{Tr}(S\theta_i) - \lambda\|\theta_i\|_1 + \beta\sum_{t=2}^{T}\Phi(\theta_i - \theta_{i-1}) \tag{2}$$

where θ_i is the network at every time stamp. α and $lambda$ control the sparsity and time consistency of dynamic networks, respectively. The convexity of both the lasso penalty and the Laplacian penalty guarantees global optimality without being plagued by local minima. The Laplacian quadratic penalty encourages smoothness across coefficients associated with the correlated predictors, which guarantees that the graphical models at adjacent time points are similar enough such that we can borrow information across time points by reweighting the observation[28], this phenomena corresponds to our resting-state fMRI time-series evolutionary pattern.

E. Alternating direction method of multipliers

The alternating direction method of multipliers (ADMM) is an algorithm that solves convex optimization problems by breaking them into smaller pieces, each of which are then easier to handle [3]. After rewriting the original equation, glasso problems is cast as a problem which is block separable. Using ADMM, equation 1 can be rewritten as

$$\begin{aligned} \max \quad & \log\det\theta - \operatorname{Tr}(S\theta) - \lambda\|\Omega\|_1 \\ \text{s.t.} \quad & \theta - \Omega = 0,\ \theta \in \mathbf{S}^p_{++} \end{aligned} \tag{3}$$

where Ω denotes consensus variables.

And the augmented Lagrangian is $\mathcal{L}(\theta,\Omega,U) = \sum_{i=1}^{T}(\log det\theta_i - \operatorname{Tr}(S\theta_i)) - \lambda\|\Omega\|_1 + \rho U^T(\theta - \Omega) + \frac{\rho}{2}\|\theta - \Omega\|_F^2$

The procedure can be summarized as algorithm 1.

Similarly, with the consensus variables $\Omega = \{\Omega_0,\Omega_1,\Omega_2\} = \{(\Omega_{1,0},\dots,\Omega_{T,0}), (\Omega_{1,1},\dots,\Omega_{T-1,1}),(\Omega_{2,2},\dots,\Omega_{T,2}\}$, the problem 2 can be rewritten as

$$\begin{aligned} \max \quad & \sum_{i=1}^{T}(\log\det\theta_i - \operatorname{Tr}(S\theta_i)) - \lambda\big\|\Omega_{i,0}\big\|_1 + \beta\sum_{i=2}^{T}\Phi(\Omega_{i,2} - \Omega_{i-1,1}) \\ \text{s.t.} \quad & \theta_i - \Omega_{i,0} = 0, \theta \in \mathbf{S}^p_{++} \quad \forall i = 1,\dots,T \\ & (\Omega_{i-1,1},\Omega_{i,2}) = (\theta_{i-1},\theta_i) \quad \forall i = 2,\dots,T \end{aligned} \tag{4}$$

By introducing the scaled dual variable $U = \{U_0,U_1,U_2\} = \{(U_{1,0},\dots,U_{T,0}), (U_{1,1},\dots,U_{T-1,1}),(U_{2,2},\dots,U_{T,2})\}$ and the ADMM penalty parameter $\rho \in \mathbb{R}^+$, the corresponding augmented Lagrangian [12] is:

$$\begin{aligned} \mathcal{L}(\theta,\Omega,U) = & \sum_{i=1}^{T}(\log det\theta_i - \operatorname{Tr}(S\theta_i)) - \lambda\big\|\Omega_{i,0}\big\|_1 \\ & + \beta\sum_{i=2}^{T}\Phi(\Omega_{i,2} - \Omega_{i-1,1}) \\ & + \rho/2\sum_{i=1}^{T}(\big\|\theta_i - \Omega_{i,0} + U_{i,0}\big\|_F^2) \\ & + \rho/2\sum_{i=2}^{T}(\big\|\theta_{i-1} - \Omega_{i-1,1} + U_{i-1,1}\big\|_F^2 \\ & - \big\|U_{i-1,1}\big\|_F^2 + \big\|\theta_i - \Omega_{i,2}\big\|_F^2) \end{aligned} \tag{5}$$

As described in algorithm 2, by separating equation 1 and 2 into two blocks of variables, θ and Ω , global convergence is guaranteed in our ADMM algorithm [3].

F. Problem scenario

The workflow of the proposed framework can be summarized in Figure ??. We formulate our problem as a binary graph classification problem, a supervised learning task: Given a static or dynamic fMRI network(s) as data input, which can be denoted as $\mathbf{x} = G$ or $\mathbf{x} = \mathcal{G} = \{G_1,G_2,\dots,G_n\}$ and corresponding label(s), we aim to assign proper labels to every graph. Specifically, given the following dataset, $\mathcal{D} = \big((\mathbf{G}_1,\mathbf{y}_1),(\mathbf{G}_2,\mathbf{y}_2),\dots,(\mathbf{G}_n,\mathbf{y}_n)\big)$ or $\big(((\mathbf{G}^{(1)},\mathbf{G}^{(2)},\dots,\mathbf{G}^{(T)})_1,\mathbf{y}_1),((\mathbf{G}^{(1)},\mathbf{G}^{(2)},\dots,\mathbf{G}^{(T)})_2,\mathbf{y}_2), \dots,((\mathbf{G}^{(1)},\mathbf{G}^{(2)},\dots,\mathbf{G}^{(T)})_n,\mathbf{y}_n)\big)$, we aim to find a non-linear and complex mapping f for the classification task: $f(\mathcal{D}) = \mathcal{Y}$ where $\mathcal{Y} = \{y_i\}_{i=1}^n$ and $y_i := \{-1,1\}$.

In data processing, to achieve interpretability, we convert the off-diagonal elements of the precision matrix into partial correlations $P = \Theta$, with elements denoted by r_{ij} and computed using

$$r_{ij} = -\frac{p_{ij}}{\sqrt{p_{ii}p_{jj}}} \tag{6}$$

Algorithm 1 ADMM for solving the graphical lasso

```
 1: Input ADMM penalty parameter $\rho$
 2: Output network $\theta$
 3: procedure GLASSO($\rho$)
 4:     Intialize $\theta^0$, $\Omega^0$, $U^0$
 5:     for k = 0 to convergence do
 6:         $\theta^{k+1} := \underset{\theta\in\mathbf{S}^P_{++}}{\operatorname{argmin}} - \log\det\theta + \rho/2\Big\|\theta - \Omega^t + \frac{1}{\rho}U^t + \frac{1}{\rho}S\Big\|_F^2$
 7:         $\Omega^{k+1} := \underset{\Omega_0,\Omega_1,\Omega_2}{\operatorname{argmin}} \lambda\|\Omega\|_1 + \rho/2\Big\|\theta^{t+1} - \Omega + \frac{1}{\rho}U^t\Big\|_F^2$
 8:         $U^{k+1} = U^k + \rho(\theta^{t+1} - \Omega^{t+1})$
 9:     end for
10:     return $\theta$
11: end procedure
```

Algorithm 2 ADMM for solving time-varying graphical lasso

```
 1: Input ADMM penalty parameter $\rho > 0$
 2: Output Time-varying network set $\Theta$
 3: procedure TVGL($\rho$)
 4:     Intialize $\theta^0$, $\Omega^0$, $U^0$
 5:     for k = 0 to convergence do
 6:         $\theta^{k+1} := \underset{\theta\in\mathbf{S}^P_{++}}{\operatorname{argmin}} \mathcal{L}(\theta,\Omega^k,U^k)$
 7:         $\Omega^{k+1} = \begin{bmatrix}\Omega_0^{k+1}\\ \Omega_1^{k+1}\\ \Omega_2^{k+1}\end{bmatrix} := \underset{\Omega_0,\Omega_1,\Omega_2}{\operatorname{argmin}} \mathcal{L}(\Theta^{k+1},\Omega,U^k)$
 8:         $U^{k+1} = \begin{bmatrix}U_0^{k+1}\\ U_1^{k+1}\\ U_2^{k+1}\end{bmatrix} := \begin{bmatrix}U_0^k\\ U_1^k\\ U_2^k\end{bmatrix} + \begin{bmatrix}\theta_0^{k+1} - \Omega_0^{k+1}\\ (\theta_1^{k+1},\dots,\theta_{T-1}^{k+1}) - \Omega_1^{k+1}\\ (\theta_2^{k+1},\dots,\theta_T^{k+1}) - \Omega_2^{k+1}\end{bmatrix}$
 9:     end for
10:     return $\Theta$
11: end procedure
```

After standardization, off-diagonal elements $\forall r_{ij}\ (i \neq j)$ are extracted and flattened into a single high-dimensional connectivity feature vector X^p or matrix $M^p = [X_1^p, X_2^p, \dots, X_k^p]$ as the input of our model, where $p = n(n-1)/2$.

Both simulation and experiments on real datasets show the advantage of our proposed approach.

G. Classifers

1) Support vector machine

Support vector machine(SVM) classifier is a popular supervised learning model due to its generalization capability and high classification accuracy. [27, 6]

In order to construct the optimum hyperplane that separates data points correctly, the functional margin is maximized such that the generalization error of the classifier is minimized.

Specifically, for any $i = 1, 2, \dots, N$ and $\xi_i \geq 0$, the optimization of margin to its support vector can be converted into a quadratic programming problem

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1 - \xi_i \end{aligned} \tag{7}$$

where ξ_i is the slack variable and parameter C represents misclassified sample of the corresponding sample of the margin hyperplane and cost of penalty, respectively.

For a non-linear classification problem, SVM can efficiently perform a non-linear $K(x_n, x_i)$ to maximum-margin hyperplanes, implicitly mapping their inputs into high-dimensional feature spaces by transforming function with the following dot product $\phi(x)$ such that data can be separated easily.

$$K(x_n, x_i) = \phi(x_n)\phi(x_i) \tag{8}$$

Therefore the hyperplane function can be written as

$$f(x_i) = \sum_{n=1}^{N} \alpha_n y_n K(x_n, x_i) + b \tag{9}$$

where x_n is support vector data. α_n is Lagrange multiplier and y_n denotes the label.

Here we choose sigmoid kernel due to its performance on our datasets, and the corresponding kernel function is

$$K(x_n, x_i) = tanh(\gamma(x_n, x_i) + r) \tag{10}$$

where C, γ, r are hyperparameters.

2) XGBoost

XGBoost is a scalable tree boosting system which achieves state-of-the-art results in machine learning challenges[5]. For the given dataset $\mathcal{G} = \{(G_i, y_i)\}$, we perform adaptive training, a.k.a boosting for classification. A tree ensemble can be denoted as follows:

$$\hat{y_i} = \phi(G_i) = \sum_{k=1}^{n} f_k(G_i) \tag{11}$$

where $f_k \in \mathcal{F}$, $\mathcal{F}$ is the space of classification and regression trees(CART)

To learn the tree ensembles in the model, a differentiable convex loss function l and penalty $\Omega = \lambda T + \frac{1}{2}\lambda\|w\|^2 (T$ is the number of leaves in the tree) are introduced, such that the following regularized objective is minimized:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(\hat{y}_i^{(t-1)} + f_t(G_i), y_i) + \Omega(f_t) \tag{12}$$

Additive training is performed by greedily adding the f_t that most improves the model. A second-order approximation can be used to quickly optimize the objective further and loss reduction after the split is used for evaluating the split candidates[5].

3) Long-short term memory

Rather than hand-crafting features, we leverage ideas from representation learning and use deep models to automatically learn relevant features from data[9].

Long short-term memory(LSTM) is a kind of recurrent neural network(RNN), which is widely used for sequential data like time series. An LSTM unit is composed of a cell, an input gate, an output gate and a forget gate.

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{13}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{14}$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{15}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{16}$$

$$h_t = o_t \odot \sigma_h(c_t) \tag{17}$$

where the operator $\odot$ denotes element-wise product and subscript t means time step. Variables can be summarized as: $x_t \in \mathbf{R}^d$ is the input vector to the LSTM unit. $f_t \in \mathbf{R}^h$, $i_t \in \mathbf{R}^h$, $o_t \in \mathbf{R}^h$ is the activation vector of forget gate, input gate, output gate respectively. $h_t \in \mathbf{R}^h$ is the hidden state vector and $c_t \in \mathbf{R}^h$ is the cell state vector. During training, the weight matrices $W \in \mathbf{R}^{h\times d}$, $U \in \mathbf{R}^{h\times h}$ and bias vector parameters $b \in \mathbf{R}^h$ are learned, where d and h refer to the number of input features and hidden units, respectively.

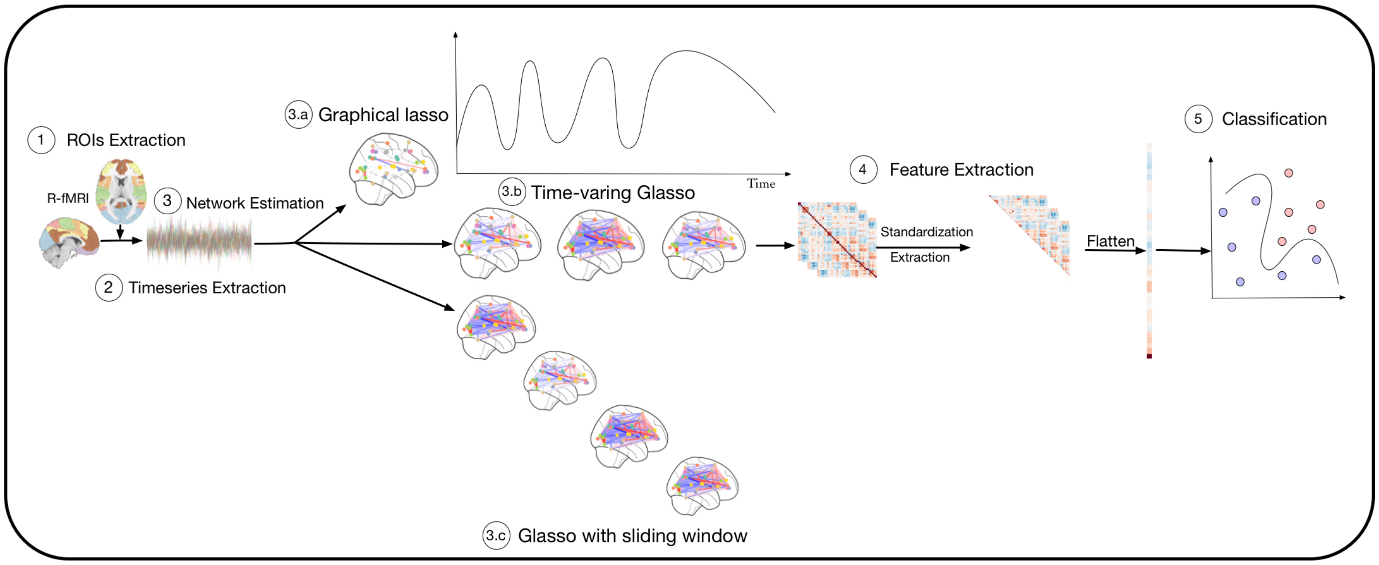
The benefit of using LSTMs for sequence classification is that they can learn from the raw time series data directly, and in turn do not require domain expertise to manually engineer input features. The model can learn an internal representation of the time series data and ideally achieve comparable performance to models fit on a version of the dataset with engineered features.

H. Parameter tuning

Due to the data scarcity, 10-fold cross validation (CV) and leave-one-out cross validation (LOOCV) are employed in the ADNI and Dyslexia dataset, respectively.

Larger λ values enforce a network structure with many zeros, while smaller values yield denser matrices. A sparse graph effectively limits the degree of freedom of the model, which makes structure recovery possible given small sample size. For the sliding window, using a wider bandwidth (window-length) parameter d provides more samples to estimate the network, but sharp changes are more likely to miss the network. In

Figure 1 | **Framework of our method.**



contrast, using a narrower bandwidth parameter makes the estimate more sensitive to sharp changes, but this also leads to larger variance due to the reduced sample size [24].

We employ the Akaike information criterion (AIC) for choosing λ and β that trades off between the fit to the data and the model complexity [21]. The λ chosen varies across subjects. λ and β empirically are around 0.01 and 0.1 [4].

For the parameters of classifiers, rather than search over a grid of parameters, we use both Bayesian optimization [23] and cross-validation for tuning. Particularly, we use 10-fold cross validation for parameter choosing in the ADNI dataset and LOOCV for the dyslexia dataset. Bayesian optimization, trading off exploration and exploitation, is proved to be an automatic approach that can optimize the performance of many learning algorithms to the problem at hand effectively [23]. More specifically, it works by treating the objective function as a random function and place a prior over it, which demonstrates our belief about the behavior of the function. After gathering the function evaluations, which are treated as data, the prior is updated to form the posterior distribution over the objective function. The posterior distribution is used to construct an acquisition function that determines the next query point. As the number of observations grows, the posterior distribution improves, and the algorithm becomes more certain of which regions in parameter space are worth exploring and which are not. At each step, a Gaussian Process is fitted to the known samples, and the posterior distribution, combined with an exploration strategy is used to determine the next point that should be explored.

III. fMRI DATA

We conduct experiments on the two fMRI datasets and the result shows the advantages of our proposed approach. For static network estimation, glasso is employed on the whole timeseires. On the other hand, a window with length $d = 40$ is specified in the dynamic cases.

Table I Classification results on ADNI dataset

Cases	Static	Dynamic1	Dynamic2	Dynamic3	TVGL
XGBoost	58,93	66,07	66,07	76,79	67,86
SVM	57,14	57,14	57,14	57,14	57,14
RF	67,89	66,07	57,14	73,21	64,29

In TVGL, we use the TVGL Python solver [12] on top of SnapVX, an open-source convex optimization package. Our solver takes as inputs the multivariate time series and the regularization parameters, and it returns the time-varying network. In both static and dynamic cases, we follow the strategy described in the method section. Specifically, for example, in the static case, we extract as input off-diagonal elements, a single 55 dimensional vector, from the Dyslexia dataset with 11 dimension.

A. *dataset overview*

The fMRI time series datasets include two parts: the small Dyslexia dataset contains 23 subjects(9 controls and 14 patients) with 11 ROIs(features) and 200 time points, while large ADNI dataset contains 56 subjects(30 controls and 26 patients) with 10 ROIs(features) and 135 time points. The ROIs chosen in both datasets are from AAL atlas.

The ADNI dataset used this paper were obtained from the ADNI database [1] We preprocessed the ADNI data using both FSL and AFNI. The detailed steps are summarized in [13].

IV. RESULTS

The experimental results show that our proposed dynamic glasso methods outperform the static one with respect to the classification accuracy.

[1](http://adni.loni.usc.edu)

Table II Classification results on Dyslexia dataset

Cases	Static	Dynamic1	Dynamic2	Dynamic3	TVGL
XGBoost	60,42	78,13	60,42	60,42	60,42
SVM	60,42	60,42	60,42	60,42	60,42
RF	63,54	76,04	70,83	70,83	70,83

A. Raw time series results

We perform minmax scale on the time series of every subject and then feed them into the LSTM unit. The LSTM achieves 58% classification accuracy in the case of 10-fold cross-validation on the ADNI dataset.

Due to the data scarcity, LSTM performs badly despite we reduce the number of hidden units and use regularization techniques like dropout[26].

B. Static results

We use LOOCV to evaluate our performance. Specifically, we use 22 training set to

For parameter choosing, like sparsity parameter α, we specify a list from 0.01 to 1.00 with step size equals to 0.01 and further use cross validation to choose among these values.

C. Dynamic results

In dynamic cases, we stack multiple consistent networks per subject into a feature vector for training Specifically, network size of 5, 10, 20 is estimated and the corresponding window length for Dyslexia and ADNI dataset is $d = 40$ and step size is 0, 5, 10, leading to dynamic network sets with the amount of 2, 9, 19 and 3, 10, 19 respectively.

V. Conclusion

In this work, we investigate the effectiveness of static and dynamic network estimation techniques in neuropsychiatric disease classification problems. We compare three kinds of network estimation methods in both static and dynamic cases with respect to classification accuracy. Experimental results prove that classification on dynamic networks consistently outperforms static ones. Our method also provides efficiency guarantees on estimations using a decentralized optimization technique, which also applies to a larger scale optimization problem.

References

[1] Amr Ahmed and Eric P Xing. “Recovering time-varying networks of dependencies in social and biological studies”. In: *Proceedings of the National Academy of Sciences* 106.29 (2009), pp. 11878–11883.

[2] Elena A Allen et al. “Tracking whole-brain connectivity dynamics in the resting state”. In: *Cerebral cortex* 24.3 (2014), pp. 663–676.

[3] Stephen Boyd et al. “Distributed optimization and statistical learning via the alternating direction method of multipliers”. In: *Foundations and Trends® in Machine learning* 3.1 (2011), pp. 1–122.

[4] Biao Cai et al. “Capturing Dynamic Connectivity from Resting State fMRI using Time-Varying Graphical Lasso”. In: *IEEE Transactions on Biomedical Engineering* (2018).

[5] Tianqi Chen and Carlos Guestrin. “Xgboost: A scalable tree boosting system”. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM. 2016, pp. 785–794.

[6] Corinna Cortes and Vladimir Vapnik. “Support-vector networks”. In: *Machine learning* 20.3 (1995), pp. 273–297.

[7] Patrick Danaher, Pei Wang, and Daniela M Witten. “The joint graphical lasso for inverse covariance estimation across multiple classes”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 76.2 (2014), pp. 373–397.

[8] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. “Sparse inverse covariance estimation with the graphical lasso”. In: *Biostatistics* 9.3 (2008), pp. 432–441.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[10] Aditya Grover and Jure Leskovec. “node2vec: Scalable feature learning for networks”. In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2016, pp. 855–864.

[11] Hao Guo et al. “Alzheimer classification using a minimum spanning tree of high-order functional network on fMRI dataset”. In: *Frontiers in neuroscience* 11 (2017), p. 639.

[12] David Hallac et al. “Network inference via the time-varying graphical lasso”. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2017, pp. 205–213.

[13] Brian Hart et al. “A longitudinal model for functional connectivity networks using resting-state fMRI”. In: *NeuroImage* 178 (2018), pp. 687–701.

[14] Mladen Kolar et al. “Estimating time-varying networks”. In: *The Annals of Applied Statistics* 4.1 (2010), pp. 94–123.

[15] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[16] Lu Meng and Jing Xiang. “Brain Network Analysis and Classification Based on Convolutional Neural Network”. In: *Frontiers in computational neuroscience* 12 (2018).

[17] Regina J Meszlényi, Krisztian Buza, and Zoltán Vidnyánszky. “Resting state fMRI functional connectivity-based classification using a convolutional neural network architecture”. In: *Frontiers in neuroinformatics* 11 (2017), p. 61.

[18] Karthik Mohan et al. “Structured learning of Gaussian graphical models”. In: *Advances in neural information processing systems*. 2012, pp. 620–628.

[19] Atif Riaz et al. “Deep fMRI: An end-to-end deep network for classification of fMRI data”. In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. IEEE. 2018, pp. 1419–1422.

[20] Maria J Rosa et al. "Sparse network-based models for patient classification using fMRI". In: *Neuroimage* 105 (2015), pp. 493–506.

[21] Yosiyuki Sakamoto, Makio Ishiguro, and Genshiro Kitagawa. "Akaike information criterion statistics". In: *Dordrecht, The Netherlands: D. Reidel* 81 (1986).

[22] Stephen M Smith et al. "Network modelling methods for FMRI". In: *Neuroimage* 54.2 (2011), pp. 875–891.

[23] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. "Practical bayesian optimization of machine learning algorithms". In: *Advances in neural information processing systems*. 2012, pp. 2951–2959.

[24] Le Song, Mladen Kolar, and Eric P Xing. "KELLER: estimating time-varying interactions between genes". In: *Bioinformatics* 25.12 (2009), pp. i128–i136.

[25] Le Song, Mladen Kolar, and Eric P Xing. "Time-varying dynamic bayesian networks". In: *Advances in neural information processing systems*. 2009, pp. 1732–1740.

[26] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[27] Johan AK Suykens and Joos Vandewalle. "Least squares support vector machine classifiers". In: *Neural processing letters* 9.3 (1999), pp. 293–300.

[28] Kilian Q Weinberger et al. "Graph Laplacian regularization for large-scale semidefinite programming". In: *Advances in neural information processing systems*. 2007, pp. 1489–1496.

[29] Dong Wen et al. "Deep learning methods to process fMRI data and their application in the diagnosis of cognitive impairment: a brief overview and our opinion". In: *Frontiers in neuroinformatics* 12 (2018), p. 23.