Joint Embedding of Graphs

Shangsi Wang, Joshua T. Vogelstein, Carey E. Priebe

Abstract—Feature extraction and dimension reduction for networks is essential in a wide variety of domains. Efficiently and accurately learning features for multiple graphs has important applications in statistical inference on graphs. In this manuscript, we propose a method to jointly embed multiple undirected graphs. Given a set of graphs, the joint embedding method identifies a linear subspace spanned by rank one symmetric matrices and projects adjacency matrices of graphs into this subspace. The projection coefficients can be treated as features of the graphs. We also propose a random graph model which can be used to model multiple graphs. We show through theory and numerical experiments that under the model the joint embedding method produces estimates of parameters with small errors. Via simulation experiments, we demonstrate that the joint embedding method produces features which lead to state of the art performance in classifying graphs. Applying the joint embedding method to human brain graphs, we find it extracting interpretable features which can be used to predict individual composite creativity index.

Index Terms—graphs, embedding, feature extraction, statistical inference

1 Introduction

In many problems arising in science and engineering, graphs arise naturally as data structures to capture complex relationships between a set of objects. Graphs have been used in various application domains as diverse as social networks [1], internet mapping [2], brain connectomics [3], political voting networks [4], and many others. These graphs are naturally high dimensional objects with complicated topological structure which makes graph clustering and classification a challenge to traditional machine learning algorithms. Therefore, feature extraction and dimension reduction techniques are helpful in the applications of graph data. In this paper, we propose an algorithm to jointly embed multiple graphs into low dimensional space. We demonstrate through theory and experiments that the joint embedding algorithm produces features which lead to state of the art performance for subsequent inference tasks on graphs.

There exist a few unsupervised approaches to extract features from graphs. First, classical principal component analysis can be applied by treating each edge of graphs as a raw feature vector [5]. This approach produces features which are linear combinations of edges, but it ignores the topological structure of graphs and features extracted are not easily interpretable. Second, features can be extracted by computing summary topological and label statistics from graphs [6]. These statistics commonly include number of edges, number of triangles, average clustering coefficient, maximum effective eccentricity, etc. In general, it is hard to know what intrinsic statistics to compute *a priori* and computing some statistics can be computational expensive. Also, many frequent subgraph mining algorithms are developed [7]. For example, the fast frequent subgraph mining algorithm can identify all connected subgraphs that occur in a large fraction of graphs in a graph data set [8]. Furthermore, spectral feature selection can also be applied to graphs. It treats each graph as a node and constructs an object graph based on a similarity measure. Features are computed through the spectral decomposition of this object graph [9].

Adjacency Spectral Embedding (ASE) and Laplacian Eigenmap (LE) are proposed to embed a single graph observation [10] [11]. The inference task considered in these papers is learning the block structure of the graph or clustering vertices. Given a set of graphs $\{G_i = (V_i, E_i)\}_{i=1}^m$, ASE and LE need to embed adjacency matrix or Laplacian matrix of G_i individually, and there is no easy way to combine multiple embeddings. The joint embedding method considers the set of graphs together. It takes a matrix factorization approach to extract features for multiple graphs. The algorithm manages to simultaneously identify a set of rank one matrices and project adjacency matrices into the linear subspace spanned by this set of matrices. The joint embedding can be understood as a generalization of ASE for multiple graphs. We demonstrate through simulation experiments that the joint embedding algorithm extracts features which lead to good performance for a variety of inference tasks. In the next section, we review some random graph models and present a model for generating multiple random graphs. In Section 3, we define the joint embedding of graphs and present an algorithm to compute it. In Section 4, we perform some theoretical analyses of our joint embedding. The theoretical results and a real data experiment are explored in Section 5. We conclude the paper with a brief discussion of implications and possible future work.

2 Setting

In this paper, we focus on embedding unweighted and undirected graphs for simplicity, although the joint embedding

- *Shangsi Wang and Carey Priebe are with the Department of Applied Mathematics and Statistics, Johns Hopkins University. E-mail: swang127@jhu.edu, cep@jhu.edu*
- *Joshua Vogelstein is with the Department of Biomedical Engineering and Institute for Computational Medicine, Johns Hopkins University E-mail: jovo@jhu.edu*

algorithm works on weighted and directed graphs as well. Let $\{G_i = (V_i, E_i)\}_{i=1}^m$ be m graphs each with n vertices and A_i be the adjacency matrix of graph G_i. We require that the vertices in these graphs are matched, which means that all the graphs have a common vertex set V. The joint embedding algorithm embeds all G_is simultaneously into $\mathbb{R}^d$ and represents G_i by a vector $\lambda_i \in \mathbb{R}^d$. Before discussing the joint embedding algorithm, we first introduce three random graph models.

Definition (Stochastic Block Model (SBM)) Let π be a prior probability vector for block membership which lies in the unit $K-1$-simplex. Denote by $\tau = (\tau_1, \tau_2, ..., \tau_n) \in [K]^n$ the block membership vector, where τ is a multinomial sequence with probability vector π. Denote by $B \in [0,1]^{K \times K}$ the block connectivity probability matrix. Suppose A is a random adjacency matrix given by,

$$P(A|\tau, B) = \prod_{i<j} B_{\tau_s,\tau_t}^{A_{s,t}} (1 - B_{\tau_s,\tau_t})^{(1-A_{s,t})}$$

Then, we say A is an adjacency matrix of a K-block stochastic block model graph, and we write $A \sim SBM(\pi, B)$. We may also treat τ as the parameter of interest, in this case we write $A \sim SBM(\tau, B)$.

SBM is a widely used model to study community structure of a graph [12] [13]. Next, we recall the notion of a random dot product graph [14].

Definition (Random Dot Product Graph (RDPG)) Let F be a distribution on a set $\mathcal{X} \in \mathbb{R}^d$ satisfying $x^T y \in [0,1]$ for all $x, y \in \mathcal{X}$. Let $X = [X_1^T, X_2^T, ..., X_n^T] \in \mathbb{R}^{n \times d}$. We say $(X, A) \sim RDPG(F)$, if the X_i are independent and identically distributed according to F, and conditioned on X, the A_{ij} are independent Bernoulli random variables,

$$A_{st} \sim Bernoulli(X_s^T X_t).$$

Alternatively,

$$P(A|X) = \prod_{s<t} X_s^T X_t^{A_{st}} (1 - X_s^T X_t)^{1-A_{st}}.$$

Also, we define $P := XX^T$ to be edge probability matrix. When we are more interested in estimating latent positions X, we treat X as parameters and write $A \sim RDPG(X)$.

The random dot product graph model is a convenient model which is designed to capture more complex structures than SBM. The adjacency spectral embedding of its adjacency matrix is well studied [15]. SBM and RDPG are models for a single graph. We propose a model for multiple random graphs which generalizes SBM and RDPG.

Definition (Multiple Random Eigen Graphs (MREG)) Let $\{h_k\}_{k=1}^d$ be a set of norm-1 vectors in $\mathbb{R}^n$, and F be a distribution on a set $\mathcal{X} \in \mathbb{R}^d$, satisfying $\sum_{k=1}^d \lambda[k] h_k h_k^T \in [0,1]^{n \times n}$ for all $\lambda \in \mathcal{X}$, where $\lambda[k]$ is the kth entry of vector λ. We say (λ_i, A_i) follows a m-graph d-dimensional multiple random eigen graphs model and write

$$\{(\lambda_i, A_i)\}_{i=1}^m \sim MREG(F, h_1, ..., h_d)$$

if the λ_i are independent and identically distributed according to F, and conditioned on λ_i, the entries of A_i are independent Bernoulli random variables,

$$A_i[s,t] \sim Bernoulli(\sum_{k=1}^d \lambda_i[k] h_k[s] h_k[t]).$$

We call $P_i := \sum_{k=1}^d \lambda_i[k] h_k h_k^T$ the edge probability matrix for graph i. In cases that we are more interested in $\{\lambda_i\}_{i=1}^m$, we treat them as parameters and write

$$\{A_i\}_{i=1}^m \sim MREG(\lambda_1, ..., \lambda_m, h_1, ..., h_d).$$

In MREG, we allow self loops to happen. This is mainly for theoretical convenience. If we ignore the self loops, we have the following relationships between three random graph models. If an adjacency matrix $A \sim SBM(\pi, B)$ and the block connectivity matrix B is positive semi-definite, A can also be written as a $RDPG(F)$ with F being a finite mixture of point masses. If an adjacency matrix $A \sim RDPG(X)$, then it is also a 1-graph $MREG(\lambda_1, h_1, ..., h_d)$ with h_k being the normalized kth column of X and λ_1 being the vector containing the squared norms of columns of X. However, a 1-graph $MREG(\lambda_1, h_1, ..., h_d)$ is not necessarily an RDPG graph since λ_1 could contain negative entries which may result in an indefinite edge probability matrix. If m RDPG graphs have edge probability matrices sharing the same eigenspace, they also follow a m-graph MREG model.

3 METHODOLOGY

3.1 Joint Embedding of Graphs

The joint embedding method considers a collection of vertex-aligned graphs, and estimates a common embedding space across all graphs and a loading for each graph. Specifically, it simultaneously identifies a subspace spanned by a set of rank one symmetric matrices and projects each adjacency matrix A_i into the subspace. The coefficients obtained by projecting A_i are denoted by $\hat{\lambda}_i \in \mathbb{R}^d$ which is called the loading for graph i. To estimate rank one symmetric matrices and loadings for graphs, the algorithm minimizes the sum of squared Frobenius distances between adjacency matrices and their projections as described below.

Definition (Joint Embedding of Graphs) Given m graphs $\{G_i\}_{i=1}^m$ with A_i being the corresponding adjacency matrix, the d-dimensional joint embedding of graphs $\{G_i\}_{i=1}^m$ is given by

$$(\hat{\lambda}_1, ..., \hat{\lambda}_m, \hat{h}_1, ..., \hat{h}_d) = \underset{\lambda_i, \|h_k\|=1}{\operatorname{argmin}} \sum_{i=1}^m \|A_i - \sum_{k=1}^d \lambda_i[k] h_k h_k^T\|^2. \quad (1)$$

Here, $\|\cdot\|$ denotes the Frobenius norm and $\lambda_i[k]$ is the kth entry of vector λ_i.

To make sure that model is identifiable, we put an extra requirement on the norm of h_k. To ease our notation, we introduce two matrices $\Lambda \in \mathbb{R}^{m \times d}$ and $H \in \mathbb{R}^{n \times d}$, where λ_i is the ith row of Λ and h_k is the kth row of H; that is, $\Lambda = [\lambda_1^T, ..., \lambda_m^T]$ and $H = [h_1, ..., h_d]$. We can rewrite equation (1) using Λ and H as

$$(\hat{\Lambda}, \hat{H}) = \underset{\Lambda, \|h_k\|=1}{\operatorname{argmin}} \sum_{i=1}^m \|A_i - \sum_{k=1}^d \Lambda_{ik} h_k h_k^T\|^2.$$

We denote the function on the left hand side of the equation by $f(\Lambda, H)$ which is explicitly a function of λ_is and h_ks. There are several alternative ways to formulate the problem. If we convert λ_i into a diagonal matrix $D_i \in \mathbb{R}^{d\times d}$ by putting entries of λ_i on the diagonal of D_i, then solving equation (1) is equivalent to solving

$$\begin{aligned} \underset{D_i, \|h_k\|=1}{\operatorname{argmin}} \quad & \sum_{i=1}^{m} \|A_i - H D_i H^T\|^2 \\ \text{subject to} \quad & D_i \text{ being diagonal.} \end{aligned}$$

We may also view (1) as a tensor factorization problem. If $\{A_i\}_{i=1}^m$ are stacked in a 3-D array $\mathbf{A} \in \mathbb{R}^{m\times n\times n}$, then solving equation (1) is also equivalent to

$$\underset{\Lambda, \|h_k\|=1}{\operatorname{argmin}} \|\mathbf{A} - \sum_{k=1}^{d} \Lambda_{*k} \otimes h_k \otimes h_k\|^2$$

where $\otimes$ denotes the tensor product and Λ_{*k} is the kth column of Λ.

The optimization problem in equation (1) is similar to principal component analysis in the sense of minimizing squared reconstruction error to recover loadings and components [5]. However, there are extra symmetries and rank constraints on the components. Similar optimization problems are also considered in the simultaneous diagonalization literature [16] [17]. The difference is that we are estimating an n-by-d matrix H by minimizing reconstruction error instead of finding a n-by-n non-singular matrix by trying to simultaneously diagonalize all matrices. The problem in equation (1) has considerably fewer parameters to optimize which makes it more stable and applicable with n being moderately large. In case of embedding only one graph, the joint embedding is equivalent to the Adjacency Spectral Embedding solved by singular value decomposition [10]. Next, we describe an algorithm to optimize the objective function $f(\Lambda, H)$.

3.2 Alternating Descent Algorithm

The joint embedding of $\{G_i\}_{i=1}^m$ is estimated by solving the optimization problem in equation (1). There are a few methods proposed to solve similar problems. Carroll and Chang [18] propose to use an alternating minimization method that ignores symmetry. The hope is that the algorithm will converge to a symmetric solution itself due to symmetry in data. Gradient approaches have also been considered for similar problems [19] [20]. We develop an alternating descent algorithm to minimize $f(\Lambda, H)$ which combines ideas from both approaches. The algorithm iteratively updates one of Λ and H while treating the other parameter as fixed. We will see that optimizing Λ when fixing H is easy, since it is essentially a least squares problem. However, optimizing H when fixing Λ is hard due to the fact that the problem is non-convex and there is no closed form solution available. In this case, we utilize gradient information and take an Armijo line search strategy to update H [21].

Instead of optimizing all columns Λ and H simultaneously, we consider a greedy algorithm which solves the optimization problem by only considering one column of Λ and H at a time. Specifically, the algorithm fixes all estimates for the first $k_0 - 1$ columns of Λ and H at iteration k_0, and then the objective function is minimized by searching through only the kth column of Λ and H. That is,

$$\begin{aligned} (\hat{\Lambda}_{*k_0}, \hat{h}_{k_0}) = & \underset{\Lambda_{*k_0}, \|h_{k_0}\|=1}{\operatorname{argmin}} \\ & \sum_{i=1}^{m} \|A_i - \sum_{k=1}^{k_0-1} \hat{\Lambda}_{ik} \hat{h}_k \hat{h}_k^T - \Lambda_{ik_0} h_{k_0} h_{k_0}^T\|^2. \end{aligned} \quad (2)$$

Let $f(\Lambda_{*k_0}, h_{k_0})$ denote the sum on the left hand side of the equation. To compute a d-dimensional joint embedding $(\hat{\Lambda}, \hat{H})$, the algorithm iteratively solves the one dimensional optimization problem above by letting k_0 vary from 1 to d.

There are a few advantages in iteratively solving one dimensional problems. First, there are fewer parameters to fit at each iteration, since the algorithm are only allowed to vary Λ_{*k_0} and h_{k_0} at iteration k_0. This makes initialization and optimization steps much easier compared to optimizing all columns of H simultaneously. Second, it implicitly enforces an ordering on the columns of H. This ordering allows us to select the top few columns of Λ and H in cases where we want to perform model selection. Third, it allows incremental computation. If we compute d and d' dimensional joint embeddings, the first $\min(d, d')$ columns of $\hat{\Lambda}$ and $\hat{H}$ will be the same. Finally, based on numerical experiments, the difference between optimizing iteratively and optimizing all the parameters when d is small is negligible; however, the iterative algorithm yields a slightly smaller objective function when d is large. The disadvantage of optimizing each column separately is that the algorithm is more likely to end up at a local minimum when the objective function is structured not in favor of embedding iteratively. In practice, this problem can be mitigated by running the joint embedding algorithm several times with random initializations.

To find Λ_{*k_0} and h_{k_0} in equation2, the algorithm needs to evaluate two derivatives: $\frac{\partial f}{\partial h_{k_0}}$ and $\frac{\partial f}{\partial \Lambda_{ik_0}}$. Denote by R_{ik_0} the residual matrix after iteration $k_0 - 1$ which is $A_i - \sum_{k=1}^{k_0-1} \hat{\Lambda}_{ik} \hat{h}_k \hat{h}_k^T$. The gradient of the objective function with respect to h_{k_0} is given by

$$\frac{\partial f}{\partial h_{k_0}} = -4 \sum_{i=1}^{m} \Lambda_{ik_0} (R_{ik} - \Lambda_{ik_0} h_{k_0} h_{k_0}^T) h_{k_0}. \quad (3)$$

The derivative of the objective function with respect to Λ_{ik_0} is given by

$$\frac{\partial f}{\partial \Lambda_{ik_0}} = -2 \langle R_{ik} - \Lambda_{ik_0} h_{k_0} h_{k_0}^T, h_{k_0} h_{k_0}^T \rangle.$$

Setting the derivative to 0 yields,

$$\hat{\Lambda}_{ik_0} = \langle R_{ik}, h_{k_0} h_{k_0}^T \rangle. \quad (4)$$

Algorithm 1 describes the general procedure to compute the d-dimensional joint embedding of graphs $\{G_i\}_{i=1}^m$. After the algorithm finishes, rows of $\hat{\Lambda}$ denoted by $\{\hat{\lambda}_i\}_{i=1}^m$ can be treated as estimates of $\{\lambda_i\}_{i=1}^m$ or features for graphs. Columns of $\hat{H}$ denoted by $\{\hat{h}_k\}_{k=1}^d$ are estimates of $\{h_k\}_{k=1}^d$

Algorithm 1 Joint Embedding Algorithm

procedure FIND JOINT EMBEDDING $\hat{\Lambda}, \hat{H}$ OF $\{A_i\}_{i=1}^m$
  Set residuals: $R_{i1} = A_i$
  for $k = 1 : d$ **do**
    Initialize h_k and Λ_{*k}
    while not convergent **do**
      Fixing Λ_{*k}, update h_k by gradient descent (3)
      Project h_k back to the unit sphere
      Fixing h_k, update Λ_{*k} by (4)
      Compute objective $\sum_{i=1}^{m} \|R_{ik} - \Lambda_{ik} h_k h_k^T\|^2$
    end while
    Update residuals: $R_{i(k+1)} = R_{ik} - \Lambda_{ik} h_k h_k^T$
  end for
  Output $\hat{\Lambda} = [\Lambda_{*1}, ..., \Lambda_{*d}]$ and $\hat{H} = [h_1, ..., h_d]$
end procedure

If a new graph G is observed with adjacency matrix A, we can project A into linear space spanned by $\{\hat{h}_k \hat{h}_k^T\}_{k=1}^d$ to obtain features for the graph. The optimization algorithm described above may not be the fastest approach to solving the problem; however, numerical optimization is not the focus of this paper. Based on results from numerical applications, our approach works well in estimating parameters and extracting features for subsequent statistical inference.

3.3 Variations

The joint embedding algorithm described above can be modified to accommodate several settings.

Variation 1. When all graphs come from the same distribution, we can force estimated loadings $\hat{\lambda}_i$ to be equal across all graphs. This is useful when the primary inference task is to extract features for vertices. Since all graphs share the same loadings, with slightly abusing notations, let Λ be a vector in $\mathbb{R}^d$ and the optimization problem becomes

$$(\hat{\Lambda}, \hat{H}) = \underset{\Lambda, \|h_k\|=1}{\operatorname{argmin}} \sum_{i=1}^{m} \|A_i - \sum_{k=1}^{d} \Lambda_k h_k h_k^T\|^2.$$

In this case, the optimization problem can be solved exactly by finding singular value decomposition of the average adjacency matrix $\frac{1}{m} \sum_{i=1}^{m} A_i$.

Variation 2. When there is a discrete label $y_i \in \mathbb{Y}$ associated with G_i available, we may require all loadings $\hat{\lambda}_i$ to be equal within class. If we let $\Lambda \in \mathbb{R}^{|\mathbb{Y}| \times d}$, the optimization problem becomes

$$(\hat{\Lambda}, \hat{H}) = \underset{\Lambda, \|h_k\|=1}{\operatorname{argmin}} \sum_{i=1}^{m} \|A_i - \sum_{k=1}^{d} \Lambda_{y_i k} h_k h_k^T\|^2.$$

In this case, when updating Λ as in equation 4, the algorithm should average $\Lambda_{y_i k}$ within the same class.

Variation 3. In some applications, we may require all Λ_{ik} to be greater than 0 as in non-negative matrix factorization. One advantage of this constraint is that graph G_i may be automatically clustered based on the largest entry of $\hat{\lambda}_i$. In this case, the optimization problem is

$$(\hat{\Lambda}, \hat{H}) = \underset{\Lambda \geq 0, \|h_k\|=1}{\operatorname{argmin}} \sum_{i=1}^{m} \|A_i - \sum_{k=1}^{d} \Lambda_{ik} h_k h_k^T\|^2.$$

To guarantee nonnegativity, the algorithm should use nonnegative least squares in updating Λ [22]. Next, we discuss some theoretical properties of joint embedding when treated as a parameter estimation procedure for MREG model.

4 THEORY

In this section, we consider a simple setting where graphs follow 1-dimensional MREG model, that is $\{(\lambda_i, A_i)\}_{i=1}^m \sim MREG(F, h_1)$. Under this MREG model, we can understand joint embedding of graphs as estimators for parameters of the model. Specifically, $\hat{\lambda}_i$ and $\hat{h}_1$ are estimates of λ_i and h. We prove the following two theorems concerning the asymptotic behavior of estimator $\hat{h}_1$ produced by joint embedding. Let $\hat{h}_1^m$ denote the estimates based on m graphs and define functions ρ, D_m and D as below:

$$\rho(A_i, h) = \|A_i - \langle A_i, hh^T \rangle hh^T\|^2,$$

$$D_m(h, h_1) = \frac{1}{m} \sum_{i=1}^{m} \rho(A_i, h),$$

$$D(h, h_1) = E(\rho(A_i, h)).$$

By equation (1), we have

$$\hat{h}_1^m = \underset{\|h\|=1}{\operatorname{argmin}} \underset{\lambda_i}{\operatorname{argmin}} \sum_{i=1}^{m} \|A_i - \lambda_i hh^T\|.$$

By equation (4), we have

$$\langle A_i, hh^T \rangle = \underset{\lambda_i}{\operatorname{argmin}} \sum_{i=1}^{m} \|A_i - \lambda_i hh^T\|.$$

Therefore,

$$\hat{h}_1^m = \underset{\|h\|=1}{\operatorname{argmin}} D_m(h, h_1).$$

The first theorem states that $\hat{h}_1^m$ converges almost surely to the global minimum of $D(h, h_1)$, given that the global minimum is unique.

Theorem 4.1. If $D(h, h_1)$ has a unique global minimum at h', then $\hat{h}_1^m$ converges almost surely to h' as m goes to infinity. That is,

$$\hat{h}_1^m \overset{a.s.}{\to} h'.$$

In Theorem 4.1, we require h' to be the unique global minimizer of $D(h, h_1)$. However, the global optimizer is definitely not unique due to the symmetry up to sign flip of h, that is $D(h, h_1) = D(-h, h_1)$ for any h. This problem can be addressed by forcing an orientation of $\hat{h}_1^m$ or stating that the convergence is up to a sign flip. It is also possible that there are multiple global minimizers of $D(h, h_1)$ which are not sign flips of each other. In this case, Theorem 4.1 does not apply. We are currently only certain that when all graphs are from the Erdos-Renyi random graph model, the global minimizer is unique up to a sign flip. The next theorem concerns the asymptotic bias of $\hat{h}_1^m$.

Theorem 4.2. If h' is a minimizer of $D(h, h_1)$, then

$$\|h' - h_1\| \leq \frac{2E(\lambda_i)}{E(\lambda_i^2)(h_1^T h')^2}.$$

To see an application of Theorem 4.2, we consider the case in which all graphs are Erdos-Renyi graphs with 100 vertices

and edge probability 0.5. Under this setting, Theorem 4.2 implies $\|h' - h_1\| \in [0, 0.04] \cup [1.28, 1.52]$. The second interval is disturbing. It is due to the fact that when $h_1^T h'$ is small the bound is useless. We provide some insights why the second interval is there and how we can get rid of it with additional assumptions. In the proof of Theorem 4.2, we show that the global optimizer h' satisfies

$$h' = \underset{\|h\|=1}{\operatorname{argmax}} E(\langle A_i, hh^T\rangle^2).$$

If we take a closer look at $E(\langle A_i, hh^T\rangle^2)$, we see

$$\begin{aligned} E(\langle A_i, hh^T\rangle^2) &= E(\langle P_i, hh^T\rangle^2) + E(\langle A_i - P_i, hh^T\rangle^2) \\ &= E(\lambda_i^2)(h_1^T h)^4 + E((h^T(A_i - P_i)h)^2). \end{aligned}$$

Therefore,

$$h' = \underset{\|h\|=1}{\operatorname{argmax}} E(\lambda_i^2)(h_1^T h)^4 + E((h^T(A_i - P_i)h)^2).$$

We can see that $E(\lambda_i^2)(h_1^T h)^4$ is maximized when $h = h_1$; however, the noise term $E((h^T(A_i - P_i)h)^2)$ is generally not maximized at $h = h_1$. If we assume n is large, we can apply a concentration inequality to $(h^T(A_i - P_i)h)^2$ and have an upper bound on $E((h^T(A_i - P_i)h)^2)$. If we further assume A_i is not too sparse, that is $E(\lambda_i^2)$ grows with n fast enough, then the sum of these two terms is dominated by the first term. This provides a way to have a lower bound on $h_1^T h'$. We may then replace the denominator of the bound in Theorem 4.2 by the lower bound. In general, if n is small, the noise term may cause h' to differ from h_1 by a significant amount. In this paper, we focus on the case that n is fixed. The case that n goes to infinity for random dot product graphs is considered in [23].

The two theorems above concern only the estimation of h_1, but not λ_i. Based on equation (4), we estimate λ_i by

$$\hat{\lambda}_i^m = \langle A_i, \hat{h}_1^m \hat{h}_1^{mT}\rangle.$$

If n is fixed, it is impossible to prove any consistency result on estimation of λ_i due to the fact we observe only one finite graph A_i associated with λ_i. When m goes to infinity, we can apply Theorem 4.1,

$$\hat{\lambda}_i^m = \langle A_i, \hat{h}_1^m \hat{h}_1^{mT}\rangle \xrightarrow{a.s.} \langle A_i, h'h'^T\rangle = h'^T A_i h'.$$

Then applying the bound on $\|h' - h_1\|$ derived in Theorem 4.2 and utilizing the fact that $h^T A_i h$ is continuous in h, we can have an upper bound on $|\hat{\lambda}_i^m - h_1^T A_i h_1|$. When A_i is large, $h_1^T A_i h_1$ is concentrated around λ_i with high probability. In the next section, we demonstrate properties and utilities of the joint embedding algorithm.

5 Experiments

Before going into details of our experiments, we want to discuss how to select the dimensionality d of the joint embedding. Estimating d is an important model selection question which has been studied for years under various settings [24]. It is not the focus of this paper, but we still face this decision in numerical experiments. In the simulation experiments of this section, we assume d is known to us and simply set the dimensionality estimate $\hat{d}$ equal to d. In the real data experiment, we recommend two approaches to determine $\hat{d}$. Both approaches require we first run the d'-dimensional joint embedding algorithm, where d' is sufficiently large and we are confident that d is less d'. We then plot the objective function versus dimension, and determine $\hat{d}$ to be where the objective starts to flatten out. Alternatively, we can plot $\{\hat{\Lambda}_{ik}\}_{i=1}^m$ for $k = 1, ..., d'$, and select $\hat{d}$ when the loadings start to look like noise with 0 mean. These two approaches should yield a similar dimensionality estimate $\hat{d}$.

5.1 Simulation Experiment 1: Joint Embedding Under a Simple Model

We present a simple numerical example to demonstrate some properties of our joint embedding procedure as the number of graphs grows. We repeatedly generate graphs with 5 vertices from 1-dimensional MREG where $\lambda_i \sim Unif(1, 2)$ and $h_1 = [0.1, 0.2, 0.3, 0.4, 0.5]^T/0.74$. We keep doubling the number of graphs m from 2^4 to 2^{16}. At each value of m, we compute the 1-dimensional joint embedding of m graphs. Let the estimated parameters based on m graphs be denoted by $\hat{\lambda}_i^m$ and $\hat{h}_1^m$. Two quantities based on $\hat{h}_1^m$ are calculated. The first is the norm difference between the current h_1 estimates and the previous estimates, namely $\|\hat{h}_1^m - \hat{h}_1^{m/2}\|$. This provides numerical evidence for the convergence of our principled estimation procedure. The second quantity is $\|\hat{h}_1^m - h_1\|$. This investigates whether $\hat{h}_1$ is an unbiased estimator for h_1. The procedure described above is repeated 100 times. The Figure 1 presents the result.

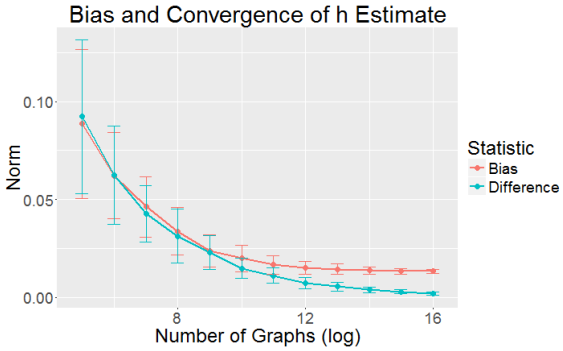

Fig. 1. Mean bias ($\|\hat{h}_1^m - h_1\|$) and mean difference between estimates ($\|\hat{h}_1^m - \hat{h}_1^{m/2}\|$) across 100 simulations are shown. The standard errors across 100 simulations are also given by error bars. The graphs are generated from a 1-dimensional MREG model as described in section 5.1. Bias stops decreasing at approximately 0.015, and difference converges to 0.

From the plot, we can see that the norm difference converges to 0 as m increases. This suggests the convergence of $\hat{h}_1^m$. Secondly, we should notice that the bias $\|\hat{h}_1^m - h_1\|$ does not converge to 0; instead, it stops decreasing at around 0.015. This suggests that $\hat{h}_1$ is an asymptotically biased estimator for h_1. The $\hat{h}_1^m$ with $m = 2^{16}$ is $[0.083, 0.186, 0.296, 0.406, 0.503]^T/0.74$. Compared to h_1, there are negative biases for small entries of h_1, and positive biases for large entries of h_1. Actually, this is as

to be expected: when there are infinitely many nuisance parameters present, Neyman and Scott demonstrate that maximum likelihood estimator is inconsistent [25]. In our case, there are infinitely many λ_i as m grows; therefore, we do not expect joint embedding to provide a consistent estimate of h_1.

In applications such as clustering or classifying multiple graphs, we may be not interested in $\hat{h}_1$. We are more interested in the $\hat{\lambda}_i$, which provide information specifically about the graphs G_i. Here, we consider two approaches to estimate λ_i. The first approach is estimating λ_i through joint embedding, that is

$$\hat{\lambda}_i = \langle A_i, \hat{h}_1^m \hat{h}_1^{mT} \rangle.$$

The second approach estimates λ_i by forcing $\hat{h}_1$ equal to h_1, that is

$$\hat{\lambda}_i = \langle A_i, h_1 h_1^T \rangle.$$

$\hat{\lambda}_i$ calculated this way can be thought of as the 'oracle' estimate, since it assumes h_1 is known. For each value of m, Figure 2 shows the differences in estimates provided by two approaches. Not surprisingly, the differences are small due to the fact that $\hat{h}_1^m$ and h_1 are close.

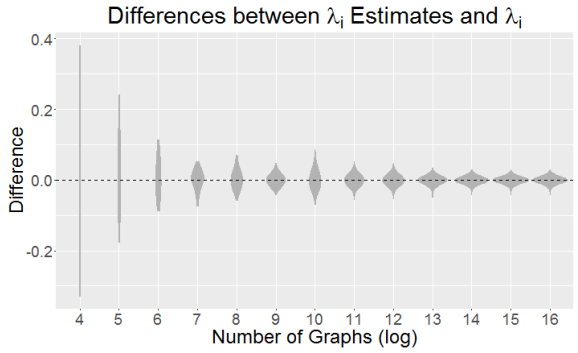

Fig. 2. Distribution of differences between $\hat{\lambda}_i$ estimated using $\hat{h}_1^m$ and h_1. The graphs are generated from a 1-dimensional MREG model as described in section 5.1. The differences are small due to the fact that $\hat{h}_1^m$ and h_1 are close.

5.2 Simulation Experiment 2: Joint Embedding to Classify Graphs

In this experiment, we consider the inference task of classifying graphs. We have m pairs $\{(A_i, Y_i)\}_{i=1}^m$ of observations. Each pair consists of an adjacency matrix $A_i \in \{0,1\}^{n \times n}$ and a label $Y_i \in [K]$. Furthermore, all pairs are assumed to be independent and identically distributed according to an unknown distribution $\mathbb{F}_{A,Y}$, that is

$$(A_1, Y_1), (A_2, Y_2), ..., (A_m, Y_m) \overset{i.i.d.}{\sim} \mathbb{F}_{A,Y}.$$

The goal is to find a classifier g which is a function $g : \{0,1\}^{n \times n} \to [K]$ which has small classification error $L_g = P(g(A) \neq Y)$.

We consider a binary classification problem where Y takes value 1 or 2. We independently generate 200 graphs with 100 vertices. The graphs are sampled from a 2-dimensional MREG model. Let h_1 and h_2 be two vectors in $\mathbb{R}^{100}$, and

$$h_1 = [0.1, ..., 0.1]^T \text{ , and } h_2 = [-0.1, ..., -0.1, 0.1, ..., 0.1]^T.$$

Here, h_2 has -0.1 as its first 50 entries and 0.1 as its last 50 entries. We generate graphs according to the MREG model,

$$A_i \sim MREG(F, h_1, h_2). \tag{5}$$

Here, F is a mixture of two point masses with equal probability,

$$F = \frac{1}{2}\mathbb{I}\{\lambda = [25, 5]\} + \frac{1}{2}\mathbb{I}\{\lambda = [22.5, 2.5]\}.$$

We let the class label Y_i indicate which point mass λ_i is sampled from. In terms of SBM, this graph generation scheme is equivalent to

$$A_i | Y_i = 1 \sim SBM((1, ..., 1, 2, ..., 2), \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.3 \end{bmatrix})$$

$$A_i | Y_i = 2 \sim SBM((1, ..., 1, 2, ..., 2), \begin{bmatrix} 0.25 & 0.2 \\ 0.2 & 0.25 \end{bmatrix}).$$

To classify graphs, we first jointly embed 200 graphs. The first two dimensional loadings are shown in Figure 3. We can see two classes are clearly separated after being jointly embedded. Then, a 1-nearest neighbor classifier

Fig. 3. Scatter plot of loadings computed by jointly embedding 200 graphs. The graphs are generated from a 2-dimensional MREG model as described in equation 5. The loadings of two classes are clearly separated after being jointly embedded.

is constructed based on loadings $\{\hat{\lambda}_i\}_{i=1}^m$. The 1-NN classification error with 200 graphs is 0.

We compare classification performance using joint embedding and Laplacian Eigenmap [11]. For Laplacian Eigenmap, we first embed each normalized Laplacian matrix and then compute Frobenious norm between embeddings. We also apply a 1-NN rule to classify graphs. We let the number of graphs m increase from 4 to 200. For each value of m, we repeat the simulation 100 times. The result is shown in Figure 4. We see joint embedding takes advantage of increasing sample size and dominates Laplacian Eigenmap at all values of m.

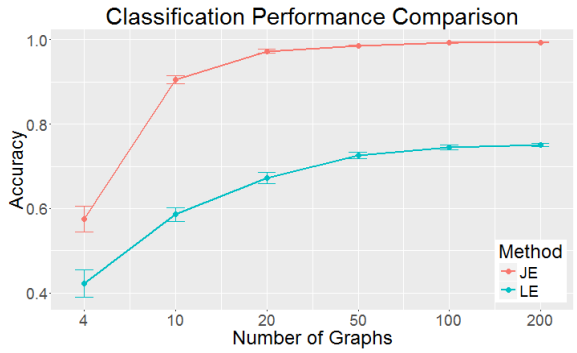

Fig. 4. Mean classification accuracy of joint embedding (JE) and Laplacian Eigenmap (LE) with their standard errors are shown. The graphs are generated from a 2-dimensional MREG model as described in equation 5. The graphs are first embedded using two approaches, and then apply a 1-NN to the embeddings. For each value of m, the simulation is repeated 100 times. Joint embedding takes advantage of increasing sample size and has nearly perfect classification performance when given more than 100 graphs. Joint embedding dominates Laplacian eigenmap at all values of m.

5.3 Real Data Experiment: Predict Composite Creativity Index

In this experiment, we study predicting individual composite creativity index (CCI) through brain connectomes which are obtained by Multimodal Magnetic Resonance Imaging [26]. Neuroimaging and creativity have been jointly investigated previously [27]. Most studies utilize a statistical testing method and find CCI significantly related or inversely related to the activity of some regions of the brain. We embrace a different approach by directly building a prediction model for CCI. First, we jointly embed brain graphs of all subjects. Then, we construct a linear regression model by treating the estimated loadings as explanatory variables and CCI as response variable.

In total, 113 healthy, young adult subjects were scanned using Siemens TrioTim scanner. 3D-MPRAGE and DTI in 35 directions of the subjects were acquired [28]. The images are then registered by Desikan-Killiany Atlas [30], and a graph of 70 vertices is constructed. The whole process transforming MRI to graphs is done by NeuroData's MRI Graphs pipeline [29]. The graphs derived have weighted edges. One example of a graph is shown in the top panel of Figure 5. For each subject, a divergent thinking measure is scored by independent judges using the Consensual Assessment Technique [31], from which the CCI is derived. To predict the CCI, we first jointly embed 113 graphs with $d = 10$, and then fit a linear model by regressing CCI on $\hat{\lambda}_i$, that is

$$CCI_i \sim \beta_0 + \lambda_i^T \beta + \epsilon_i.$$

Next, we assess the statistical significance of this model by comparing to the null model with only intercept.

If only $\hat{\lambda}_i[1]$ is used as explanatory variable, the top panel of Figure 6 shows the result. We can see a significant positive linear relationship between CCI and the first dimensional loadings. The first dimensional loadings generally capture the overall connectivity of graphs. In our case, the correlation between the first dimensional loadings and the sum of edge weights is around 0.98. This model implies that individual tends to be more creative when there is more brain connectivity. The R-square of this model is 0.07248, and the model is statistically significantly better when compared to the null model with a p-value 0.0039 according to F-test. If fit CCI to all the 10 dimensional loadings, a summary of the linear model is provided in appendix and a scatter plot of fitted CCI versus true CCI is provided in the bottom panel of Figure 6. The R-square is 0.2325 and the model is statistically significantly better than the null model with a p-value 0.0018 according to F-test. It is also significantly better than the model with only $\hat{\lambda}_i[1]$. Although there is still a positive relationship between CCI and the first dimensional loadings, it is no longer significant due to including more explanatory variables. In this model, there is a significant negative relationship between CCI and $\hat{\lambda}_i[6]$ based on the t-test. The scatter plot of CCI against $\hat{\lambda}_i[6]$ is given in the middle panel of Figure 6. We look into the rank one matrix $\hat{h}_6^T \hat{h}_6$ which is shown in the bottom panel of figure 5. It has positive connectivity within each hemisphere of the brain, but negative connectivity across hemispheres. This suggests that compared to within-hemisphere connectivity, across-hemisphere connectivity tends to have a more positive impact on the human creativity.

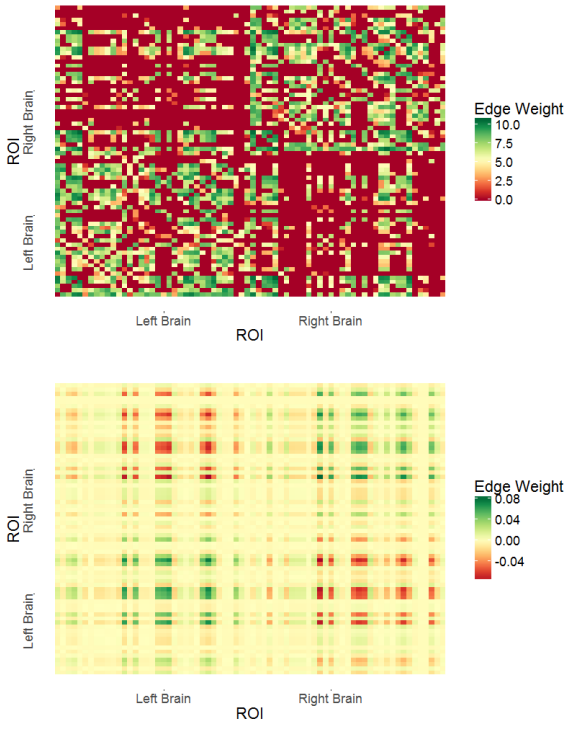

Fig. 5. The top panel shows the graph derived from a typical subject. There is much more neural connectivity within each hemisphere. The bottom panel shows the rank one matrix $\hat{h}_6^T \hat{h}_6$, which has positive connectivity within each hemisphere, but negative connectivity across hemispheres.

6 CONCLUSION

In summary, we develop a joint embedding method which can simultaneously embed multiple graphs into low dimen-

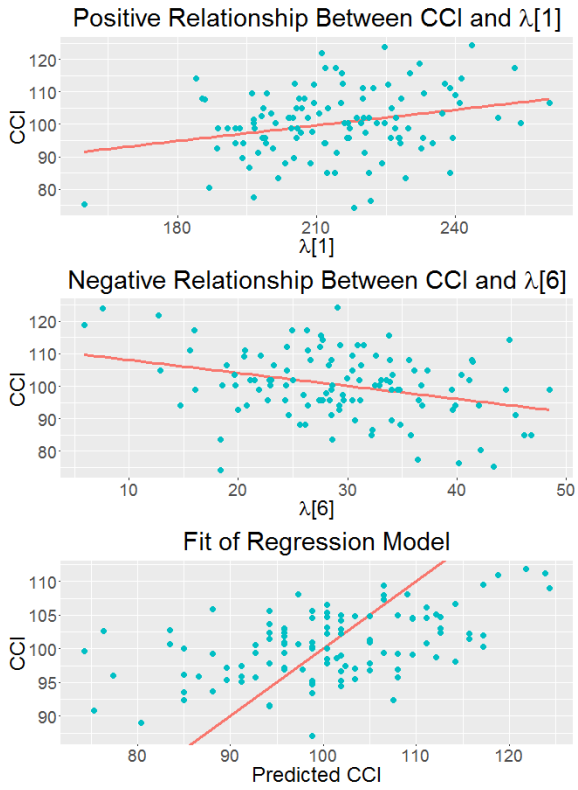

Fig. 6. The top panel shows the scatter plot of CCI against $\hat{\lambda}_i[1]$ with the regression line. There is a positive relationship between CCI and first dimensional loadings. The middle panel shows the scatter plot of CCI against $\hat{\lambda}_i[6]$ with regression line. There is a negative relationship between CCI and sixth dimensional loadings. The bottom panel shows the predicted CCI versus true CCI with the identity line.

sional space. Joint embedding can be used to estimate simple and intuitive statistics for inference problems on multiple vertex matched graphs. Learning on multiple graphs has significant applications in diverse fields and our results have both theoretical and practical implications for the problem. As the real data experiment illustrates, joint embedding is a practically viable inference procedure. We also propose a Multiple Random Eigen Graphs model. It can be seen as a generalization of Random Dot Product Graph model or Stochastic Block Model for multiple random graphs. We analyze the performance of joint embedding on this model under simple settings. It can be shown that our method provides estimates with bounded error. Our approach is intimately related to other matrix factorization approaches such as singular value decomposition and tensor factorization. Indeed, we all try to estimate a low dimensional representation of high dimensional objects. We are currently investigating joint embedding with more or less regularizations on parameters and under different set ups. We are optimistic that our method provides a viable tool for analyzing multiple graphs and can contribute to a deeper understanding of the joint structure of networks.

APPENDIX A

Proof of Theorem 4.1 First, we show that $|D_n(h, h_1) - D(h, h_1)|$ converges uniformly to 0. We notice three facts:

(1) the set $\{h : \|h\| = 1\}$ is compact;
(2) for all h, the function $\rho(\cdot, h)$ is continuous
(3) for all h, the function $\rho(\cdot, h)$ is bounded by n^2.

Therefore, by the uniform law of large numbers [32], we have

$$\sup_h |D_m(h, h_1) - D(h, h_1)| \xrightarrow{a.s.} 0.$$

To prove the claim of theorem, we use a technique similar to that employed by Bickel and Doksum [33]. By definition, we have $D_m(\hat{h}_1^m, h) \leq D_m(h', h)$ and $D(h', h) \leq D(\hat{h}_1^m, h)$. From these two inequalities we obtain

$$\begin{aligned} D_m(h', h) - D(h', h) &\geq D_m(\hat{h}_1^m, h) - D(h', h) \\ &\geq D_m(\hat{h}_1^m, h) - D(\hat{h}_1^m, h). \end{aligned}$$

Therefore,

$$\begin{aligned} |D_m(\hat{h}_1^m, h) - D(h', h)| \leq \max(&|D_m(h', h) - D(h', h)|, \\ &|D_m(\hat{h}_1^m, h) - D(\hat{h}_1^m, h)|). \end{aligned}$$

This implies

$$|D_m(\hat{h}_1^m, h) - D(h', h)| \leq \sup_h |D_m(h, h_1) - D(h, h_1)|.$$

Hence, $|D_m(\hat{h}_1^m, h) - D(h', h)|$ must converge almost surely to 0, that is

$$|D_m(\hat{h}_1^m, h) - D(h', h)| \xrightarrow{a.s.} 0.$$

If $\hat{h}_1^m$ does not converge almost surely h', then $\|\hat{h}_1^m - h'\| \geq \epsilon$ for some ϵ and infinitely many values of m. Since h' is the unique global minimum, $|D(\hat{h}_1^m, h) - D(h', h)| > \epsilon'$ for infinitely many values of m and some ϵ'. This contradicts with the previous equation. Therefore, $\hat{h}_1^m$ must converge almost surely to h'.

Proof of Theorem 4.2 The first part of proof involves showing that h' is the eigenvector corresponding to the largest eigenvalue of $E(\langle A_i, h'h'^T\rangle A_i)$. Then, we show $E(\langle A_i, h'h'^T\rangle A_i)$ is close to a properly scaled $E(P_i)$. To conclude, we apply the Davis-Kahan theorem to the top eigenvector of both matrices to obtain the desired result. First, we notice that

$$\begin{aligned} \min_{\|h\|=1} D(h, h_1) &= \min_{\|h\|=1} E(\|A_i - \langle A_i, hh^T\rangle hh^T\|^2) \\ &= \min_{\|h\|=1} E(\langle A_i, A_i\rangle - \langle A_i, hh^T\rangle^2) \\ &= E(\langle A_i, A_i\rangle) - \max_{\|h\|=1} E(\langle A_i, hh^T\rangle^2). \end{aligned}$$

Therefore,

$$h' = \operatorname*{argmin}_{\|h\|=1} D(h, h_1) = \operatorname*{argmax}_{\|h\|=1} E(\langle A_i, hh^T\rangle^2). \quad (6)$$

Taking the derivative of $E(\langle A_i, hh^T\rangle^2) + c(h^T h - 1)$ with respect to h, we have

$$\begin{aligned} \frac{\partial E(\langle A_i, hh^T\rangle^2) + c(h^T h - 1)}{\partial h} &= E(\frac{\partial \langle A_i, hh^T\rangle^2}{\partial h}) + 2ch \\ &= 4E(\langle A_i, hh^T\rangle A_i)h + 2ch. \end{aligned}$$

Setting this expression to 0 yields,

$$E(\langle A_i, h'h'^T\rangle A_i)h' = -\frac{1}{2}ch'.$$

Using the fact that, $\|h'\| = 1$, we can solve for c:

$$c = -2h'^T E(\langle A_i, h'h'^T\rangle A_i)h' = -2E(\langle A_i, h'h'^T\rangle^2).$$

Then, substituting for c, we obtain

$$E(\langle A_i, h'h'^T\rangle A_i)h' = E(\langle A_i, h'h'^T\rangle^2)h'. \quad (7)$$

Therefore, we see that h' is an eigenvector of $E(\langle A_i, h'h'^T\rangle A_i)$ and the corresponding eigenvalue is $E(\langle A_i, h'h'^T\rangle^2)$. Furthermore, $E(\langle A_i, h'h'^T\rangle^2)$ must be the eigenvalue with the largest magnitude. For if not, then there exists a h'' with norm 1 such that

$$|h''^T E(\langle A_i, h'h'^T\rangle A_i)h''| = |E(\langle A_i, h'h'^T\rangle\langle A_i, h''h''^T\rangle)| > E(\langle A_i, h'h'^T\rangle^2);$$

however, by Cauchy-Schwarz inequality we must have

$$E(\langle A_i, h''h''^T\rangle^2)E(\langle A_i, h'h'^T\rangle^2) > |E(\langle A_i, h'h'^T\rangle\langle A_i, h''h''^T\rangle)|^2,$$

implying $E(\langle A_i, h''h''^T\rangle^2) > E(\langle A_i, h'h'^T\rangle^2)$ which contradicts equation (6). Thus we conclude that h' is the eigenvector corresponding to the largest eigenvalue of $E(\langle A_i, h'h'^T\rangle A_i)$.Next, we compute $E(\langle A_i, h'h'^T\rangle A_i)$.

$$\begin{aligned}
&E(\langle A_i, h'h'^T\rangle A_i|P_i)\\
&= E(\langle A_i - P_i, h'h'^T\rangle(A_i - P_i)|P_i)\\
&\quad + E(\langle A_i - P_i, h'h'^T\rangle P_i)|P_i)\\
&\quad + E(\langle P_i, h'h'^T\rangle(A_i - P_i)|P_i) + E(\langle P_i, h'h'^T\rangle P_i|P_i)\\
&= E(\langle A_i - P_i, h'h'^T\rangle(A_i - P_i)|P_i) + \lambda_i(h_1^T h')^2 P_i\\
&= 2h'h'^T * P_i * (J - P_i) - DIAG(h_1 h_1^T * P_i * (J - P_i))\\
&\quad + \lambda_i(h_1^T h')^2 P_i.
\end{aligned}$$

Here, DIAG() means only keep the diagonal of matrix; $*$ means the Hadamard product, and J is a matrix of all ones. Using the fact that $P_i = \lambda_i h_1 h_1^T$, we have

$$\begin{aligned}
&E(\langle A_i, h'h'^T\rangle A_i) - E(\lambda_i^2)(h_1^T h')^2 h_1 h_1^T\\
&= E(E(\langle A_i, h'h'^T\rangle A_i|P_i) - \lambda_i(h_1^T h')^2 P_i)\\
&= E(2h'h'^T * P_i * (J - P_i) - DIAG(h'h'^T * P_i * (J - P_i))).
\end{aligned}$$

If we consider the norm difference between $E(\langle A_i, h'h'^T\rangle A_i)$ and $E(\lambda_i^2)(h_1^T h')^2 h_1 h_1^T$, we have

$$\begin{aligned}
&\|E(\langle A_i, h'h'^T\rangle A_i) - E(\lambda_i^2)(h_1^T h')^2 h_1 h_1^T\|\\
&= \|E(2h'h'^T * P_i * (J - P_i) -\\
&\qquad DIAG(h'h'^T * P_i * (J - P_i)))\|\\
&\le E(\|2h'h'^T * P_i * (J - P_i) -\\
&\qquad DIAG(h'h'^T * P_i * (J - P_i))\|)\\
&\le E(\|2h'h'^T * P_i * (J - P_i)\|)\\
&\le E(\|2h'h'^T * P_i\|)\\
&\le 2E(\lambda_i)\|h'h'^T * h_1 h_1^T\|\\
&= 2E(\lambda_i).
\end{aligned}$$

Notice that the only non-zero eigenvector of $E(\lambda_i^2)(h_1^T h')^2 h_1 h_1^T$ is h_1 and the corresponding eigenvalue is $E(\lambda_i^2)(h_1^T h')^2$. We apply the Davis-Kahan theorem [34] to the eigenvector corresponding to the largest eigenvalue of matrices $E(\langle A_i, h'h'^T\rangle A_i)$ and $E(\lambda_i^2)(h_1^T h')^2 h_1 h_1^T$, yielding

$$\|h' - h_1\| \le \frac{2E(\lambda_i)}{E(\lambda_i^2)(h_1^T h')^2}.$$

APPENDIX B

Linear Regression Model Summary

```
> model<-lm(cci~Lambda+1)
> summary(model)

Call:
lm(formula = cci ~ Lambda + 1)

Residuals:
Min        1Q    Median        3Q       Max
-26.3432  -6.144  -0.7578  7.1032   16.9004

Coefficients: Estimate   Pr(>|t|)
(Intercept)   1.275e+02   0.000275 ***
Lambda1       2.421e-04   0.997981
Lambda2      -2.326e-01   0.070110 .
Lambda3      -3.716e-02   0.822592
Lambda4       8.049e-02   0.687628
Lambda5      -2.925e-01   0.421858
Lambda6      -4.285e-01   0.009088 **
Lambda7      -1.745e-01   0.590533
Lambda8      -3.465e-01   0.240093
Lambda9      -8.970e-01   0.007999 **
Lambda10     -8.955e-01   0.052839 .
---
Signif. codes:  0    ***    0.001    **
0.01    *    0.05    .    0.1          1

Residual standard error: 9.437
on 102 degrees of freedom
Multiple R-squared:  0.2325,
Adjusted R-squared:  0.1572
F-statistic:  3.09 on 10 and 102 DF,
p-value: 0.001795
```

REFERENCES

[1] E. Otte and R. Rousseau, "Social network analysis: a powerful strategy, also for the information sciences," *Journal of information Science*, vol. 28, no. 6, pp. 441–453, 2002.

[2] R. Govindan and H. Tangmunarunkit, "Heuristics for internet map discovery," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3. IEEE, 2000, pp. 1371–1380.

[3] E. T. Bullmore and D. S. Bassett, "Brain graphs: graphical models of the human brain connectome," *Annual review of clinical psychology*, vol. 7, pp. 113–140, 2011.

[4] M. D. Ward, K. Stovel, and A. Sacks, "Network analysis and political science," *Annual Review of Political Science*, vol. 14, pp. 245–264, 2011.

[5] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.

[6] G. Li, M. Semerci, B. Yener, and M. J. Zaki, "Graph classification via topological and label attributes," in *Proceedings of the 9th international workshop on mining and learning with graphs (MLG), San Diego, USA*, vol. 2, 2011.

[7] C. Jiang, F. Coenen, and M. Zito, "A survey of frequent subgraph mining algorithms," *The Knowledge Engineering Review*, vol. 28, no. 01, pp. 75–105, 2013.

[8] J. Huan, W. Wang, and J. Prins, "Efficient mining of frequent subgraphs in the presence of isomorphism," in *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. IEEE, 2003, pp. 549–552.

[9] Z. Zhao and H. Liu, "Spectral feature selection for supervised and unsupervised learning," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 1151–1157.

[10] D. L. Sussman, M. Tang, D. E. Fishkind, and C. E. Priebe, "A consistent adjacency spectral embedding for stochastic blockmodel graphs," *Journal of the American Statistical Association*, vol. 107, no. 499, pp. 1119–1128, 2012.

[11] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[12] B. Karrer and M. E. Newman, "Stochastic blockmodels and community structure in networks," *Physical Review E*, vol. 83, no. 1, p. 016107, 2011.

[13] V. Lyzinski, M. Tang, A. Athreya, Y. Park, and C. E. Priebe, "Community detection and classification in hierarchical stochastic blockmodels," *arXiv preprint arXiv:1503.02115*, 2015.

[14] S. J. Young and E. R. Scheinerman, "Random dot product graph models for social networks," in *Algorithms and models for the web-graph*. Springer, 2007, pp. 138–149.

[15] D. L. Sussman, M. Tang, and C. E. Priebe, "Consistent latent position estimation and vertex classification for random dot product graphs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 1, pp. 48–57, 2014.

[16] B. N. Flury and W. Gautschi, "An algorithm for simultaneous orthogonal transformation of several positive definite symmetric matrices to nearly diagonal form," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 1, pp. 169–184, 1986.

[17] A. Ziehe, P. Laskov, G. Nolte, and K.-R. MÃžller, "A fast algorithm for joint diagonalization with non-orthogonal transformations and its application to blind source separation," *Journal of Machine Learning Research*, vol. 5, no. Jul, pp. 777–800, 2004.

[18] J. D. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[19] W. Tang, Z. Lu, and I. S. Dhillon, "Clustering with multiple graphs," in *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*. IEEE, 2009, pp. 1016–1021.

[20] T. G. Kolda, "Numerical optimization for symmetric tensor decomposition," *Mathematical Programming*, vol. 151, no. 1, pp. 225–248, 2015.

[21] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[22] H. Kim and H. Park, "Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method," *SIAM journal on matrix analysis and applications*, vol. 30, no. 2, pp. 713–730, 2008.

[23] A. Athreya, C. Priebe, M. Tang, V. Lyzinski, D. Marchette, and D. Sussman, "A limit theorem for scaled eigenvectors of random dot product graphs," *Sankhya A*, pp. 1–18, 2013.

[24] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, no. 2, 1995, pp. 1137–1145.

[25] J. Neyman and E. L. Scott, "Consistent estimates based on partially consistent observations," *Econometrica: Journal of the Econometric Society*, pp. 1–32, 1948.

[26] D. Koutra, J. T. Vogelstein, and C. Faloutsos, "D elta c on: A principled massive-graph similarity function," in *Proceedings of the SIAM International Conference in Data Mining. Society for Industrial and Applied Mathematics*. SIAM, 2013, pp. 162–170.

[27] R. Arden, R. S. Chavez, R. Grazioplene, and R. E. Jung, "Neuroimaging creativity: a psychometric view," *Behavioural brain research*, vol. 214, no. 2, pp. 143–156, 2010.

[28] M. Brant-Zawadzki, G. D. Gillan, and W. R. Nitz, "Mp rage: a three-dimensional, t1-weighted, gradient-echo sequence–initial experience in the brain." *Radiology*, vol. 182, no. 3, pp. 769–775, 1992.

[29] G. Kiar, W. Gray Roncal, D. Mhembere, E. Bridgeford, R. Burns, and J. Vogelstein, "ndmg: Neurodata's mri graphs pipeline," Aug. 2016, open-source code. [Online]. Available: http://dx.doi.org/10.5281/zenodo.60206

[30] R. S. Desikan, F. Ségonne, B. Fischl, B. T. Quinn, B. C. Dickerson, D. Blacker, R. L. Buckner, A. M. Dale, R. P. Maguire, B. T. Hyman *et al.*, "An automated labeling system for subdividing the human cerebral cortex on mri scans into gyral based regions of interest," *Neuroimage*, vol. 31, no. 3, pp. 968–980, 2006.

[31] T. M. Amabile, "The social psychology of creativity: A componential conceptualization." *Journal of personality and social psychology*, vol. 45, no. 2, p. 357, 1983.

[32] R. I. Jennrich, "Asymptotic properties of non-linear least squares estimators," *The Annals of Mathematical Statistics*, vol. 40, no. 2, pp. 633–643, 1969.

[33] P. J. Bickel and K. A. Doksum, *Mathematical Statistics: Basic Ideas and Selected Topics, volume I*. CRC Press, 2015, vol. 117, ch. 6.

[34] C. Davis and W. M. Kahan, "The rotation of eigenvectors by a perturbation. iii," *SIAM Journal on Numerical Analysis*, vol. 7, no. 1, pp. 1–46, 1970.