Joint Embedding of Graphs

Shangsi Wang, Carey E. Priebe, Joshua T. Vogelstein, Avanti Athreya, Minh Tang, Vince Lyzinski, Youngser Park

Abstract—In this paper, we propose a method to jointly embed multiple undirected graphs. Given a set of graphs $\{G_i = (V_i, E_i)\}_{i=1}^m$, our method identifies a linear subspace spanned by rank one symmetric matrices and projects adjacency matrices of graphs into the subspace. The projection coefficients can be treated as features of graphs which are helpful in graph clustering and classification. We also propose a random graph model which can be used to model multiple random graphs. It can be shown that under the model our method produces estimates of parameters with small errors. We demonstrate through simulation experiments that our joint embedding method extracts features which lead to state of the art performance in classifying multiple graphs. We further conduct a real data experiment which shows that an individual composite creativity index can be predicted by the neural connectivity of the human brain through joint embedding.

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

1 Introduction

In many problems arising in science, graphs appear naturally as data 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. The 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 study of graphs. In this paper, we propose an algorithm to jointly embed multiple graphs into low dimensional space. We demonstrate through theory and experiments that our 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, some filtering criteria based on frequency of a subgraph are proposed. 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 [7]. 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 [8].

Adjacency spectral embedding (ASE) and laplacian eigenmap (LE) are proposed to embed a single graph observation [9] [10]. The inference task considered is usually learning the block structure of the graph. Given a set of graphs $\{G_i = (V_i, E_i)\}_{i=1}^m$, ASE and LE need to embed each G_i individually, and there is no easy way to combine multiple embeddings. Our joint embedding method considers the set 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. Joint embedding can be understood as a generalization of ASE for multiple graphs. We demonstrate through simulation experiments that our 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 our 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 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. 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,

$$P(A|\tau, B) = \prod_{i<j} B_{\tau_i,\tau_j}^{A_{i,j}}(1 - B_{\tau_i,\tau_j})^{(1-A_{i,j})}$$

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 [11] [12]. Next, we recall the notion of a random dot product graph [13].

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,

$$A_{ij} \sim Bernoulli(X_i^T X_j).$$

Alternatively,

$$P(A|X) = \prod_{i<j} X_i^T X_j^{A_{ij}}(1 - X_i^T X_j)^{1-A_{ij}}.$$

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 [14]. SBM and RDPG are models for a single graph. We propose a model for multiple random graphs which generalizes SBM and RDPG.0

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 $\{(\lambda_i, A_i)\}_{i=1}^m$ 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. It is mainly for the convenience of proving theories. 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 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

Our 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, we minimize 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 n vertices, let $A_i \in \{0,1\}^{n\times n}$ denote the adjacency matrix of graph G_i. 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 the 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

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

We denote the function on the left hand side of the equation by $f(\Lambda, H)$ which is implicitly 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 of 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

$$\underset{D_i, \|h_k\|=1}{\operatorname{argmin}} \sum_{i=1}^m \|A_i - H D_i H^T\|^2 \text{ , subject to } D_i \text{ is diagonal.}$$

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 also similar to principal component analysis in the sense of minimizing squared reconstruction error to recover loadings and components. However, there are extra symmetries and rank constraints on the components. In case of embedding only one graph, the joint embedding is equivalent to the adjacency spectral embedding solved by singular value decomposition [9]. Next, we describe an algorithm to optimize the objective function $f(\Lambda, H)$.

3.2 Alternating Descent Algorithm

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 [15] 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 [16] [17]. 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 square 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 use gradient information and take an Armijo line search strategy to update H [18].

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, we fix all estimates for first $k_0 - 1$ columns of Λ and H at iteration k_0, and then the objective function is minimized by searching through only kth column of Λ and H, that is

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

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 $(\hat{\Lambda}, \hat{H})$, we recursively solve the one dimensional optimization problem above by letting k_0 to vary from 1 to d.

There are a few advantages in recursively solving one dimensional problem. First, there are less parameters to fit at each iteration, since we are only allowed to vary Λ_{*k_0} and h_{k_0}. It makes initialization and optimization procedure much easier compared to optimizing all columns of H at the same time. Second, it implicitly enforces an ordering on columns of H. The ordering allows us to select the top few columns of Λ and H in cases we want to perform model selection. Third, it allows incremental computation. If we compute d and d' dimensional joint embedding, the first $\min(d, d')$ columns of $\hat{\Lambda}$ and $\hat{H}$ will be the same. The disadvantage of optimizing each column separately is that we are more likely to end up at a local minimum when the objective function is structured not in favor of embedding separately. In practice, this problem can be mitigated by running the joint embedding algorithm several times with random initialization.

To find Λ_{*k_0} and h_{k_0} in equation2, we need 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

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

Set the derivative to 0, we have

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

Algorithm 1 describes the general procedure to compute d-dimensional joint embedding of graphs $\{G_i\}_{i=1}^m$.

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)
      Fixing h_k, update Λ_{*k} by (4)
      Compute objective $\sum_{i=1}^{m} \|R_{ik} - \Lambda_{ik}h_kh_k^T\|^2$
    end while
    Update residuals: $R_{i(k+1)} = R_{ik} - \Lambda_{ik}h_kh_k^T$
  end for
  Output $\hat{\Lambda} = [\Lambda_{*1}, ..., \Lambda_{*d}]$ and $\hat{H} = [h_1, ..., h_d]$
end procedure

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 H denoted by $\{\hat{h}_k\}_{k=1}^d$ are estimates of $\{h_k\}_{k=1}^d$ 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. We should admit that the optimization algorithm described above may not be the

fastest approach to solve the problem; however, it is not the focus of this paper. Based on results from numerical applications, it 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 model, we can force estimated loadings $\hat{\lambda}_i$ to be equal across all graphs. This is useful when 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, we should average $\Lambda_{y_i k}$ within the same class.

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

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

Next, we discuss 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 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 fixed 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 flip of each other. In this case, theorem 4.1 does not apply. We are currently only certain that when all graphs are from 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 a case that 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 more assumptions. In the proof, we show that

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

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

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

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 see $E(\lambda_i^2)(h_1^T h)^4$ is clearly 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 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 are 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' differs with h_1 by a significant amount. In this paper, we focus on the case that n is fixed in this paper. The case that n goes to infinity for random inner product graph is considered in [19].

The two theorems above only concern 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 only observe one finite graph A_i associated with λ_i. When m goes to infinity, we can apply theorem 4.1,

$$\hat{\lambda}_i^m \to \langle A_i, h' h'^T \rangle = h'^T A_i h'.$$

Then applying theorem 4.2 and utilizing the fact that $h^T A_i h$ is continuous in h, we may have an upper bound on $|\hat{\lambda}_i^m - h_1^T A_i h_1|$. In general, $h_1^T A_i h_1$ is concentrated about λ_i with high probability.

5 Experiments

Before going into details of experiments, we want to discuss how to select the dimensionality of joint embedding d. Estimating d is an important model selection question which has been studied for years under various settings [20]. 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 the d. In the real data experiment, we recommend two approaches to determine $\hat{d}$. Both approaches require we first run 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 $\hat{d}$ where the objective start 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 usually yield 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 as it increases from 2^4 to 2^{16}. At each value of m, we compute 1-dimensional joint embedding of m graphs. Let the estimated parameters based on m graphs denoted by $\hat{\lambda}_i^m$ and $\hat{h}_1^m$. Two quantities based on $\hat{h}_1^m$ are calculated. The first one is the norm difference between the current h_1 estimates and the previous estimates, namely $\|\hat{h}_1^m - \hat{h}_1^{m/2}\|$. It tries to seek for numerical evidences for the convergence of our principled estimation procedure. The second quantity is $\|\hat{h}_1^m - h_1\|$. It tries to see whether $\hat{h}_1$ is an unbiased estimator for h_1. The figure 1 demonstrates the result.

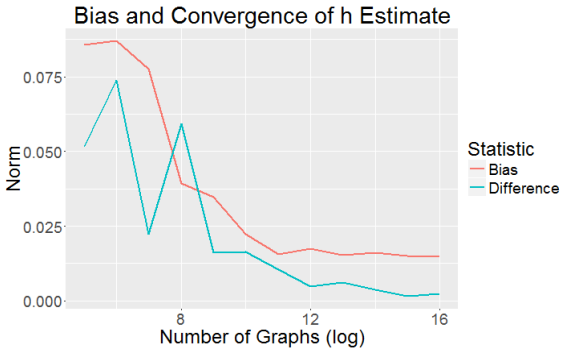

Fig. 1. Bias ($\|\hat{h}_1^m - h_1\|$) and difference between estimates ($\|\hat{h}_1^m - \hat{h}_1^{m/2}\|$) when increasing the number of graphs. The graphs are generated from a 1-dimensional MREG model as described in section 5.1. Bias stops to decrease at the level of around 0.015, and difference converges to 0.

From the plot, we can see that the norm difference converges to 0 as m increases. It 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 to decrease at around 0.03. 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, there are some theoretical evidences for this phenomenon. When there are infinitely many nuisance parameters present, Neyman and Scott gives an example that maximum likelihood estimator can be inconsistent as well [21]. 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 like clustering or classifying multiple graphs, we may be not interested in $\hat{h}_1$. We are more interested in $\hat{\lambda}_i$ which provides information specifically about graph 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' estimates, 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.

5.2 Simulation Experiment 2: Joint Embedding to Classify Graphs

In this experiment, we consider the inference task of classifying graphs. The set up is m pairs $\{(A_i, Y_i)\}_{i=1}^m$ are observed.

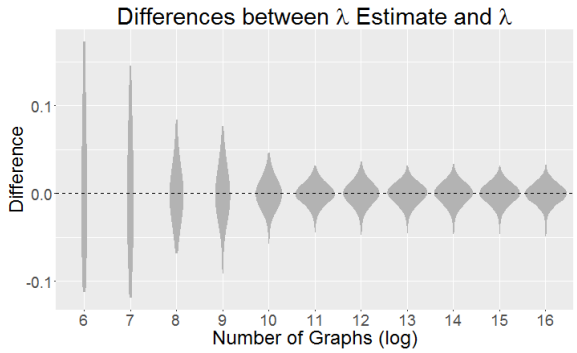

Fig. 2. Distribution of differences between $\hat{\lambda}_i$ estimated using $\hat{h}_1^m$ and h_1. The graphs are generated from a 1-dimensional MREG model as described in section 5.1. The differences are small due to the fact that $\hat{h}_1^m$ and h_1 are close.

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.

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 can only take 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. Conditioned on the value of Y, we generate graphs according to two MREG models, that is

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

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 to 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 the figure 3. We can see two classes are clearly separated after being jointly embedded. Then, an 1-nearest neighbor classifier 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 [10]. 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 to 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.

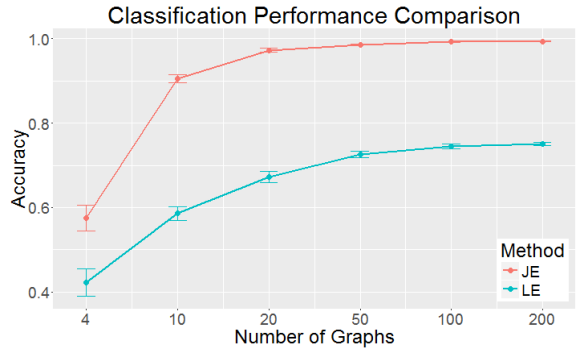

Fig. 4. Classification accuracy of joint embedding (JE) and laplacian eigenmap (LE) with varying number of graphs. The graphs are generated from a 2-dimensional MREG model as described in equation 5. The graphs are first embedding using two approaches, and then apply a 1-NN to the embeddings. For each value of m, the simulation is repeated 100 times. The mean classification accuracy and standard error are plotted. 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 consider predicting individual composite creativity index (CCI) through functional magnetic resonance imaging. Neuroimaging and creativity have been jointly investigated previously [22]. Most of studies utilize a statistical testing method and find CCI significant related or inversely related to the activity of some regions of brain. We embrace a different approach by directly building a prediction model for CCI. First,

we jointly embed brain images of all subjects. Then, we construct a linear regression model by treating the estimated loadings as explanatory variables and CCI as response variable.

113 healthy, young adult subjects were scanned using scanner with functional magnetic resonance imaging. The image is then registered by Desikan atlas [23] and a graph of 70 vertices is constructed through computing pairwise correlations across regions. The graphs constructed are weighted; however, it doesn't stop us using our joint embedding procedure. One example of a graph is shown in the top panel of figure 5. DT measures were scored by independent judges using the Consensual Assessment Technique [24], from which the CCI was derived. To predict the CCI, we jointly embed 113 graphs with $d = 10$. We 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 test the statistical significance of this model when compared to the null model with only intercept.

If only $\hat{\lambda}_i[1]$ is used as predictor, the top panel of figure 6 shows the result. We can see a significant positive linear relationship between CCI and the first dimensional loadings. Since the first dimensional loadings usually captures the overall connectivity of graphs, this suggests that individual tends to be more creative when there are more brain connectivity. The R-square of this model is 0.07248 and the model is statistically significant better when compared to the null model with a p-value 0.0039. If fit CCI to all the loadings, a summary of the linear model is provided in appendix. The R-square is 0.2325 and the model is statistically significant better than the null model with a p-value 0.0018. 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. There is a significant negative relationship between CCI and $\hat{\lambda}_i[6]$ as shown in the model summary. The scatter plot of CCI against $\hat{\lambda}_i[6]$ is given in the bottom 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 half of brain, but negative connectivity across half brains. This suggests that individual tends to be more creative when there are more across half brains connectivity than within half brains connectivity.

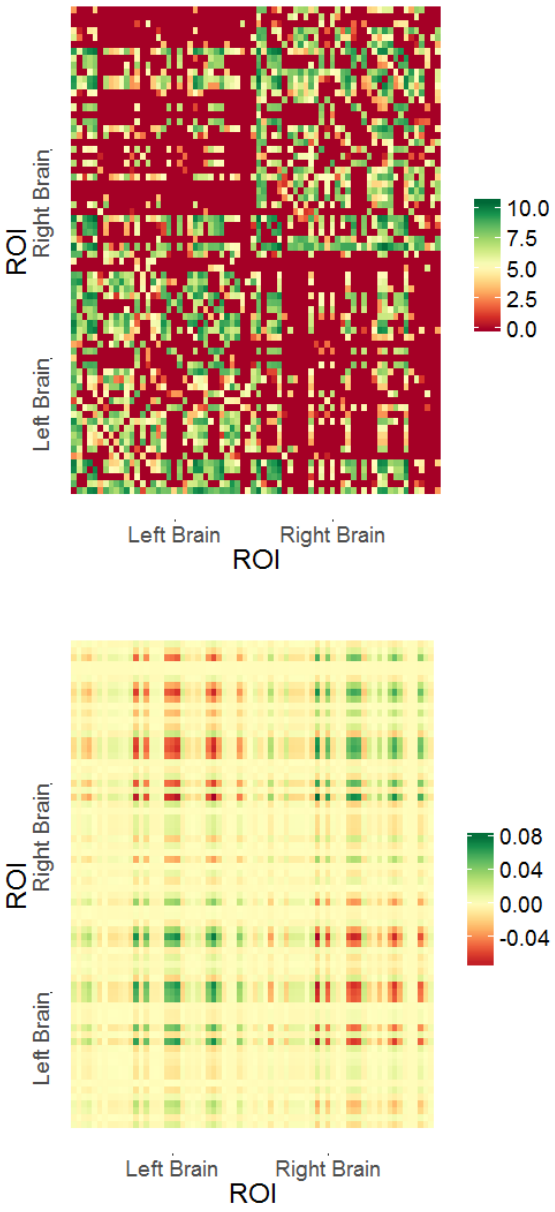

Fig. 5. The top panel shows the graph derived from a typical subject. There are much more neural connectivity within each half of the brain. The bottom panel shows the rank one matrix $\hat{h}_6^T \hat{h}_6$, which has positive connectivity within each half of brain, but negative connectivity across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

6 CONCLUSION

In summary, we develop a joint embedding method which can simultaneously embed multiple graphs into low dimensional 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 illustrate, 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 our method provides estimates with bounded error.

Our approach is intimately related other matrix factorization approaches like singular value decomposition and tensor factorization. Indeed, they all try to estimate a low dimensional representation of high dimensional objects. We are currently investigate joint embedding with more or less regularizations on parameters under difference real set ups.

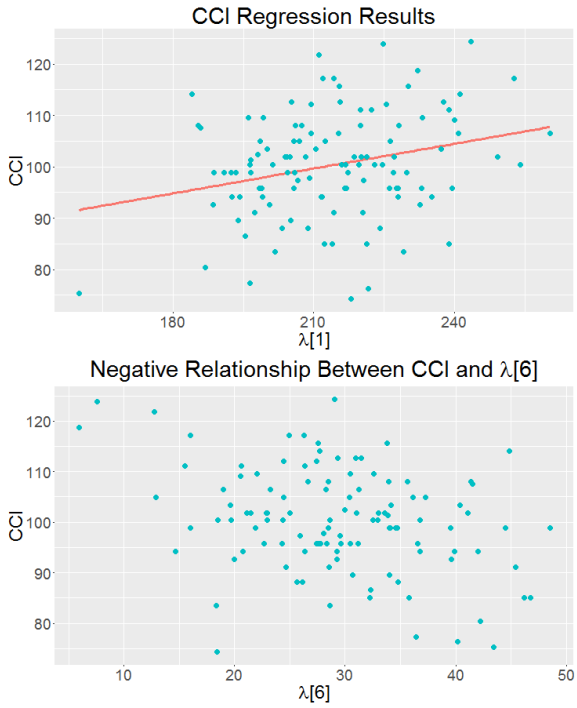

Fig. 6. The top panel shows the scatter plot of CCI against $\hat{\lambda}_i[1]$ with regression line. There is a positive relationship between CCI and first dimensional loadings. The bottom panel shows the scatter plot of CCI against $\hat{\lambda}_i[6]$. There is a positive relationship between CCI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

APPENDIX

Proof of Theorem 4.1 First, we are going to 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 a compact set.
(2) The function $\rho(A_i, h)$ is continuous for all h.
(3) For all h, $\rho(A_i, h)$ is bounded by n^2.

Therefore, by the uniform law of large numbers [25], 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 similar technique employed by Bickel and Doksum [26]. By definition, we have $D_m(\hat{h}_1^m, h) \leq D_m(h', h)$ and $D(h', h) \leq D(\hat{h}_1^m, h)$. Using these two inequalities we have

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

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', $\|\hat{h}_1^m - h'\| \geq \epsilon$ 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. This contradicts with the previous equation. Therefore, $\hat{h}_1^m$ must converges almost surely h'.

Proof of Theorem 4.2 The first part of proof is showing that h' is the eigenvector corresponds 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 Davis-Kahan theorem to the top eigenvector of both matrices and get 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 $E(\langle A_i, hh^T\rangle^2) + c(h^Th - 1)$ with respect to h, we have

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

Set it to 0,

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

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

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

Then, substitute c

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

Therefore, we see 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. Assume not, then there exists a h'' with norm 1 such that

$$\begin{aligned} |h''^T E(\langle A_i, h'h'^T\rangle A_i)h''| &= |E(\langle A_i, h'h'^T\rangle \langle A_i, h''h''^T\rangle)| \\ &> E(\langle A_i, h'h'^T\rangle^2). \end{aligned}$$

However, by Cauchy-Schwarz inequality we must have

$$\begin{aligned} E(\langle A_i, h''h''^T\rangle^2)E(\langle A_i, h'h'^T\rangle^2) > \\ |E(\langle A_i, h'h'^T\rangle \langle A_i, h''h''^T\rangle)|^2. \end{aligned}$$

This implies $E(\langle A_i, h''h''^T\rangle^2) > E(\langle A_i, h'h'^T\rangle^2)$ which contradicts equation (6). This concludes that h' is the eigenvector corresponds 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 Davis-Kahan theorem [27] to the eigenvector corresponding to the largest eigenvalue of matrices $E(\langle A_i, h'h'^T\rangle A_i)$ and $E(\lambda_i^2)(h_1^T h')^2 h_1 h_1^T$, we have

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

CCI Linear 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] 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.

[8] 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.

[9] 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.

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

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.
[19] 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.
[20] 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.
[21] J. Neyman and E. L. Scott, "Consistent estimates based on partially consistent observations," *Econometrica: Journal of the Econometric Society*, pp. 1–32, 1948.
[22] 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.
[23] 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.
[24] T. M. Amabile, "The social psychology of creativity: A componential conceptualization." *Journal of personality and social psychology*, vol. 45, no. 2, p. 357, 1983.
[25] R. I. Jennrich, "Asymptotic properties of non-linear least squares estimators," *The Annals of Mathematical Statistics*, vol. 40, no. 2, pp. 633–643, 1969.
[26] P. J. Bickel and K. A. Doksum, *Mathematical Statistics: Basic Ideas and Selected Topics, volume I*. CRC Press, 2015, vol. 117.
[27] 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.