Joint Embedding of Graphs

Shangsi Wang, Jesús Arroyo, Joshua T. Vogelstein, and Carey E. Priebe, *Senior Member, IEEE*

Abstract—Feature extraction and dimension reduction for networks is critical in a wide variety of domains. Efficiently and accurately learning features for multiple graphs has important applications in statistical inference on graphs. 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, while the embedding components can represent vertex features. We also propose a random graph model for multiple graphs that generalizes other classical models for 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 extracts interpretable features with good prediction accuracy in different tasks.

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

1 Introduction

In many problems arising in science and engineering, graphs arise naturally as data structure 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 applications of learning 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 a graph as a raw feature [5]. This approach produces features which are linear combinations of edges, but it ignores the topological structure of graphs and the features extracted are not easily interpretable. Second, features can be extracted by computing summary topological and label statistics from graphs [6], [7]. 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 computationally expensive. Third, many frequent subgraph mining algorithms are developed [8]. 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 [9]. Finally, 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 [10].

Adjacency Spectral Embedding (ASE) and Laplacian Eigenmap (LE) are proposed to embed a single graph observation [11], [12]. The inference task considered in these papers is learning of 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 an 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 real data experiments are explored in Section 5. We conclude the paper with a brief discussion of implications and possible future work.

- *S. Wang and C. Priebe are with the Department of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, MD 21218 USA. E-mail: {swang127, cep}@jhu.edu.*
- *J. Arroyo is with the Center for Imaging Science, Johns Hopkins University, Baltimore, MD 21218 USA. E-mail: jesus.arroyo@jhu.edu.*
- *J. T. Vogelstein is with the Department of Biomedical Engineering, Institute for Computational Medicine, Kavli Neuroscience Discovery Institute, Johns Hopkins University, Baltimore, MD 21218 USA. E-mail: jovo@jhu.edu.*

Manuscript received 4 Dec. 2018; revised 12 Sept. 2019; accepted 12 Oct. 2019. Date of publication 31 Oct. 2019; date of current version 4 Mar. 2021. (Corresponding author: Jesús Arroyo.)
Recommended for acceptance by D. Lowd.
Digital Object Identifier no. 10.1109/TPAMI.2019.2948619

2 SETTING

We focus on embedding unweighted and undirected graphs for simplicity, although the joint embedding algorithm works on weighted graphs, and directed graphs with some modifications. Let $\{G_i = (V_i, E_i)\}_{i=1}^m$ be m graphs, each with n vertices, and $\mathbf{A}_i$ be the adjacency matrix of graph G_i. The vertices in these graphs should be 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 need a random graph model on multiple graphs, on which the theoretical analysis is based. Let us first recall a model on a single graph: Random Dot Product Graph [13].

Definition Random Dot Product Graph (RDPG). *Let $\mathcal{X}$ be a subset of $\mathbb{R}^d$ such that $x^T y \in [0,1]$ for all $x, y \in \mathcal{X}$. Let $\mathbf{X} = [x_1^T, x_2^T, \ldots, x_n^T] \in \mathcal{X}^n$ be a $n \times d$ matrix, and given $\mathbf{X}$, suppose that $\mathbf{A}$ is a random $n \times n$ adjacency matrix such that*

$$\mathbf{A}_{st} \sim Bernoulli(x_s^T x_t).$$

Alternatively,

$$P(\mathbf{A}|\mathbf{X}) = \prod_{s<t} (x_s^T x_t)^{\mathbf{A}_{st}} (1 - x_s^T x_t)^{1-\mathbf{A}_{st}}.$$

Also, define $\mathbf{P} := \mathbf{X}\mathbf{X}^T$ to be edge probability matrix. We write $\mathbf{A} \sim \mathrm{RDPG}(\mathbf{X})$ to denote the distribution of a random dot product graph *with latent positions $\mathbf{X}$. When the rows of $\mathbf{X}$ are not fixed, but instead are random variables with a distribution F on $\mathcal{X}$, $(\mathbf{X}, \mathbf{A}) \sim \mathrm{RDPG}(F)$ denotes the distribution of a* random dot product graph *with latent positions distributed according to F.*

The RDPG is a convenient model which is designed to capture the relationship between the vertices of a graph using latent positions, and some extensions have been proposed to capture more general connectivity structures [14]. Moreover, other popular models, including the stochastic block model (SBM) [15] and mixed membership SBM [16] are special cases of the RDPG and its generalizations. The RDPG can be further generalized to a Latent Position Graph by replacing the inner product by a kernel [17]. The Adjacency Spectral Embedding of a RDPG adjacency matrix is well studied [18]. Next, we propose a new random graph model which generalizes the RDPG to multiple graphs.

Definition Multiple Random Eigen Graphs (MREG). *Let $h_1, \ldots, h_d$ be vectors in $\mathbb{R}^n$ with $\|h_i\|_2 = 1, i = 1, \ldots, d$, and denote by $\mathcal{X} \subset \mathbb{R}^d$ the set of vectors 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. The random adjacency matrices $\mathbf{A}_1, \ldots, \mathbf{A}_m$ follow a d-dimensional* multiple random eigen graphs model, *denoted by*

$$\{\mathbf{A}_i\}_{i=1}^m \sim \mathrm{MREG}(\{\lambda_i\}_{i=1}^m, h_1, \ldots, h_d),$$

if the entries of A_i are independent Bernoulli random variables,

$$\mathbf{A}_i[s,t] \sim Bernoulli\left(\sum_{k=1}^d \lambda_i[k] h_k[s] h_k[t]\right).$$

$\mathbf{P}_i := \sum_{k=1}^d \lambda_i[k] h_k h_k^T$ is defined to be the edge probability matrix for graph i. In cases that $\{\lambda_i\}_{i=1}^m$ are of primary interest, they are treated as parameters. When the $\{\lambda_i\}_{i=1}^m$ are random variables, F denotes their distribution on $\mathcal{X}$, and the model is expressed as

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

Compared to the RDPG model, MREG is designed to model multiple graphs. The vectors $\{h_k\}_{k=1}^d$ are shared across graphs and represent joint latent positions of the vertices, where $(h_j[1], \ldots, h_j[d]) \in \mathbb{R}^d$ represents the position of vertex j; a λ_i represents the parameter of an individual graph relative to the latent positions $\{h_k\}_{k=1}^d$. Note that for a single graph, the edge probability matrix can be written as $\mathbf{P}_i = \mathbf{H}\mathbf{\Lambda}_i\mathbf{H}^T$, where $\mathbf{H} = [h_1 \cdots h_d] \in \mathbb{R}^{n \times d}$ and $\mathbf{\Lambda}_i = \mathrm{diag}(\lambda_i)$. When the entries of λ_i are non-negative, then $\mathbf{X}_i = \mathbf{H}\mathbf{\Lambda}_i^{1/2}$ are the latent positions of graph i, and hence, on a single graph, RDPG and MREG are equivalent if the edge probability matrix is positive semidefinite. In MREG, we allow self loops to happen. This is mainly for theoretical convenience. Next, we introduce another random graph model: Stochastic Block Model [15], which generalizes the Erdos-Renyi (ER) model [19] that corresponds to a SBM with only one block. SBM is a widely used model to study the community structure of a graph [20], [21].

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 $\tau = (\tau_1, \tau_2, \ldots, \tau_n) \in [K]^n$ the block membership vector, where τ is a multinomial sequence with probability vector π. Denote by $\mathbf{B} \in [0,1]^{K \times K}$ the block connectivity probability matrix. Suppose $\mathbf{A}$ is a random adjacency matrix given by,*

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

Then, $\mathbf{A}$ is an adjacency matrix of a K-block stochastic block model graph, and the notation is $\mathbf{A} \sim SBM(\pi, \mathbf{B})$. Sometimes, τ may also be treated as the parameter of interest, in this case the notation is $\mathbf{A} \sim SBM(\tau, \mathbf{B})$.

The top panel of Fig. 1 shows the relationships between three random graph models defined above and the ER model on 1 graph. The models considered are those conditioned on latent positions, that is τ, $\mathbf{X}$ and λ in SBM, RDPG and MREG respectively are treated as parameters; furthermore, loops are ignored in MREG. If an adjacency matrix $\mathbf{A} \sim SBM(\tau, \mathbf{B})$ and the block connectivity matrix $\mathbf{B}$ is positive semidefinite, $\mathbf{A}$ can also be written as an $RDPG(\mathbf{X})$ with $\mathbf{X}$ having at most K distinct rows. If an adjacency matrix $\mathbf{A} \sim RDPG(\mathbf{X})$, then it is also a 1-graph $MREG(\lambda_1, h_1, \ldots, h_d)$ with h_k being the normalized kth column of $\mathbf{X}$ and λ_1 being the vector containing the squared norms of columns of $\mathbf{X}$. However, a 1-graph $MREG(\lambda_1, h_1, \ldots, h_d)$ is not necessarily an RDPG graph since λ_1 could contain negative entries which may result in an indefinite edge probability matrix.

The bottom panel of Fig. 1 shows the relationships between the models on multiple graphs. For RDPG, the graphs are sampled i.i.d. with the same parameters. MREG has the flexibility to have λ differ across graphs, which leads to a more generalized model for multiple graphs. A d

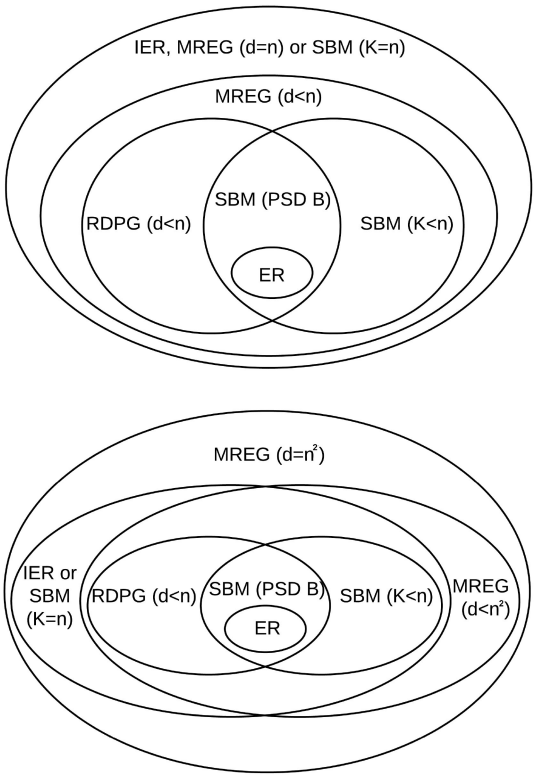


Fig. 1. Relationships between random graph models on 1 graph and multiple graphs. The top panel shows the relationships between the random graph models on 1 graph. The models considered are those conditioned on latent positions, that is τ, $\mathbf{X}$ and λ in SBM, RDPG and MREG respectively are treated as parameters. ER is a 1-block SBM. If a graph follows SBM with a positive semidefinite edge probability matrix, it also follows the RDPG model. Any SBM and RDPG graph can be represented by a d-dimensional MREG model with d being less than or equal to the number of blocks or the dimension of RDPG. On one graph, inhomogeneous ER (IER), n-dimensional MREG and n-block SBM are equivalent. The bottom panel shows the relationships between the random graph models on multiple graphs. The models considered are those conditioned on latent positions, and for ER, SBM and RDPG graphs are sampled i.i.d. with the same parameters. In this case, MREG has the flexibility to have λ differ across graphs, which leads to a more generalized model for multiple graphs.

dimensional MREG can represent any SBM with $K \leq d$ blocks, in which the block memberships are the same across all the graphs in the population, but possible different connectivity matrices $\mathbf{B}_i$ for each graph, a common assumption in modeling multilayer and time-varying networks [22], [23]. Other models in this setting also allow some vertices of the SBM to change their block memberships over time [22], [24]; in this case, the MREG can still represent those models, but the dimension d may need to increase. Actually, it turns out that if d is allowed to be as large as $\frac{n(n-1)}{2}$, MREG can represent any distribution on binary graphs, which includes distributions in which edges are not independent.

Theorem 2.1. *Given any distribution $\mathcal{F}$ on graphs and a random adjacency matrix $\mathbf{A} \sim \mathcal{F}$, there exists a dimension d, a distribution F on $\mathbb{R}^d$, and a set of vectors $\{h_k\}_{k=1}^d$, such that $\mathbf{A} \sim MREG(F, h_1, \ldots, h_d)$.*

Theorem 2.1 implies that MREG is really a semi-parametric model, which can capture any distribution on graphs. One can model any set of graphs by MREG with the guarantee that the true distribution is in the model with d being large enough. However, in practice, a smaller d may lead to better inference performance due to reduction in the dimensionality.

In the next section, we consider the joint embedding algorithm which can be understood as a parameter estimation procedure for MREG.

3 Joint Embedding

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 $\mathbf{A}_i$ into the subspace. The coefficients obtained by projecting $\mathbf{A}_i$ 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 (JE). *Given m graphs $\{G_i\}_{i=1}^m$ with $\mathbf{A}_i$ being the corresponding adjacency matrix, the d-dimensional joint embedding of graphs $\{G_i\}_{i=1}^m$ is given by*

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

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 and avoid the scaling factor, h_k is required to have norm 1. In addition, $\{h_k h_k^T\}_{k=1}^d$ must be linearly independent to avoid identifiability issues in estimating λ_i; however, $\{h_k\}_{k=1}^d$ needs not to be linearly independent or orthogonal. To ease the notations, let us introduce two matrices $\mathbf{\Lambda} \in \mathbb{R}^{m \times d}$ and $\mathbf{H} \in \mathbb{R}^{n \times d}$, where λ_i is the ith row of Λ and h_k is the kth row of $\mathbf{H}$; that is, $\mathbf{\Lambda} = [\lambda_1^T, \ldots, \lambda_m^T]$ and $\mathbf{H} = [h_1, \ldots, h_d]$. The Equation (1) can be rewritten using $\mathbf{\Lambda}$ and $\mathbf{H}$ as

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

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

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

Equation (1) can be also viewed as a tensor factorization problem. If $\{\mathbf{A}_i\}_{i=1}^m$ are stacked in a 3-D array $\mathbb{A} \in \mathbb{R}^{m \times n \times n}$, then solving Equation (1) is also equivalent to

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

where $\otimes$ denotes the tensor product and $\mathbf{\Lambda}_{*k}$ is the kth column of $\mathbf{\Lambda}$. It is well known in the tensor factorization community that the solution to Equation 1 may not necessarily exist for $d \geq 2$. This phenomenon was first found by Bini et al. [25], and Silva and Lim gives a characterization all such tensors in the order-3 rank-2 case [26]. Although there may not exist a global minimum, finding the local solution in a compact region still provide significant insights to the data. We design an algorithm which is guaranteed to converge, and provide analysis under the $d = 1$ case.

The joint embedding algorithm assumes the graphs are vertex-aligned and undirected. The vertex-aligned graphs are common in many applications of interest such as neuroimaging [27], multilayer networks [28] or time-varying graphs [7]. In case that the graphs are not aligned, graph matching should be performed before the joint embedding [29], [30]. The mis-alignments of some vertices will have adverse effects in estimating corresponding latent positions in $\mathbf{H}$; however, a small number of mis-aligned vertices should not have a big impact in estimating $\mathbf{\Lambda}$. If the graphs have weighted edges, the joint embedding can still be applied. Also, the MREG model can be easily extended to weighted graphs by replacing the Bernoulli distribution with other proper distributions. In fact, in the experiment of Section 5.3, the graphs are weighted, where the edge weights are the log of fiber counts across regions of brains. In case of directed graphs, to apply the joint embedding, one can symmetrize the graph by removing the direction of edges. Alternatively, $h_k h_k^T$ in Equation (1) can be replaced by $h_k g_k^T$, with h_k and g_k representing the in and out latent positions respectively. With this modification, Equation (1) becomes the tensor factorization problem [31].

The optimization problem in Equation (1) is similar to Principal Component Analysis (PCA) 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. Specifically, if $h_k h_k^T$ is replaced by a matrix $\mathbf{S}_k$ in Equation (1)

$$(\hat{\lambda}_1, \ldots, \hat{\lambda}_m, \hat{\mathbf{S}}_1, \ldots, \hat{\mathbf{S}}_d) = \underset{\lambda_i,\mathbf{S}}{\operatorname{argmin}} \sum_{i=1}^{m} \|\mathbf{A}_i - \sum_{k=1}^{d} \lambda_i[k]\mathbf{S}_k\|^2, \tag{2}$$

the problem can be solved by applying PCA on vectorized adjacency matrices. In this case, there is a $\mathbf{S}_k$ to estimate for each latent dimension which has $\frac{n(n+1)}{2}$ parameters. Compared to PCA, the joint embedding estimates a rank one matrix $h_k h_k^T$ for each latent dimension which has n parameters, and h_k can be treated as latent positions for vertices, but the joint embedding yields a larger approximation error due to the extra constraints. Similar optimization problems have also been considered in the simultaneous diagonalization literature [32], [33]. The difference is that the joint embedding is estimating an n-by-d matrix $\mathbf{H}$ 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 [11]. Next, we describe an algorithm to optimize the objective function $f(\mathbf{\Lambda}, \mathbf{H})$.

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. Alternating least squares (ALS) is a popular method to solve similar problems [31], [34], but often ignore symmetric constraints. Gradient approaches have also been considered for similar problems [35], [36]. We develop an alternating descent algorithm to minimize $f(\mathbf{\Lambda}, \mathbf{H})$ that combines ideas from both approaches [37]. The algorithm can also be understood as a block coordinate descent method with $\mathbf{\Lambda}$ and $\mathbf{H}$ being the two blocks [38], [39]. The algorithm iteratively updates one of $\mathbf{\Lambda}$ and $\mathbf{H}$ while treating the other parameter as fixed. Optimizing $\mathbf{\Lambda}$ when fixing H is straightforward, since it is essentially a least squares problem. However, optimizing $\mathbf{H}$ when fixing $\mathbf{\Lambda}$ is hard due to the fact that the problem is non-convex and there is no closed form solution available. In this case, the joint embedding algorithm utilizes gradient information and takes an Armijo backtracking line search strategy to update $\mathbf{H}$ [40].

Instead of optimizing all columns $\mathbf{\Lambda}$ and $\mathbf{H}$ simultaneously, we consider a greedy algorithm which solves the optimization problem by only considering one column of $\mathbf{H}$ at a time. Specifically, the algorithm fixes all estimates for the first $k_0 - 1$ columns of $\mathbf{\Lambda}$ and $\mathbf{H}$ at iteration k_0, and then the objective function is minimized by searching through only the k_0th column of $\mathbf{\Lambda}$ and $\mathbf{H}$. That is,

$$(\hat{\mathbf{\Lambda}}_{*k_0}, \hat{h}_{k_0}) = \underset{\mathbf{\Lambda}_{*k_0},\|h_{k_0}\|=1}{\operatorname{argmin}} \sum_{i=1}^{m} \|\mathbf{A}_i - \sum_{k=1}^{k_0-1} \hat{\mathbf{\Lambda}}_{ik}\hat{h}_k\hat{h}_k^T - \mathbf{\Lambda}_{ik_0}h_{k_0}h_{k_0}^T\|^2. \tag{3}$$

When $m = 1$, by the Eckart-Young theorem, the solution for $(\hat{\mathbf{\Lambda}}_{1,k_0}, \hat{h}_{k_0})$ is given by the leading eigenvalue and eigenvector of $\mathbf{A}_1 - \sum_{k=1}^{k_0-1} \hat{\mathbf{\Lambda}}_{1k}\hat{h}_k\hat{h}_k^T$, and hence the solution obtained by our greedy approach coincides with the eigendecomposition of $\mathbf{A}_1$. If $m > 1$ there is no closed-form solution in general, so we develop an optimization method next.

Let $f(\mathbf{\Lambda}_{*k_0}, h_{k_0})$ denote the sum on the left hand side of the equation. To compute a d-dimensional joint embedding $(\hat{\mathbf{\Lambda}}, \hat{\mathbf{H}})$, the algorithm iteratively finds one component at a time by solving the optimization for the one dimensional embedding problem (3). Once a new component k_0 is found, the matrix of loadings $\hat{\mathbf{\Lambda}}$ is updated by minimizing the optimization problem (1) with the first k_0 components fixed at $\hat{h}_1, \ldots, \hat{h}_{k_0}$, resulting in the least squares problem

$$(\hat{\mathbf{\Lambda}}_{*1}, \ldots, \hat{\mathbf{\Lambda}}_{*k_0}) = \underset{\mathbf{\Lambda}\in\mathbb{R}^{m\times k_0}}{\operatorname{argmin}} \sum_{i=1}^{m} \|\mathbf{A}_i - \sum_{k=1}^{k_0} \mathbf{\Lambda}_{ik}\hat{h}_k\hat{h}_k^T\|^2. \tag{4}$$

This step finds the projection of the graphs into the space spanned by the first k_0 components $\{\hat{h}_k\hat{h}_k^T\}_{k=1}^{k_0}$. The

algorithm solves the d-dimensional embedding problem by iteratively updating $\hat{\mathbf{H}}$ and $\hat{\mathbf{\Lambda}}$ using Equations (3) and (4).

There are a few advantages in iteratively solving one dimensional embedding problems. First, there are fewer parameters to fit at each iteration, since the algorithm is only allowed to vary h_{k_0} at iteration k_0. This makes initialization and optimization steps much easier compared to optimizing all columns of $\mathbf{H}$ simultaneously. Second, it implicitly enforces an ordering on the columns of $\mathbf{H}$. This ordering allows us to select the top few columns of $\mathbf{\Lambda}$ and $\mathbf{H}$ in cases where model selection is needed after the joint embedding. Third, it allows incremental computation. If d and d' dimensional joint embeddings are both computed, the first $\min(d, d')$ columns of $\hat{\mathbf{H}}$ will be the same. Fourth, although the global rank-d tensor approximation problem does not always have a solution for $d \geq 2$, a one dimensional embedding that corresponds to a rank-1 tensor approximation always have a solution [26]. 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 $\mathbf{\Lambda}_{*k_0}$ and h_{k_0} in Equation (3), the algorithm needs to evaluate two derivatives: $\frac{\partial f}{\partial h_{k_0}}$ and $\frac{\partial f}{\partial \mathbf{\Lambda}_{ik_0}}$. Denote by $\mathbf{R}_{ik_0}$ the residual matrix after iteration $k_0 - 1$ which is $\mathbf{A}_i - \sum_{k=1}^{k_0-1} \hat{\Lambda}_{ik} \hat{h}_k \hat{h}_k^T$. The gradient of the objective function with respect to h_{k_0} is given by

$$\frac{\partial f}{\partial h_{k_0}} = -4 \sum_{i=1}^{m} \mathbf{\Lambda}_{ik_0} (\mathbf{R}_{ik} - \mathbf{\Lambda}_{ik_0} h_{k_0} h_{k_0}^T) h_{k_0}. \tag{5}$$

The derivative of the objective function with respect to $\mathbf{\Lambda}_{ik_0}$ is given by

$$\frac{\partial f}{\partial \mathbf{\Lambda}_{ik_0}} = -2 \langle \mathbf{R}_{ik} - \mathbf{\Lambda}_{ik_0} h_{k_0} h_{k_0}^T, h_{k_0} h_{k_0}^T \rangle.$$

Setting the derivative to 0 yields

$$\hat{\mathbf{\Lambda}}_{ik_0} = \langle \mathbf{R}_{ik}, h_{k_0} h_{k_0}^T \rangle, \tag{6}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product.

Once a new component $\hat{h}_{k_0} \hat{h}_{k_0}^T$ is obtained, the algorithm proceeds to update $\mathbf{\Lambda}$ by solving the optimization problem (4). Note that the problem can be split into m least square subproblems, one for each graph $\mathbf{A}_i$, of the form

$$(\hat{\mathbf{\Lambda}}_{i1}, \ldots, \hat{\mathbf{\Lambda}}_{ik_0}) = \underset{\mathbf{\Lambda} \in \mathbb{R}^{k_0}}{\operatorname{argmin}} \|\mathbf{A}_i - \sum_{k=1}^{k_0} \mathbf{\Lambda}_k \hat{h}_k \hat{h}_k^T\|^2,$$

for each $i = 1, \ldots, m$. By obtaining the derivative with respect to $\mathbf{\Lambda}$ for all $i = 1, \ldots, m$ and setting them equal to zero, $\mathbf{\Lambda}$ is updated by solving m systems of linear equations with k_0 variables each. Let $\mathbf{\Gamma}$ be a $k_0 \times k_0$ matrix and $\mathbf{\Psi}$ be a $k_0 \times m$ matrix such that $\mathbf{\Gamma}_{kj} = (\hat{h}_j^T \hat{h}_k)^2$ and $\mathbf{\Psi}_{ki} = (\hat{h}_k^T \mathbf{A}_i \hat{h}_k)$. The algorithm updates the first k_0 columns of $\mathbf{\Lambda}$ by solving

$$\mathbf{\Gamma} \hat{\mathbf{\Lambda}}_{*,1:k_0}^T = \mathbf{\Psi}. \tag{7}$$

The joint embedding algorithm alternates between updating $\hat{\mathbf{\Lambda}}_{*k_0}$ and $\hat{h}_{k_0}$ according to Equations (5) and (6), after which the values of all the loadings $(\hat{\mathbf{\Lambda}}_{*1}, \ldots, \hat{\mathbf{\Lambda}}_{*k_0})$ are updated using Equation (7). Algorithm 1 describes the general procedure to compute the d-dimensional joint embedding of graphs $\{G_i\}_{i=1}^m$. The algorithm outputs two matrices: $\hat{\mathbf{\Lambda}}$ and $\hat{\mathbf{H}}$. The rows of $\hat{\mathbf{\Lambda}}$ denoted by $\{\hat{\lambda}_i\}_{i=1}^m$ can be treated as estimates of $\{\lambda_i\}_{i=1}^m$ in MREG and features for graphs. Columns of $\hat{\mathbf{H}}$ 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 $\mathbf{A}$, $\mathbf{A}$ can be projected into the linear space spanned by $\{\hat{h}_k \hat{h}_k^T\}_{k=1}^d$ to obtain features for the graph.

In case of $\mathbf{A}_i$ being large, the updating Equations (5) and (6) are not practical due to $h_k h_k^T$ and $\mathbf{R}_{ik}$ being large and dense. However, they can be rearranged to avoid explicit computation of $h_k h_k^T$ and $\mathbf{R}_{ik}$. Equation (5) becomes

$$\frac{\partial f}{\partial h_{k_0}} = -4 \sum_{i=1}^{m} \mathbf{\Lambda}_{ik_0} \mathbf{A}_i h_{k_0} + 4 \sum_{i=1}^{m} \mathbf{\Lambda}_{ik_0} \sum_{k=1}^{k_0-1} \mathbf{\Lambda}_{ik} (h_k^T h_{k_0}) h_k + 4 \times \sum_{i=1}^{m} \mathbf{\Lambda}_{ik_0}^2 h_{k_0}.$$

Similarly, Equation (6) can be rewritten as

$$\hat{\mathbf{\Lambda}}_{ik_0} = h_{k_0}^T \mathbf{A}_i h_{k_0} - \sum_{k=1}^{k_0-1} \mathbf{\Lambda}_{ik} (h_{k_0}^T h_k)^2.$$

Based on the rearranged equations, efficiently evaluating matrix vector product $\mathbf{A}_i h_{k_0}$ is needed to calculate the derivatives. This can be completed for a variety of matrices, in particular, sparse matrices [41].

The Algorithm 1 is guaranteed to converge to a stationary point. Specifically, at the termination of iteration k_0, $\frac{\partial f}{\partial h_{k_0}} \approx 0$ and $\frac{\partial f}{\partial \mathbf{\Lambda}_{ik_0}} \approx 0$. First, $\frac{\partial f}{\partial \mathbf{\Lambda}_{ik_0}} \approx 0$ is ensured due to exact updating by Equation (6). Second notice that updating according to Equations (5) and (6) always decreases the objective function. Due to the fact that h_{k_0} lies on the unit sphere and the objective is twice continuous differentiable, $\frac{\partial f}{\partial h_{k_0}}$ is Lipschitz continuous. This along with Armijo backtracking line search guarantees a "sufficient" decrease $c\|\frac{\partial f}{\partial h_{k_0}}\|^2$ in objective function each time when the algorithm updates h_{k_0} [40], where c is a constant independent of h_{k_0}. Since the objective function is bounded below by 0, this implies convergence of gradient, that is $\frac{\partial f}{\partial h_{k_0}} \to 0$.

The joint embedding of graphs model requires for identifiability that the basis of the embedding space $\{h_k h_k^T\}_{k=1}^m$ is linearly independent. This condition also ensures that Equation (4) has a unique solution. Although the optimization problem (3) does not directly enforce this constraint, the next theorem guarantees that the components obtained by our method are linearly independent. The proof is given in the appendix.

Theorem 3.1. *Suppose that* $\hat{\mathbf{\Lambda}} = (\hat{\mathbf{\Lambda}}_{*1}, \ldots, \hat{\mathbf{\Lambda}}_{*k_0})$ *and* $\hat{\mathbf{H}} = [\hat{h}_1, \ldots, \hat{h}_{k_0}]$ *are obtained by iteratively solving (3) and (4) for* k_0 *components, and the first* $k_0 - 1$ *components* $\{\hat{h}_k \hat{h}_k^T\}_{k=1}^{k_0-1}$ *are linearly independent. Then* $\{\hat{h}_k \hat{h}_k^T\}_{k=1}^{k_0}$ *are also linearly independent, or* $\hat{\mathbf{\Lambda}}_{ik_0} = 0$ *for all* $i = 1, \ldots, m$.

In general, the objective function may have multiple stationary points due to non-convexity. Therefore, the joint embedding algorithm is sensitive to initializations. Actually, like many of the problems in tensor factorization, finding the global minimum in joint embedding is NP-Hard [42]. When time permits, we recommend running the joint embedding several times including many random initializations. In Section 5.1, we study the effects of different initialization approaches through a numerical simulation experiment. For other simulation and real experiments, we initialize $\hat{h}_{k_0}$ using the top left singular vector of the average residual matrix $\sum \mathbf{R}_{ik_0}/m$. Computing this vector is computationally cheap compared to the computational cost of the joint embedding when the number of graphs m is larger than 1. When $m = 1$, the SVD initialization and the joint embedding solution coincide, and when $m > 1$ the SVD initialization gives a good approximation to the solution if all the graphs are identically distributed. 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. Next, we discuss some variations of the joint embedding algorithm.

Algorithm 1. Joint Embedding

```
procedure FIND JOINT EMBEDDING Λ̂, Ĥ 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 (5)
      Project h_k back to the unit sphere
      Fixing h_k, update Λ_{*k} by (6)
      Compute objective Σ_{i=1}^m ‖R_{ik} − Λ_{ik} h_k h_k^T‖^2
    end while
    Update Λ by solving (7)
    Update residuals: R_{i(k+1)} = A_i − Σ_{j=1}^k Λ_{ij} h_j h_j^T
  end for
  Output Λ̂ = [Λ_{*1}, ..., Λ_{*d}] and Ĥ = [h_1, ..., h_d]
end procedure
```

3.3 Variations

The joint embedding algorithm described in the previous section can be modified to accommodate several different 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 $\mathbf{\Lambda}$ be a vector in $\mathbb{R}^d$ and the optimization problem becomes

$$(\hat{\mathbf{\Lambda}}, \hat{\mathbf{H}}) = \underset{\mathbf{\Lambda}, \|h_k\|=1}{\operatorname{argmin}} \sum_{i=1}^{m} \|\mathbf{A}_i - \sum_{k=1}^{d} \mathbf{\Lambda}_k h_k h_k^T\|^2,$$

which is equivalent to

$$(\hat{\mathbf{\Lambda}}, \hat{\mathbf{H}}) = \underset{\mathbf{\Lambda}, \|h_k\|=1}{\operatorname{argmin}} \|\frac{1}{m}\sum_{i=1}^{m} \mathbf{A}_i - \sum_{k=1}^{d} \mathbf{\Lambda}_k h_k h_k^T\|^2.$$

Therefore, the optimization problem can be solved exactly by finding the singular value decomposition of the average adjacency matrix $\frac{1}{m}\sum_{i=1}^{m} \mathbf{A}_i$.

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

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

In this case, when updating $\mathbf{\Lambda}$ as in Equation (6), the algorithm should average $\mathbf{\Lambda}_{yk}$ within the same class, that is

$$\hat{\mathbf{\Lambda}}_{yk} = \left(\sum_{i=1}^{m} \mathbb{I}\{y_i = y\} \langle \mathbf{R}_{ik}, h_{k_0} h_{k_0}^T \rangle\right) \Big/ \left(\sum_{i=1}^{m} \mathbb{I}\{y_i = y\}\right).$$

Variation 3. In some applications, we may require all $\mathbf{\Lambda}_{ik}$ 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

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

To guarantee nonnegativity, the algorithm should use nonnegative least squares in updating $\mathbf{\Lambda}$ [43]. Furthermore, a constraint on the number of non-zero elements in ith row of $\mathbf{\Lambda}$ can be added as in K-SVD [44], and a basis pursuit algorithm could be used to update $\mathbf{\Lambda}$ [45], [46]. Next, we discuss some theoretical properties of the MREG model and joint embedding when treated as a parameter estimation procedure for the model.

4 Theoretical Results

In this section, we consider a simple setting where graphs follow a 1-dimensional MREG model, that is $\{(\lambda_i, \mathbf{A}_i)\}_{i=1}^m \sim MREG(F, h_1)$. The 1-dimensional joint embedding is well defined in this case, that is $\hat{\lambda}_i$ and $\hat{h}_1$ defined in Equation 1 is guaranteed to exist. Under this MREG model, the joint embedding of graphs can be understood as estimators for parameters of the model. Specifically, $\hat{\lambda}_i$ and $\hat{h}_1$ are estimates of λ_i and h. We prove 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:

$$\rho(\mathbf{A}_i, h) = \|\mathbf{A}_i - \langle \mathbf{A}_i, hh^T \rangle hh^T\|^2,$$

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

$$D(h, h_1) = \mathbb{E}(\rho(\mathbf{A}_i, h)).$$

One can understand D_m and D as sample and population approximation errors respectively. By Equation (1),

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

By Equation (6),

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

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 a global minimum of $D(h, h_1)$. Alternatively, the theorem implies the sample minimizer converges to the population minimizer.

Theorem 4.1. *The estimator $\hat{h}_1^m$ converges almost surely to the set of global minimizers of $D(h, h_1)$ as m goes to infinity. That is,*

$$\mathbb{P}\left(\lim_{m\to\infty} \hat{h}_1^m \in \underset{h}{\operatorname{argmin}} D(h, h_1) \right) = 1.$$

Theorem 4.1 ensures that, in the limit, $\hat{h}^m$ is arbitrarily close to parameters in the set of global minimizer of $D(h, h_1)$. However, the global minimizer 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. 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 h'. It gives a bound on the difference between the population minimizer h' and the truth h_1.

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

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

To see an application of Theorem 4.2, let us consider the case in which all graphs are Erdos-Renyi graphs with 100 vertices and edge probability of 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 as to 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

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

Taking a closer look at $E(\langle \mathbf{A}_i, hh^T \rangle^2)$,

$$\begin{aligned}\mathbb{E}(\langle \mathbf{A}_i, hh^T \rangle^2) &= \mathbb{E}(\langle \mathbf{P}_i, hh^T \rangle^2) + \mathbb{E}(\langle \mathbf{A}_i - \mathbf{P}_i, hh^T \rangle^2) \\ &= \mathbb{E}(\lambda_i^2)(h_1^T h)^4 + \mathbb{E}((h^T(\mathbf{A}_i - \mathbf{P}_i)h)^2).\end{aligned}$$

Therefore,

$$h' = \underset{\|h\|=1}{\operatorname{argmax}} \mathbb{E}(\lambda_i^2)(h_1^T h)^4 + \mathbb{E}((h^T(\mathbf{A}_i - \mathbf{P}_i)h)^2).$$

We can see that $\mathbb{E}(\lambda_i^2)(h_1^T h)^4$ is maximized when $h = h_1$; however, the noise term $\mathbb{E}((h^T(\mathbf{A}_i - \mathbf{P}_i)h)^2)$ is generally not maximized at $h = h_1$. If n is large, we can apply a concentration inequality to $(h^T(\mathbf{A}_i - \mathbf{P}_i)h)^2$ and have an upper bound on $\mathbb{E}((h^T(\mathbf{A}_i - \mathbf{P}_i)h)^2)$. If we further assume A_i is not too sparse, that is $\mathbb{E}(\lambda_i^2)$ 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 chapter, we focus on the case that n is fixed. The case that n goes to infinity for Random Dot Product Graph is considered in [47].

The two theorems above concern only the estimation of h_1, but not λ_i. Based on Equation (6), the joint embedding estimates λ_i by

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

When m goes to infinity, we can apply Theorem 4.1,

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

Then, applying the bound on $\|h' - h_1\|$ derived in Theorem 4.2 and utilizing the fact that $h^T \mathbf{A}_i h$ is continuous in h, we can obtain an upper bound on $|\hat{\lambda}_i^m - h_1^T \mathbf{A}_i h_1|$. When $\mathbf{A}_i$ is large, $h_1^T \mathbf{A}_i h_1$ is concentrated around λ_i with high probability. As a consequence, with high probability $|\hat{\lambda}_i^m - \lambda_1|$ is small. In the next section, we demonstrate properties and utilities of the joint embedding algorithm through experiments.

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 that has been studied for years under various settings [48]. Model selection is not the focus of this paper, but we still face this problem 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 first running the d'-dimensional joint embedding algorithm, where d' is sufficiently large. 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 $\{\hat{\mathbf{\Lambda}}_{ik}\}_{i=1}^m$ for $k = 1, \ldots, d'$, and select $\hat{d}$ when the loadings start to look like noise with 0 mean. These two approaches should yield a similar dimensionality estimate of $\hat{d}$.

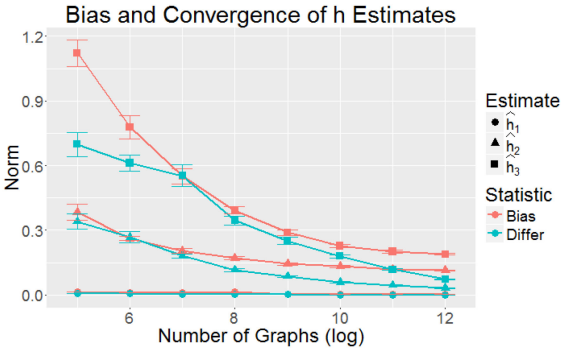


Fig. 2. Mean bias ($\|\hat{h}_k^m - h_k\|$) and mean difference between estimates ($\|\hat{h}_k^m - \hat{h}_k^{m/2}\|$) across 20 simulations are shown. The standard errors are also given by error bars. The graphs are generated from a 3-dimensional MREG model as described in Section 5.1. $\hat{h}_k^m$ has small asymptotic bias; however, it seems to converge as m increases.

5.1 Simulation Experiment 1: Joint Embedding Under a Simple Model

In the first experiment, we present a simple numerical example to demonstrate some properties of the joint embedding procedure as the number of graphs grows. We repeatedly generate graphs with 20 vertices from 3-dimensional MREG, where $\lambda_i[1] \sim \text{Uniform}(8, 16)$, $\lambda_i[2] \sim \text{Uniform}(0, 2)$ and $\lambda_i[3] \sim \text{Uniform}(0, 1)$, with

$$h_1 = [1, 1, 1, \ldots, 1]/\sqrt{20}$$
$$h_2 = [1, -1, 1, -1, 1, -1, \ldots, -1]/\sqrt{20}$$
$$h_3 = [1, 1, -1, -1, 1, 1, -1, -1, \ldots, -1]/\sqrt{20}.$$

We keep doubling the number of graphs m from 2^4 to 2^{12}. At each value of m, we compute the 3-dimensional joint embedding of graphs. Let the estimated parameters based on m graphs be denoted by $\hat{\lambda}_i^m$ and $\hat{h}_k^m$. Two quantities based on $\hat{h}_k^m$ are calculated. The first is the norm difference between the current h_k estimates and the previous estimates, namely $\|\hat{h}_k^m - \hat{h}_k^{m/2}\|$. This provides numerical evidence for the convergence of our principled estimation procedure. The second quantity is $\|\hat{h}_k^m - h_k\|$. This investigates whether $\hat{h}_k$ is an unbiased estimator for h_k. The procedure described above is repeated 20 times. Fig. 2 presents the result.

Based on the plot, the norm of differences $\|\hat{h}_k^m - \hat{h}_k^{m/2}\|$ seem to converge to 0 as m increases. This suggests the convergence of $\hat{h}_1^m$. Second, we notice that the bias $\|\hat{h}_2^m - h_2\|$ and $\|\hat{h}_3^m - h_3\|$ do not converge to 0; instead, it stops decreasing at around 0.1 and 0.2 respectively. This suggests that $\hat{h}_k$ is an asymptotically biased estimator for h_k. 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 [49]. In our case, there are infinitely many λ_i as m grows; therefore, we do not expect the joint embedding to provide an asymptotic consistent estimate of h_k.

In applications such as clustering or classifying multiple graphs, we may be not interested in $\hat{h}_k$. $\hat{\lambda}_i$ is of primary interest, which provides information specifically about the graphs G_i. Here, we consider two approaches to estimate

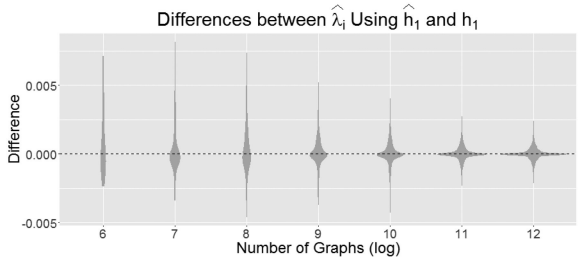


Fig. 3. Distribution of differences between $\hat{\lambda}_i[1]$ estimated using $\hat{h}_1^m$ and h_1. The graphs are generated from the three-dimensional MREG model as described in section 5.1. The differences are small due to the fact that $\hat{h}_1^m$ and h_1 are close.

$\lambda_i[1]$. The first approach is estimating $\lambda_i[1]$ through joint embedding, that is

$$\hat{\lambda}_i[1] = \langle \mathbf{A}_i, \hat{h}_1^m \hat{h}_1^{mT} \rangle.$$

The second approach estimates λ_i by assuming h_1 is known. In this case, Equation (6) gives

$$\hat{\lambda}_i[1] = \langle \mathbf{A}_i, h_1 h_1^T \rangle.$$

$\hat{\lambda}_i[1]$ calculated this way can be thought as the 'oracle' estimate. Fig. 3 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.

Next, we investigate the effects of four different initialization approaches. The first approach utilizes SVD on average residual matrix to initialize h_k at each iteration. The second approach randomly samples independent Gaussian variable for each entry of h_k. The third approach takes the best initialization among 10 random initializations. The fourth approach initializes h_k using the truth. To compare these approaches, we generate 16 graphs from the MREG model and jointly embed them with four different initializations. Then, another 16 graphs are generated and the objective function on these graphs are evaluated using $\hat{\mathbf{H}}$ estimated by joint embedding. This procedure is repeated 100 times. Mean objective function and total running time with standard error of these four approaches are shown in Table 1. Based on Wilcoxon signed-rank test, SVD and truth initializations are significantly better than random initializations on this scenario. For the rest experiments, the initialization is completed by SVD.

TABLE 1
Objective Function and Running Time of Four Initialization Approaches

Initialization	Objective	Running time (sec.)
SVD	375.22(1.21)	8.3(1.0)
1 Random	383.29(1.60)	9.2(1.4)
Best of 10 Random	379.63(1.39)	96.5(5.3)
Truth	374.69(1.22)	7.8(1.0)

The standard error is shown in parenthesis. SVD and truth initializations are significantly better than random initializations on this scenario.

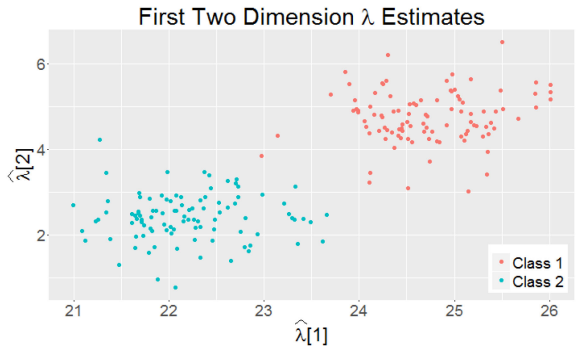


Fig. 4. Scatter plot of loadings computed by jointly embedding 200 graphs. The graphs are generated from the two-dimensional MREG model as described in Equation (8). The loadings of two classes are separated after being jointly embedded.

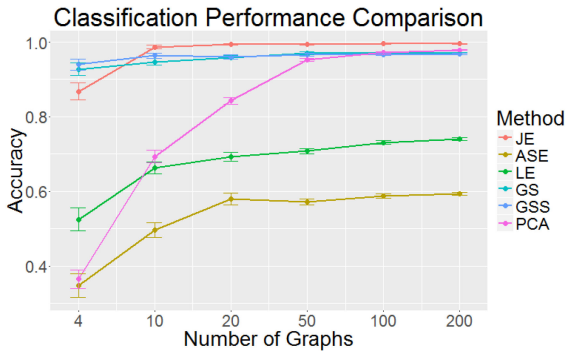


Fig. 5. Mean classification accuracy of joint embedding, Adjacency Spectral Embedding, Laplacian Eigenmap, Graph Statistics, Graph Spectral Statistics, and PCA with their standard errors are shown. The graphs are generated from a two-dimensional MREG model as described in the Equation (8). The features are first extracted using methods described above; subsequently, we apply a one-NN to classify graphs. For each value of m, the simulation is repeated 100 times. ASE, LE, GS, and GSS do not take advantage of increasing sample size in the feature extraction step. PCA has poor performance when the sample size is small. Joint embedding takes advantage of increasing sample size and outperforms other approaches when given more than 10 graphs.

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 $\{(\mathbf{A}_i, y_i)\}_{i=1}^m$ of observations. Each pair consists of an adjacency matrix $\mathbf{A}_i \in \{0,1\}^{n\times n}$ and a label $y_i \in [K]$. Furthermore, all pairs are assumed to be independent and identically distributed according to an unknown distribution $\mathbb{F}_{\mathbf{A},y}$, that is

$$(\mathbf{A}_1, y_1), (\mathbf{A}_2, y_2), \ldots, (\mathbf{A}_m, y_m) \overset{i.i.d.}{\sim} \mathbb{F}_{A,y}.$$

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

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

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

Here, h_2 has -0.1 as its first 50 entries and 0.1 as its last 50 entries. Graphs are generated according to the MREG model,

$$\{(\lambda_i, \mathbf{A}_i)\}_{i=1}^{200} \sim MREG(F, h_1, h_2), \tag{8}$$

where 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

$$A_i | y_i = 1 \sim SBM\left((1, \ldots, 1, 2, \ldots, 2), \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.3 \end{bmatrix}\right)$$

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

To classify graphs, we first jointly embed 200 graphs. The first two dimensional loadings are shown in Fig. 4. We can see two classes are separated after being jointly embedded.

Then, a 1-nearest neighbor classifier (1-NN) is constructed based on loadings $\{\hat{\lambda}_i\}_{i=1}^m$.

We compare classification performances of using the joint embedding to extract features to five other feature extraction approaches: Adjacency Spectral Embedding, Laplacian Eigenmap, Graph Statistics, Graph Spectral Statistics, and PCA. For Adjacency Spectral Embedding (ASE) and Laplacian Eigenmap (LE), we first embed each adjacency matrix or normalized Laplacian matrix and then compute the Procrustes distance between embeddings. For Graph Statistics (GS), we compute topological statistics of graphs considered by Park et al. in [7]. For Graph Spectral Statistics (GSS), we compute the eigenvalues of adjacency matrices and treat them as features [50]. For PCA, we vectorize the adjacency matrices and compute the factors through SVD. After the feature extraction step, 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. Fig. 5 shows the result. The joint embedding takes advantage of increasing sample size and outperforms other approaches when given more than 10 graphs.

5.3 Real Data Experiment 1: Subject Classification on HNU1 Data

In this section, we use the HNU1 data [51] to classify graphs from different subjects. These data consist of diffusion tensor imaging (DTI) records of the brain of 30 healthy different subjects, each of which was scanned 10 times over a period of one month. Each scan was processed with the NeuroData's MRI to Graphs (NDMG) pipeline [52] using the Talairach atlas [53] to register the vertices, and we obtained a sample of 300 graphs (10 graphs per subject), each of which is composed of 1105 vertices.

It has been suggested that the information encoded in patterns of brain connectivity can uniquely identify different subjects [54], [55], and there is some evidence of low-rank structure in those differences [56], [57]. We study how

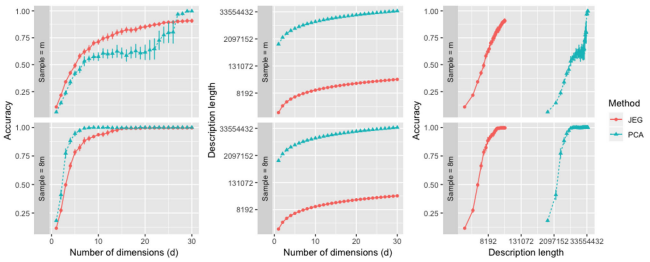


Fig. 6. Comparison of average classification accuracy and model complexity (number of embedding dimensions and description length) for JEG and PCA on the HNU1 data. A d dimensional representation is estimated using a training sample (top row: one graph per subject, bottom row: eight graphs per subject), after which all data is embedded, and the test data is classified using one-NN. The representations obtained by JEG are more accurate with fewer model parameters than PCA, especially when the sample size is small.

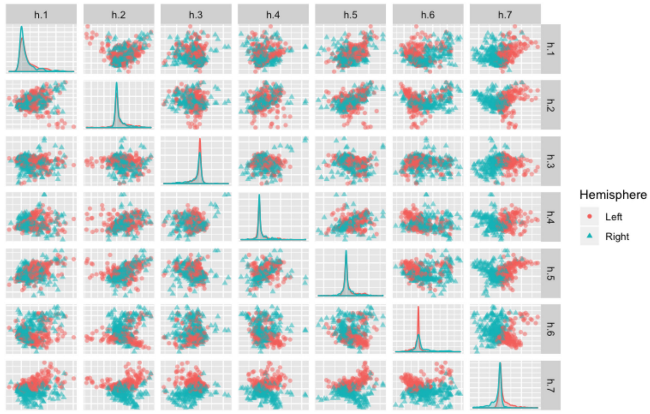


Fig. 7. Latent positions of the vertices found by JEG for the first seven dimensions using the HNU1 data. The color indicates the hemisphere side according to the Talairach atlas. The off-diagonal panels contain scatter plots of the vertices and diagonal plots show the density. Several dimensions of the embedding show a relationship between the latent positions and the hemisphere side.

low-dimensional representations can capture inter-individual variability by using the HNU1 data to classify subject scans. The joint embedding of graphs is an ideal method for this task, since it provides a low-rank representation of each of the graphs which is jointly learned from the sample.

The information encoded in the adjacency matrices can be used to accurately discriminate between different individuals. This can be confirmed by a 1-NN classifier using the vectorized adjacency matrices, which gives an almost perfect classification accuracy. However, our goal here is to study the accuracy of low-dimensional representations of the graphs to discriminate between subjects. Thus, we measure how does our joint dimensional embedding method perform as the number of embedding dimensions grows. We compare the performance of our method with PCA, as another dimensionality reduction method. Note that the joint embedding of graphs directly imposes a rank-one restriction on the components, enforcing a low-rank representation of the graphs. PCA instead represents the adjacency matrices using components that are usually full rank, and requires many more parameters to represent those components.

Before computing the embedding, all graphs are centered by the mean, that is, we compute $\tilde{\mathbf{A}}_i = \mathbf{A}_i - \frac{1}{300}\sum_{j=1}^{300} \mathbf{A}_j$. Using a 5-fold cross-validation, the data of each subject is divided into training and test sets. We measure the effect of the sample size using two different scenarios, the first uses 30 graphs on the training set (one scan per subject) and the second uses 240 graphs (eight scans per subject). The rest of the graphs are used in the test set to evaluate the accuracy, and the average over the 5 folds is computed. We follow the same procedure as in experiment 1. Using the training set, we compute an embedding of the graphs into d dimensions, either by doing JEG or PCA on the vectorized adjacency matrices. After that, all the data is projected into the d-dimensional embedding, and we use 1-NN to estimate the labels of the test data.

Fig. 6 shows a comparison of the average classification accuracy and model complexity for both methods. We measure model complexity as the number of embedding dimensions d, and as the total number of parameters used in each instance ($d(n+m)$ for JEG and $d(n^2+m)$ for PCA). In general, both methods are able to perform very accurate classification when d is large enough, but JEG uses far fewer parameters in the components for the same number of dimensions (see middle plot). The advantage of JEG is especially remarkable when the sample size is small (first row). In this case, JEG shows a better performance than PCA when d is not large. For values close to 30, PCA shows almost perfect performance, which is not surprising, since the maximum number of principal components that can be obtained by PCA is $m = 30$, and thus PCA is not really performing any dimensionality reduction when d is close to this value, but rather projecting onto the training data itself. JEG on the other hand is able to provide very accurate classification error in all scenarios using low-rank representations of the graphs with a fewer number of parameters that are interpretable. These can be observed in Fig. 7, which shows the latent positions of the vertices obtained by JEG for the first seven dimensions. To construct these plots, only the vertices that are labeled as left (507 of them) or right (525) were considered. As it can be observed, several dimensions of the latent positions show a structure that is clearly related to the hemisphere side. This structure is specially highlighted on the 7th dimension of the embedding.

5.4 Real Data Experiment 2: Joint Embedding to Cluster Vertices

In the previous experiments, we focus on feature extraction for graphs. Here, we consider a different task, that is spectral clustering through the joint embedding. In general, spectral clustering first computes (generalized) eigenvalues and eigenvectors of adjacency matrix or Laplacian matrix, then clusters the latent positions into groups [11], [12]. The cluster identities of latent positions become the cluster identities of vertices of the original graph. Adjacency Spectral Embedding (ASE) is one of the spectral clustering algorithms used to find the latent positions of the vertices to fit the SBM and RDPG [58], which are special cases of our MREG model. When applied to one graph, the joint embedding is equivalent to Adjacency Spectral Embedding (ASE). When given multiple graphs, the joint embedding can estimate latent positions for graph i as $[\hat{\lambda}_i[1]^{\frac{1}{2}}\hat{h}_1, \hat{\lambda}_i[2]^{\frac{1}{2}}\hat{h}_2, \ldots, \hat{\lambda}_i[d]^{\frac{1}{2}}\hat{h}_d]$ or equivalently $\hat{\mathbf{H}}\hat{\mathbf{D}}_i^{\frac{1}{2}}$. Alternatively, the matrix of components

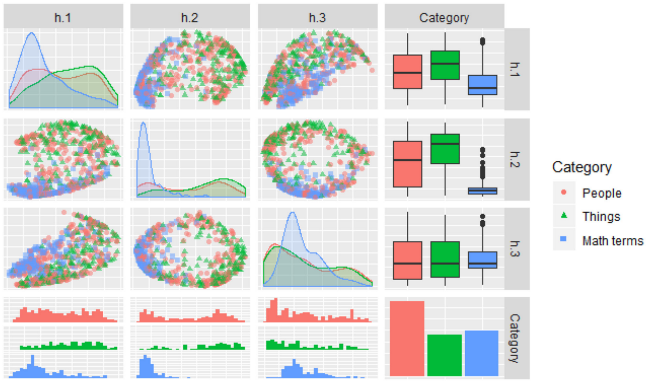


Fig. 8. The joint latent positions of $\hat{\mathbf{H}}$ estimated by the JE are shown. The first three plots on the diagonal are density estimates of latent positions for each dimension and category, and the last plot shows the number of points from each category. The first three plots of the last row show latent positions histograms for each dimension and category, and the first three plots of the last column are the corresponding box plots. The pairs plots of latent positions are given in the off-diagonal panels. The latent positions of math terms are separated from the other two clusters.

$\hat{H} = [\hat{h}_1, \ldots, \hat{h}_d]$ gives joint latent positions for all the graphs. Then, a clustering algorithm can be applied to the latent positions to obtain communities.

We apply this spectral clustering approaches to two Wikipedia graphs [59]. The vertices of these graphs represent Wikipedia article pages. Two vertices are connected by an edge if either of the associated pages hyperlinks to the other. Two graphs are constructed based on English webpages and French webpages. The full graph has 1382 vertices which represents articles within 2-neighborhood of "Algebraic Geometry". Based on the content of the associated articles, they are grouped by hand into 6 categories: people, places, dates, things, math terms, and categories.

We consider a subset of vertices from 3 categories: people, things, and math terms. After taking the induced subgraph of these vertices and removing isolated vertices, there are $n = 704$ vertices left. Specifically, 326, 181, and 197 vertices are from people, things and math terms respectively. We consider approaches to embed the graphs to obtain latent positions. First, we consider clustering of each graph separately by doing ASE on the English graph $\mathbf{A}_{en}$ (ASE +EN), or equivalently, JE on $\mathbf{A}_{en}$, and ASE on the French Graph $\mathbf{A}_{fr}$ (ASE+FR), and compare with the individual latent positions obtained by JE $\hat{\mathbf{H}}\hat{\mathbf{D}}_{en}^{\frac{1}{2}}$ (JE+EN) and $\hat{\mathbf{H}}\hat{\mathbf{D}}_{fr}^{\frac{1}{2}}$ (JE +FR). We also consider methods to estimate joint latent positions, by doing ASE on the mean of both graphs $\bar{\mathbf{A}} = (\mathbf{A}_{en} + \mathbf{A}_{fr})/2$ (ASE+(EN+FR)) [56], and the matrix $\hat{\mathbf{H}}$ obtained by JE on both graphs (JE+(EN,FR)). The dimension d is set to 3 for all approaches, and the latent positions are scaled to have norm 1 for degree correction. Then, we apply 3-means algorithm to the latent positions.

The joint latent positions of the graphs $\hat{\mathbf{H}}$ estimated by the joint embedding are provided on Fig. 8. The latent positions of math terms are separated from the other two clusters. However, the latent positions of people and things are mixed. Table 2 shows the clustering results measured by adjusted rand index and the purity of clustering [60], [61]. The standard error is estimated through repeatedly clustering bootstrapped latent positions. All methods yield clustering results which are significantly better than random. The English graph demonstrates clearer community structure than the French graph. The joint embedding produces latent positions that combine the information in both graphs, and leads to better clustering performance. The JE and ASE give similar results on the English graph, but JE is able to improve the clustering performance on the French graph significantly. The JE also shows better performance than ASE on $\bar{\mathbf{A}}$, which also uses the information of both graphs. These results show that our method provide interpretable representations for the vertices, with an accuracy comparable to state of the art methods for spectral clustering.

TABLE 2
Clustering Performance on Wikipedia Graphs

Method	ARI	Purity
ASE+EN	0.152 (0.02)	**0.555** (0.018)
JE+EN	**0.154** (0.019)	0.55 (0.019)
ASE+FR	0.114 (0.019)	0.521 (0.017)
JE+FR	**0.154** (0.019)	**0.549** (0.017)
ASE+(EN+FR)	0.155 (0.02)	0.547 (0.016)
JE+(EN, FR)	**0.156** (0.02)	**0.551** (0.021)

The adjusted rand index (ARI) and the purity of clustering of different spectral clustering approaches are shown. The first two scenarios consider each graph separately, and the third considers them jointly. The standard error (included in parenthesis) is estimated through repeatedly clustering bootstrapped latent positions. The joint embedding estimates latent positions which combine the information in both graphs with good clustering performance.

6 Conclusion

In summary, we developed a joint embedding method that can simultaneously embed multiple graphs into low dimensional space. The joint embedding can be utilized to estimate features 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, the joint embedding is a practically viable inference procedure. We also proposed a Multiple Random Eigen Graphs model. It can be understood as a generalization of the Random Dot Product Graph model or the Stochastic Block Model for multiple random graphs. We analyzed the performance of joint embedding on this model under simple settings. We demonstrated that the joint embedding method provides estimates with bounded error. Our approach is intimately related to other matrix and tensor factorization approaches such as singular value decomposition and CP decomposition. Indeed, the joint embedding and these algorithms all try to estimate a low dimensional representation of high dimensional objects through minimizing a reconstruction error. We are currently investigating the utility of 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.

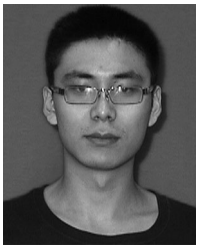
ACKNOWLEDGMENTS

The authors gratefully acknowledge support from the GRAPHS program of the Defense Advanced Research Projects Agency (DARPA) contract N66001-14-1-4028, the DARPA SIMPLEX program contract N66001-15-C-4041, the DARPA D3M program administered through AFRL contract FA8750-17-2-0112, and the DARPA MAA program contract FA8750-18-2-0035. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory and DARPA or the U.S. Government. Shangsi Wang and Jesús Arroyo equally contributed to the project.

REFERENCES

[1] E. Otte and R. Rousseau, "Social network analysis: a powerful strategy, also for the information sciences," *J. Inf. Sci.*, vol. 28, no. 6, pp. 441–453, 2002.
[2] R. Govindan and H. Tangmunarunkit, "Heuristics for internet map discovery," in *Proc. 19th Annu. Joint Conf. IEEE Comput. Commun. Societies*, 2000, vol. 3, pp. 1371–1380.
[3] E. T. Bullmore and D. S. Bassett, "Brain graphs: graphical models of the human brain connectome," *Annu. Rev. Clinical Psychol.*, vol. 7, pp. 113–140, 2011.
[4] M. D. Ward, K. Stovel, and A. Sacks, "Network analysis and political science," *Annu. Rev. Political Sci.*, vol. 14, pp. 245–264, 2011.
[5] I. Jolliffe, *Principal Component Analysis*. Hoboken, NJ, USA: Wiley, 2002.
[6] G. Li, M. Semerci, B. Yener, and M. J. Zaki, "Graph classification via topological and label attributes," in *Proc. 9th Int. Workshop Mining Learn. Graphs*, San Diego, USA, Aug. 2011, vol. 2.
[7] Y. Park, C. E. Priebe, and A. Youssef, "Anomaly detection in time series of graphs using fusion of graph invariants," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 1, pp. 67–75, Feb. 2013.
[8] C. Jiang, F. Coenen, and M. Zito, "A survey of frequent subgraph mining algorithms," *Knowl. Eng. Rev.*, vol. 28, no. 01, pp. 75–105, 2013.
[9] J. Huan, W. Wang, and J. Prins, "Efficient mining of frequent subgraphs in the presence of isomorphism," in *Proc. 3rd IEEE Int. Conf. Data Mining*, 2003, pp. 549–552.
[10] Z. Zhao and H. Liu, "Spectral feature selection for supervised and unsupervised learning," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 1151–1157.
[11] D. L. Sussman, M. Tang, D. E. Fishkind, and C. E. Priebe, "A consistent adjacency spectral embedding for stochastic blockmodel graphs," *J. Amer. Statistical Assoc.*, vol. 107, no. 499, pp. 1119–1128, 2012.
[12] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, 2003.
[13] S. J. Young and E. R. Scheinerman, "Random dot product graph models for social networks," in *Proc. Int. Workshop Algorithms Models Web-Graph*, 2007, pp. 138–149.
[14] P. Rubin-Delanchy, C. E. Priebe, M. Tang, and J. Cape, "A statistical interpretation of spectral embedding: The generalised random dot product graph," 2017, *arXiv preprint arXiv:1709.05506*.
[15] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Social Netw.*, vol. 5, no. 2, pp. 109–137, 1983.
[16] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed membership stochastic blockmodels," in *Proc. Conf. Advances Neural Inf. Process. Syst.*, 2009, vol. 9, pp. 34–41.
[17] M. Tang, D. L. Sussman, and C. E. Priebe, "Universally consistent vertex classification for latent positions graphs," *Ann. Statistics*, vol. 41, no. 3, pp. 1406–1430, 2013.
[18] D. L. Sussman, M. Tang, and C. E. Priebe, "Consistent latent position estimation and vertex classification for random dot product graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 1, pp. 48–57, Jan. 2014.
[19] P. Erdős and A. Rényi, "On random graphs," *Publicationes Mathematicae Debrecen*, vol. 6, pp. 290–297, 1959.
[20] B. Karrer and M. E. Newman, "Stochastic blockmodels and community structure in networks," *Physical Rev. E*, vol. 83, no. 1, 2011, Art. no. 016107.
[21] V. Lyzinski, M. Tang, A. Athreya, Y. Park, and C. E. Priebe, "Community detection and classification in hierarchical stochastic blockmodels," *IEEE Trans. Netw. Sci. Eng.*, vol. 4, no. 1, pp. 13–26, Mar. 2017.
[22] Q. Han, K. S. Xu, and E. M. Airoldi, "Consistent estimation of dynamic and multi-layer block models," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2014, pp. 1511–1520.
[23] T. P. Peixoto, "Inferring the mesoscale structure of layered, edge-valued, and time-varying networks," *Physical Rev. E*, vol. 92, no. 4, 2015, Art. no. 042807.
[24] A. Ghasemian, P. Zhang, A. Clauset, C. Moore, and L. Peel, "Detectability thresholds and optimal algorithms for community structure in dynamic networks," *Physical Rev. X*, vol. 6, no. 3, 2016, Art. no. 031005.
[25] D. Bini, M. Capovani, F. Romani, and G. Lotti, "O (n2. 7799) complexity for n× n approximate matrix multiplication," *Inf. Process. Lett.*, vol. 8, no. 5, pp. 234–235, 1979.
[26] V. De Silva and L.-H. Lim, "Tensor rank and the ill-posedness of the best low-rank approximation problem," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 1084–1127, 2008.
[27] J. Nolte, "The human brain: An introduction to its functional anatomy (No. 798)," *Mosby Inc.*, 1999.
[28] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter, "Multilayer networks," *J. Complex Netw.*, vol. 2, no. 3, pp. 203–271, 2014.
[29] J. Yan, Y. Li, W. Liu, H. Zha, X. Yang, and S. M. Chu, "Graduated consistency-regularized optimization for multi-graph matching," in *Proc. Eur. Conf. Comput. Vision*, 2014, pp. 407–422.
[30] H.-M. Park and K.-J. Yoon, "Encouraging second-order consistency for multiple graph matching," *Mach. Vision Appl.*, vol. 27, no. 7, pp. 1021–1034, 2016.
[31] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, 2009.
[32] B. N. Flury and W. Gautschi, "An algorithm for simultaneous orthogonal transformation of several positive definite symmetric matrices to nearly diagonal form," *SIAM J. Scientific Statistical Comput.*, vol. 7, no. 1, pp. 169–184, 1986.
[33] A. Ziehe, P. Laskov, G. Nolte, and K.-R. Müller, "A fast algorithm for joint diagonalization with non-orthogonal transformations and its application to blind source separation," *J. Mach. Learn. Res.*, vol. 5, pp. 777–800, 2004.
[34] 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.
[35] W. Tang, Z. Lu, and I. S. Dhillon, "Clustering with multiple graphs," in *Proc. 9th IEEE Int. Conf. Data Mining*, 2009, pp. 1016–1021.
[36] T. G. Kolda, "Numerical optimization for symmetric tensor decomposition," *Math. Program.*, vol. 151, no. 1, pp. 225–248, 2015.
[37] J. C. Bezdek and R. J. Hathaway, "Convergence of alternating optimization," *Neural Parallel Scientific Computations*, vol. 11, no. 4, pp. 351–368, 2003.
[38] S. J. Wright, "Coordinate descent algorithms," *Math. Program.*, vol. 151, no. 1, pp. 3–34, 2015.
[39] A. Beck and L. Tetruashvili, "On the convergence of block coordinate descent type methods," *SIAM J. Optimization*, vol. 23, no. 4, pp. 2037–2060, 2013.
[40] J. Nocedal and S. Wright, *Numerical Optimization*. Berlin, Germany: Springer, 2006.
[41] N. Bell and M. Garland, "Implementing sparse matrix-vector multiplication on throughput-oriented processors," in *Proc. Conf. High Performance Comput. Netw. Storage Anal.*, 2009, Art. no. 18.
[42] C. J. Hillar and L.-H. Lim, "Most tensor problems are NP-hard," *J. ACM*, vol. 60, no. 6, 2013, Art. no. 45.
[43] H. Kim and H. Park, "Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 2, pp. 713–730, 2008.
[44] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.
[45] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Rev.*, vol. 43, no. 1, pp. 129–159, 2001.

[46] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, May 2007.
[47] 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.
[48] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proc. 14th Int. Joint Conf. Artif. Intell.*, 1995, vol. 14, pp. 1137–1145.
[49] J. Neyman and E. L. Scott, "Consistent estimates based on partially consistent observations," *Econometrica*, vol. 16, no. 1, pp. 1–32, 1948.
[50] S. N. Dorogovtsev, A. V. Goltsev, J. F. Mendes, and A. N. Samukhin, "Spectra of complex networks," *Physical Rev. E*, vol. 68, no. 4, 2003, Art. no. 046109.
[51] X.-N. Zuo et al., "An open science resource for establishing reliability and reproducibility in functional connectomics," *Scientific Data*, vol. 1, 2014, Art. no. 140049.
[52] G. Kiar, W. G. Roncal, D. Mhembere, E. Bridgeford, R. Burns, and J. T. Vogelstein, "ndmg: Neurodata's mri graphs pipeline pipeline," Zenodo, 2016.
[53] J. Lancaster, "The Talairach Daemon, a database server for Talairach atlas labels," *Neuroimage*, vol. 5, no. 4, pp. 238–242, 1997.
[54] S. Wang, C. Priebe, C. Craddock, X.-N. Zuo, and M. Milham, "Optimal Design for Discovery Science: Applications in Neuroimaging," *Figshare Poster*, 2015. [Online]. Available: https://doi.org/10.6084/m9.figshare.1515021.v1
[55] E. S. Finn et al., "Functional connectome fingerprinting: Identifying individuals using patterns of brain connectivity," *Nature Neuroscience*, vol. 18, no. 11, 2015, Art. no. 1664.
[56] R. Tang, et al., "Connectome smoothing via low-rank approximations," *IEEE Trans. Medical Imaging*, vol. 38, no. 6, pp. 1446–1456, 2016.
[57] C. Sripada et al., "Fundamental units of inter-individual variation in resting state connectomes," *BioRxiv*, 2018, Art. no. 326082.
[58] V. Lyzinski, D. L. Sussman, M. Tang, A. Athreya, and C. E. Priebe, "Perfect clustering for stochastic blockmodel graphs via adjacency spectral embedding," *Electron. J. Statistics*, vol. 8, no. 2, pp. 2905–2922, 2014.
[59] S. Suwan, D. S. Lee, R. Tang, D. L. Sussman, M. Tang, and C. E. Priebe, "Empirical bayes estimation for the stochastic blockmodel," *Electron. J. Statistics*, vol. 10, no. 1, pp. 761–782, 2016.
[60] D. Steinley, "Properties of the hubert-arable adjusted rand index," *Psychological Methods*, vol. 9, no. 3, 2004, Art. no. 386.
[61] E. Rendón, I. Abundez, A. Arizmendi, and E. Quiroz, "Internal versus external cluster validation indexes," *Int. J. Comput. Commun.*, vol. 5, no. 1, pp. 27–34, 2011.



Shangsi Wang received the BS degree in mathematics and actuarial science from the University of Waterloo, Waterloo, Canada, in 2012, and the PhD degree from the Department of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, Maryland, in 2019. He is currently working in the financial services industry. His research interests include statistical inference on random netowrks and pattern recognition in graph datasets.

Jesús Arroyo received the BS degrees in applied mathematics and computer engineering from the Instituto Tecnológico Autónomo de México, Mexico City, Mexico, in 2013, and the MA and PhD degrees from the Department of Statistics, University of Michigan, Ann Arbor, Michigan, in 2018. He is postdoctoral fellow with the Center for Imaging Science, Johns Hopkins University, Baltimore, Maryland, since 2018. His research interests include statistical analysis of graphs and high dimensional data and applications to neuroimaging.

Joshua T. Vogelstein received the BS degree from the Department of Biomedical Engineering (BME), Washington University in St. Louis, in 2002, the MS degree from the Department of Applied Mathematics and Statistics, Johns Hopkins University (JHU), Baltimore, Maryland, in 2009, and the PhD degree from the Department of Neuroscience, Johns Hopkins University, in 2009. He was a postdoctoral fellow in AMS at JHU from 2009 until 2011, when he was appointed an assistant research scientist, and became a member of the Institute for Data Intensive Science and Engineering. He spent two years at Information Initiative at Duke, before becoming assistant professor in BME at JHU, and core faculty in the Institute for Computational Medicine and the Center for Imaging Science. His research interests include computational statistics, focusing on ultrahigh-dimensional, and non-Euclidean neuroscience data, especially connectomics.

Carey E. Priebe received the BS degree in mathematics from Purdue University, West Lafayette, Indiana, in 1984, the MS degree in computer science from San Diego State University, San Diego, CA, in 1988, and the PhD degree in information technology (computational statistics) from George Mason University, Fairfax, Virginia in 1993. From 1985 to 1994, he worked as a mathematician and scientist in the US Navy research and development laboratory system. Since 1994, he has been a professor in the Department of Applied Mathematics and Statistics, Johns Hopkins University (JHU), Baltimore, Maryland. His research interests include computational statistics, kernel and mixture estimates, statistical pattern recognition, statistical image analysis, dimensionality reduction, model selection, and statistical inference for high-dimensional and graph data. He is a senior member of the IEEE, lifetime member of the Institute of Mathematical Statistics, elected member of the International Statistical Institute, and a fellow of the American Statistical Association.