Joint Embedding of Graphs

Shangsi Wang

September 29, 2015

Abstract

In this paper, we propose an algorithm to jointly embed multiple graphs into lower dimensional space. The graphs represented in lower dimensional space can be classified or clustered easily with various standard algorithms. We demonstrate through simulation studies that our embedding algortihm leads to the best classification performance compared to the other existing embedding approaches.

1 Introduction

In many problems arising in science, graphs naturally appears as data to model complex relationship between objects. The graphs are high dimensional object with comlicated topological structure which makes graphs classification and clustering a challenge to traditional machine learning algorithms. In this paper, we propose an algorithm to jointly embed multiple graphs into lower dimensional space. The graphs represented in lower dimensional space then can be classified or clustered easily with various standard learning algorithms. We demonstrate through simulation studies that our graph embedding algortihm leads to the best classification performance compared to the other existing embedding approaches.

2 Setting

The joint embedding algorithm consumes a set of vertex mathced graphs. Although the algorithm works on weighted and directed graphs, we will focus on simple, unweighted and undirected graphs for simplicity. Let $\{G_i\}_{i=1}^n$ be n m-vertex graphs. We require the vertices in these graphs are matched which is all the graphs have the common vertex set V : $G_i = (V, E_i)$. The joint embedding algorithm tries to embed of all G_is simultaneously to $\mathbb{R}^d$. Before we introduce the joint embedding algorithm, we first introduce two random graph models which provide basis for statistical analysis and are also used to generate simulation data.

Definition (Stochasitic Block Model (SBM)) Let π be prior probability for block membership. Denote $\tau = (\tau_1, \tau_2, ..., \tau_m) \sim\in [K]^m$ be a block membership vector of vertices which follows multinomial distribution with probability π. $B \in [0,1]^{K\times K}$ is the block edge 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.

Definition (Random Dot Product Graph (RDPG)) Let $X = [X_1^T, X_2^T, ..., X_m^T] \in \mathbb{R}^{n\times d}$ be the latent positions for vertices. Suppose A is a random adjacency matrix given by,

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

Then we say that A is the adjacency matrix of a random dot product graph with latent position X of rank d.

3 Methodology

The joint embedding algorithm operates on the adjacency matrices of graphs. It tries to express all adjacency matrices as linear combinations of rank one symmetric matrices.

Definition (Joint Embedding) Given n graphs $\{G_i\}_{i=1}^n$. Let A_i be the associated adjacency matrix of G_i. The d-dimensional joint embeddings of graphs $\{G_i\}_{i=1}^n$ are represented by $\lambda \in R_{n\times d}$ and $H \in R_{m\times d}$ and they are given by,

$$(\lambda, H) = \arg\min_{\lambda,H} \sum_{i=1}^{n} \|A_i - \sum_{k=1}^{d} \lambda_{ik} h_k h_k^T\|_F^2$$

$\|\cdot\|_F$ denote the Frobenious norm and h_k is the kth column of H.

Here, $h_k h_k^T$ can be understood as the basis of graphs, and each graph G_i is represented as a linear combination of basis with loadings λ_{i*}. The joint embeddings of (λ, H) can be recovered by solving the optimization problem. At the first glance, we could try to minimize the objective through gradiant descent on λ and H. The problem is that the optimization problem is not convex in H. Therefore, if we cannot innitialize H carefullt, we will very likely end up in a bad local minimum. This problem can be mitigated a little by perform gradiant descent algorithm several times with random innitialization. We propose a greedy algorithm which solves the optimization problem dimension by dimension. Specifically at iteration k_0, we find k_0th basis and loadings λ_{k_0} by treating all previous dimensions fixed that is,

$$(\lambda_{k_0}, h_{k_0}) = \arg\min_{\lambda_{k_0},h_{k_0}} \sum_{i=1}^{n} \|A_i - \sum_{k=1}^{k_0-1} \lambda_{ik} h_k h_k^T - \lambda_{ik_0} h_{k_0} h_{k_0}^T\|_F^2$$

In general, to compute d dimensional embeddings λ and H, we recursively solve the one dimensional problem above. To solve the one dimensional problem, we take a gradient descent approach which iteratively update λ_{k_0} and h_{k_0}. If we denote the objective function on the right hand side as $f(\lambda_{k_0}, h_{k_0})$ and the redidual matrix $R_{ik_0} = A_i - \sum_{k=1}^{k_0-1} \lambda_{ik} h_k h_k^T$. The gradiant of the objective with repect to h_{k_0} are given by,

$$\frac{\partial f}{\partial h_{k_0}} = \sum_{i=1}^{n} 4(R_{ik} - \lambda_{ik_0} h_{k_0} h_{k_0}^T) h_{k_0}$$

When updating λ_{k_0}, we can just regress R_{ik_0} on $h_{k_0} h_{k_0}$. It yields,

$$\arg\min_{\lambda_{k_0}} f(\lambda_{k_0}, h_{k_0}) = \frac{< h_{k_0} h_{k_0}^T, R_{ik} >}{\| h_{k_0} h_{k_0}^T \|_F^2}$$

Algorithm 1 describes framework to find joint embeddings of graphs.

Algorithm 1 Joint Embedding Algorithm

```
1: procedure FIND JOINT EMBEDDINGS $\lambda, H$ OF $\{A_i\}_{i=1}^n$
2:     Set residuals: $R_{i1} = A_i$
3:     for $k = 1 : d$ do
4:         Innitialize $h_k$ and $\lambda_k$
5:         while not convergent do
6:             Compute objective $\sum_{i=1}^{n} \|R_{ik} - \lambda_{ik} h_k * h_k^T\|_F^2$
7:             Fixing $h_k$, update $\lambda_k$ by regression
8:             Fixing $\lambda_k$, update $h_k$ through gradient descent
9:         end while
10:        Update residuals: $R_{i(k+1)} = R_{ik} - \lambda_{ik} h_k * h_k^T$
11:    end for
12:    Output $\lambda = [\lambda_1, \lambda_2, ..., \lambda_d]$ and $H = [h_1, h_2, ..., h_d]$
13: end procedure
```

The basic idea of the algorthm is to recursively find the best rank 1 approximation to the residual. It is similar to the least angle regression algorithm. The difference between least angle regression and joint embedding is that joint embedding need also compute its predictors $h_k * h_k^T$. In the case of only one graph, the joint embedding is equivalent to adjacency spectral embedding algorithm (ASE). The basic joint embedding algorithm above can be modified to accomodate several settings.

Variation 1. When we are confident all graphs comes from the same model, we can force all λ_{ik}s to be equal across graphs. This is useful when primary inference task is to learn embeddings of vertices. Since all graphs share the same loadings, we only need $\lambda \in \mathbb{R}^d$ and the optimization problem becomes,

$$(\lambda, H) = \arg\min_{\lambda, H} \sum_{i=1}^{n} \|A_i - \sum_{k=1}^{d} \lambda_k h_k h_k^T\|_F^2$$

In this case, the optimization problem can be solved exactly by singular value decomposition of the average matrix of pointwise square of A_i.

Variation 2. When we have a label $Y_i \in \mathbb{Y}$ associated with graph G_i, we may require all λ_{ik}s to be equal within class. That is $\lambda_{ik} = \lambda_{yk}$ for all i with $Y_i = y$.

If we let $\lambda \in \mathbb{R}^{|\mathbb{Y}|,d}$, the optimization problem becomes,

$$(\lambda, H) = \arg\min_{\lambda,H} \sum_{i=1}^{n} \|A_i - \sum_{k=1}^{d} \lambda_{Y_i k} h_k h_k^T\|_F^2$$

In this case, when updating λ algorithm 1 should to regress all R_{ik} from the same class simultaneously.
Variation 3. We may require all λ_{ik}s to be greater than 0 like in the non-negative matrix fatorization. The advantage of this constrain is that we may automatically cluster graph G_i based on largest λ_{ik}.

$$(\lambda, H) = \arg\min_{\lambda \geq 0,H} \sum_{i=1}^{n} \|A_i - \sum_{k=1}^{d} \lambda_{ik} h_k h_k^T\|_F^2$$

To solve this problem, the algorithm can be modified to apply iterative shrinkage thresholding on λ.

4 An Simple Experiment

In this section, we consider two simple experiments which demonstrates joint embedding can effectively embed graphs. In particular, we will look into performance of classification using embeddings of graphs.

4.1 Experiment 1

In this experiment, we randomly independently generate 100 100-vertex graphs from two stochastic block models. Specifically,

$$\{G_i\}_{i=1}^{100} \sim SBM([0.5, 0.5], \begin{bmatrix} 0.5 & 0.2 \\ 0.2 & 0.5 \end{bmatrix}) \text{ , and}$$

$$\{G_i\}_{i=101}^{200} \sim SBM([0.5, 0.5], \begin{bmatrix} 0.6 & 0.2 \\ 0.2 & 0.5 \end{bmatrix})$$

The 200 graphs are jointly embedded into 2 dimensonal Euclidean space. The λ is as shown in the figure 1. By computing pairwise distances of λ and fitting a 3-NN classifier, the classification accuracy is 1.

4.2 Experiment 2

In this experiment, we randomly independently generate 200 100-vertex graphs from random dot product graph models. Specifically,

$$\{G_i\}_{i=1}^{100} \sim RDPG(X_i) \text{ , here } X_i \sim Dirchlet([1, 1]/c) \text{ , and}$$

$$\{G_i\}_{i=101}^{200} \sim RDPG(X_i) \text{ , here } X_i \sim Dirchlet([1, 2]/c)$$

Here, c is a concentration parameter which controls how much latent positons X_i are concentrated around its mode. The 200 graphs are then jointly embedded into 2 dimensonal Euclidean space. We compare joint embedding to two other embedding approaches. The first one is adjacency spectral embedding with procrust distance; the second one is Omni-embedding with Euclidean distance. Again, we use a k-NN classifier to compute classification accuracy. In addition, reliability of resulting embeddings is reported. Figure 2 shows the results. We can see from the result that joint embedding is clearly the best.

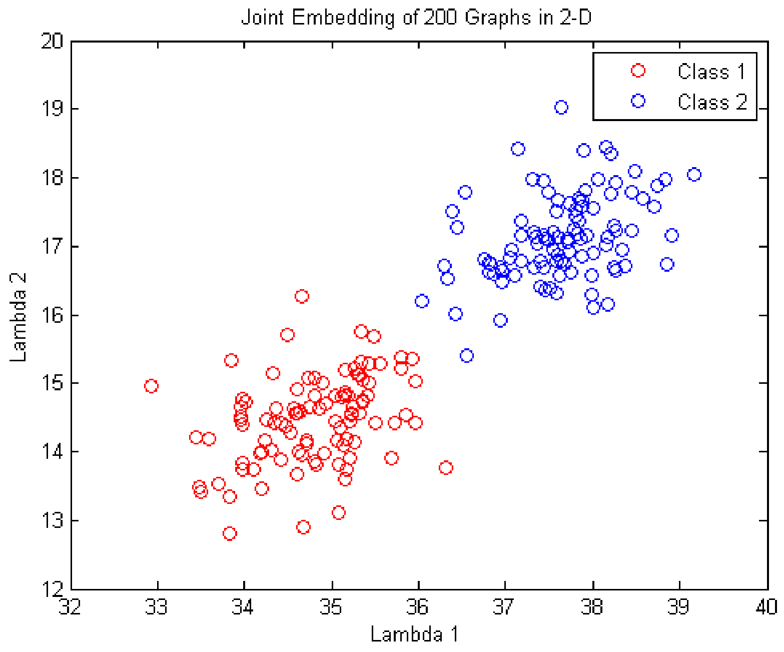

Figure 1: Loadings of Graphs

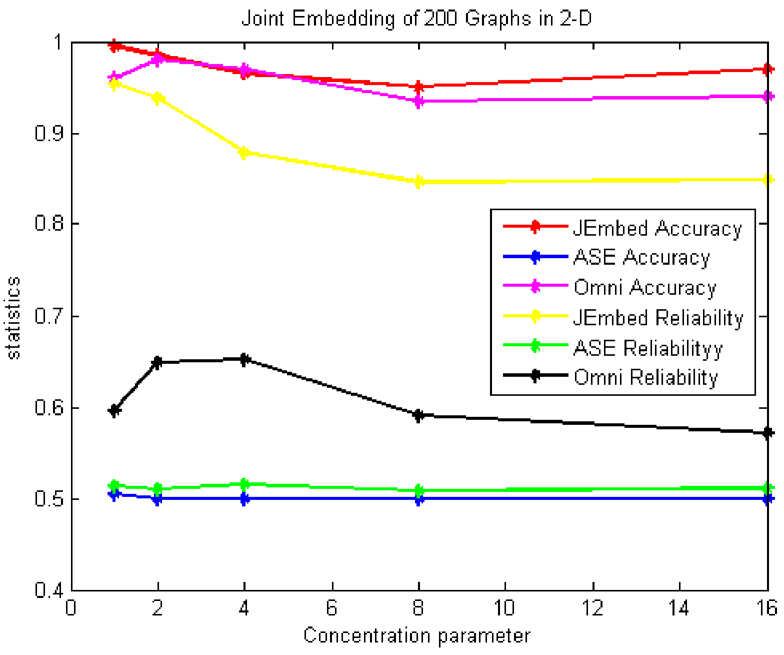

Figure 2: Classification Accracy of Embeddings