

DeepWalking Backwards: From Embeddings Back to Graphs

Sudhanshu Chanpuriya^{*} Cameron Musco[†] Konstantinos Sotiropoulos[‡]
Charalampos E. Tsourakakis[§]

Abstract

Low-dimensional node embeddings play a key role in analyzing graph datasets. However, little work studies exactly *what information is encoded* by popular embedding methods, and how this information correlates with performance in downstream learning tasks. We tackle this question by studying whether embeddings can be *inverted* to (approximately) recover the graph used to generate them. Focusing on a variant of the popular DeepWalk method [Perozzi et al., 2014, Qiu et al., 2018], we present algorithms for accurate embedding inversion – i.e., from the low-dimensional embedding of a graph G , we can find a graph \tilde{G} with a very similar embedding. We perform numerous experiments on real-world networks, observing that significant information about G , such as specific edges and bulk properties like triangle density, is often lost in \tilde{G} . However, community structure is often preserved or even enhanced. Our findings are a step towards a more rigorous understanding of exactly what information embeddings encode about the input graph, and why this information is useful for learning tasks.

1 Introduction

Low-dimensional node embeddings are a primary tool in graph mining and machine learning. They are used for node classification, community detection, link prediction, and graph generative models. Classic approaches like spectral clustering [Shi and Malik, 2000, Ng et al., 2002], Laplacian eigenmaps [Belkin and Niyogi, 2003], IsoMap [Tenenbaum et al., 2000], and locally linear embeddings [Roweis and Saul, 2000] use spectral embeddings derived for the graph Laplacian, adjacency matrix, or their variants. Recently, neural-network and random-walk-based embeddings have become popular due to their superior performance in many settings. Examples include DeepWalk [Perozzi et al., 2014], node2vec [Grover and Leskovec, 2016], LINE [Tang et al., 2015], NetMF [Qiu et al., 2018], and many others [Cao et al., 2016, Kipf and Welling, 2016, Wang et al., 2016]. In many cases, these methods can be viewed as variants on classic spectral methods, producing an approximate low-dimensional factorization of an implicit matrix representing graph structure [Qiu et al., 2018].

Problem definition. We focus on the following high-level question:

What graph properties are encoded in and can be recovered from node embeddings?
How do these properties correlate with learning tasks?

We study the above question on undirected graphs with non-negative edge weights. Let \mathcal{G} denote the set of all such graphs with n nodes. We formalize the question via Problems 1 and 2 below.

^{*}UMass Amherst. schanpuriya@umass.edu

[†]UMass Amherst. cmusco@cs.umass.edu

[‡]Boston University. ksotirop@bu.edu

[§]ISI Foundation & Boston University. tsourolampis@gmail.com

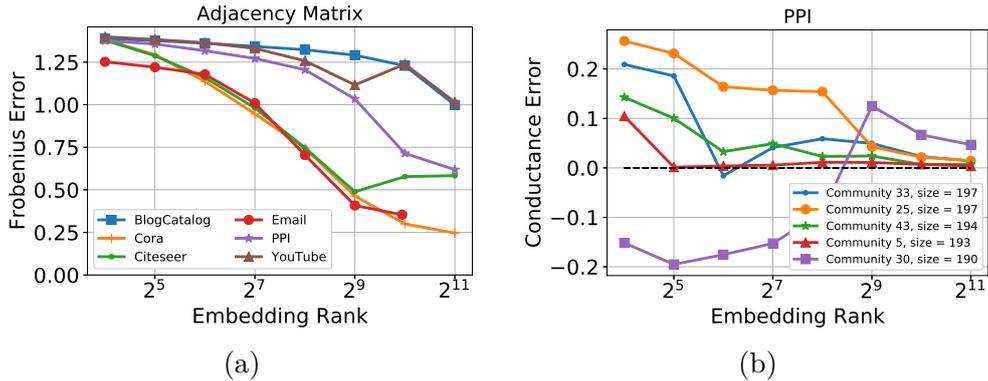


Figure 1: (a) Relative Frobenius error $\|A - \tilde{A}\|_F / \|A\|_F$ between the adjacency matrices of G and \tilde{G} . (b) Relative error between G and \tilde{G} for the conductances of the five largest communities (corresponding to biological states) in a human protein-protein interaction network.

Problem 1 (Embedding Inversion). *Given an embedding algorithm $\mathcal{E} : \mathcal{G} \rightarrow \mathbb{R}^{n \times k}$ and the embedding $\mathcal{E}(G)$ for some $G \in \mathcal{G}$, produce $\tilde{G} \in \mathcal{G}$ with $\mathcal{E}(\tilde{G}) = \mathcal{E}(G)$ or such that $\|\mathcal{E}(\tilde{G}) - \mathcal{E}(G)\|$ is small for some norm $\|\cdot\|$.*

We refer to k as the *embedding dimension*. A solution to Problem 1 lets us approximately invert the embedding $\mathcal{E}(G)$ to obtain a graph. It is natural to ask what structure is common between G, \tilde{G} . Using the same notation as Problem 1, our second problem is as follows.

Problem 2 (Graph Recovery). *Given G, \tilde{G} such that $\|\mathcal{E}(\tilde{G}) - \mathcal{E}(G)\|$ is small for some matrix norm $\|\cdot\|$, how close are G, \tilde{G} in terms of common edges, degree sequence, triangle counts, and community structure?*

Answering Problems 1 and 2 is an important step towards a better understanding of a node embedding method \mathcal{E} . We focus on the popular DeepWalk method of Perozzi et al. [2014]. DeepWalk embedding can be interpreted as low-rank approximation of a pointwise mutual information (PMI) matrix based on node co-occurrences in random walks [Goldberg and Levy, 2014]. The NetMF method of Qiu et al. [2018] directly implements this low-rank approximation using SVD, giving a variant with improved performance in many tasks. Due to its mathematically clean definition, we use this variant. Many embedding methods can be viewed similarly – as producing a low-rank approximation of some graph-based similarity matrix. We expect our methods to extend to such embeddings.

Our contributions. We make the following findings:

- We prove that when the embedding dimension k is equal to n and the node embedding method is NetMF in the limit as the co-occurrence window size parameter goes to infinity, then solving a linear system can provably recover G from $\mathcal{E}(G)$, i.e., find $\tilde{G} = G$.
- We present two algorithms for solving Problem 1 on NetMF embeddings in typical parameter regimes. The first is inspired by the above result, and relies on solving a linear system. The second is based on minimizing $\|\mathcal{E}(G) - \mathcal{E}(\tilde{G})\|_F$, where $\|\cdot\|_F$ is the matrix Frobenius norm, using gradient based optimization.
- Despite the non-convex nature of the above optimization problem, we show empirically that our approach successfully solves Problem 1 on a variety of real word graphs, for a range of embedding dimensions used frequently in practice. We show that, typically our optimization based algorithm outperforms the linear system approach with respect to producing a graph \tilde{G} with embeddings closer to those of the input graph G .

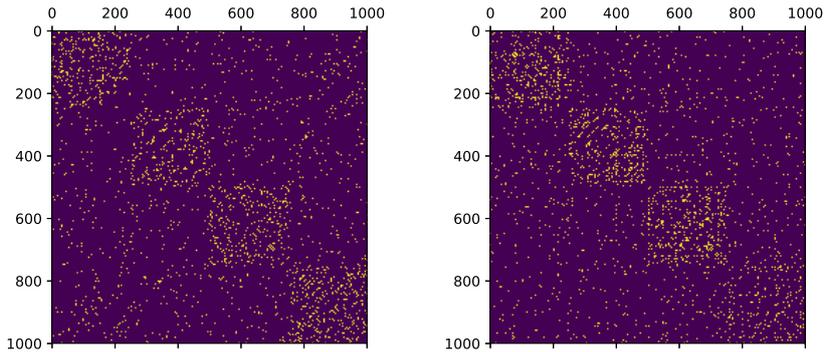


Figure 2: G (left), a stochastic block model graph with 1000 nodes and 4 clusters, and \tilde{G} (right), a reconstruction of G from a 32-dimensional NetMF embedding. While G and \tilde{G} differ in the exact edges they contain, we can see that the community structure is preserved.

- We study Problem 2 by applying our optimization algorithm to NetMF embeddings for a variety of real world graphs. We compare the input graph G and the output of our inversion algorithm \tilde{G} across different criteria. Our key findings include the following:

1. **Fine-Grained Edge Information.** As the embedding dimension k increases *up to a certain point* \tilde{G} tends closer to G , i.e., the Frobenius norm of the difference of the adjacency matrices gets smaller. After a certain point, the recovery algorithm is trying unsuccessfully to reconstruct fine grained edge information that is “washed-out” by NetMF. Figure 1(a) illustrates this finding for a popular benchmark of datasets (see Section 4 for more details).
2. **Graph properties.** We focus on two fundamental graph properties, counts of triangles and community structure. Surprisingly, while the number of triangles in G and \tilde{G} can differ significantly, community structure is well-preserved. In some cases this structure is actually enhanced/emphasized by the embedding method. I.e., the conductance of the same community in \tilde{G} is even lower than in G .

Figure 1(b) shows the relative error between the conductance of a ground-truth community in G and the conductance of the same community in \tilde{G} vs. k for the five largest communities in a human protein-protein interaction network.

Figure 2 provides another visual summary of the above findings. Specifically, it shows on the left the spy plot of a stochastic block model graph with 1000 nodes and four clusters, and on the right the spy plot of the output of our reconstruction algorithm from a 32-dimensional NetMF embedding of the former graph. The two graphs differ on exact edges, but the community structure is preserved.

2 Related work

Graph recovery from embeddings. To the best of our knowledge, Problem 1 has not been studied explicitly in prior work. Hoskins et al. [2018] study graph recovery using a partial set of effective resistance measurements between nodes – equivalent to Euclidean distances for a certain embedding, see Section 4 of [Spielman and Srivastava, 2011]. Close to our work lies recent work on node embedding privacy, and in particular graph reconstruction attacks on these embeddings. Ellers et al. [2019] identify neighbors of a given node v with good accuracy by considering the change in embeddings of the other nodes in G and $G \setminus v$. Duddu et al. [2020] study a graph

reconstruction attack that inverts a simple spectral embedding using a neural network. Training this network requires knowledge of a random subgraph of G , used as training data, and can be viewed as solving Problem 1, but with some auxiliary information provided on top of $\mathcal{E}(G)$.

Graph sketching algorithms study the recovery of information about G (e.g., approximations to all its cuts or shortest path distances) from linear measurements of its edge-vertex incidence matrix [McGregor, 2014]. These linear measurements can be thought of as low-dimensional node embeddings. However, generally they are designed specifically to encode certain information about G , and they differ greatly from the type of embeddings used in graph learning applications. Recently, Chanpuriya et al. [2020] showed that any graph with degree bounded by Δ admits an embedding into $2\Delta + 1$ dimensions that can be *exactly inverted*. These exact embeddings allow for a perfect encoding of the full graph structure in low-dimensions, and circumvent limitations of a large family of embeddings that cannot capture triangle richness and edge sparsity *provably* in low dimensions [Seshadhri et al., 2020].

DeepWalk and NetMF. We focus on inverting embeddings produced by the Qiu et al. [2018] NetMF variant of the popular DeepWalk method of Perozzi et al. [2014]. Consider an undirected, connected, non-bipartite graph G , with adjacency matrix $A \in \{0, 1\}^{n \times n}$, diagonal degree matrix $D \in \mathbb{R}^{n \times n}$ and volume $v_G = \text{tr}(D) = \sum_{i,j} A_{i,j}$. Qiu et al. show that, for window size hyperparameter T (typical settings are $T = 10$ or $T = 1$), DeepWalk stochastically factorizes the *pointwise mutual information (PMI) matrix*:

$$\hat{M}_T = \log \left(\frac{v_G}{T} \sum_{r=1}^T (D^{-1}A)^r D^{-1} \right),$$

where the logarithm is applied entrywise to its $n \times n$ argument. Note that if the diameter of G exceeds T , then at least one entry of $\sum_{r=1}^T (D^{-1}A)^r D^{-1}$ will be 0. To avoid taking the logarithm of 0, NetMF instead employs the *positive pointwise mutual information (PPMI) matrix*:

$$M_T = \log \left(\max \left(1, \frac{v_G}{T} \sum_{r=1}^T (D^{-1}A)^r D^{-1} \right) \right). \quad (1)$$

Via truncated eigendecomposition of M_T , one can find an eigenvector matrix $V \in \mathbb{R}^{n \times k}$ and a diagonal eigenvalue matrix $W \in \mathbb{R}^{k \times k}$ such that $M_{T,k} = VWV^\top$ is the best possible k -rank approximation of M_T in the Frobenius norm. The NetMF embedding is set to the eigenvectors scaled by the square roots of the eigenvalue magnitudes. I.e., $\mathcal{E}(G) = V\sqrt{|W|}$, where the absolute value and the square root are applied entrywise. In practice, these node embeddings perform at least as well as DeepWalk in downstream tasks. Further, their deterministic nature lets us to define a straightforward optimization model to invert them.

3 Proposed methods

In Sections 3.1 and 3.2 we present our two proposed NetMF embedding inversion methods. The first is inspired by our constructive proof of Theorem 1 and relies on solving an appropriately defined linear system. The second is based on optimizing a natural objective using a gradient descent algorithm. Since the NetMF embedding $\mathcal{E}(G)$ encodes the best k -rank approximation $M_{T,k} = VWV^\top$ to the positive pointwise mutual information (PPMI) matrix M_T , we will assume throughout that we are given $M_{T,k}$ directly and seek to recover \tilde{G} from this matrix. We also assume knowledge of the number of edges in G in terms of the volume v_G .

While all networks used in our experiments are unweighted, simple, undirected graphs, i.e., their adjacency matrices are binary ($A \in \{0, 1\}^{n \times n}$), our inversion algorithms produce \tilde{G} with $\tilde{A} \in [0, 1]^{n \times n}$. The real valued edge weights in \tilde{G} can be thought of as representing edge probabilities. We will also convert \tilde{G} to an unweighted graph with binary adjacency matrix $\tilde{A}_b \in \{0, 1\}^{n \times n}$. We describe the binarization process in detail in the following sections.

3.1 Analytical Approach

We leverage a recent asymptotic result of Chanpuriya and Musco [2020], which shows that as the number of samples and the window size T for DeepWalk/NetMF tend to infinity, the PMI matrix tends to the limit:

$$\lim_{T \rightarrow \infty} T \cdot \hat{M}_T = \hat{M}_\infty = v_G \cdot D^{-1/2}(\bar{L}^+ - I)D^{-1/2} + J, \quad (2)$$

where $\bar{L} = I - D^{-1/2}AD^{-1/2}$ is the normalized Laplacian, \bar{L}^+ is the Moore-Penrose pseudoinverse of this matrix, and J is the all-ones matrix. Our first observation is that if, in addition to \hat{M}_∞ , we are given the degrees of the vertices in G , then we know both D and v_G , and we can simply invert equation (2) as follows:

$$\begin{aligned} \bar{L} &= \left(D^{1/2} \left(\frac{\hat{M}_\infty - J}{v_G} \right) D^{1/2} + I \right)^+ \\ A &= D^{1/2} (I - \bar{L}) D^{1/2}. \end{aligned} \quad (3)$$

In Appendix A.1, we show using just the graph volume v_G , that one can perfectly recover the degree matrix D from \hat{M}_∞ via a linear system, provided the adjacency matrix of G is full-rank. Combining this fact with Equations (2) and (3) we obtain the following:

Theorem 1 (Limiting Invertibility of Full-Rank PMI Embeddings). *Let G be an undirected, connected, non-bipartite graph with full-rank adjacency matrix $A \in \{0, 1\}^{n \times n}$ and volume v_G . Let \hat{M}_T be the PMI matrix of G which is produced with window size T . There exists an algorithm that takes only \hat{M}_T and v_G as input and recovers A exactly in the limit as $T \rightarrow \infty$.*

In our embedding inversion task, rather than the exact limiting PMI matrix \hat{M}_∞ , we are given the low-rank approximation $M_{T,k}$ of the finite- T PPMI matrix, through the NetMF embeddings. Our first algorithm is based on essentially ignoring this difference. We use $M_{T,k}$ to obtain an approximation to \hat{M}_∞ , which we then plug into (3). This approximation is based on inverting the following limit, shown by Chanpuriya and Musco [2020]:

$$\lim_{T \rightarrow \infty} \hat{M}_T = \log \left(\frac{1}{T} \hat{M}_\infty + J \right), \quad (4)$$

where the logarithm is applied entrywise.

Due to the various approximations used, the elements of the reconstructed adjacency matrix \tilde{A} may not be in $\{0, 1\}$, and may not even be in $[0, 1]$; for this reason, as in Seshadhri et al. [2020], we apply an entrywise clipping function, $\text{clip}(x) = \min(\max(0, x), 1)$, after the inversion steps from Equations (3) and (4). The overall procedure is given in Algorithm 1.

Algorithm 1 DeepWalking Backwards (Analytical)

input approximation $M_{T,k}$ of true T -step PPMI, window-size T , degree matrix D , graph volume v_G

output reconstructed adjacency matrix $\tilde{A} \in [0, 1]^{n \times n}$

- 1: $\tilde{M}_\infty \leftarrow T \cdot (\exp(M_{T,k}) - J)$ \triangleright exp is applied entrywise, J is the all-ones matrix
 - 2: $\tilde{\bar{L}} \leftarrow \left(D^{1/2} \left(\frac{\tilde{M}_\infty - J}{v_G} \right) D^{1/2} + I \right)^+$
 - 3: $\tilde{A} \leftarrow \text{clip} \left(D^{1/2} (I - \tilde{\bar{L}}) D^{1/2} \right)$
 - 4: **return** \tilde{A}
-

Binarization. To produce a binary adjacency matrix $\tilde{A}_b \in \{0, 1\}^{n \times n}$ from \tilde{A} , we use a slight modification of Algorithm 1: rather than clipping, we set the highest v_G off-diagonal entries above the diagonal to 1, and their symmetric counterparts below the diagonal to 1. This ensures that the matrix represents an undirected graph \tilde{G} with the same number of edges as G .

3.2 Optimization Approach

Our gradient based approach parameterizes the entries of a real valued adjacency matrix $\tilde{A} \in (0, 1)^{n \times n}$ with independent logits for each potential edge, and leverages the differentiability of Equation (1). Based on \tilde{A} , we compute the PPMI matrix \tilde{M}_T , and then the squared PPMI error loss, i.e., the squared Frobenius error between \tilde{M}_T and the low-rank approximation $M_{T,k}$ of the true PPMI, given by the NetMF embeddings. We differentiate through these steps, update the logits, and repeat. Pseudocode is given in Algorithm 2.

Since the input to the algorithm is a low-rank approximation of the true PPMI, and since this approximation is used for the computation of error, it may seem more appropriate to also compute a low-rank approximation of the reconstructed PPMI matrix \tilde{M}_T prior to computing the error; we skip this step since eigendecomposition within the optimization loop is both computationally costly and unstable to differentiate through.

Note that we invoke a “shifted logistic” function σ_v which constructs an adjacency matrix with a given target volume. The pseudocode for this function is given in Algorithm 3. This algorithm is an application of Newton’s method. We find that 10 iterations are sufficient for convergence in our experiments.

Our implementation uses PyTorch [Paszke et al., 2019] for automatic differentiation and minimizes the loss using the SciPy [Jones et al., 2001] implementation of L-BFGS [Liu and Nocedal, 1989, Zhu et al., 1997] with default hyperparameters and a maximum of 500 iterations.

Algorithm 2 DeepWalking Backwards (Optimization)

input approximation $M_{T,k}$ of true T -step PPMI, window-size T , graph volume v_G , number of iters. N

output reconstructed adjacency matrix $\tilde{A} \in (0, 1)^{n \times n}$

- 1: Initialize elements of $X \in \mathbb{R}^{(n \times n)}$ to 0 ▷ logits of the reconstructed adjacency matrix
 - 2: **for** $i \leftarrow 1$ to N **do**
 - 3: $\tilde{A} \leftarrow \sigma_{v_G}(X)$ ▷ construct adjacency matrix with target volume, see Algorithm 3
 - 4: $\tilde{M}_T \leftarrow \text{PPMI}(\tilde{A})$ via Eq. (1)
 - 5: $L \leftarrow \|\tilde{M}_T - M_{T,k}\|_F^2$ ▷ squared error of PPMI
 - 6: Calculate $\partial_X L$ via automatic differentiation through Steps 3 to 5
 - 7: Update X to minimize L using $\partial_X L$
 - 8: **end for**
 - 9: **return** $\sigma_v(X)$
-

Algorithm 3 Shifted Logistic Function σ_v

input logit matrix $X \in \mathbb{R}^{(n \times n)}$, target sum $v \in (0, n^2)$, number of iterations I

output matrix $A \in (0, 1)^{n \times n}$ which sums approximately to v

- 1: $s \leftarrow 0$
 - 2: **for** $i \leftarrow 1$ to I **do**
 - 3: $A \leftarrow \sigma(X + s)$ ▷ σ is the logistic function applied entrywise
 - 4: $s \leftarrow s + \frac{v - \Sigma(A)}{\Sigma(A \circ (1 - A))}$ ▷ Σ sums over all elements, and \circ is an entrywise product
 - 5: **end for**
 - 6: **return** $\sigma(X + s)$
-

Binarization. We binarize the reconstructed $\tilde{A} \in (0, 1)^{n \times n}$ differently from the prior approach. We treat each element of \tilde{A} as the parameter of a Bernoulli distribution and sample independently to produce $\tilde{A}_b \in \{0, 1\}^{n \times n}$. Since we set \tilde{A} ’s volume to be approximately v_G using the σ_v function, the number of edges in the binarized network after sampling is also $\approx v_G$.

4 Experimental results

4.1 Experimental setup

Datasets. We apply the NetMF inversion algorithms described in Section 3 to a benchmark of networks, summarized in Table 1. As part of our investigation of how well the output \tilde{G} of our methods matches the underlying graph G , we examine how community structure is preserved. For this reason, we choose only test graphs with labeled ground-truth communities. All datasets we use are publicly available: see Qiu et al. [2018] for BLOGCATALOG and PPI, Sen et al. [2008] for CITESEER and CORA, and SNAP [Leskovec and Krevl, 2014] for EMAIL and YOUTUBE. The YOUTUBE graph we use is a sample of 20 communities from the raw network of Leskovec and Krevl [2014]. For all networks, we consider only the largest connected component. The community labels that we report for various datasets, such as those reported in the legends of Figure 5, refer to the labels as given in the input datasets.

Name	Nodes	Edges	# Labels
BLOGCATALOG	10,312	333,983	39
E-MAIL	986	16,064	42
PPI	3,852	76,546	50
CORA	2,485	10,138	7
CITESEER	2,110	7,388	6
YOUTUBE	10,617	55,864	20

Table 1: Datasets used in our experiments.

Hyperparameter settings. We experiment with a set of different values for the embedding dimension k , starting from 2^4 and incrementing in powers of 2, up to $2^{11} = 2048$, except for the EMAIL dataset, which has fewer than 2^{10} nodes. For this dataset we only test for k up to 2^9 . Throughout the experiments, we set the window-size T to 10, as this is the most commonly used value in downstream machine learning tasks.

Evaluation. Our first step is to evaluate how well the two algorithms proposed in Section 3 solve embedding inversion (Problem 1). To do this, we measure the error in terms of the relative Frobenius error between the rank- k approximations of the true and reconstructed PPMI matrices, $M_{T,k}$ and $\tilde{M}_{T,k}$ respectively. These matrices represent the NetMF embeddings of G and \tilde{G} . The relative Frobenius error for two matrices X and \tilde{X} is simply $\|X - \tilde{X}\|_F / \|X\|_F$.

We next study how the reconstructed graph \tilde{G} obtained via embedding inversion compares with the true G (Problem 2). Here, we binarize the reconstructed adjacency matrix to produce \tilde{A}_b . See Sections 3.1 and 3.2 for details. Thus, like G , \tilde{G} is an undirected, unweighted graph. Most directly, we measure the relative Frobenius error between G 's adjacency matrix A and \tilde{G} 's adjacency matrix \tilde{A}_b . We also measure the reconstruction error for three other key measures:

- **Number of triangles (τ).** The total number of 3-cliques, i.e., triangles, in the graph.
- **Average path length (ℓ).** The average path length between any two nodes in the graph.
- **Conductance (ϕ) of ground-truth communities.** For a community S , the conductance is defined as: $\phi(S) = \frac{e(S;\bar{S})}{\min(\text{vol}(S), \text{vol}(\bar{S}))}$ where $e(S;\bar{S})$ is the number of edges leaving community S and $\text{vol}(S)$ is number of edges induced by S . \bar{S} is the complement $V \setminus S$.

For the above measures we report the relative error between the measure x for the true network and the one of the recovered network \tilde{x} , defined as $(\tilde{x} - x)/x$.

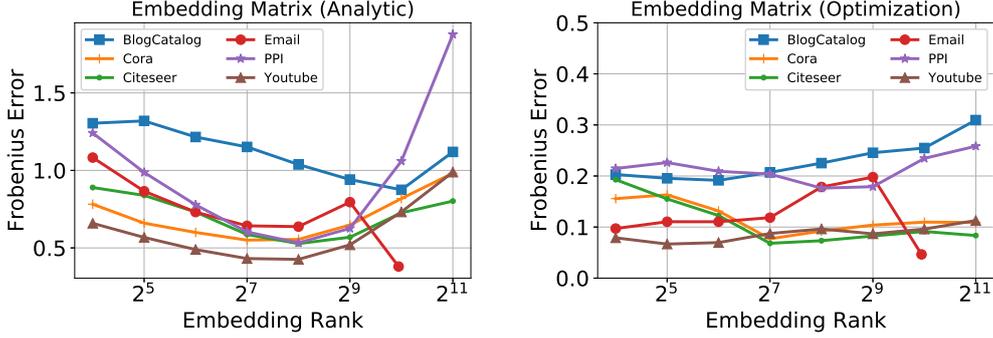


Figure 3: Relative Frobenius error vs. embedding rank k for the low-rank PPMI matrices of the graphs reconstructed using the inversion algorithms: the analytical approach, Alg. 1 (left), and the optimization approach, Alg. 2 (right). For details, see Section 4.2.

Finally, we evaluate how well \tilde{G} 's low-dimensional embeddings perform in classification, where the goal is to infer the labels of the nodes of G . We train a linear model using a fraction of the labeled nodes of G and the low-dimensional embedding of \tilde{G} , and try to infer the labels of the remaining nodes. We report accuracy in terms of micro F1 score and compare it with the accuracy when using the low-dimensional embedding of G itself. For this task, we use both the recovered real-valued adjacency matrix of \tilde{G} and its binarized version. We observe that, contrary to the previous measures, performance is sensitive to binarization.

Code. All code is written in Python and is available at https://github.com/konsotirop/Invert_Embeddings.

Summary of findings. Before we delve into details, we summarize our key findings.

- The optimization approach (Alg. 2), significantly outperforms the analytical approach (Alg. 1), in terms of how closely the NetMF embeddings of the reconstructed graph \tilde{G} match those of the true graph G (i.e., in solving Problem 1). See Figure 3.
- Focusing on \tilde{G} produced by Algorithm 2, the NetMF embedding is close to the input at all ranks. The adjacency matrix error of \tilde{G} trends downwards as the embedding rank k increases. However, for small k , the two graph topologies can be very different in terms of edges and non-edges. See Figure 4.
- \tilde{G} preserves and or even enhances the community structure present in G , and tends to preserve the average path length. However, the number of triangles in \tilde{G} greatly differs from that in G when the embedding rank k is low. See Figure 4.
- \tilde{G} 's NetMF embeddings perform essentially identically to G 's in downstream classification on G . However, binarization has a significant effect: if we first binarize \tilde{G} 's edge weights, and then produce embeddings, there is a drop in classification performance.
- Overall, we are able to invert NetMF embeddings as laid out in Problem 1 and, in the process, recover \tilde{G} with similar community structure to the true graph G . Surprisingly, however, \tilde{G} and G can be very different graphs in terms of both specific edges and broader network properties, despite their similar embeddings.

4.2 Analytical vs. Optimization Based Inversion

Figure 3 reports the relative Frobenius error of the analytical method (Alg. 1) and the optimization approach (Alg. 2) in embedding inversion as we range k . We can see that Alg. 2 *significantly* outperforms Alg. 1. While Alg. 1 comes with strong theoretical guarantees (Theorem 1) in asymptotic settings (i.e., $T \rightarrow \infty$, $k = n$), it performs poorly when these conditions are violated.

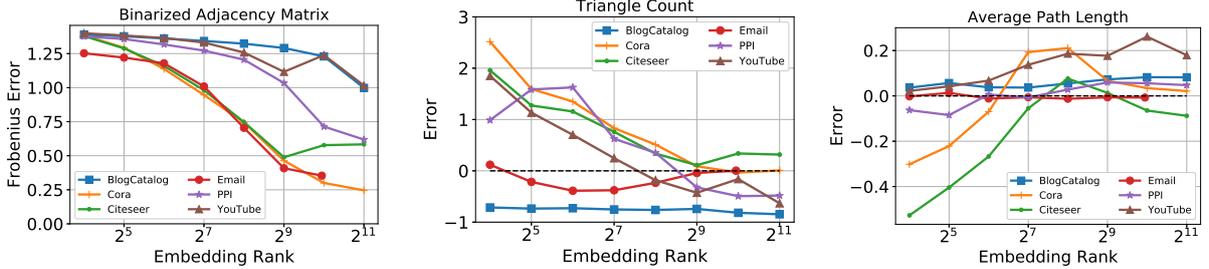


Figure 4: From left to right: Relative Frobenius error for the binarized adjacency matrix; relative error for the number of triangles; and relative error for the average path length. All plots are versus the embedding rank.

In practice, the embedding dimension k is always set to be less than n (typical values are 128 or 256), and T is finite (T is often set to 10). At these settings, the approximations used in Alg. 1 seem to severely limit its performance.

Given the above, in the following sections we focus our attention on the optimization approach. This approach makes no assumption on the rank k , or the window-size T . We can see in Figure 3 that the embedding error stays low across different values of k when using Alg. 2, indicating that performance is insensitive to the dimension parameter.

4.3 Evaluating Graph Recovery

Adjacency matrix reconstruction. We next examine how closely the output of Alg. 2, the binarized adjacency matrix \tilde{A}_b , matches the original adjacency matrix A , especially as we vary the embedding dimensionality k . As can be seen in Figure 4, at low ranks, the relative Frobenius error is often quite high – near 1. In combination with Figure 3 (left), this shows an interesting finding: two graphs may be very different topologically, but still have very similar low-dimensional node embeddings (i.e., low-rank PPMI matrices). We do observe that as the embedding dimension grows, the adjacency matrix error decreases. This aligns with the message of Theorem 1 that, in theory, high dimensional node embeddings yield enough information to facilitate full recovery of the underlying graph G . We remark that, by construction, G and \tilde{G} have approximately the same number of edges. Thus, the incurred Frobenius error is purely due to a reorientation of the specific edges between the true and the reconstructed networks.

Recovery of graph properties. Bearing in mind that the recovered \tilde{G} differs substantially from the input graph G in the specific edges it contains, we next investigate whether the embedding inversion process at least recovers bulk graph properties.

Figure 4 shows the relative error of the triangle count versus embedding dimensionality k . We observe that the number of triangles can be hugely different among the true and the reconstructed networks when k is small. In other words, there exist networks with similar low-dimensional NetMF embeddings that differ significantly in their total number of triangles. This is surprising: since the number of triangles is an important measure of local connectivity, one might expect it to be preserved by the node embeddings. In contrast, for another important global property, the average path length, the reconstruction error is always relatively low (also shown in Figure 4).

In Figure 5, we plot the relative errors for the conductances of the five most populous communities of the networks under consideration. We see that the conductance of ground-truth communities is generally preserved in the reconstructed networks, with the error becoming negligible after rank $2^7 = 128$, an embedding rank which is often used in practice. This finding is intuitive – since NetMF embeddings are used for node classification and community detection, it is to be expected that they preserve community structure.

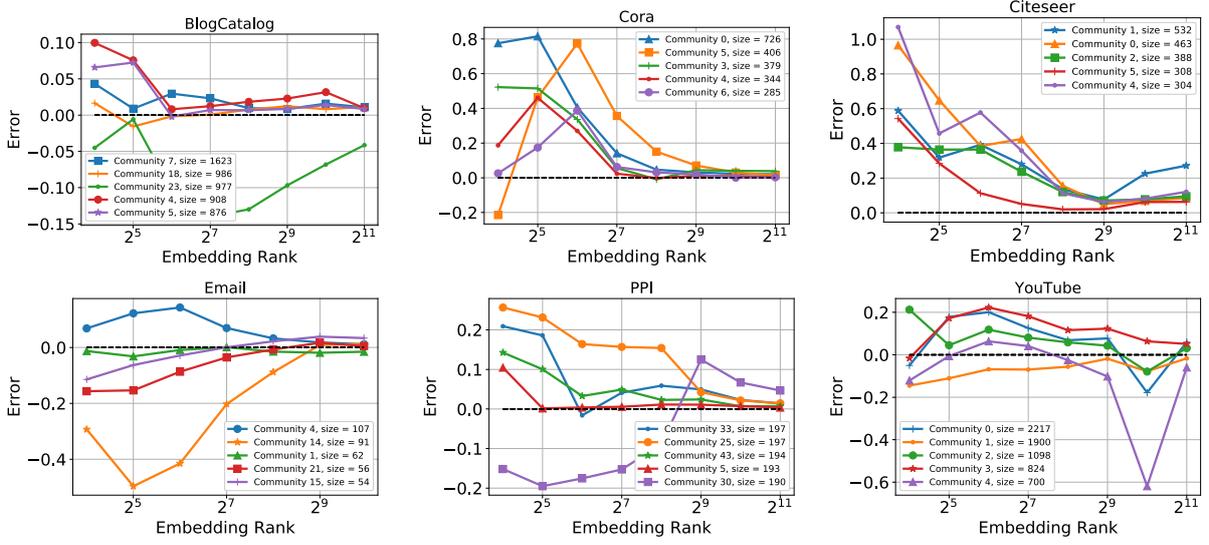


Figure 5: Relative error for the conductances of the five largest communities for each of the selected networks.

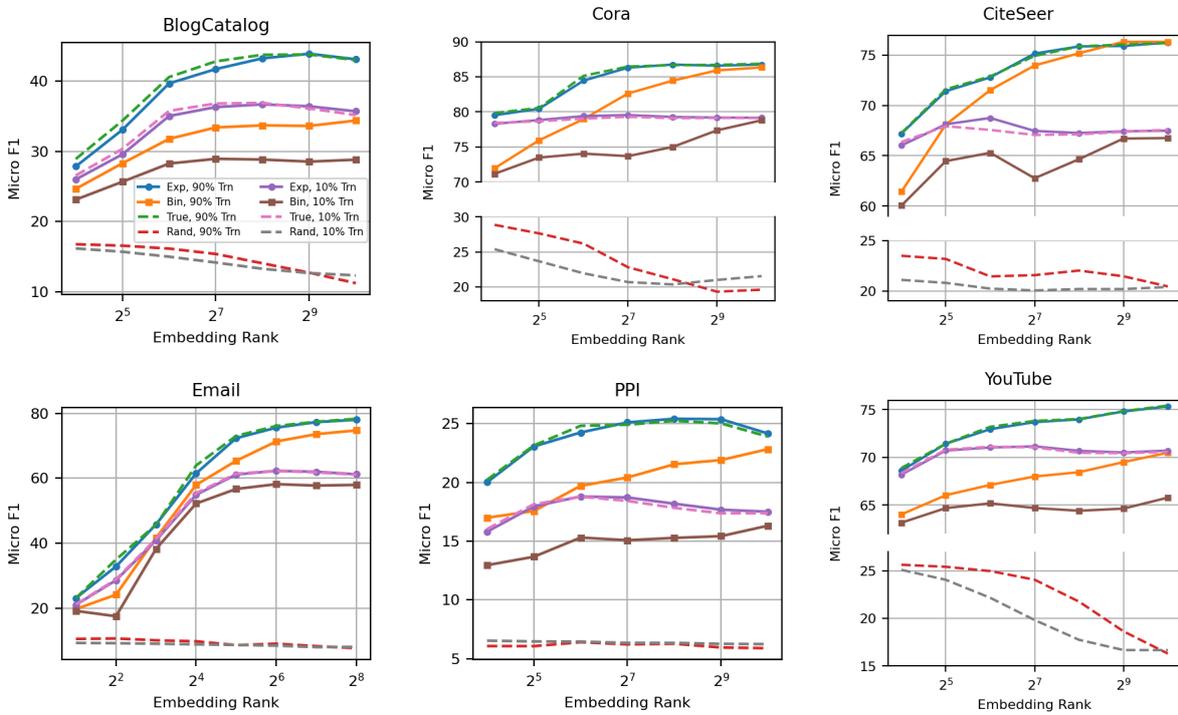


Figure 6: Multi-label classification using embeddings from reconstructed networks. Performance when using embeddings from a random graph is included as a baseline.

Node classification. In a typical classification setting for a graph G , when we know only a fraction of the labels of its nodes and want to infer the rest, we can use a low-dimensional embedding of its nodes as our feature matrix and employ a linear classifier to infer the labels for the remaining nodes. While our reconstructed networks \tilde{G} differ from G edge-wise, they have similar low-dimensional NetMF embeddings. As another indicator of the preservation of community structure, we measure the performance in this node classification task when using the embeddings $\mathcal{E}(\tilde{G})$ as our feature matrix in place of $\mathcal{E}(G)$. We report the performance of two embeddings made from reconstructed networks: by applying NetMF to \tilde{G} before and after binarizing its edges as described in Section 3.2.

Our classification setting is the same as that of Qiu et al. [2018]: we use a one-vs-rest logistic regression classifier, sampling a certain portion of the nodes as the training set. We repeat this sampling procedure 10 times and report the mean micro F1 scores. We also repeat the experiments as we vary the embedding dimensionality k and as we change the ratio of labeled examples from 10% to 90%.

As shown in Figure 6, when we use $\mathcal{E}(\tilde{G})$ generated from the non-binarized (i.e., expected) \tilde{G} as the input to our logistic regression classifier, we achieve almost equal performance to when we use the true embedding $\mathcal{E}(G)$. This finding can be interpreted in two ways. First, it shows that the low error observed in Figure 3 (left) extends beyond the Frobenius norm metric, to the perhaps more directly meaningful metric of comparable performance in classification. Second, it makes clear that losing local connectivity properties in the inversion process (like total triangle count and the existence of specific edges) does not significantly effect classification performance. The reconstructed networks seem to preserve more global properties that are important for node classification, like community structure.

While binarization does not significantly affect other metrics used to compare \tilde{G} to G (e.g., adjacency error, triangles), the classification task seems to be more sensitive, as performance falls when we use the embedding for the binarized \tilde{G} . It is an interesting open direction to investigate this phenomenon, and generally how the low-dimensional embeddings of a probabilistic adjacency matrix change when that matrix is sampled to produce an unweighted graph.

Synthetic graphs. We repeat the above experiments using several synthetic networks produced by the stochastic block model (SBM) [Abbe et al., 2015]. This random graph model assigns each node to a single cluster, and an edge between two nodes appears with probability p_{in} if the nodes belong to the same cluster and p_{out} otherwise, where generally it sets $p_{out} < p_{in}$. The configurations are summarized in Table 2. All networks have 1000 nodes, and, within each network, each cluster has the same size.

Name	# of Clusters	p_{in}	p_{out}
SBM 1	4	0.1	0.02
SBM 2	2	0.06	0.015
SBM 3	2	0.1	0.055
SBM 4	2	0.1	0.01
SBM 5	2	0.07	0.04

Table 2: Configuration of SBM networks; all networks have 1000 nodes.

As with the real-world networks, we include plots for the error of the NetMF embedding matrix and the binarized adjacency matrix; (Figure 7; the error of triangles count, and average path length (Figure 8); the error of the conductances of the top communities (Figure 9); and the node classification performance using embeddings made from the reconstructed networks (Figure 10). For the node classification task, each node is a member of a single ground-truth community which corresponds to its cluster in the SBM.

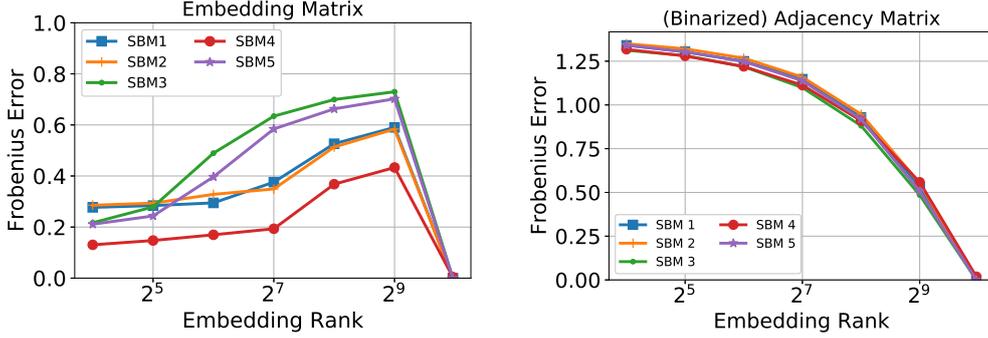


Figure 7: Relative Frobenius error for the low-rank PPMI matrices of reconstructions of the synthetic SBM networks (left) and the binarized adjacency matrix (right).

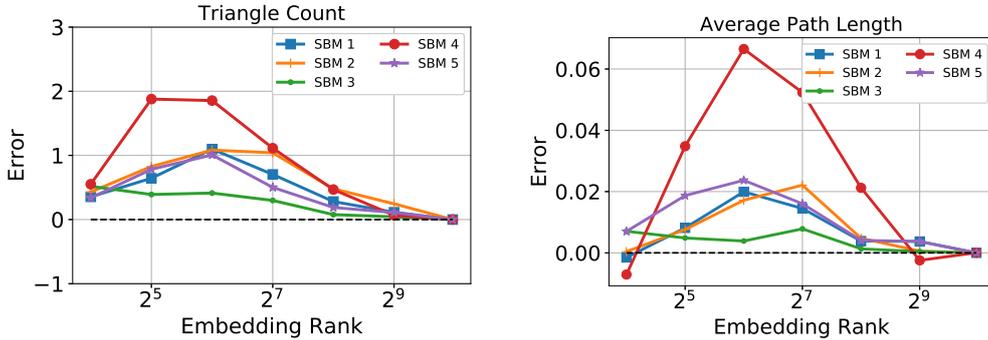


Figure 8: Graph reconstruction errors for synthetic SBM networks. Relative error for the number of triangles (left) and for the average path length (right).

The results here largely match those of the real-world networks: the networks recovered by applying NetMF embedding inversion differ substantially from the true networks in terms of adjacency matrix and triangle count. However, we observe that community structure is well preserved – see Figure 2 for a visual depiction.

Finally, we note that when our input is the full rank PPMI matrix (i.e., $k = n$), we succeed in reconstructing G exactly (i.e., $\tilde{G} = G$) for the SBM networks. This further supports the message of Theorem 1 that, when embedding dimensionality is sufficiently high, node embeddings can be exactly inverted. However, at low dimensions, the embeddings seem to capture some important global properties, including community structure, while washing out more local structure.

5 Conclusion

Node embeddings have been instrumental in achieving state-of-the-art results for graph-based machine learning tasks. Our work is a step towards a deeper understanding of why this is the case. We initiate the study of node embedding inversion as a tool to probe the information encoded in these embeddings. For the NetMF embedding method, we propose two approaches based on different techniques, and we show that the inversion problem can be effectively solved. Building on this, we show that while these embeddings seem to wash out local information in the underlying graph, they can be inverted to recover a graph with similar community structure to the original. Two interesting questions are whether our framework can be extended beyond the NetMF method, and whether we can formalize our empirical findings mathematically. We believe that our framework can be extended to the broader family of node embeddings that are based on low-rank factorization of graph similarity matrices. We hope that comparing the invertibility of such embeddings can shed light on the differences and similarities between them.

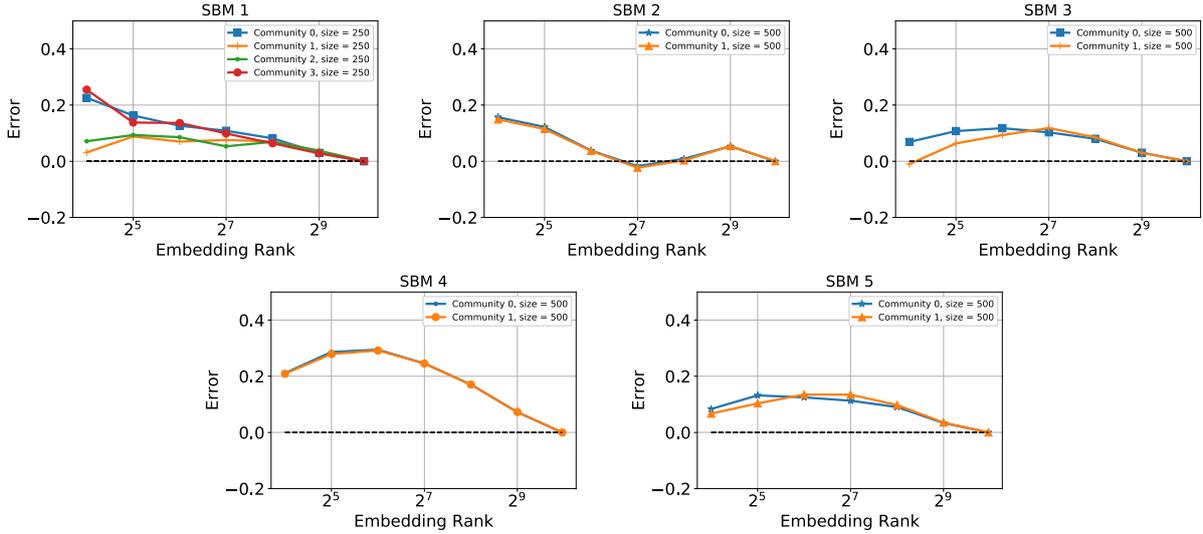


Figure 9: Relative error for the conductances of the five most populous communities for each synthetic SBM network.

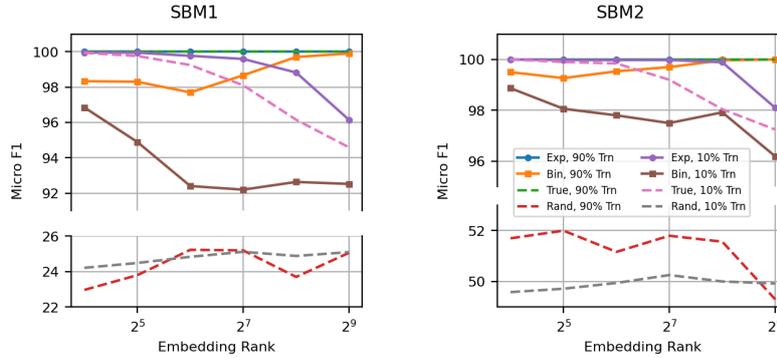


Figure 10: Multi-label classification using embeddings from reconstructions of two of the synthetic SBM networks.

References

- Emmanuel Abbe, Afonso S Bandeira, and Georgina Hall. Exact recovery in the stochastic block model. *IEEE Transactions on Information Theory*, 62(1):471–487, 2015.
- Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.
- Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*, 2016.
- Sudhanshu Chanpuriya and Cameron Musco. InfiniteWalk: Deep network embeddings as Laplacian embeddings with a nonlinearity. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2020.
- Sudhanshu Chanpuriya, Cameron Musco, Konstantinos Sotiropoulos, and Charalampos E Tsourakakis. Node embeddings and exact low-rank representations of complex networks. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

- Vasisht Duddu, Antoine Boutet, and Virat Shejwalkar. Quantifying privacy leakage in graph embedding. *arXiv:2010.00906*, 2020.
- Michael Ellers, Michael Cochez, Tobias Schumacher, Markus Strohmaier, and Florian Lemmerich. Privacy attacks on network embeddings. *arXiv:1912.10979*, 2019.
- Yoav Goldberg and Omer Levy. word2vec explained: deriving Mikolov et al.’s negative-sampling word-embedding method. *arXiv:1402.3722*, 2014.
- Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 855–864. ACM, 2016.
- Jeremy G Hoskins, Cameron Musco, Christopher Musco, and Charalampos E Tsourakakis. Learning networks from random walk-based node similarities. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, 2018.
- Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001. URL <http://www.scipy.org/>.
- Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv:1609.02907*, 2016.
- Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. <http://snap.stanford.edu/data>, 2014.
- Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528, 1989.
- Andrew McGregor. Graph stream algorithms: a survey. *ACM SIGMOD Record*, 43(1):9–20, 2014.
- Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 15 (NeurIPS)*, pages 849–856, 2002.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pages 8024–8035. 2019.
- Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 701–710, 2014.
- Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying DeepWalk, LINE, PTE, and node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 459–467, 2018.
- Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.
- Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–93, 2008.

- C Seshadhri, Aneesh Sharma, Andrew Stolman, and Ashish Goel. The impossibility of low-rank representations for triangle-rich complex networks. *Proceedings of the National Academy of Sciences*, 117(11):5631–5637, 2020.
- Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.
- Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.
- Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. LINE: Large-scale information network embedding. In *Proceedings of the 24th International World Wide Web Conference (WWW)*, pages 1067–1077, 2015.
- Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.
- Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1225–1234, 2016.
- Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560, 1997.

A Appendix

A.1 Recovery of Degrees from Limiting PMI

For an undirected graph G with adjacency matrix A and unnormalized Laplacian L , let \mathbf{d} be the vector with i^{th} entry equal to the i^{th} node's degree and $\mathbf{d}^{1/2}$ be its entrywise square root. Note that

$$\bar{L}\mathbf{d}^{1/2} = D^{-1/2}LD^{-1/2}\mathbf{d}^{1/2} = D^{-1/2}L\mathbf{1} = \mathbf{0}$$

since the all-ones vector $\mathbf{1}$ is in the null space of the unnormalized Laplacian L .

Suppose we have the limiting PMI matrix M_∞ and the graph volume v_G . We subtract the all-ones matrix J from M_∞ and multiply by \mathbf{d}/v_G :

$$\begin{aligned} (M_\infty - J)(\mathbf{d}/v_G) &= v_G \cdot D^{-1/2}(\bar{L}^+ - I)D^{-1/2}(\mathbf{d}/v_G) \\ &= D^{-1/2}\bar{L}^+\mathbf{d}^{1/2} - D^{-1/2}I\mathbf{d}^{1/2} \\ &= \mathbf{0} - \mathbf{1} = -\mathbf{1}. \end{aligned}$$

Thus, if we solve the linear system $(M_\infty - J)\mathbf{x} = -\mathbf{1}$ for \mathbf{x} , we should get $\mathbf{x} = \mathbf{d}/v_G$, from which we can determine all nodes' degrees. Note that without v_G , we can still recover the degrees up to a constant factor.

The only issue with the above approach occurs when $(M_\infty - J)$ is singular and the linear system does not have a unique solution. $(M_\infty - J)$ is singular iff $(\bar{L}^+ - I)$ is singular, and this only occurs when \bar{L}^+ and hence \bar{L} has an eigenvalue equal to 1. $\bar{L} = I - D^{-1/2}AD^{-1/2}$, so this requires that $D^{-1/2}AD^{-1/2}$ has a zero eigenvalue. Thus, $\bar{L}^+ - I$ is singular exactly when A is singular.