DeepWalking Backwards: From Node Embeddings Back to Graphs

Sudhanshu Chanpuriya [1] **Cameron Musco** [1]
Konstantinos Sotiropoulos [2] **Charalampos E. Tsourakakis** [2 3]

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 & Malik, 2000; Ng et al., 2002), Laplacian eigenmaps (Belkin & Niyogi, 2003), IsoMap (Tenenbaum et al., 2000), and locally linear embeddings (Roweis & 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 & Leskovec, 2016), LINE (Tang et al., 2015), NetMF (Qiu et al., 2018), and many others (Cao et al., 2016; Kipf & 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. In this work 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 $\mathscr{G}$ denote the set of all such graphs with n nodes. We formalize the question through Problems 1 and 2 below.

Problem 1 (Embedding Inversion)**.** *Given an embedding algorithm* $\mathscr{E} : \mathscr{G} \to \mathbb{R}^{n\times k}$ *and the embedding* $\mathscr{E}(G)$ *for some* $G \in \mathscr{G}$*, produce* $\tilde{G} \in \mathscr{G}$ *with* $\mathscr{E}(\tilde{G}) = \mathscr{E}(G)$ *or such that* $\|\mathscr{E}(\tilde{G}) - \mathscr{E}(G)\|$ *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 $\mathscr{E}(G)$ 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* $\|\mathscr{E}(\tilde{G}) - \mathscr{E}(G)\|$ *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 $\mathscr{E}$. In this work, we focus on the popular DeepWalk method of Perozzi et al. (2014). DeepWalk embedding can be interpreted as

[1] College of Information and Computer Sciences, University of Massachusetts Amherst, Amherst, MA, USA. [2] Department of Computer Science, Boston University, Boston, MA, USA. [3] ISI Foundation, Turin, Italy. Correspondence to: Sudhanshu Chanpuriya <umass>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

low-rank approximation of a point-wise mutual information (PMI) matrix based on node co-occurrences in random walks (Goldberg & 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 focus on 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.

• 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 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 2 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.

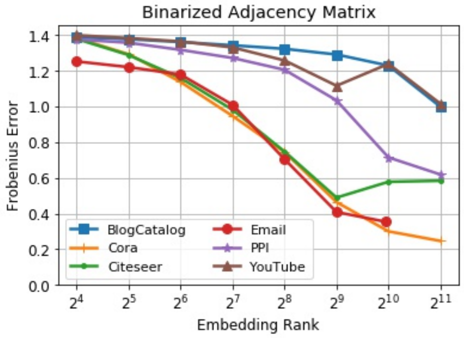

Figure 1: Relative Frobenius error $\|A - \tilde{A}\|_F / \|A\|_F$ between the adjacency matrices of G and $\tilde{G}$.

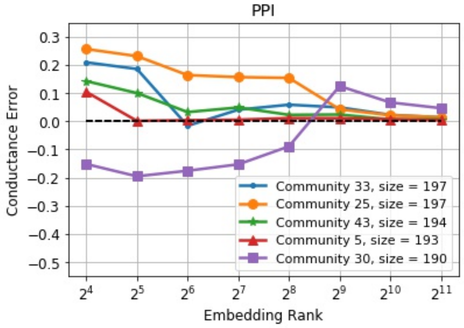

Figure 2: Relative error between G and $\tilde{G}$ for the conductance of the five largest communities (corresponding to biological states) in a human protein-protein interaction network.

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 (Spielman & 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

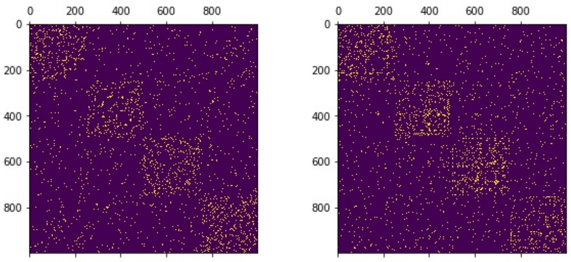

Figure 3: G (left), a stochastic block model graph with 1000 nodes and 4 clusters, and $\tilde{G}$ (right), a reconstruction of G from a 32-dimensional NetMF embedding. While G and $\tilde{G}$ differ in the exact edges they contain, we can see that the community structure is preserved.

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 $\mathscr{E}(G)$.

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 $v_G = \operatorname{tr}(D) = \sum_{i,j} A_{i,j}$. Qui et al. show that, for window size hyperparameter T (typical settings are $T = 10$ or $T = 1$), DeepWalk stochastically factorizes the *point-wise 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 entry-wise 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 the issue of taking the logarithm of 0, NetMF instead employs the *positive point-wise 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., $\mathscr{E}(G) = V\sqrt{|W|}$, where the absolute value and the square root are applied entry-wise. 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 $\mathscr{E}(G)$ encodes the best k-rank approximation $M_{T,k} = VWV^T$ to the positive point-wise 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 & 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:

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

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 $\hat{M}_\infty$, 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:

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

In Appendix A.1, we show using just the graph volume v_G, that one can perfectly recover the degree matrix D from $\hat{M}_\infty$ 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 $T \to \infty$.*

In our embedding inversion task, rather than the exact limiting PMI matrix $\hat{M}_\infty$, 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 $\hat{M}_\infty$, which we then plug into (3). This approximation is based on inverting the following limit, shown by Chanpuriya & Musco (2020):

$$\lim_{T\to\infty} \hat{M}_T = \log\left(\tfrac{1}{T}\hat{M}_\infty + J\right), \tag{4}$$

where the logarithm is applied entry-wise.

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 entry-wise 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.

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. Note that we invoke a function σ_v which constructs an adjacency matrix with a given target volume. The details of the latter can be found in Appendix A.2.

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: $\tilde{M}_\infty \leftarrow T \cdot \left(\exp\left(M_{T,k}\right) - J\right)$
     ▷ exp is applied entry-wise, $J$ 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: $\tilde{A} \leftarrow \text{clip}\left(D^{1/2}\left(I - \tilde{\bar{L}}\right)D^{1/2}\right)$
4: return $\tilde{A}$
```

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 Appendix A
4:     $\tilde{M}_T \leftarrow \text{PPMI}\left(\tilde{A}\right)$ 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 $\sigma_v(X)$
```

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.

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

Binarization. We binarize the reconstructed adjacency matrix $\tilde{A} \in (0,1)^{n\times n}$ differently from the prior approach. Here, 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 approximately v_G.

3.3. Baseline Approach

As a point of comparison to our proposed methods, we also evaluate the following natural baseline, which is inspired by Yin et. al (Yin & Wei, 2019). Namely, we view the input matrix, the low-rank approximation $M_{T,k}$ of the PPMI matrix, as a generic node similarity matrix, and take as edges of the reconstructed adjacency matrix $\tilde{A}$ the indices (i, j) that correspond to the top-v_G entries of $M_{T,k}$.

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 & Krevl, 2014) for EMAIL and YOUTUBE. The YOUTUBE graph we use is a sample of 20 communities from the raw network of (Leskovec & 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 6, refer to the labels as given in the input datasets.

Table 1: Datasets used in our experiments

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

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 graph G (Problem 2). Here, we binarize the reconstructed adjacency matrix to produce $\tilde{A}_b$. See Section 3.1 and Section 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: $\phi(S) = \frac{e(S:\bar{S})}{\min(\mathrm{vol}(S),\mathrm{vol}(\bar{S}))}$ where $e(S:\bar{S})$ is the number of edges leaving community S and $\mathrm{vol}(S)$ 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 $(\tilde{x} - x)/x$.

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 experimental findings.

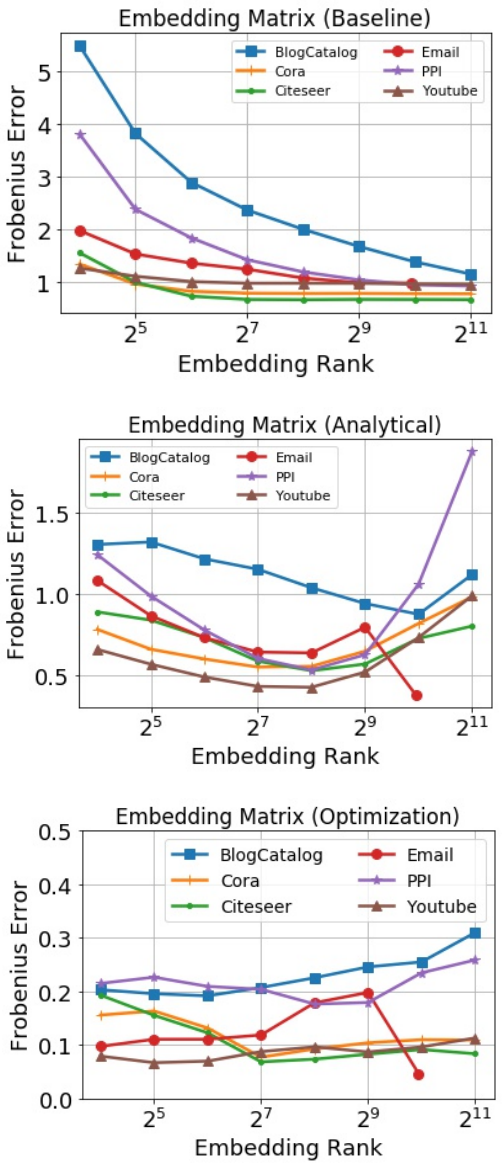

Figure 4: Relative Frobenius error vs. embedding rank k for the low-rank PPMI matrices of the graphs reconstructed using the inversion algorithms: the baseline (top), the analytical approach, Alg. 1 (middle), and the optimization approach, Alg. 2 (bottom). For details, see Section 4.2.

- 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). Both of the proposed approaches generally outperform the baseline. See Figure 4.

- 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 5.

- $\tilde{G}$ 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 5.

- $\tilde{G}$'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 4 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., $T \to \infty$, $k = n$), it performs poorly when these conditions are violated. 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. Additionally, both Alg. 1 and Alg. 2 significantly outperform the baseline of Section 3.3, except at the highest of the embedding ranks which we evaluate, where Alg. 1 performs comparably to the baseline.

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 4 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 5, at low ranks, the relative Frobenius error is often quite high – near 1. In combination with Figure 4 (middle), 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 ma-

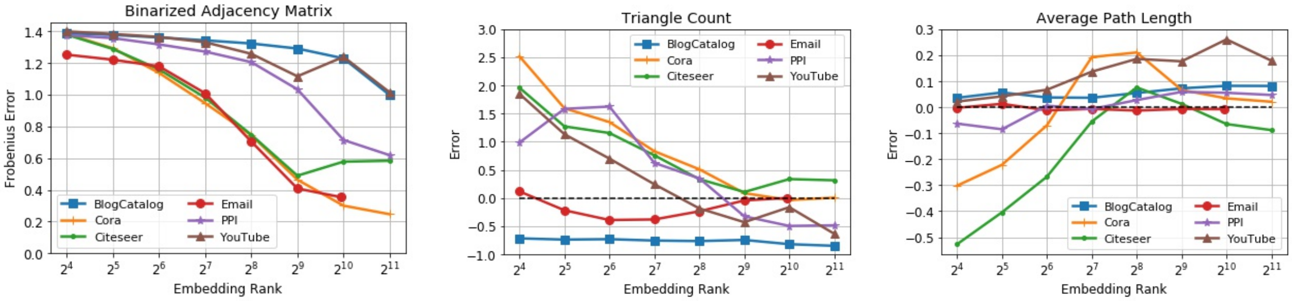

Figure 5: From left to right: Relative Frobenius error for the binarized adjacency matrix; relative error for the number of triangles; and relative error for the average path length. All plots are versus the embedding rank.

trices). 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 5 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 5).

In Figure 6, we plot the relative errors for the conductance 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.

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 $p_{out} < p_{in}$. We observe similar results for these graphs, with figures included in Appendix A.

As with the real-world datasets, the networks recovered by applying NetMF embedding inversion to these synthetic datasets 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 3 for a visual depiction.

Finally, we note that when we use the full rank PPMI matrix as our input (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.

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 $\mathscr{E}(\tilde{G})$ as our feature matrix in place of $\mathscr{E}(G)$. 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

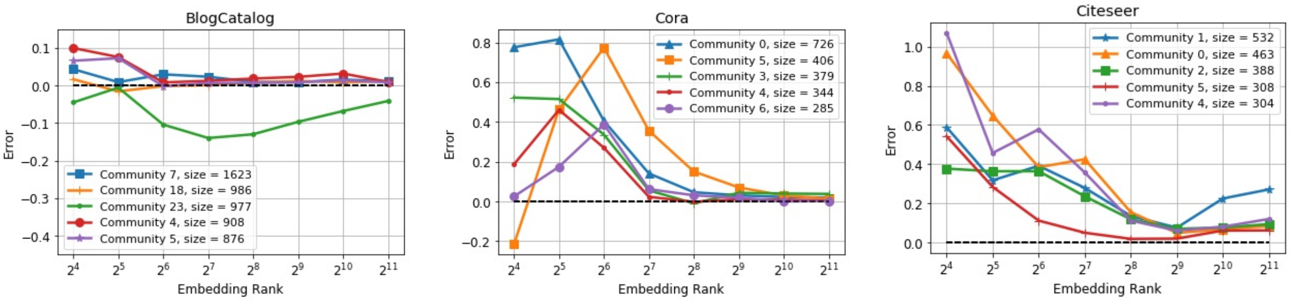

Figure 6: Relative error for the conductance of the five largest communities for three selected networks.

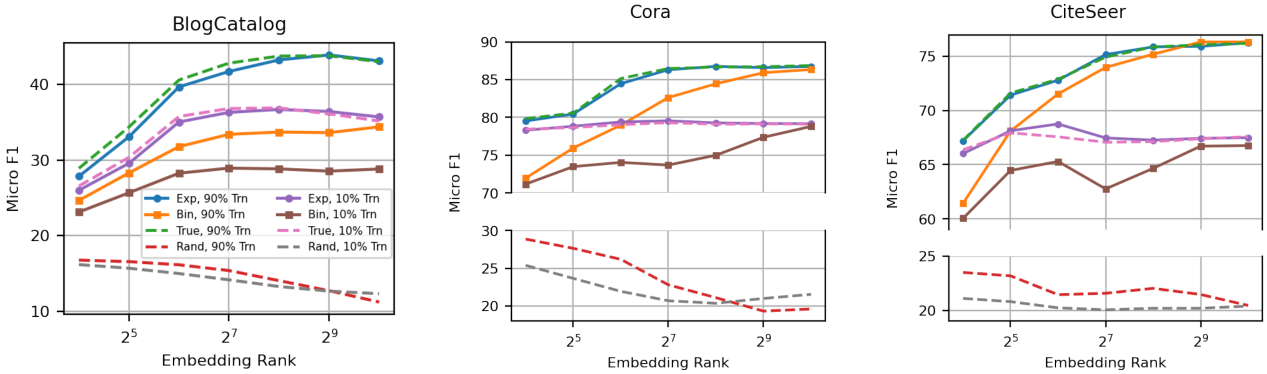

Figure 7: Multi-label classification using embeddings from reconstructed networks. Performance when using embeddings from a random graph is included as a baseline.

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 7, when we use $\mathscr{E}(\tilde{G})$ 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 $\mathscr{E}(G)$. This finding can be interpreted in two ways. First, it shows that the low error observed in Figure 4 (bottom) 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 affect 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, etc.), 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.

4.4. Additional Considerations

Impact of Window Size T

For the experiments discussed previously, we set the window size to $T = 10$, as this is the value most commonly used for skip-gram node embeddings. We also investigate whether the performance of Algorithm 2 is sensitive to the window size used for the embeddings. These results are presented in Figure 8 for the PPI network. We find that the related Frobenius error for the reconstructed adjacency matrix is not very sensitive by the window size T. The same is true for the average path length. By contrast, we find that triangle count does differ significantly as window size is varied. These plots, along with more complete results for other datasets, can be found in the Appendix.

Effect of network structure on embedding inversion

As discussed previously, when inverting the low-dimensional embedding of a stochastic block model (SBM) network, we can effectively capture its community structure, even if more fined grained information (like exact edges) is lost. One question is how well we can approximate the embeddings of random graphs that lack the community

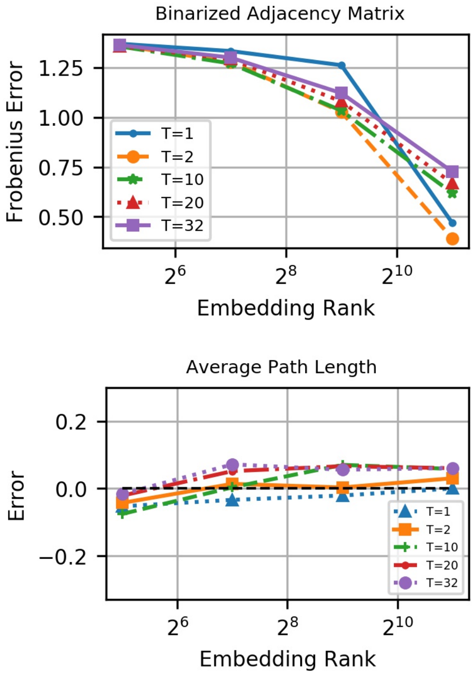

Figure 8: Impact of window size T on the reconstruction of the adjacency matrix and the average path length for the PPI network.

structure of the SBM. To investigate this, we again use the SBM and fix the probability of an edge between two nodes belonging to the same cluster as $p_{\text{in}} = 0.1$. Then, we generate different networks, by varying the probability of an edge between nodes in different clusters, $p_{\text{out}} = [0.01, \ldots, 0.1]$. Notice that as p_{out} approaches p_{in}, the SBM approaches an Erdős-Rényi (ER) graph. We observed that as p_{out} approaches p_{in} (so the graph approaches an ER graph), the embedding error increases. We conjecture that this is because the graph no longer has low-rank community structure. A better understanding of the embeddings of random networks that lack a clear community structure (as the ER model, or the Barabási-Albert model) is an interesting direction for future research.

Table 2: Embedding error for a range of values of p_{out}. p_{in} is fixed at 0.100. The networks have 1,000 nodes, and the embedding rank is 32.

$p_{\textbf{out}}$	0.005	0.010	0.030	0.050	0.100
Error	0.009	0.033	0.191	0.204	0.192

5. Conclusion

Modern node embeddings have been instrumental in achieving state-of-the-art empirical results for many 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 methods 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 node embedding method, and whether we can formalize our empirical findings mathematically. We believe that our framework can be used for the broader family of node embedding methods that are based on low-rank factorization of graph similarity matrices. In this way, we hope to shed further light on the differences and similarities between various node embedding methods.

Acknowledgments

We would like to thank the anonymous reviewers for their suggestions that helped improve the quality of the paper. CT acknowledges support from Intesa Sanpaolo Innovation Center. CM was partially supported by NSF grant no. 1763618 and an Adobe Research Grant. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

References

Abbe, E., Bandeira, A. S., and Hall, G. Exact recovery in the stochastic block model. *IEEE Transactions on Information Theory*, 62(1):471–487, 2015.

Belkin, M. and Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

Cao, S., Lu, W., and Xu, Q. Deep neural networks for learning graph representations. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*, 2016.

Chanpuriya, S. and Musco, C. InfiniteWalk: Deep network embeddings as Laplacian embeddings with a nonlinearityalk: 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.

Chanpuriya, S., Musco, C., Sotiropoulos, K., and Tsourakakis, C. E. Node embeddings and exact low-rank representations of complex networks. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

Duddu, V., Boutet, A., and Shejwalkar, V. Quantifying privacy leakage in graph embedding. *arXiv:2010.00906*, 2020.

Ellers, M., Cochez, M., Schumacher, T., Strohmaier, M., and Lemmerich, F. Privacy attacks on network embeddings. *arXiv:1912.10979*, 2019.

Goldberg, Y. and Levy, O. word2vec explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv:1402.3722*, 2014.

Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 855–864. ACM, 2016.

Hoskins, J. G., Musco, C., Musco, C., and Tsourakakis, C. E. Learning networks from random walk-based node similarities. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, 2018.

Jones, E., Oliphant, T., Peterson, P., et al. SciPy: Open source scientific tools for Python, 2001. URL http://www.scipy.org/.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv:1609.02907*, 2016.

Leskovec, J. and Krevl, A. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, 2014.

Liu, D. C. and Nocedal, J. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528, 1989.

McGregor, A. Graph stream algorithms: a survey. *ACM SIGMOD Record*, 43(1):9–20, 2014.

Ng, A. Y., Jordan, M. I., and Weiss, Y. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 15 (NeurIPS)*, pp. 849–856, 2002.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pp. 8024–8035. 2019.

Perozzi, B., Al-Rfou, R., and Skiena, S. DeepWalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 701–710, 2014.

Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., and Tang, J. 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*, pp. 459–467, 2018.

Roweis, S. T. and Saul, L. K. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290 (5500):2323–2326, 2000.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI Magazine*, 29(3):93–93, 2008.

Seshadhri, C., Sharma, A., Stolman, A., and Goel, A. The impossibility of low-rank representations for triangle-rich complex networks. *Proceedings of the National Academy of Sciences*, 117(11):5631–5637, 2020.

Shi, J. and Malik, J. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

Spielman, D. A. and Srivastava, N. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6): 1913–1926, 2011.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. LINE: Large-scale information network embedding. In *Proceedings of the 24th International World Wide Web Conference (WWW)*, pp. 1067–1077, 2015.

Tenenbaum, J. B., De Silva, V., and Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

Wang, D., Cui, P., and Zhu, W. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1225–1234, 2016.

Yin, Y. and Wei, Z. Scalable graph embeddings via sparse transpose proximities. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1429–1437, 2019.

Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. 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. Supplemental Appendix

A.1. Recovery of Degrees from Limiting PMI

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

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

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 $\boldsymbol{d}/v_G$:

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

Thus, if we solve the linear system $(M_\infty - J)\boldsymbol{x} = -\mathbf{1}$ for $\boldsymbol{x}$, we should get $\boldsymbol{x} = \boldsymbol{d}/v_G$, 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.

A.2. Shifted Logistic Function σ_v

The pseudocode of the shifted logistic function σ_v that is used in Section 3.2 is given in Algorithm 3. This implementation is an application of Newton's method. We find that 10 iterations are sufficient for convergence in our experiments.

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 $v$
$s \leftarrow 0$
for $i \leftarrow 1$ to $I$ do
    $A \leftarrow \sigma(X + s)$    ▷ $\sigma$ is the logistic function applied entry-wise
    $s \leftarrow s + \frac{v - \Sigma(A)}{\Sigma(A \circ (1-A))}$    ▷ $\Sigma$ sums over all elements, and $\circ$ is an entry-wise product
end for
return $\sigma(X + s)$
```

Table 3: Configuration of SBM networks; all networks have 1,000 nodes.

Name	# of Clusters	p_{in}	p_{out}
SBM 1	4	0.10	0.020
SBM 2	2	0.06	0.015
SBM 3	2	0.10	0.055
SBM 4	2	0.10	0.010
SBM 5	2	0.07	0.040

A.3. Omitted Plots for Real-World Networks

We include the relative conductance error (Figure 9) and classification performance (Figure 10) for the networks we omitted for brevity from the main section: EMAIL, PPI, and YOUTUBE.

A.4. Stochastic Block Model Experiments

We also perform the experiments we conducted for real-world networks for a number of synthetic networks, focusing only on the optimization approach (Alg. 2). These synthetic networks are created using the stochastic block model (SBM). The configurations are summarized in Table 3. All networks have 1,000 nodes, and, within each network, each cluster has the same size. As with the real-world networks, we include plots for the error of the NetMF embedding matrix (Figure 11); the error of the binarized adjacency matrix, triangle count, and average path length (Figure 12); the error of the conductance of the top communities (Figure 13); and the node classification performance using embeddings made from the reconstructed networks (Figure 14). For the node classification task, each node is a member of a single ground-truth community which corresponds to its cluster in the SBM.

A.5. Effect of binarization on triangle counts

In the main text, we presented our results for different metrics on the binarized versions of the reconstructed networks. We noted that the effect of binarization on triangle counts is generally not . This is especially true at higher embedding ranks, where the reconstructions converge to the true value. In Figure 15, we plot these results for three of our networks.

A.6. Effect of window size T on different measures

In Figures 16, 17, and 18, we plot the adjacency error, average path length error, and triangle count error, respectively, for a range of values of T. As discussed in the main text, only the third measure seems sensitive to T.

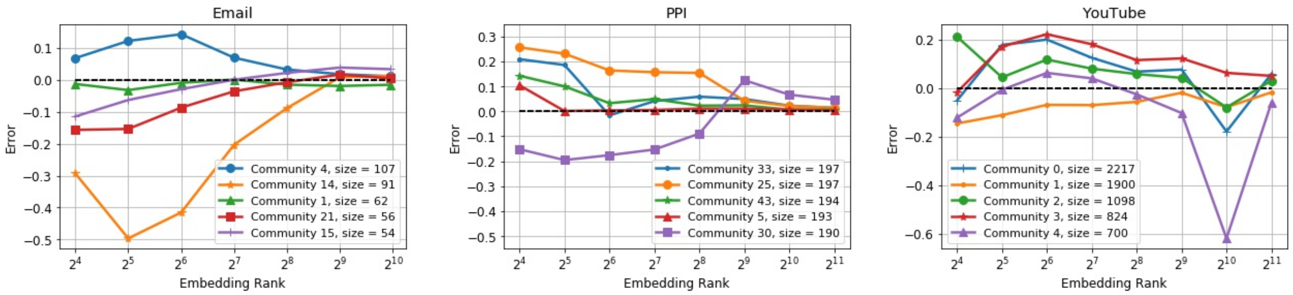

Figure 9: Relative error for the conductance of the five most populous communities for each network.

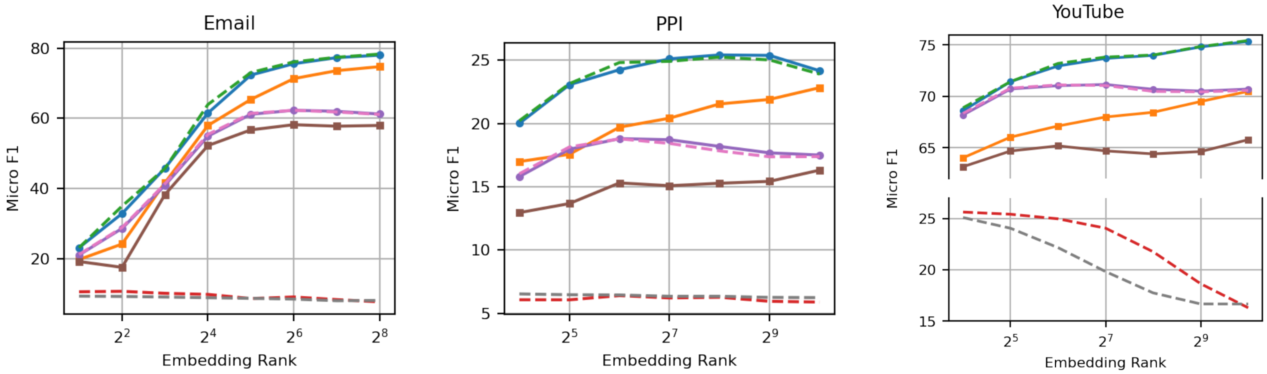

Figure 10: Multi-label classification using embeddings from reconstructed networks. Performance when using embeddings from a random graph is included as a baseline.

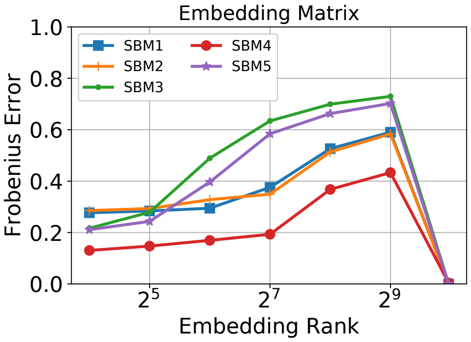

Figure 11: Relative Frobenius error for the low-rank PPMI matrices of reconstructions of the synthetic SBM networks.

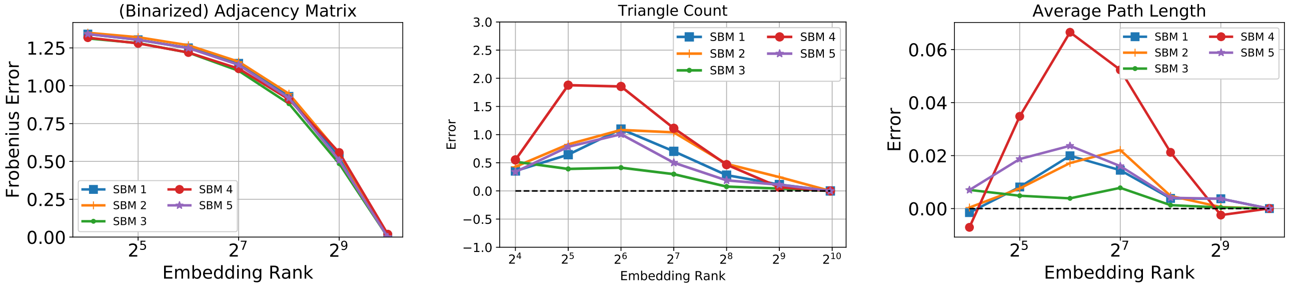

Figure 12: Various reconstruction errors for synthetic networks. From left to right: relative Frobenius error for the binarized adjacency matrix, relative error for the number of triangles, and relative error for the average path length.

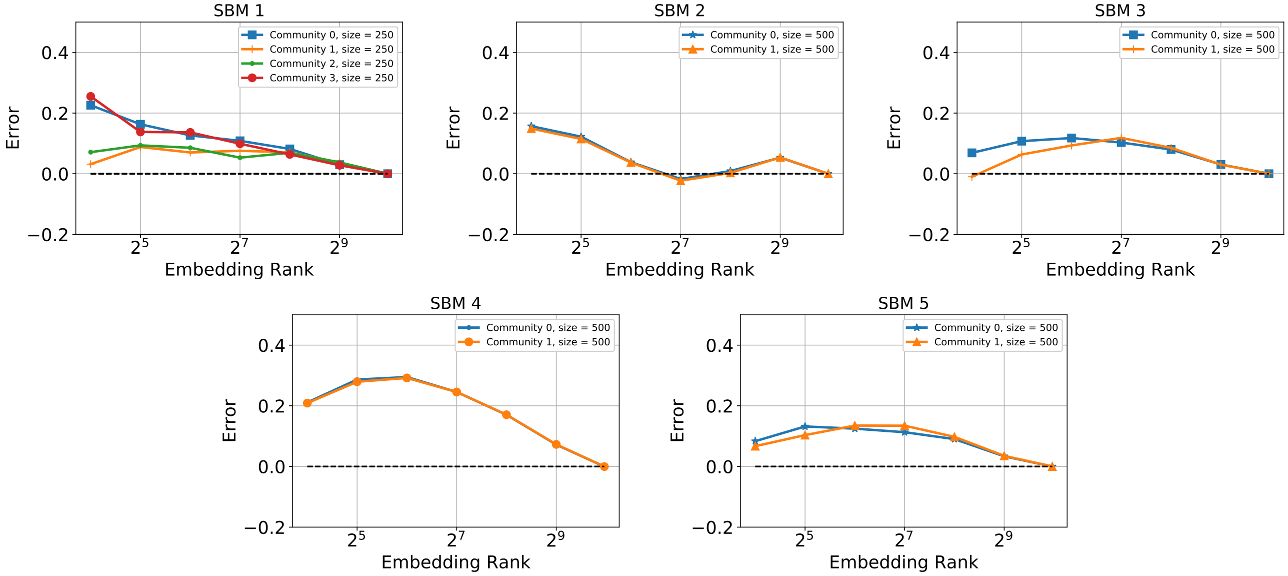

Figure 13: Relative error for the conductance of the five most populous communities for each synthetic network.

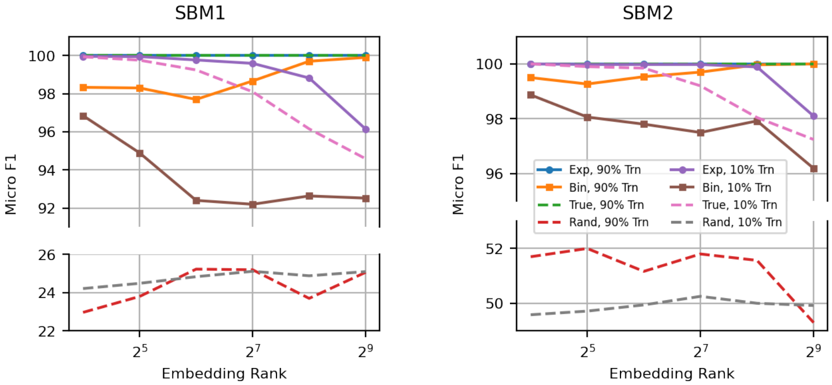

Figure 14: Multi-label classification using embeddings from reconstructions of two of the synthetic SBM networks.

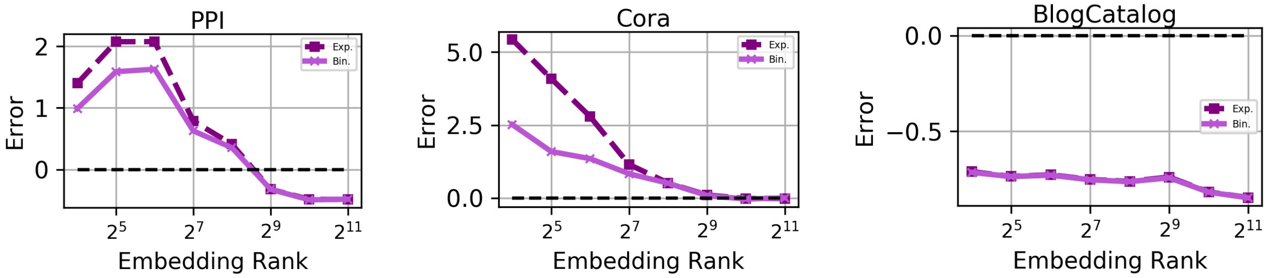

Figure 15: Expected number of triangles in the reconstructed network and total number of triangles after binarizing.

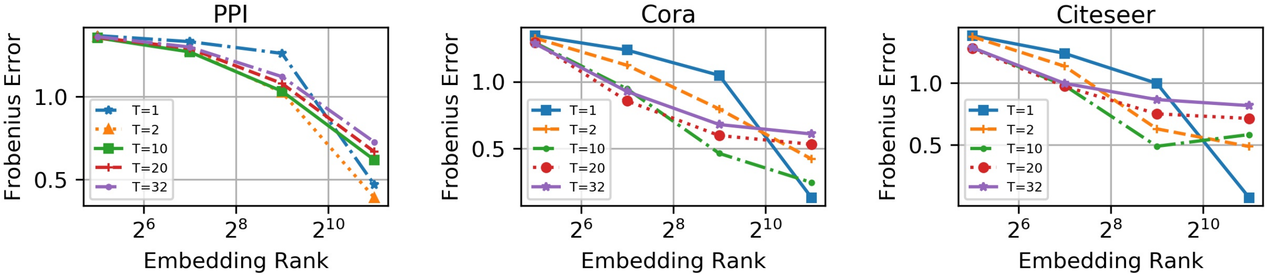

Figure 16: Adjacency error of reconstructed networks for a range of values of window size T.

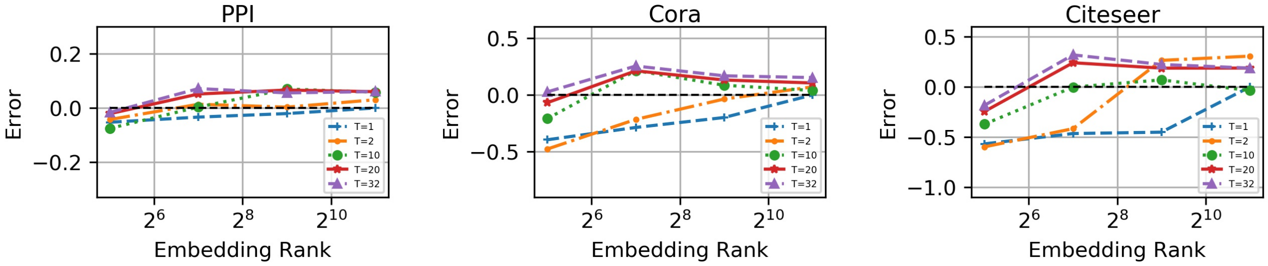

Figure 17: Average path length error of reconstructed networks for a range of values of window size T.

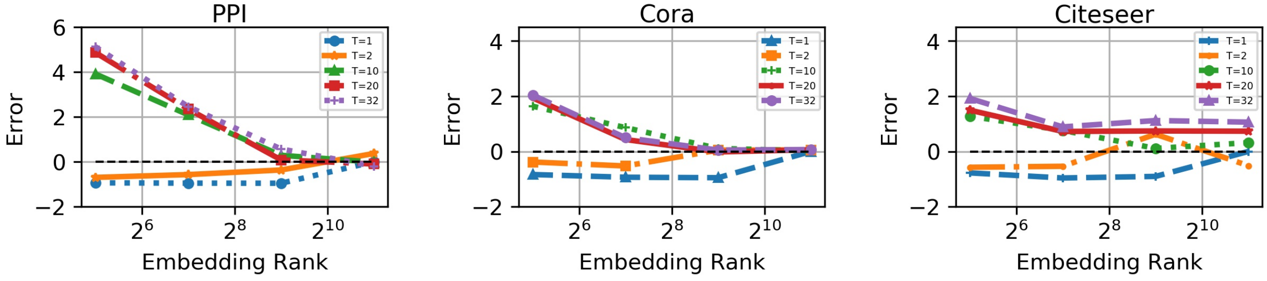

Figure 18: Triangle count error of reconstructed networks for a range of values of window size T.