Foundations of Node Representation Learning

Sudhanshu Chanpuriya
University of Massachusetts Amherst

FOUNDATIONS OF NODE REPRESENTATION LEARNING

A Dissertation Presented

by

SUDHANSHU CHANPURIYA

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2023

Robert and Donna Manning College of
Information and Computer Sciences

FOUNDATIONS OF NODE REPRESENTATION LEARNING

A Dissertation Presented

by

SUDHANSHU CHANPURIYA

Approved as to style and content by:

ACKNOWLEDGMENTS

I would like to thank the many people who helped me complete this thesis. I am especially grateful to my advisor, Cameron Musco. He has been a constant source of encouragement and helpful advice, and I am incredibly lucky to have had him as my advisor. I would also like to thank Charalampos Tsourakakis, not only for being on my committee, but also for his collaboration and guidance on much of my graduate research. I also thank my other committee members, Andrew McGregor and Andrew McCallum, for their time and feedback. During the past few years, I have been fortunate to work with some great co-authors, including Konstantinos Sotiropoulos and Ryan Rossi, to whom I am also grateful. Finally, I would like to thank Eileen Hamel for her help with administrative issues throughout my graduate studies.

ABSTRACT

FOUNDATIONS OF NODE REPRESENTATION LEARNING

SEPTEMBER 2023

SUDHANSHU CHANPURIYA

B.Sc., DARTMOUTH COLLEGE

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Cameron Musco

Low-dimensional node representations, also called node embeddings, are a cornerstone in the modeling and analysis of complex networks. In recent years, advances in deep learning have spurred development of novel neural network-inspired methods for learning node representations which have largely surpassed classical 'spectral' embeddings in performance. Yet little work asks the central questions of this thesis: *Why do these novel deep methods outperform their classical predecessors, and what are their limitations?*

We pursue several paths to answering these questions. To further our understanding of deep embedding methods, we explore their relationship with spectral methods, which are better understood, and show that some popular deep methods are equivalent to spectral methods in a certain natural limit. We also introduce the problem of *inverting* node embeddings in order to probe what information they contain. Further, we propose a simple, non-deep method for node representation learning, and find it to often be competitive with modern deep graph networks in downstream performance.

To better understand the limitations of node embeddings, we prove some upper and lower bounds on their capabilities. Most notably, we prove that node embeddings are capable of *exact* low-dimensional representation of networks with bounded max degree or arboricity, and we further show that a simple algorithm can find such exact embeddings for real-world networks. By contrast, we also prove inherent bounds on random graph models, including those derived from node embeddings, to capture key structural properties of networks without simply memorizing a given graph.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Graphs naturally model a wide variety of complex systems including computer networking, social networks, protein-protein interaction, and co-authorship [22, 165, 81, 206, 159, 107, 129, 120, 80]. Understanding and analyzing such networks lies at the heart of computer science. Underlying almost all graph machine learning are methods which learn useful representations of nodes and reveal interpretable structure in the graph, such as communities of nodes. The learned node representations can be used as input for many downstream tasks, such as network analysis, visualization, clustering, classification, and link prediction. In this thesis, we analyze the capacity and limitations of existing methods, both by empirical investigation of real-world networks and by proving theoretical bounds.

More specifically, we focus on node 'embedding' techniques, which map the nodes of a graph to low-dimensional Euclidean space in such way that the geometry of the embedding reflects important structure in the graph. A node embedding method takes as input a graph G with n nodes $v_1, \ldots, v_n$ and maps each node v_i to a vector $\boldsymbol{x}_i \in \mathbb{R}^k$, where k is an embedding dimension typically with $k \ll n$.

Low-dimensional node embeddings have a long and rich history. They have played a major role in theoretical computer science, and specifically in the development of approximation algorithms for NP-hard problems. Spectral clustering relies on node embeddings based on a small number of extremal eigenvectors of a matrix representation of the graph (e.g., top eigenvectors of the adjacency matrix [124], or bottom eigenvectors of the Laplacian [11]) to find good cuts, e.g., [133, 169, 154]. Metric embeddings have been used in multi-commodity flow algorithms [102, 113]. These methods embed a graph in a Euclidean space so that the distances between nodes in the graph are close to the geometric distances between the embeddings. In particular, by Bourgain's theorem, every metric space with n elements (including every n node graph metric) can be embedded in an $\mathcal{O}(\log n)$ dimensional space with $\mathcal{O}(\log n)$ distortion [24]. In their seminal work on the maximum cut problem [70], Goemans and Williamson introduced a large family of semidefinite programming relaxations for NP-hard problems that embed nodes in a Euclidean space, and round these embeddings to obtain a near-optimal solution.

Many of the most classic random network models, including the stochastic block model [84, 7] and random dot-product models [206] are based on low-dimensional node embeddings. Each pair of nodes v_i, v_j is connected with probability depending on the similarity between their embeddings (e.g., the dot product $\boldsymbol{x}_i^\top \boldsymbol{x}_j$.). Many machine learning methods learn a latent low-dimensional embedding by maximizing a likelihood function that depends on a probabilistic model of this form [92, 127, 138, 73]. Relatedly, work in non-linear dimensionality reduction learns node embeddings that capture general dataset structure. Classical methods such as Laplacian eigenmaps, IsoMap, and locally linear embeddings [17, 184, 156] associate a graph G with a generic high-dimensional dataset (e.g., by forming a k-nearest neighbors graph) and apply variants of spectral embedding on G to find an informative embedding for the original data points.

In recent years, the overall successes of deep learning [99] have inspired new methods for graph machine learning. Several neural network architectures that work with graph data in an end-to-end fashion have been developed, including the graph convolutional network (GCN) [94, 46] and many descendents [36, 199, 188]. In this thesis, we mostly focus on 'unsupervised' node representations, which are derived solely from a graph's structure, without optimizing directly for a downstream objective, e.g., to target some given node classes. There are several neural network-based methods that fall in this category, including DeepWalk, node2Vec, LINE, PTE, SDNE, and many more [144, 182, 181, 75, 28, 192, 29, 139, 193]. These modern methods have overall surpassed classical methods in downstream performance and become the node embeddings of choice in practice. This raises the following questions, which are the core questions of this thesis:

> *Why do these novel methods outperform their classical predecessors, and what are their limitations?*

Some prior work approaches a better understanding of these methods from various directions. Notably, a line of work [109, 110, 147] which overlaps with the related task of learning word representations in natural language processing (NLP) has found an interesting connection: many of these methods learn embeddings that implicitly form a low-dimensional factorization of a certain matrix corresponding to the graph G. This interpretation is striking given that these methods are based on the typical deep learning framework of stochastic gradient descent (SGD) on some probabilistic objective function, which does not obviously equate to matrix factorization. The implicit matrix contains the pointwise mutual information (PMI) of co-occurences between nodes when taking random walks on the graph; roughly, its elements correspond to positive and negative latent correlations between nodes. In NLP, explaining the structure of the PMI matrix and word embeddings derived thereof, especially as it relates to their effectiveness in word analogies (e.g., that "*man* is to *king* as *woman* is to *queen*" may be reflected in the embeddings of these four words), has attracted much attention [15, 68, 8, 9]. Other work at the intersection of machine learning and privacy empirically investigates the information contained in deep node embeddings to try to quantify how much can be inferred about, e.g., users in a social network, if a malicious party is provided node embeddings from the network [54, 51].

Broadly, the preceding research works towards characterizing the nature and power of deep node embeddings, but there is also some prior work on the limitations of embeddings. [128] provides a certain negative result for graph neural networks (GNNs), and hence embeddings derived thereof, by relating them to the 1-dimensional Weisfeiler-Leman graph isomorphism heuristic (1-WL) [74]. Specifically, they prove that 'vanilla' GNNs, which form each node's representation by iteratively aggregating those of its neighbors, are no more expressive than 1-WL in terms of distinguishing between graphs. As more directly relates to node embeddings, [167] recently showed that a natural graph model in which, roughly speaking, the probability of an edge between two nodes increases with the dot product of their embeddings, cannot simultaneously exhibit sparsity and high triangle density, which are common characteristics of real-world networks. This result can be seen as a strong instance in a line of work showing that certain random graph models fail to capture empirically-observed properties of networks [195, 146].

We contribute to many of these lines of work as we attempt to answer our core questions. In doing so, we proceed along several paths, which are summarized in the next section.

1.1 Summary of Contributions

We now present a summary of the main contributions of this thesis.

1.1.1 Understanding Modern Node Embedding Methods

In Chapter 2, we investigate the first core question concerning the greater effectiveness of deep learning-based methods for node embedding compared to the older spectral methods. We approach this question in Section 2.1 by refining our understanding of the relationship between spectral methods and the popular DeepWalk method, which perhaps best exemplifies the more recent family of methods. At the core of spectral methods is factorizing the graph's adjacency matrix or Laplacian matrix. Although DeepWalk's algorithm for node embedding seems totally different – it involves stochastic gradient descent (SGD) on a probabilistic objective involving co-occurrences of nodes in random walks on the graph – we show that they are actually closely related. We build on work from Qiu et al. [147], who provide a closed-form expression for the DeepWalk objective function, which is based on factorizing the pointwise mutual information (PMI) matrix discussed previously. In this objective, the "window size" T within which nodes are considered to co-occur is a key hyper-parameter. We study the objective in the $T \to \infty$ limit, which allows us to simplify the expression from [147]. We prove that this limiting objective corresponds to factorizing a simple transformation of the pseudoinverse $\boldsymbol{L}^+$ of the graph Laplacian $\boldsymbol{L}$. Specifically, we show that the limiting PMI matrix has the form

$$\boldsymbol{M}_\infty = \mathrm{tr}(\boldsymbol{D}) \cdot \mathbf{D}^{-1/2} \left(\tilde{\mathbf{L}}^+ - \mathbf{I} \right) \mathbf{D}^{-1/2} + \mathbf{J},$$

where $\mathbf{D}$ is the degree matrix, $\tilde{\mathbf{L}}$ is the normalized Laplacian (i.e., $\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$), and $\mathbf{J}$ is the all-ones matrix. This result tightly links DeepWalk to spectral embeddings: it shows that in this limit, as DeepWalk factorizes $\boldsymbol{M}_\infty$, it essentially computes the top eigenvectors of a simple transformation of $\tilde{\boldsymbol{L}}$, which is exactly the approach of classical spectral embeddings. Beyond this limiting result, we show that by applying a simple nonlinear entrywise transformation to $\boldsymbol{M}_\infty$, we recover a good approximation of the finite-T PMI matrix, yielding embeddings that are competitive with those from DeepWalk and related methods on a downstream task. Surprisingly, we find that even simple binary thresholding of $\boldsymbol{L}^+$ is often competitive, suggesting that the core advancement of recent methods is a nonlinearity on top of the classical spectral embedding approach. This finding has implications for the design and analysis of new methods.

We also investigate the effectiveness of modern node embedding methods via a another route: by exploring exactly what information is encoded by these methods, and how this information correlates with performance in downstream learning tasks. We tackle this investigation in Section 2.2 by introducing a novel problem: studying whether embeddings can be *inverted* to (approximately) recover the graph used to generate them. Again focusing on DeepWalk, 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. Interestingly, based on the result from Section 2.1, we can prove that in a certain limiting sense, as the window size T and dimensionality of embeddings are both increased, DeepWalk embeddings become exactly invertible:

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

In the more practical situation of finite T and low-dimensionality embeddings, 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. These 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.

1.1.2 Power and Limitations of Embeddings

We approach the second core question involving the limitations of low-dimensional node embeddings in Chapter 3. To do so, we investigate whether embeddings are capable of representing structures of interest in graphs. This work is largely in response to work from Seshadhri et al. [167] which suggests that such embeddings, regardless of whether they are derived from classical spectral methods or modern neural network-inspired methods, cannot capture local structure arising in complex networks. In particular, they show that any network generated from a natural low-dimensional model cannot be both sparse and have high triangle density (high clustering coefficient), two hallmark properties of many real-world networks. In Section 3.1, we show that the results of [167] are intimately connected to the model they use rather than the low-dimensional structure of complex networks: we prove that a minor relaxation of their model can generate sparse graphs with high triangle density. Surprisingly, we can show that this same model leads to *exact low-dimensional factorizations* of many real-world networks, formalized as follows:

Theorem (Exact Embeddings for Bounded-Degree Graphs)**.** *Let $\boldsymbol{A} \in \{0,1\}^{n\times n}$ be the adjacency matrix of a graph G with maximum degree c. Then there exist embeddings $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n\times(2c+1)}$ such that $\boldsymbol{A} = \left[\boldsymbol{X}\boldsymbol{Y}^\top > 0\right]$, where $[z > 0]$, which is 1 if z is positive and 0 otherwise, is applied entry-wise to $\boldsymbol{X}\boldsymbol{Y}^T$.*

The salient difference between our graph model and that of [167] is that ours has two factors $\boldsymbol{X}$ and $\boldsymbol{Y}$, corresponding to two embedding vectors per node, rather than one; this drastically increases the model's expressiveness. Beyond our theoretical results, we give a simple algorithm based on a logistic variant of principal component analysis (PCA) that succeeds in finding such exact embeddings, and a large number of experiments verify the ability of very low-dimensional embeddings to exactly represent local structure in real-world networks.

In Section 3.2, we extend the results of Section 3.1 in two ways. First, we show that, in addition to the exact representation guarantee for bounded *degree* graphs, a similar result applies to graphs with bounded *arboricity*: for graphs with arboricity α, there are exact embeddings with embedding dimensionality $\mathcal{O}(\alpha^2)$. This result is more applicable to real-world networks, which are typically sparse (corresponding to low arboricity), but often have high max degree. Second, we show that the exact embedding result applies not only to the above factorization, but also to the following one: $\boldsymbol{A} \approx \left[\boldsymbol{B}\boldsymbol{B}^\top - \boldsymbol{C}\boldsymbol{C}^\top > 0\right]$, for nonnegative factors $\boldsymbol{B}, \boldsymbol{C}$. For undirected graphs, which are the main focus of this work, this factorization is more interpretable in that its symmetry reflects the undirectedness (i.e., the symmetry of the adjacency matrix), and also since the nonnegativity allows the embeddings to be interpreted as community assignments in a community detection framework: intuitively, each entry of the nonnegative embedding vector of a node represents the intensity with which the node participates in a community. On the empirical side, we evaluate the performance of this factorization for downstream tasks like community detection and link prediction.

Overall, this work adds to a growing body of literature on expressiveness guarantees for representations of relational data. While we focus in this thesis on embeddings in Euclidean space with standard dot products, we also note that there is a wealth of prior work [153, 119, 150, 77, 90, 163, 37, 190, 161, 19, 196, 23] on guarantees for other kinds of embeddings of graphs. This includes spherical embeddings, which use a non-Euclidean, positively-curved space; hyperbolic embeddings, which use a negatively-curved space and are well-suited to tree-like graphs; and box embeddings, which represent each node with an axis-aligned box in Euclidean space such that two boxes overlap iff the corresponding nodes are adjacent. For example, [23] show that a variant of box embeddings can exactly represent any directed acyclic graph (DAG) with boxes in a $\mathcal{O}\left(c \cdot \log(n)\right)$-dimensional space, where c is the max degree and n is the number of nodes; note that the scaling is linear in the max degree, as in our result. We note that [23] also specifies the bits of precision needed for their exact representation result (i.e., the bit complexity). In our guarantees, we do not consider precision, and effectively assume infinite precision; providing a guarantee with both dimension and bit complexity is a key future direction.

1.1.3 Power and Limitations of Random Graph Models

The strength of our results concerning exact representation actually suggests a possible weakness of low-dimensional node embeddings: such embeddings are generally intended to capture meaningful network structure (such as node communities), but are they instead simply attempting to memorize the input graph? The tension between representation and memorization motivates Chapter 4, which approaches this investigation of the second core question from the following, more abstract direction.

In Section 4.1, we study the inherent limitations of *edge independent random graph models*, in which each edge is added to the graph independently with some probability:

Definition (Edge Independent Graph Model)**.** *For any symmetric matrix* $\boldsymbol{P} \in [0,1]^{n\times n}$*, let* $\mathcal{G}(\boldsymbol{P})$ *be the distribution over undirected unweighted graphs where* $G \sim \mathcal{G}(\boldsymbol{P})$ *and its edge set* $E(G)$ *contains edge* (i,j) *independently, with probability* $\boldsymbol{P}_{ij}$*. That is,* $\mathbb{P}(G) = \prod_{(i,j)\in E(G)} \boldsymbol{P}_{ij} \cdot \prod_{(i,j)\notin E(G)} (1-\boldsymbol{P}_{ij})$*.*

Such models include both the classic Erdös-Rényi [56] and stochastic block models [84], as well as modern generative models such as NetGAN [20], variational graph autoencoders [93], and CELL [151]. We prove that subject to a *bounded overlap* condition, which ensures that the model does not simply memorize a single graph, edge independent models are inherently limited in their ability to generate graphs with high triangle and other subgraph densities. Notably, such high densities are known to appear in real-world social networks and other graphs. In particular, we define *overlap* as follows:

Definition (Expected Overlap)**.** *For symmetric* $\boldsymbol{P} \in [0,1]^{n\times n}$*, let*

$$\mathrm{Ov}(\boldsymbol{P}) := \frac{\mathbb{E}_{G_1,G_2\sim\mathcal{G}(\boldsymbol{P})}|E(G_1)\cap E(G_2)|}{\mathbb{E}_{G\sim\mathcal{G}(\boldsymbol{P})}|E(G)|}.$$

That is, for any $\boldsymbol{P} \in [0,1]^{n \times n}$, $\mathrm{Ov}(\boldsymbol{P}) \in [0,1]$ is the ratio of the expected number of edges shared by two graphs drawn independently from $\mathcal{G}(\boldsymbol{P})$ to the expected number of edges in a graph drawn from $\mathcal{G}(\boldsymbol{P})$. Our main result is that for any edge independent model with bounded overlap, $G \sim \mathcal{G}(\boldsymbol{P})$ cannot have too many triangles in expectation:

Theorem (Overlap Upper Bounds Triangles)**.** *For a graph G, let $\Delta(G)$ denote the number of triangles in G. For symmetric $\boldsymbol{P} \in [0,1]^{n \times n}$, $\mathbb{E}_{G \sim \mathcal{G}(\boldsymbol{P})}[\Delta(G)] = \mathcal{O}\left(n^3 \cdot \mathrm{Ov}(\boldsymbol{P})^3\right)$.*

We complement our negative results with a simple generative model that balances overlap and accuracy, performing comparably to more complex models in reconstructing many graph statistics.

In Section 4.2, we extend this investigation on limitations of *edge independent* random graph models to more general classes of graph models which allow for edge dependency in the graph distributions. We define *fully dependent* random graph models, which allow for arbitrary dependencies (i.e., an arbitrary distribution over graphs), and *node independent* random graph models, which serve as a medium between the two prior classes of models. Specifically, the concept of the node independent model is that each node has a distribution over embeddings, that all nodes' embeddings are sampled independently, and that the presence of an edge between two nodes is determined by a pairwise function of their embeddings:

Definition (Node Independent Graph Model)**.** *A distribution $\mathcal{A}$ over symmetric adjacency matrices $\boldsymbol{A} \in \{0,1\}^{n \times n}$, where, for some embedding space $\mathcal{E}$, some mutually independent random variables $\boldsymbol{x_1}, \ldots, \boldsymbol{x_n} \in \mathcal{E}$, and some symmetric function $e : \mathcal{E} \times \mathcal{E} \to [0,1]$, the entries of $\boldsymbol{A}$ are Bernoulli random variables $\boldsymbol{A}_{ij} = Bernoulli(e(\boldsymbol{x_i}, \boldsymbol{x_j}))$ that are mutually independent conditioned on $\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}$.*

We prove analogous bounds for these latter two new classes of models as we prove in Section 4.1 for edge independent models. Again, the main results concern upper bounds on triangle count in terms of our notion of *overlap*, which generalizes naturally to (not necessarily edge independent) graph distributions $\mathcal{A}$. Letting $\text{Ov}(\mathcal{A})$ denote this generalized overlap, recall that for edge independent models, the expected number of triangles is bounded above by $\mathcal{O}(n^3 \cdot \text{Ov}(\mathcal{A})^3)$. We can show that this bound for fully dependent models is relaxed to $\mathcal{O}(n^3 \cdot \text{Ov}(\mathcal{A})^1)$, and that the bound for node independent models sits in between at $\mathcal{O}(n^3 \cdot \text{Ov}(\mathcal{A})^{1.5})$. A summary of the main results in this section is given in Table 1.1. This section broadly works towards expanding the applicability of our theoretical bounds to practical algorithms, some of which violate edge dependency, as well as a more precise characterization of the capabilities of these more powerful models. The results in this chapter overall, together with those about exact embeddings, begin to shape an understanding of the capabilities of node embedding methods in general, beyond just classical spectral methods.

Table 1.1. Preview of results for Section 4.2. The level of edge dependency in graph generative models inherently limits the range of graph statistics, such as triangle counts, that they can produce. Note that overlap $\text{Ov}(\mathcal{A}) \in [0, 1]$, so a higher power on $\text{Ov}(\mathcal{A})$ means a tighter bound on the number of triangles.

Model	Upper Bound on Δ/n^3	Examples
Edge Independent	$\text{Ov}(\mathcal{A})^3$	• Erdős–Rényi • SBM • NetGAN [20, 152]
Node Independent	$\text{Ov}(\mathcal{A})^{3/2}$	• Variational Graph Auto-Encoder (VGAE) [94]
Fully Dependent	$\text{Ov}(\mathcal{A})$	• GraphVAE [170] • GraphVAE-MM [208] • Exponential Random Graph Models (ERGMs) [62]

1.1.4 Simplifying Deep Graph Networks

While much of our work characterizes the greater strength of modern deep methods for learning node representations relative to classical spectral ones, in Chapter 5, we also question the need for deep learning in certain settings. Note that in this chapter only, the graphs we work with are also associated with node feature vectors, that is, the graph data comprise not only an $n \times n$ adjacency matrix $\boldsymbol{A}$, but also a node feature matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$. The methods in this chapter use $\boldsymbol{A}$ in different ways to process $\boldsymbol{X}$ into learned node representations $\boldsymbol{X}' \in \mathbb{R}^{n \times d'}$, which can be used for a downstream task like node classification. We build on work of Wu et al. [197], who propose a method called Simple Graph Convolution (SGC), which eschews deep learning. SGC's approach is to produce $\boldsymbol{X}'$ by simply smoothing the features across the edges of the graph. This is much faster and simpler than deep methods like graph convolutional networks (GCNs) [94], which are based on the usual deep learning pipeline of backpropagation through an end-to-end model, but SGC is surprisingly found to be competitive with GCNs on benchmark tasks. However, SGC's feature smoothing operation is known [135] to be sensible only when the graph exhibits the common but not universal characteristic of homophily, wherein nodes mostly link to similar nodes. We ask whether a simple, non-deep approach like SGC can also handle heterophilous graph structure, wherein nodes mostly link to dissimilar nodes.

We propose a new method, Adaptive Simple Graph Convolution (ASGC), which processes each feature separately in a manner that can be either smoothing or non-smoothing, and thus can adapt to both homophilous and heterophilous structures. Like SGC, ASGC is not a deep model, instead being based on linear least squares, and hence is fast, scalable, and interpretable. Our approach is a sort of spectral method for feature processing: roughly, for each node feature $\boldsymbol{x} \in \mathbb{R}^n$, ASGC constructs a Krylov subspace generated by $\boldsymbol{A}$ and $\boldsymbol{x}$, that is, span $\left\{\boldsymbol{A}\boldsymbol{x}, \boldsymbol{A}^2\boldsymbol{x}, \ldots, \boldsymbol{A}^K\boldsymbol{x}\right\}$ for some hyperparameter K, and projects $\boldsymbol{x}$ onto this subspace to generate the fil-

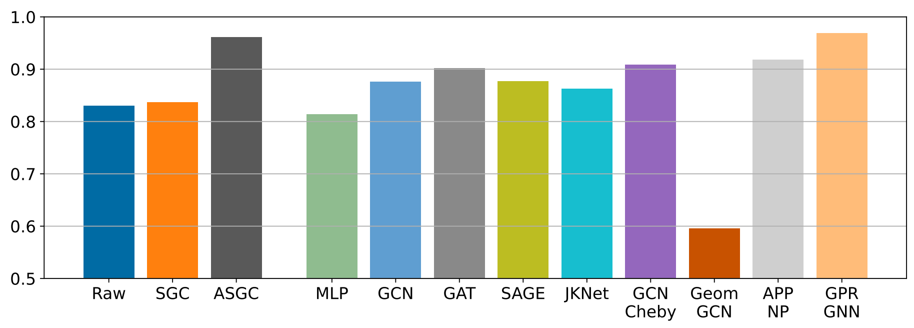


Figure 1.1. Node classification test accuracy with 12 methods as a proportion of the best method's accuracy; mean performance over 5 homophilous and 5 heterophilous networks. The three non-deep methods are grouped on the left. ASGC is competitive with the deep filtering method GPR-GNN and outperforms the other methods.

tered $\boldsymbol{x}'$, which is directly passed to a linear classifier. Empirically, ASGC is often competitive with recent deep models at node classification on a benchmark of real-world datasets – see Figure 1.1 for test accuracy results on the node classification task, aggregated across 5 homophilous and 5 heterophilous datasets. Note that this competitive performance is despite that our method, like SGC, is not a *feature learning* method – classification is linear following application of fixed linear algebraic filtering. This means that, in contrast to the deep methods, which form node embeddings *with* access to the training class labels, our method forms embeddings *without* these labels, but is still comparably effective on the classification task even with this barrier. Also in contrast to deep methods, the simplicity of our approach enables us to to prove performance guarantees on natural synthetic data models. The SGC paper questioned whether the complexity of graph neural networks is warranted for common graph problems involving homophilous networks; our results similarly suggest that, while deep learning often achieves the highest performance, heterophilous structure alone does not necessitate these more involved methods. Together with SGC, these results call into question the near-ubiquity of deep learning in recently-proposed graph machine learning methods.

1.2 Bibliography Notes

Most of the content presented in the thesis is based on work that is previously published work, is to be published, or is currently under submission:

1. Section 2.1 has been published with the title "InfiniteWalk: Deep Network Embeddings as Laplacian Embeddings with a Nonlinearity" at the Conference on Knowledge Discovery in Databases (KDD), 2020 [30].

2. Section 2.2 has been published with the title "DeepWalking Backwards: From Embeddings Back to Graphs" at the International Conference on Machine Learning (ICML), 2021 [32].

3. Section 3.1 has been published with the title "Node Embeddings and Exact Low-Rank Representations of Complex Networks" at the Conference on Neural Information Processing Systems, 2020 [34].

4. Section 4.1 has been published with the title "On the Power of Edge Independent Graph Models" at the Conference on Neural Information Processing Systems, 2021 [33].

5. Section 5.1 is to be published with the title "Simplified Graph Convolution with Heterophily" at the Conference on Neural Information Processing Systems, 2022 [31].

6. The work in Section 3.2 on "Nonnegative Symmetric Representations of Sparse Networks" is under submission.

7. The work in Section 4.2 on "The Role of Edge Dependency in Random Graph Models" is under submission.

CHAPTER 2

UNDERSTANDING MODERN NODE EMBEDDING METHODS

In this chapter, we pursue two paths to understanding the action of modern node embedding methods as exemplified by DeepWalk [144]. We first derive a simplified expression for the DeepWalk objective which clarifies its relationship with classical spectral embeddings. We also propose and pursue the problem of *inverting* node embedding, that is, taking node embeddings as input and outputting a graph with approximately these embeddings; we then probe the graph to see what information is captured in the embeddings.

2.1 Deep Network Embeddings as Laplacian Embeddings with a Nonlinearity

The primary classical approach for the node embedding task is spectral embedding: nodes are represented by their corresponding values in the smallest eigenvectors of the graph Laplacian. Spectral embedding methods include the Shi-Malik normalized cuts algorithm [169], Laplacian Eigenmaps [17], and spectral partitioning methods for stochastic block models [124]. They also include many spectral clustering methods, which apply spectral embeddings to general datasets by first transforming them into a graph based on data point similarity [133]. The spectral embedding approach has recently been exceeded in predictive performance on many tasks by methods inspired by Word2vec [126], which performs the related word embedding task. Word2vec forms representations for words based on the frequency with which they co-occur with other words, called context words, within a fixed distance T in natural text. The DeepWalk [144], LINE [182], and node2vec [75] algorithms, among others, adapt this idea to network data. In particular, DeepWalk takes several random walks on the network, treating the nodes as words, and treating the walks as sequences of words in text. It has been shown in [109] that the representations learned by this approach are implicitly forming a low-dimensional factorization of a known matrix, which contains the pointwise mutual information (PMI) of co-occurences between nodes in the random walks. Work by Qiu et al. [147] gives a closed form expression for this matrix and shows a connection to the normalized graph Laplacian. This motivates their NetMF algorithm, which performs a direct factorization, improving on the performance of DeepWalk on a number of tasks.

In this section, we consider DeepWalk in the limit as the window size T goes to infinity. We derive a simple expression for the PMI matrix in this limit:

$$\boldsymbol{M}_{\boldsymbol{\infty}} = v_G \cdot \mathbf{D}^{1/2} \left(\tilde{\mathbf{L}}^{+} - \mathbf{I} \right) \mathbf{D}^{1/2} + \mathbf{J}, \tag{2.1}$$

where $\mathbf{D}$ is the degree matrix, v_G is the trace of $\mathbf{D}$ (twice the number of edges in G), $\tilde{\mathbf{L}}$ is the normalized Laplacian (i.e., $\mathbf{I} - \mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2}$), and $\mathbf{J}$ is the all-ones matrix. $\tilde{\mathbf{L}}^{+}$ is the pseudoinverse of $\tilde{\mathbf{L}}$. One can show that $\mathbf{D}^{1/2}\tilde{\mathbf{L}}^{+}\mathbf{D}^{1/2}$ is equal to the unnormalized Laplacian pseudoinverse $\mathbf{L}^{+}$, the central object in classic spectral embeddings, up to a rank-2 component (see Equation 2.6 in Section 2.1.2.3). Thus, $\boldsymbol{M}_{\boldsymbol{\infty}}$ is equal to $\mathbf{L}^{+}$ plus at most a rank-3 component and a diagonal matrix.

Not surprisingly, embeddings formed directly using a low dimensional factorization of $\boldsymbol{M}_{\boldsymbol{\infty}}$ itself perform poorly on downstream tasks compared to DeepWalk and other skip-gram methods. However, we show that after an element-wise application of a linear function followed by a logarithm, in particular, $x \to \log(1 + x/T)$, $\boldsymbol{M}_{\boldsymbol{\infty}}$ approximates the finite-T PMI matrix. Embeddings formed by factorizing this transformed matrix are competitive with DeepWalk and nearly competitive with the

NetMF method of Qiu et al. [147], when evaluated on multi-label node classification tasks. We call our method InfiniteWalk.

Note that the window hyperparameter T only appears in the entrywise nonlinearity in InfiniteWalk and not in the formula for $\boldsymbol{M_\infty}$. This is perhaps surprising, as T is a key hyperparameter in existing methods. Our results suggest that T's importance lies largely in determining the shape of the nonlinearity applied. Since $\boldsymbol{M_\infty}$ is closely related to the Laplacian pseudoinverse, the key difference between DeepWalk and classic spectral embedding seems to be the application of this nonlinearity.

In more detail, note that our results show that InfiniteWalk well approximates DeepWalk by forming a low-rank factorization of a nonlinear entrywise transformation of $\boldsymbol{M_\infty}$. Classic spectral embedding and clustering methods [169, 124, 17] embed nodes using the eigenvectors corresponding to the smallest eigenvalues of the Laplacian $\mathbf{L}$ (or a variant of this matrix), which are the largest eigenvalues of $\mathbf{L}^+$. Thus, these methods can be viewed as embedding nodes using an optimal low-dimensional factorization of $\mathbf{L}^+$ (lying in the span of $\mathbf{L}^+$'s top eigenvectors), without applying an entrywise nonlinearity.

Inspired by this observation, we simplify the idea of a nonlinear transformation of the Laplacian pseudoinverse even further: thresholding it to a binary matrix. In particular, we form the binary matrix

$$[\mathbf{L}^+ \geq c],$$

where c is an element of $\mathbf{L}^+$ itself of some hyperparameter quantile (e.g. the median or the 95th percentile element). Surprisingly, embeddings from factorizing this binary matrix are also competitive with DeepWalk and the method of Qiu et al. on many tasks.

Broadly, our results strengthen the theoretical connection between classical methods based on factorizing the graph Laplacian and more recent "deep" methods for node embedding. They suggest that these methods are not as different conceptually as they may seem at first glance.

2.1.1 Background

We begin by surveying prior work on skip-gram-based network embeddings and their connections to matrix factorization, which our work builds on.

2.1.1.1 Skip-Gram

In word embedding models, words are typically treated as both words and contexts. A context is simply a word that appears within a window of length T of another word. As formalized in [71], given a dataset of words $w \in \mathcal{W}$, contexts $c \in \mathcal{C}$ (typically $\mathcal{W} = \mathcal{C}$), and (w, c) word-context co-occurrence pairs $(w, c) \in \mathcal{D}$, the "skip-gram" model for training word and context embeddings $\mathbf{v}_w$, $\mathbf{v}_c$ [126] has the following log-likelihood objective:

$$\arg\max_{\{\mathbf{v}_c\}_\mathcal{C}, \{\mathbf{v}_w\}_\mathcal{W}} \sum_{(w,c)\in\mathcal{D}} \log \Pr(c|w), \quad \text{where} \quad \Pr(c|w) = \frac{e^{\mathbf{v}_c \cdot \mathbf{v}_w}}{\sum_{c' \in \mathcal{C}} e^{\mathbf{v}_{c'} \cdot \mathbf{v}_w}}.$$

We can see that the objective encourages co-occuring pairs $(w, c) \in \mathcal{D}$ to have similar embeddings with large dot product $\mathbf{v}_c \cdot \mathbf{v}_w$. This exact objective is not used as repeatedly evaluating the partition function is too computationally expensive; [126] proposes a substitute: the skip-gram with negative sampling (SGNS). Here, an auxiliary 'negative' dataset $\mathcal{D}'$ consisting of random (w, c) pairs not appearing in $\mathcal{D}$ is used. Pairs in this negative set are encouraged to have dissimilar embeddings with small $\mathbf{v}_c \cdot \mathbf{v}_w$.

2.1.1.2 Implicit PMI Matrix

Levy and Goldberg [109] prove that SGNS implicitly factors a matrix whose entries gave the pointwise mutual information (PMI) of occurrence of a word w_i and occurrence of a context c_j. Given a dataset of these co-occurrences, an element of this matrix $\mathbf{M}$ is given by

$$\begin{aligned}\mathbf{M}_{ij} &= \log\left(\frac{\Pr(w_i, c_j)}{\Pr(w_i)\Pr(c_j)}\right) - \log(b)\\ &= \log\left(\frac{\#(w,c)\cdot|\mathcal{D}|}{\#(w)\cdot\#(c)}\right) - \log(b).\end{aligned}$$

where $b = |\mathcal{D}'|/|\mathcal{D}|$ is the ratio of negative samples to positive samples. Their proof assumes that the dimensionality of the embedding is at least the cardinality of the word/context set; the analysis of Li et al. [110] relaxes assumptions under which this equivalence holds. Several works, including [15] and [68], propose generative models for word-context co-occurrence to explain the effectiveness of PMI-based methods in linearizing semantic properties of words. More recently, the analysis of Allen et al. in [8] and [9] has provided explanations of this phenomenon based on geometric properties of the PMI matrix, without the strong assumptions required by the generative models. The extensive research and results on the skip-gram PMI matrix make it an intrinsically interesting object in representation learning.

2.1.1.3 Networks

The DeepWalk method [144] applies the SGNS model to networks, where the word and context sets are the nodes in the network, and the co-occuring pairs $\mathcal{D}$ are node pairs that appear within a window of length T hops in a set of length L random walks run on the graph. Qiu et al. [147] derived the following expression for the PMI matrix in the context of random walks on undirected, connected, and non-bipartite graphs for DeepWalk: in the limit as the number of walks originating at each node $\gamma \to \infty$ and walk length $L \to \infty$, it approaches

$$\mathbf{M_T} = \log\left(v_G\left(\tfrac{1}{T}\sum_{k=1}^{T}\mathbf{P}^k\right)\mathbf{D}^{-1}\right) - \log b, \tag{2.2}$$

where log is an element-wise natural logarithm, v_G (the "volume" of the graph) is the sum of the degrees of all of the nodes, and $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$ is the random walk transition matrix.

Rather than sampling from random walks as in DeepWalk, the NetMF algorithm of Qiu et al. explicitly calculates and factors this matrix to produce embeddings, outperforming DeepWalk significantly on multi-label node classification tasks. For low T, NetMF manually computes the exact sum, whereas for high T, it computes a low-rank approximation via SVD of the symmetrized transition matrix, $\tilde{\mathbf{P}} = \mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2}$. While Qiu et al. analyze the effect of increasing T on the spectrum of the resulting matrix, they do not pursue the $T \to \infty$ limit, stopping at $T = 10$ as in the original DeepWalk paper. We show that this limiting matrix is both meaningful and simple to express.

2.1.1.4 Other Approaches

Some other node embedding algorithms share significant similarities with DeepWalk. Qiu et al [147] showed the LINE method to also be implicit matrix factorization, though its algorithm is based on edge sampling rather than sampling from random walks. In particular, its factorized matrix is a special case of the DeepWalk matrix with $T = 1$. We include the performance of LINE in our empirical results. node2vec [75] is a generalization of DeepWalk which uses second-order random walks: the distribution of the following node in node2vec walks depends on the current and preceding nodes rather than only the current node as in DeepWalk. Hyperparameters allow the walk to approach BFS-like or DFS-like behavior as desired, which the authors assert extract qualitatively different information about node similarities.

Several architectures for applying convolutional neural networks to network data in an end-to-end fashion have been developed in the past few years, including the graph convolutional networks (GCNs) of [94] and [46], and some methods leverage these architectures to produce node embeddings: for example, Deep Graph Infomax [189] uses GCNs to maximize a mutual information objective involving patches around nodes. Recent work from Wu et al. [197] shows that much of the complexity of GCNs comes from components inspired by other forms of deep learning that have limited utility for network data. In the same way, we seek to further the investigation of the core principles of "deep" network embeddings apart from their inspiration in word embedding and neural networks. We note that, like DeepWalk, and the related methods, we focus on unsupervised embeddings, derived solely from a graph's structure, without training, e.g., on node labels.

2.1.2 Methodology

We now present our main contributions, which connect DeepWalk in the infinite window limit to classic spectral embedding with a nonlinearity. We discuss how this viewpoint clarifies the role of the window size parameter T in DeepWalk and motivates a very simple embedding technique based on a binary thresholding of the graph Laplacian pseudoinverse.

2.1.2.1 Derivation of Limiting PMI Matrix

We start by showing how to simplify the expression in Equation 2.2 for the DeepWalk PMI given by [147] in the limit $T \to \infty$. We first establish some well-known facts about random walks on graphs. First, $\mathbf{P}^\infty$ is well-defined under our assumption that the graph is undirected, connected, and non-bipartite. It is rank-1 and equal to $\mathbf{1}\tilde{\boldsymbol{d}}^\top$, where $\mathbf{1}$ is a column vector of ones and $\tilde{\boldsymbol{d}}$ is the probability mass of each node in the stationary distribution of the random walk as a column vector. Note that $\tilde{\boldsymbol{d}}_i = \mathbf{D}_{ii}/v_G$. That is, the probability mass of a node in the stationary distribution is proportional to its degree. We let $\tilde{\mathbf{D}} = \mathbf{D}/v_G$ denote the diagonal matrix with entries $\tilde{\mathbf{D}}_{ii} = \tilde{\boldsymbol{d}}_i$.

Let λ_i and $\mathbf{w_i}$ be the i^{th} eigenvalue and eigenvector of the symmetrized transition matrix $\tilde{\mathbf{P}} = \mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2}$. We have $\lambda_1 = 1$ and $\mathbf{w_1} = \left(\sqrt{\tilde{\boldsymbol{d}}_1}, \ldots, \sqrt{\tilde{\boldsymbol{d}}_n}\right)^\top$. From [108], for any positive integer k,

$$\mathbf{P}^k = \mathbf{P}^\infty + \sum_{j=2}^{n} \lambda_j^k \mathbf{v_j}\mathbf{v_j}^\top \tilde{\mathbf{D}}, \tag{2.3}$$

where $\mathbf{v_i} = \tilde{\mathbf{D}}^{1/2}\mathbf{w_i}$. We rewrite the expression in Equation 2.3 for $\mathbf{P}^k$ and the expression Equation 2.2 of Qiu et al. for the PMI matrix, setting the negative sampling ratio b to 1 for the latter (i.e., one negative sample per positive sample):

$$\mathbf{P}^k = \mathbf{1}\tilde{\boldsymbol{d}}^\top + \tilde{\mathbf{D}}^{1/2} \sum_{j=2}^{n} \lambda_j^k \mathbf{w_j}\mathbf{w_j}^\top \tilde{\mathbf{D}}^{1/2} \quad \text{and}$$
$$\mathbf{M_T} = \log\left(T^{-1} \sum_{k=1}^{T} \mathbf{P}^k \tilde{\mathbf{D}}^{-1}\right).$$

Substituting the former into the latter, then rearranging the order of the summations and applying the geometric series formula yields

$$\begin{aligned}\mathbf{M_T} &= \log\left(\mathbf{1}\mathbf{1}^\top + T^{-1}\tilde{\mathbf{D}}^{1/2}\left(\sum_{k=1}^{T}\sum_{j=2}^{n} \lambda_j^k \mathbf{w_j}\mathbf{w_j}^\top\right)\tilde{\mathbf{D}}^{1/2}\right)\\ &= \log\left(\mathbf{1}\mathbf{1}^\top + T^{-1}\tilde{\mathbf{D}}^{1/2}\left(\sum_{j=2}^{n} \frac{\lambda_j(1-\lambda_j^T)}{1-\lambda_j} \mathbf{w_j}\mathbf{w_j}^\top\right)\tilde{\mathbf{D}}^{1/2}\right).\end{aligned}$$

Now we consider the limit as $T \to \infty$. Define

$$\boldsymbol{M_\infty} = \lim_{T\to\infty} T \cdot \mathbf{M_T}.$$

Since $|\lambda_j| < 1$ for $j \neq 1$ [108], the $(1 - \lambda_j^T)$ terms go to 1 as $T \to \infty$, and we have:

$$\boldsymbol{M_\infty} = \lim_{T\to\infty} T \cdot \log\left(\mathbf{1}\mathbf{1}^\top + T^{-1}\tilde{\mathbf{D}}^{1/2}\left(\sum_{j=2}^{n} \frac{\lambda_j}{1-\lambda_j} \mathbf{w_j}\mathbf{w_j}^\top\right)\tilde{\mathbf{D}}^{1/2}\right).$$

Since $\log(1+\epsilon) \to \epsilon$ as $\epsilon \to 0$, for any constant real number c, $T\log(1+T^{-1}c) \to c$ as $T \to \infty$. We apply this identity element-wise, then simplify:

$$\begin{aligned}
\boldsymbol{M_\infty} &= \tilde{\mathbf{D}}^{1/2}\left(\sum_{j=2}^{n} \frac{\lambda_j}{1-\lambda_j}\mathbf{w_j}\mathbf{w_j}^\top\right)\tilde{\mathbf{D}}^{1/2} \\
&= \tilde{\mathbf{D}}^{1/2}\left(\sum_{j=2}^{n} \frac{1}{1-\lambda_j}\mathbf{w_j}\mathbf{w_j}^\top - \sum_{j=2}^{n} \frac{1-\lambda_j}{1-\lambda_j}\mathbf{w_j}\mathbf{w_j}^\top\right)\tilde{\mathbf{D}}^{1/2} \\
&= \tilde{\mathbf{D}}^{1/2}\left(\sum_{j=2}^{n} \frac{1}{1-\lambda_j}\mathbf{w_j}\mathbf{w_j}^\top + \mathbf{w_1}\mathbf{w_1}^\top - \sum_{j=1}^{n} \mathbf{w_j}\mathbf{w_j}^\top\right)\tilde{\mathbf{D}}^{1/2} \\
&= \tilde{\mathbf{D}}^{1/2}\left(\tilde{\mathbf{L}}^{+} + \mathbf{w_1}\mathbf{w_1}^\top - \mathbf{I}\right)\tilde{\mathbf{D}}^{1/2} \\
&= \mathbf{1}\mathbf{1}^\top + \tilde{\mathbf{D}}^{1/2}\left(\tilde{\mathbf{L}}^{+} - \mathbf{I}\right)\tilde{\mathbf{D}}^{1/2},
\end{aligned}$$

where the last step follows from $\mathbf{w_1}$ being the element-wise square root of $\tilde{\boldsymbol{d}}$. Note that the above equation gives the expression for $\boldsymbol{M_\infty}$ in Equation 2.1, since $\tilde{\mathbf{D}} = \mathbf{D}/v_G$. Similar analysis also leads to the following general expressions for the finite-T PMI matrix:

$$\begin{aligned}
\mathbf{M_T} &= \log\left(\mathbf{1}\mathbf{1}^\top + T^{-1}\tilde{\mathbf{D}}^{1/2}\tilde{\mathbf{P}}\tilde{\mathbf{L}}^{+}(\mathbf{I}-\tilde{\mathbf{P}}^{T})\tilde{\mathbf{D}}^{1/2}\right) \\
&= \log\left(\mathbf{1}\mathbf{1}^\top + T^{-1}\left(\mathbf{1}\mathbf{1}^\top + \tilde{\mathbf{D}}^{1/2}\left(\tilde{\mathbf{L}}^{+}\left(\mathbf{I}-\tilde{\mathbf{P}}^{T+1}\right) - \mathbf{I}\right)\tilde{\mathbf{D}}^{1/2}\right)\right). \qquad (2.4)
\end{aligned}$$

2.1.2.2 Approximation of Finite-T PMI Matrix via Limiting PMI Matrix

Note that the expression Equation 2.4 above for the finite-T matrix, when multiplied by T^{-1} differs from the limiting matrix only by the term $\tilde{\mathbf{D}}^{1/2}\tilde{\mathbf{L}}^{+}\tilde{\mathbf{P}}^{T+1}\tilde{\mathbf{D}}^{1/2}$, which vanishes as $T \to \infty$. So, we may approximate the finite-T matrix by simply dropping this term as follows:

$$\begin{aligned}
\mathbf{M_T} &\approx \log\left(R\left(\mathbf{1}\mathbf{1}^\top + T^{-1}\left(\mathbf{1}\mathbf{1}^\top + \tilde{\mathbf{D}}^{1/2}\left(\tilde{\mathbf{L}}^{+} - \mathbf{I}\right)\tilde{\mathbf{D}}^{1/2}\right)\right)\right) \\
&= \log\left(R\left(\mathbf{1}\mathbf{1}^\top + T^{-1}\boldsymbol{M_\infty}\right)\right), \qquad (2.5)
\end{aligned}$$

where $R(x)$ is any ramp function to ensure that the argument of the logarithm is positive. In our implementation, we use the function $R_\epsilon(x) = \max(\epsilon, x)$. We use the 64-bit floating-point machine precision limit ($\sim e^{-36}$) for ϵ. Note that the NetMF method of [147] uses $R_1(x)$; we find that a small positive value consistently performs better. Both ramping functions can be interpreted as producing the positive shifted PMI matrix (shifted PPMI) matrix introduced by Levy and Goldberg [109]. Other methods of avoiding invalid arguments to the logarithm are an interesting avenue for future work.

Note that the accuracy of Equation 2.5 is limited by the second largest eigenvalue of $\tilde{\mathbf{P}}$, which is known as the Fiedler eigenvalue. Smaller magnitudes of this eigenvalue are correlated with a faster mixing rate [108], the rate at which $\mathbf{P}^k \to \mathbf{P}^\infty$ as k increases. In Section 2.1.3.2 we show that for typical graphs, the Fiedler eigenvalue is relatively small, and so the approximation is very accurate for large T, e.g., $T = 10$, which is a typical setting for DeepWalk. The approximation is less accurate for small T, e.g., $T = 1$ (See Table 2.2.)

Effect of the Window Size T. Intuitively, the effect of T in Equation 2.5 is to control the "strength" of the logarithm nonlinearity, since, as noted previously, for any real constant c, $\log(1+T^{-1}c) \to T^{-1}c$ as $T \to \infty$. That is, the logarithm becomes a simple linear scaling in this limit. As we will see, even when the approximation of Equation 2.5 in inaccurate (when T is very low) this approximated matrix qualitatively has similar properties to the actual T-window PMI matrix, and produces similar quality embeddings, as measured by performance in downstream classification tasks. This finding suggests that the strength of the logarithm nonlinearity can influence the locality of the embedding (as modulated in DeepWalk by the window size T) independently of modifying the arguments of the nonlinearity, which contain the actual information from the network as filtered by length-T windows.

2.1.2.3 Binarized Laplacian Pseudoinverse

Motivated by the view of DeepWalk as a variant of classic Laplacian factorization with an entrywise nonlinearity, we investigate a highly simplified version of InfiniteWalk. We construct a binary matrix by 1) computing the pseudoinverse of the unnormalized Laplacian $\mathbf{L}^+$, 2) picking a quantile hyperparameter $q \in (0, 1)$, 3) determining the quantile q element value, and 4) setting all elements less than this value to 0 and others to 1. In other words, an element of this matrix $\mathbf{B}$ is given by $\mathbf{B}_{ij} = [(\mathbf{L}^+)_{ij} \geq c]$, where c is the q quantile element of $\mathbf{L}^+$. We then produce embeddings by partially factoring this matrix $\mathbf{B}$ as with the PMI matrices. Interestingly, this can be interpreted as factorizing the adjacency matrix of an implicit derived network whose sparsity is determined by q. Gaining a better understanding of the structure and interpretation of this network is an interesting direction from future work.

Note that in this method, we use the unnormalized Laplacian $\mathbf{L}$ rather than the normalized Laplacian $\tilde{\mathbf{L}} = \mathbf{D}^{1/2}\mathbf{L}\mathbf{D}^{1/2}$, which appears in the expression Equation 2.1 for $\boldsymbol{M}_{\boldsymbol{\infty}}$. This is because, as we will show, the limiting PMI matrix is equal to the pseudoinverse of the unnormalized Laplacian up to a rank-3 term and a diagonal adjustment. We can rewrite our expression for the limiting matrix by expanding the normalized Laplacian in terms of the normalized Laplacian as follows:

$$\mathbf{M}_\infty = \mathbf{1}\mathbf{1}^T + v_G \left(\underbrace{\mathbf{D}^{1/2} \left(\mathbf{D}^{1/2}\mathbf{L}\mathbf{D}^{1/2}\right)^+ \mathbf{D}^{1/2}} - \mathbf{D}^{-1} \right).$$

Consider the underbraced term above containing $\mathbf{L}$. If this term had a true inverse rather than a pseudoinverse, the four factors involving the degree matrix would simply cancel. Instead, application of a variant of the Sherman-Morrison formula for pseudoinverses [125] results in the following expression for this term:

$$\mathbf{D}^{1/2} \left(\mathbf{D}^{1/2}\mathbf{L}\mathbf{D}^{1/2}\right)^+ \mathbf{D}^{1/2} = (\mathbf{I} - \mathbf{1}\tilde{\boldsymbol{d}}^\top)\mathbf{L}^+(\mathbf{I} - \tilde{\boldsymbol{d}}\mathbf{1}^\top).$$

This yields the following alternate expression for the limiting PMI matrix:

$$\mathbf{M}_\infty = \mathbf{1}\mathbf{1}^T + v_G \left((\mathbf{I} - \mathbf{1}\tilde{\boldsymbol{d}}^\top)\mathbf{L}^+(\mathbf{I} - \tilde{\boldsymbol{d}}\mathbf{1}^\top) - \mathbf{D}^{-1} \right). \tag{2.6}$$

In our context of binarizing $\mathbf{L}^+$ by a quantile, note that addition by the all-ones matrix and multiplication by v_G does not affect the ranks of the elements within the matrix, and the subtraction by the diagonal matrix $\mathbf{D}^{-1}$ affects relatively few elements. Hence we might expect binarizing $\mathbf{L}^+$ by thresholding on quantiles to have a similar effect as binarizing the limiting PMI matrix itself.

Binarization is arguably one of the simplest possible methods of augmenting the Laplacian with a nonlinearity. As we will see, this method has good downstream performance compared to DeepWalk and related methods. We argue that this suggests that the core advancement of deep node embeddings over classical spectral embedding methods based on factorizing the Laplacian is application of a nonlinearity.

2.1.3 Experimental Setup

We now discuss how we empirically validate the performance of the limiting PMI matrix method presented in Section 2.1.2.2 and the Laplacian pseudoinverse binarization method of Section 2.1.2.3.

2.1.3.1 Data Sets

We use three of the four datasets used in the evaluation of the NetMF algorithm [147]. Table 2.1 provides network statistics. Figure 2.1 provides the eigenvalue distribution of the symmetrized random walk matrix $\tilde{\mathbf{P}}$ for each network.

BlogCatalog [4] is a social network of bloggers. The edges represent friendships between bloggers, and node labels represent group memberships corresponding to interests of bloggers.

Dataset	Nodes	Edges	Fiedler Eigval.
BlogCatalog	10,312	333,983	0.568
PPI	3,852	76,546	0.800
Wikipedia	4,777	184,812	0.504

Table 2.1. Network statistics for experiments in Section 2.1.

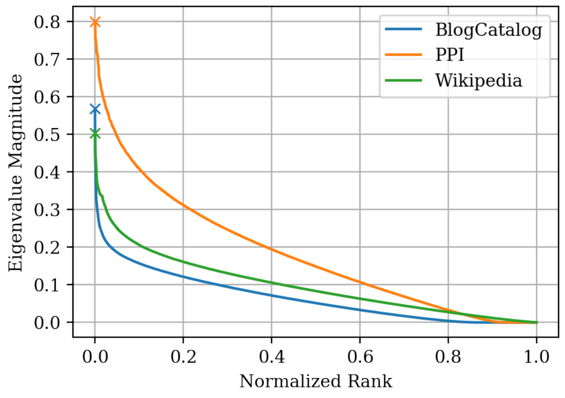


Figure 2.1. Sorted eigenvalues of $\tilde{\mathbf{P}}$ for each network. The top eigenvalue of 1 is excluded, and the Fiedler eigenvalues are marked with X's.

Protein-Protein Interaction (PPI) [177] is a subgraph of the PPI network for Homo Sapiens. nodes represent proteins, edges represent interactions between proteins, and node labels represent biological states. We use only the largest connected component, which has over 99% of the nodes.

Wikipedia is a co-occurrence network of words from a portion of the Wikipedia dump. Nodes represent words, edges represent co-occurrences within windows of length 2 in the corpus, and labels represent inferred part of speech (POS) tags.

2.1.3.2 Procedure

Implementation. See Algorithm 1 for the pseudocode of our limiting PMI matrix method. We use the NumPy [136] and SciPy [89] libraries for our implementation. The most expensive component of the algorithm is the pseudoinversion of the graph

Laplacian. While there is significant literature on approximating this matrix, or vector products with it [175, 96, 91, 148], we simply use the dense SVD-based function from NumPy. For graphs of larger scale, this method would not be practical. The truncated sparse eigendecomposition is handled via SciPy's packer to ARPACK [101], which uses the Implicitly Restarted Lanczos Method. As in [147], to generate a d-dimensional embedding, we return the singular vectors corresponding to the d largest singular values, scaling the dimensions of the singular vectors by the square roots of the singular values. For classification, we use the scikit-learn [141] packer to LIBLINEAR [58]. Demo code for InfiniteWalk is available at github.com/schariya/infwalk.

Algorithm 1 InfiniteWalk

```
1: Compute $\mathbf{M}_\infty = \mathbf{1}\mathbf{1}^\top + \tilde{\mathbf{D}}^{1/2}\left(\tilde{\mathbf{L}}^{+} - \mathbf{I}\right)\tilde{\mathbf{D}}^{1/2}$
2: Compute $\mathbf{M} = \log\left(R_\epsilon\left(\mathbf{1}\mathbf{1}^\top + T^{-1}\mathbf{M}_\infty\right)\right)$
3: Rank-$d$ approximation by truncated eigendecomposition: $\mathbf{M} \approx \mathbf{V} \times \text{diag}(\mathbf{w}) \times \mathbf{V}^\top$
4: return $\mathbf{V} \times \text{diag}(\sqrt{|\mathbf{w}|})$ as node embeddings
```

Evaluation Setting. To investigate the usefulness and meaningfulness of the limiting PMI matrix, we evaluate the quality of embeddings generated by its truncated SVD after applying the element-wise ramp-logarithm described in Section 2.1.2.2. For this task, we closely follow the same procedure as in [144] and [147]. We use one-vs-rest logistic regression on the embeddings for multi-label prediction on the datasets. The classifiers employ L2 regularization with inverse regularization strength $C = 1$. Classifiers are trained on a portion of the data, with the training ratio being varied from 10% to 90%; the remainder is used for testing. As in [144] and [147], we assume that the number of labels for each test example is given. In particular, given that a node is assigned k labels, the classifier predicts exactly the k classes to which it assigned the highest probability. We use the mean scores over 10 random splits of the training and test data for each training ratio. We evaluate the micro-F1 and macro-F1 scores of classifiers using our embedding.

Hyperparameter Choices. We compare our results to those of DeepWalk [144], LINE [182], and NetMF [147] as reported in [147]. The hyperparameters used for DeepWalk are the preferred default settings of its authors: window size $T = 10$, walk length $L = 40$, and number of walks starting from each node $\gamma = 80$. Results from the second-order variant of LINE are reported. As the authors of NetMF report results for window sizes $T = 1$ and $T = 10$, we similarly report results for InfiniteWalk with $T = 1$ and $T = 10$. We expect the results of InfiniteWalk, as an approximation of the NetMF method in the high T limit, to at least be roughly similar for the higher $T = 10$ setting. We also include results with our limiting $T \to \infty$ matrix, though only for illustrative purposes. As the limiting matrix is essentially a simple linear transformation of the graph Laplacian's pseudoinverse, we expect embeddings derived thereof to perform relatively poorly. The entrywise nonlinearity seems to be critical. The embedding dimensionality is 128 for all methods as in both [144] and [147].

2.1.3.3 Binarized Laplacian Pseudoinverse

We implement and evaluate the simplified method of factorizing a binarized version of the unnormalized Laplacian pseudoinverse (described in Section 2.1.2.3) in the same way. We present results for the best values of quantile hyperparameter q found by rough grid search. As with the window size T, the best value is expected to vary across networks. We compare to the performance of NetMF, the sampling methods LINE and DeepWalk, and classical methods - since the normalized and unnormalized Laplacians themselves both perform poorly on these tasks, we compare to factorizing the adjacency matrix itself. Again, since inverting and binarizing is one of the simplest possible nonlinearities to apply the Laplacian, good downstream performance suggests that the addition of a nonlinearity is the key advancement of deep node embeddings from classical embeddings of the Laplacian itself.

Dataset	Error ($T = 1$)	Error ($T = 10$)	Ramped Elts. ($T = 1$)	Ramped Elts. ($T = 10$)
BlogCatalog	2.456	0.001273	0.18340	0.0004901
PPI	2.588	0.041520	0.14400	0.0025210
Wikipedia	1.355	0.004892	0.08892	0.0005943

Table 2.2. PMI Approximation Error. The first two columns give the Frobenius norm of the difference between the true PPMI matrix $\mathbf{M}_T$ and our approximation based on $\mathbf{M}_\infty$ (see Equation 2.5), divided by the norm of $\mathbf{M}_T$. The log-ramp nonlinearity with R_1, as used in the NetMF method, is applied to both matrices. The last two columns give the number of elements that are affected by the ramping component of the nonlinearity in one matrix but not the other, normalized by the size of the matrices.

2.1.4 Results

We now discuss our experimental results on both the limiting PMI-based algorithm and the simple Laplacian binarization algorithm.

2.1.4.1 PMI Approximation Error

In Table 2.2, we show how closely the PMI approximation given by Equation 2.5 matches the true PMI matrix. We can see from Table 2.1 that the Fiedler eigenvalues of our graphs are bounded away from 1. Thus, as expected, the approximation of the finite-T PMI matrix via the limiting matrix is quite close at $T = 10$, but not so at $T = 1$. Additionally, at $T = 10$, the elements which are affected by the ramping component of the nonlinearity are similar between our approximation and the true PMI matrix. The accurate approximation at $T = 10$ explains why InfiniteWalk performs similarly on downstream classification tasks. Interestingly, at $T = 1$, InfiniteWalk performs competitively, in spite of inaccurate approximation.

2.1.4.2 Multi-Label Classification

In Figure 2.2 we show downstream performance of embeddings based on the limiting $\mathbf{M}_\infty$ approximation, compared to other methods. Across both metrics and all datasets, NetMF and InfiniteWalk are generally or par with or outperform the sampling-based methods, LINE and DeepWalk. As observed in [147], DeepWalk and NetMF with $T = 10$ have better overall performance than LINE and NetMF with $T = 1$ on the BlogCatalog and PPI networks, while the inverse is true for the Wikipedia network. This suggests that shorter-range dependencies better capture Wikipedia's network structure. As expected, InfiniteWalk with the $T = 10$ nonlinearity performs better than the version with the $T = 1$ nonlinearity on the former two datasets, while the inverse is true for Wikipedia. In all cases, the factorization of the $\boldsymbol{M}_\infty$ PMI matrix itself performs poorly. These findings support our hypothesis that changing the strength of the logarithm nonlinearity can largely emulate the effect of actually changing the window size T in sampling and deterministic approaches.

While maximizing downstream performance is not the focus of our work, we observe that, overall, InfiniteWalk has performance competitive with if slightly inferior to NetMF (see Figure 2.3). On BlogCatalog, InfiniteWalk underperforms relative to NetMF. On PPI, InfiniteWalk outperforms NetMF when less than half the data is used for training, but underperforms when given more training data. On Wikipedia, InfiniteWalk underperforms relative to NetMF on macro-F1 score, but outperforms NetMF on micro-F1 score.

Binarized Laplacian Pseudoinverse. In Figure 2.4 we show down stream performance of our simple method based on factorizing the binarized Laplacian pseudoinverse. This method performs remarkably well on both $T = 10$ networks. On PPI, it matches the performance of NetMF, and on BlogCatalog, it is nearly on par again, accounting for most of the improvement from the classical method. On the $T = 1$ network, Wikipedia, it is less successful, especially on Macro-F1 error, but still improves on the classical method. These result again support our hypothesis that the key ingredient of improved node embeddings seems to be the application of a nonlinearity to the Laplacian pseudoinverse.

Elements of Limiting PMI Matrices. Since we are introducing the limiting PMI matrix as an object for future investigation, we also give a preliminary qualitative description of its elements. See Figure 2.5 for a visualization of the element distribution for the three networks we investigate. Across these networks, these matrices tend to contain mostly small negative elements, corresponding to weak negative correlations between nodes, as well as some large positive values, corresponding to strong positive correlations. The distributions overall have similar shapes, and, interestingly, have roughly the same ratios of negative values to positive values, corresponding to roughly the same ratios of negative correlations to positive correlations.

2.1.5 Conclusion

In this section we have simplified known expressions for the finite-T network PMI matrix and derived an expression for the $T \to \infty$ matrix in terms of the pseudoinverse of the graph Laplacian. This expression strengthens connections between SGNS-based and classic spectral embedding methods.

We show that, with a simple scaled logarithm nonlinearity, this limiting matrix can be used to approximate finite-T matrices which yield competitive results on downstream node classification tasks. This finding suggests that the core mechanism of SGNS methods as applied to networks is a simple nonlinear transformation of classical Laplacian embedding methods. We even find that just binarizing the Laplacian pseudoinverse by thresholding often accounts for most of the performance gain from classical approaches, suggesting again the important of the nonlinearity.

We view this work as a step in understanding the core mechanisms of SGNS-based embedding methods. However many open questions remain.

For example, one may ask why the scaled logarithm non-linearity is a good choice. Relatedly, how robust is performance to changes in the nonlinearity? Our results on binarization of the Laplacian pseudoinverse indicate that it may be quite robust,

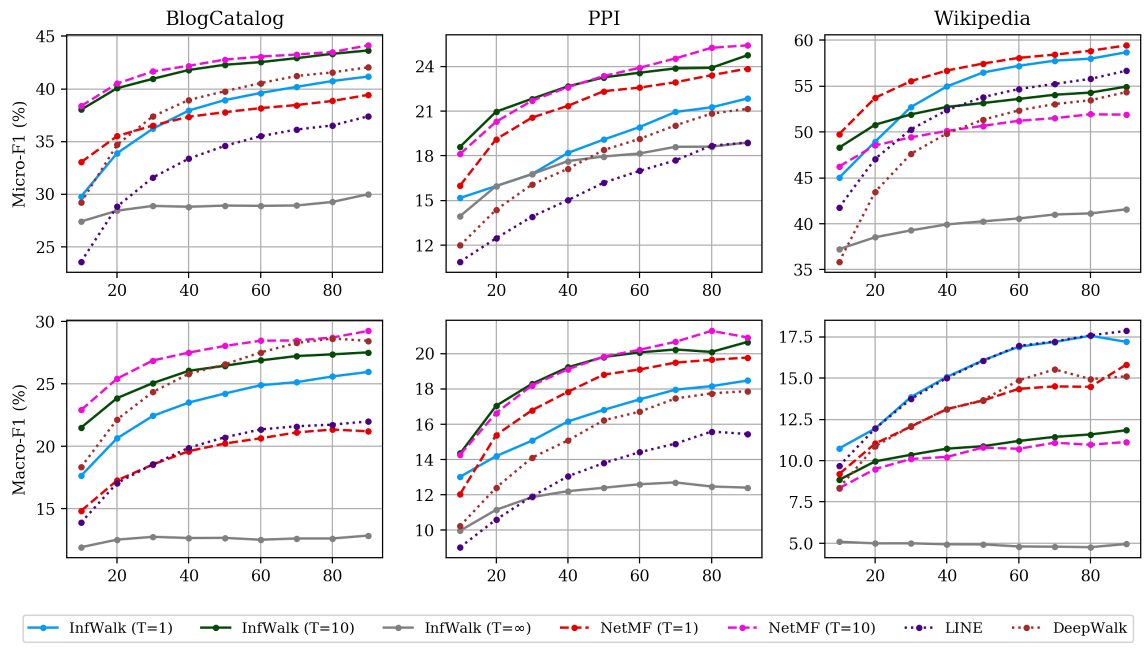


Figure 2.2. Multi-label classification performance on the BlogCatalog, PPI, and Wikipedia networks. Micro-F1 score (top) and Macro-F1 score (bottom) versus percent of data used for training. Results for InfiniteWalk (Algorithm 1) all appear as solid lines.

but this is worth further exploration. Finally, as discussed, our binarization method can be viewed as producing the adjacency matrix of a graph based on the Laplacian pseudoinverse, and then directly factoring this matrix. Understanding how this derived graph relates to the input graph would be a very interesting next step in understanding the surprisingly competitive performance of this method.

Additionally, as discussed previously, node2vec [75] is a generalization of DeepWalk that adds additional hyperparameters to create second-order random walks. Qiu et al. [147] also provide an expression for the matrix that is implicitly factored by node2vec, so pursuing the $T \to \infty$ limit of this matrix may provide insight into node2vec and an interesting generalization of DeepWalk's limiting PMI matrix.

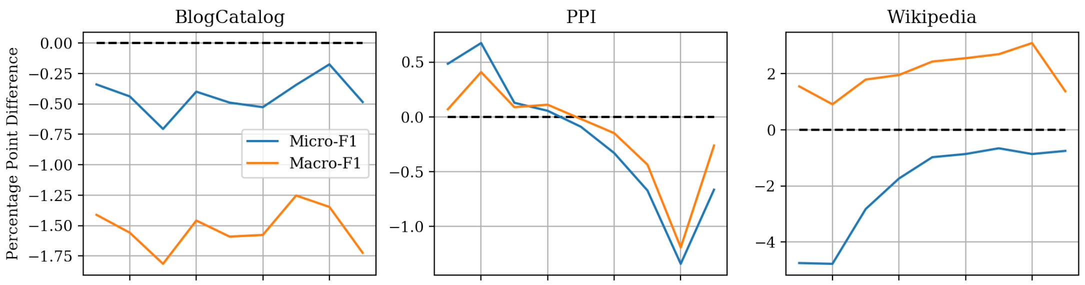


Figure 2.3. Performance of InfiniteWalk relative to NetMF [147]. F1 score (%) of InfiniteWalk minus F1 score of NetMF versus percent of data used for training. For both methods, $T = 10$ is used for BlogCatalog and PPI, and $T = 1$ is used for Wikipedia.

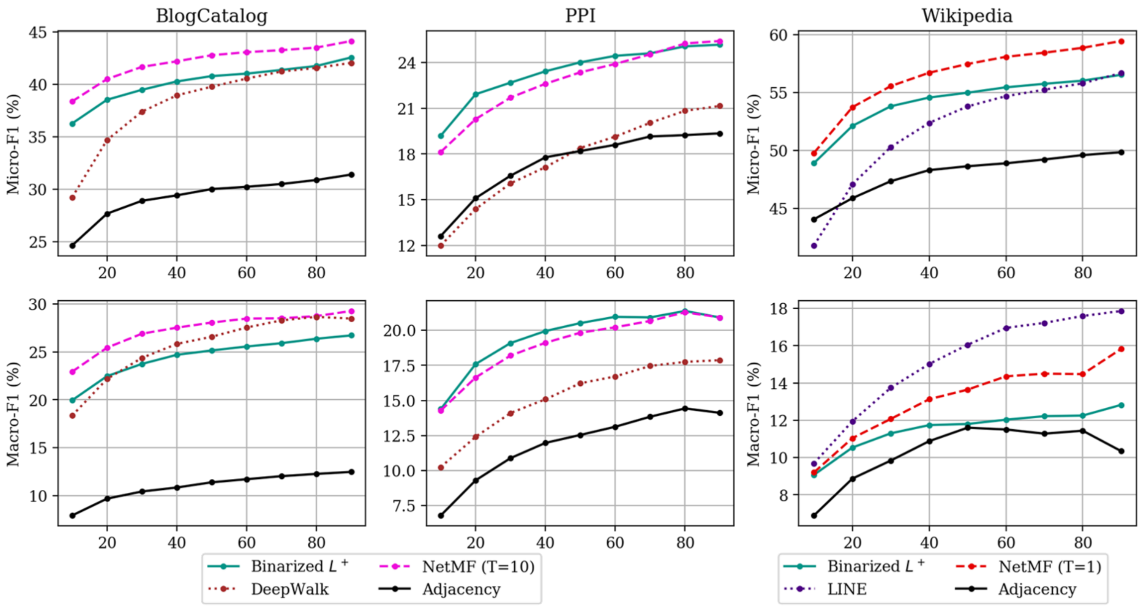


Figure 2.4. Performance of the binarized Laplacian pseudoinverse method relative to NetMF, sampling-based methods, and embedding by factorizing the adjacency matrix. 0.95 is used as the thresholding quantile for BlogCatalog and PPI, and 0.50 is used for Wikipedia. The more suitable setting of T and the more suitable sampling method is plotted for each network.

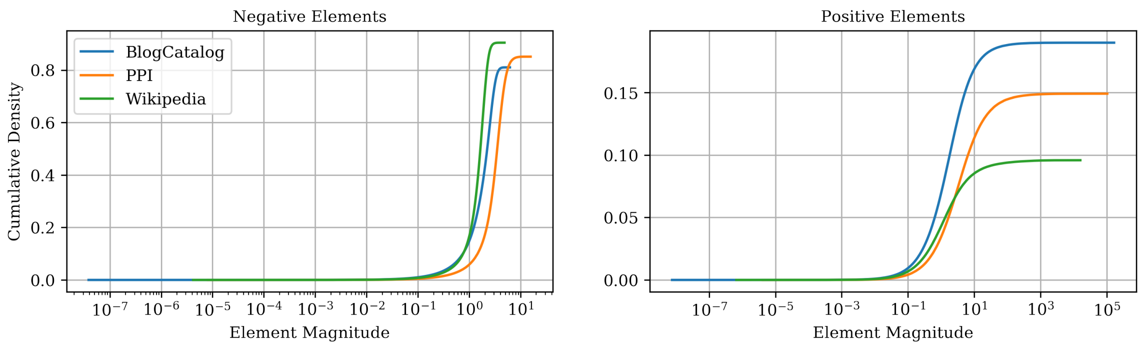


Figure 2.5. Distribution of elements of the limiting PMI matrices $\mathbf{M}_\infty$ of the three networks. The distributions are separated between negative and positive elements corresponding to negative and positive correlations between nodes.

2.2 From Embeddings Back to Graphs

In this section, we focus on the following high-level question:

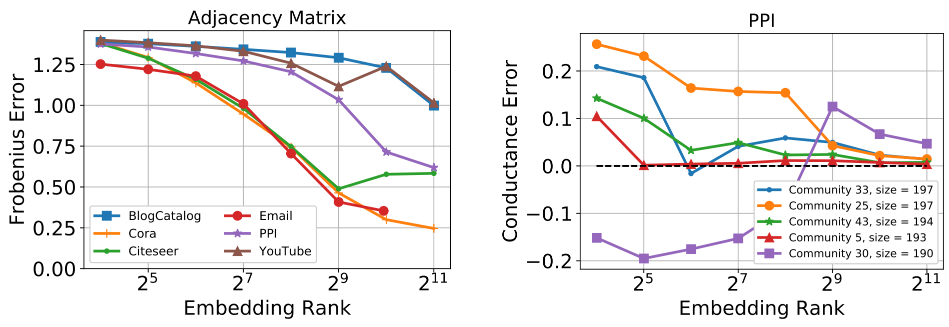
> What graph properties are encoded in and can be recovered from node embeddings? How do these properties correlate with learning tasks?

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.

Problem 1 (Embedding Inversion)**.** *Given an embedding algorithm $\mathcal{E} : \mathcal{G} \to \mathbb{R}^{n\times k}$ 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 $\left\|\mathcal{E}(\tilde{G}) - \mathcal{E}(G)\right\|$ 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 $\left\|\mathcal{E}(\tilde{G}) - \mathcal{E}(G)\right\|$ 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?*



(a) (b)

Figure 2.6. (a) Relative Frobenius error $\left\| \boldsymbol{A} - \tilde{\boldsymbol{A}} \right\|_F / \left\| \boldsymbol{A} \right\|_F$ 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.

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 [144]. As discussed in Section 2.1, DeepWalk embedding can be interpreted as low-rank approximation of a pointwise mutual information (PMI) matrix based on node co-occurrences in random walks [71]. The NetMF method [147] 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 $\left\|\mathcal{E}(G) - \mathcal{E}(\tilde{G})\right\|_F$, 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 2.6(a) illustrates this finding for a popular benchmark of datasets (see Section 2.2.3 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

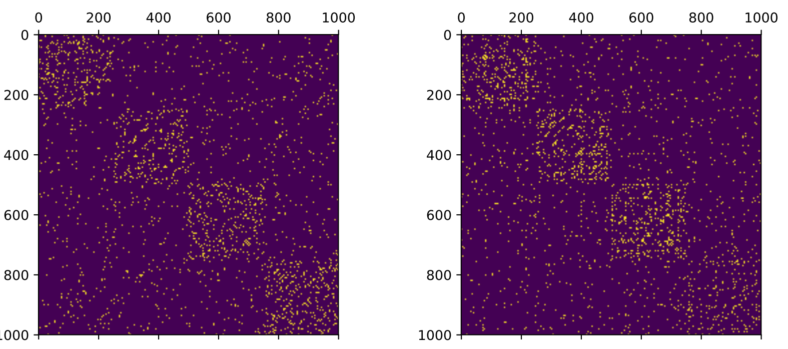


Figure 2.7. G (left), a stochastic block model graph with 1000 nodes and 4 clusters, and $\tilde{G}$ (right), a reconstruction of G from a 32-dimensional NetMF embedding. While G and $\tilde{G}$ differ in the exact edges they contain, we can see that the community structure is preserved.

the embedding method. That is, the conductance of the same community in $\tilde{G}$ is even lower than in G.

Figure 2.6(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 2.7 provides another visual summary of the above findings. Specifically, it shows on the left the spy plot of a stochastic block model graph with 1 000 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.

2.2.1 Related work

Graph recovery from embeddings. To the best of our knowledge, Problem 1 has not been studied explicitly in prior work. [85] 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 [174]. Close to our work lies recent work on node embedding privacy, and in particular graph reconstruction attacks on these embeddings. [54] identify neighbors of a given node v with good accuracy by considering the change in embeddings of the other nodes in G and $G \setminus v$. [51] 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 [122]. 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. In Section 3.1, we show 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 [167].

DeepWalk and NetMF. We focus on inverting embeddings produced by the NetMF variant [147] of the popular DeepWalk method [144]. Consider an undirected, connected, non-bipartite graph G, with adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n\times n}$, diagonal degree matrix $\boldsymbol{D} \in \mathbb{R}^{n\times n}$ and volume $v_G = \operatorname{tr}(\boldsymbol{D}) = \sum_{i,j} \boldsymbol{A}_{ij}$. Qui 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*:

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

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 $\sum_{r=1}^{T}(\boldsymbol{D}^{-1}\boldsymbol{A})^r\boldsymbol{D}^{-1}$ will be 0. To avoid taking the logarithm of 0, NetMF instead employs the *positive pointwise mutual information (PPMI)* matrix:

$$\boldsymbol{M}_T = \log\left(\max\left(1, \frac{v_G}{T}\sum_{r=1}^{T}(\boldsymbol{D}^{-1}\boldsymbol{A})^r\boldsymbol{D}^{-1}\right)\right). \tag{2.7}$$

Via truncated eigendecomposition of $\boldsymbol{M}_T$, one can find an eigenvector matrix $\boldsymbol{V} \in \mathbb{R}^{n\times k}$ and a diagonal eigenvalue matrix $\boldsymbol{W} \in \mathbb{R}^{k\times k}$ such that $\boldsymbol{M}_{T,k} = \boldsymbol{V}\boldsymbol{W}\boldsymbol{V}^\top$ is the best possible k-rank approximation of $\boldsymbol{M}_T$ in the Frobenius norm. The NetMF embedding is set to the eigenvectors scaled by the square roots of the eigenvalue magnitudes. I.e., $\mathcal{E}(G) = \boldsymbol{V}\sqrt{|\boldsymbol{W}|}$, 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.

2.2.2 Proposed methods

In Sections 2.2.2.1 and 2.2.2.2 we present our two proposed NetMF embedding inversion methods. The first is inspired by our constructive proof of Theorem 2.2.1 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 $\boldsymbol{M}_{T,k} = \boldsymbol{V}\boldsymbol{W}\boldsymbol{V}^T$ to the positive pointwise mutual information (PPMI) matrix $\boldsymbol{M}_T$, 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 ($\boldsymbol{A} \in \{0,1\}^{n\times n}$), our inversion algorithms produce $\tilde{G}$ with $\tilde{\boldsymbol{A}} \in [0,1]^{n\times n}$. 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 $\tilde{\boldsymbol{A}}_b \in \{0,1\}^{n\times n}$. We describe the binarization process in detail in the following sections.

2.2.2.1 Analytical Approach

We leverage our asymptotic result from Section 2.1, which states that as the number of samples and the window size T for DeepWalk/NetMF tend to infinity, the PMI matrix tends to the limit:

$$\lim_{T\to\infty} T \cdot \hat{\boldsymbol{M}}_T = \hat{\boldsymbol{M}}_\infty = v_G \cdot \boldsymbol{D}^{-1/2}(\bar{\boldsymbol{L}}^+ - \boldsymbol{I})\boldsymbol{D}^{-1/2} + \boldsymbol{J}, \tag{2.8}$$

where $\bar{\boldsymbol{L}} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ is the normalized Laplacian, $\bar{\boldsymbol{L}}^+$ is the Moore-Penrose pseudoinverse of this matrix, and $\boldsymbol{J}$ is the all-ones matrix. Our first observation is that if, in addition to $\hat{\boldsymbol{M}}_\infty$, we are given the degrees of the vertices in G, then we know both D and v_G, and we can simply invert Equation 2.8 as follows:

$$\begin{aligned} \bar{\boldsymbol{L}} &= \left(\boldsymbol{D}^{1/2}\left(\frac{\hat{\boldsymbol{M}}_\infty - \boldsymbol{J}}{v_G}\right)\boldsymbol{D}^{1/2} + \boldsymbol{I}\right)^+ \\ \boldsymbol{A} &= \boldsymbol{D}^{1/2}\left(\boldsymbol{I} - \bar{\boldsymbol{L}}\right)\boldsymbol{D}^{1/2}. \end{aligned} \tag{2.9}$$

Recovery of Degrees from Limiting PMI. We now show that using just the graph volume v_G, one can perfectly recover the degree matrix $\boldsymbol{D}$ from $\hat{\boldsymbol{M}}_\infty$ via a linear system, provided the graph's adjacency matrix is full-rank. For an undirected graph G with adjacency matrix $\boldsymbol{A}$ and unnormalized Laplacian $\boldsymbol{L}$, 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 entrywise square root. Note that

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

since the all-ones vector $\mathbf{1}$ is in the null space of the unnormalized Laplacian $\boldsymbol{L}$. Suppose we have the limiting PMI matrix $\boldsymbol{M}_\infty$ and the graph volume v_G. We subtract the all-ones matrix $\boldsymbol{J}$ from $\boldsymbol{M}_\infty$ and multiply by $\boldsymbol{d}/v_G$:

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

Thus, if we solve the linear system $(\boldsymbol{M}_\infty - \boldsymbol{J})\boldsymbol{x} = -\mathbf{1}$ 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 $(\boldsymbol{M}_\infty - \boldsymbol{J})$ is singular and the linear system does not have a unique solution. $(\boldsymbol{M}_\infty - J)$ is singular iff $(\bar{\boldsymbol{L}}^+ - \boldsymbol{I})$ is singular, and this only occurs when $\bar{\boldsymbol{L}}^+$ and hence $\bar{\boldsymbol{L}}$ has an eigenvalue equal to 1. $\bar{\boldsymbol{L}} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$, so this requires that $\boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ has a zero eigenvalue. Thus, $\bar{\boldsymbol{L}}^+ - \boldsymbol{I}$ is singular exactly when $\boldsymbol{A}$ is singular.

Combining this fact with Equations 2.8 and 2.9 we obtain the following:

Theorem 2.2.1 (Limiting Invertibility of Full-Rank PMI Embeddings)**.** *Let G be an undirected, connected, non-bipartite graph with full-rank adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n\times n}$ and volume v_G. Let $\hat{\boldsymbol{M}}_T$ be the PMI matrix of G which is produced with window size T. There exists an algorithm that takes only $\hat{\boldsymbol{M}}_T$ and v_G as input and recovers $\boldsymbol{A}$ exactly in the limit as $T \to \infty$.*

Approximation. In our embedding inversion task, rather than the exact limiting PMI matrix $\hat{\boldsymbol{M}}_\infty$, we are given the low-rank approximation $\boldsymbol{M}_{T,k}$ of the finite-T PPMI matrix, through the NetMF embeddings. Our first algorithm is based on essentially ignoring this difference. We use $\boldsymbol{M}_{T,k}$ to obtain an approximation to $\hat{\boldsymbol{M}}_\infty$, which we then plug into Equation 2.9. This approximation is based on inverting the following limit from Section 2.1:

$$\lim_{T\to\infty} \hat{\boldsymbol{M}}_T = \log\left(\tfrac{1}{T}\hat{\boldsymbol{M}}_\infty + \boldsymbol{J}\right), \tag{2.10}$$

where the logarithm is applied entrywise.

Due to the various approximations used, the elements of the reconstructed adjacency matrix $\tilde{\boldsymbol{A}}$ may not be in $\{0, 1\}$, and may not even be in $[0, 1]$; for this reason, as in [167], we apply an entrywise clipping function, $\text{clip}(x) = \min(\max(0, x), 1)$, after the inversion steps from Equations 2.9 and 2.10. The overall procedure is given in Algorithm 2.

Algorithm 2 DeepWalking Backwards (Analytical)

```
input approximation $\boldsymbol{M}_{T,k}$ of true $T$-step PPMI, window-size $T$, degree matrix $\boldsymbol{D}$, graph volume $v_G$
output reconstructed adjacency matrix $\tilde{\boldsymbol{A}} \in [0,1]^{n\times n}$
1: $\tilde{\boldsymbol{M}}_\infty \leftarrow T \cdot (\exp(\boldsymbol{M}_{T,k}) - J)$  ▷ exp is applied entrywise, $\boldsymbol{J}$ is the all-ones matrix
2: $\tilde{\tilde{\boldsymbol{L}}} \leftarrow \left(\boldsymbol{D}^{1/2}\left(\frac{\tilde{\boldsymbol{M}}_\infty - \boldsymbol{J}}{v_G}\right)\boldsymbol{D}^{1/2} + \boldsymbol{I}\right)^+$
3: $\tilde{\boldsymbol{A}} \leftarrow \text{clip}\left(\boldsymbol{D}^{1/2}\left(\boldsymbol{I} - \tilde{\tilde{\boldsymbol{L}}}\right)\boldsymbol{D}^{1/2}\right)$
4: return $\tilde{\boldsymbol{A}}$
```

Binarization. To produce a binary adjacency matrix $\tilde{\boldsymbol{A}}_b \in \{0, 1\}^{n\times n}$ from $\tilde{\boldsymbol{A}}$, we use a slight modification of Algorithm 2: 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.

2.2.2.2 Optimization Approach

Our gradient based approach parameterizes the entries of a real valued adjacency matrix $\tilde{\boldsymbol{A}} \in (0, 1)^{n\times n}$ with independent logits for each potential edge, and leverages the differentiability of Equation 2.7. Based on $\tilde{\boldsymbol{A}}$, we compute the PPMI matrix

$\tilde{\boldsymbol{M}}_T$, and then the squared PPMI error loss, i.e., the squared Frobenius error between $\tilde{\boldsymbol{M}}_T$ and the low-rank approximation $\boldsymbol{M}_{T,k}$ of the true PPMI, given by the NetMF embeddings. We differentiate through these steps, update the logits, and repeat. Pseudocode is given in Algorithm 3.

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 $\tilde{\boldsymbol{M}}_T$ 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 4. 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 [140] for automatic differentiation and minimizes the loss using the SciPy [89] implementation of L-BFGS [115, 210] with default hyperparameters and a maximum of 500 iterations.

Algorithm 3 DeepWalking Backwards (Optimization)

```
input approximation $\boldsymbol{M}_{T,k}$ of true $T$-step PPMI, window-size $T$, graph volume $v_G$, number of iters. $N$
output reconstructed adjacency matrix $\tilde{\boldsymbol{A}} \in (0,1)^{n \times n}$
1: Initialize elements of $\boldsymbol{X} \in \mathbb{R}^{(n \times n)}$ to 0    ▷ logits of the reconstructed adj. matrix
2: for $i \leftarrow 1$ to $N$ do
3:     $\tilde{\boldsymbol{A}} \leftarrow \sigma_{v_G}(\boldsymbol{X})$    ▷ construct adj. matrix with target volume, see Algorithm 4
4:     $\tilde{\boldsymbol{M}}_T \leftarrow \mathrm{PPMI}\left(\tilde{\boldsymbol{A}}\right)$ via Eq. 2.7
5:     $L \leftarrow \|\tilde{\boldsymbol{M}}_T - \boldsymbol{M}_{T,k}\|_F^2$    ▷ squared error of PPMI
6:     Calculate $\partial_{\boldsymbol{X}} L$ via automatic differentiation through Steps 3 to 5
7:     Update $X$ to minimize $L$ using $\partial_{\boldsymbol{X}} L$
8: return $\sigma_v(\boldsymbol{X})$
```

Algorithm 4 Shifted Logistic Function σ_v

```
input logit matrix $\boldsymbol{X} \in \mathbb{R}^{(n\times n)}$, target sum $v \in (0, n^2)$, number of iterations $I$
output matrix $\boldsymbol{A} \in (0,1)^{n\times n}$ which sums approximately to $v$
1: $s \leftarrow 0$
2: for $i \leftarrow 1$ to $I$ do
3:     $\boldsymbol{A} \leftarrow \sigma(\boldsymbol{X} + s)$        ▷ $\sigma$ is the logistic function applied entrywise
4:     $s \leftarrow s + \frac{v - \Sigma(\boldsymbol{A})}{\Sigma(\boldsymbol{A} \circ (\boldsymbol{J} - \boldsymbol{A}))}$        ▷ $\Sigma$ sums over all elements, $\circ$ is an entrywise product
5: return $\sigma(\boldsymbol{X} + s)$
```

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

2.2.3 Experimental results

We now detail the setup and results for applying our inversion algorithms to real-world and synthetic graphs.

2.2.3.1 Experimental setup

Datasets. We apply the NetMF inversion algorithms described in Section 2.2.2 to a benchmark of networks, summarized in Table 2.3. 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 [147] for BlogCatalog and PPI, [166] for Citeseer and Cora, and SNAP [106] for Email and Youtube. The YouTube graph we use is a sample of 20 communities from the raw network of [106]. For all networks, we consider only the largest connected component. The community labels that we report for various datasets, such as those

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 2.3. Network statistics for experiments in Section 2.2.

reported in the legends of Figure 2.10, refer to the labels as given in the input datasets.

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 2.2.2 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, $\boldsymbol{M}_{T,k}$ and $\tilde{\boldsymbol{M}}_{T,k}$ respectively. These matrices represent the NetMF embeddings of G and $\tilde{G}$. The relative Frobenius error for two matrices X and $\tilde{\boldsymbol{X}}$ is simply $\left\|\boldsymbol{X} - \tilde{\boldsymbol{X}}\right\|_F / \left\|\boldsymbol{X}\right\|_F$.

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 $\tilde{\boldsymbol{A}}_b$. See Sections 2.2.2.1 and 2.2.2.2 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 $\boldsymbol{A}$ and $\tilde{G}$'s adjacency matrix $\tilde{\boldsymbol{A}}_b$. 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 for this section 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. 3), significantly outperforms the analytical approach (Alg. 2), 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 2.8.

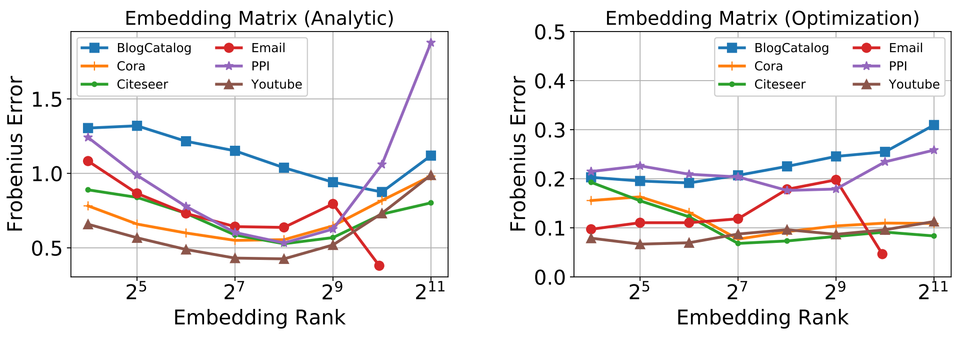


Figure 2.8. Relative Frobenius error vs. embedding rank k for the low-rank PPMI matrices of the graphs reconstructed using the inversion algorithms: the analytical approach, Alg. 2 (left), and the optimization approach, Alg. 3 (right). For details, see Section 2.2.3.2.

- Focusing on $\tilde{G}$ produced by Algorithm 3, 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 2.9.

- $\tilde{G}$ 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 2.9.

- $\tilde{G}$'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.

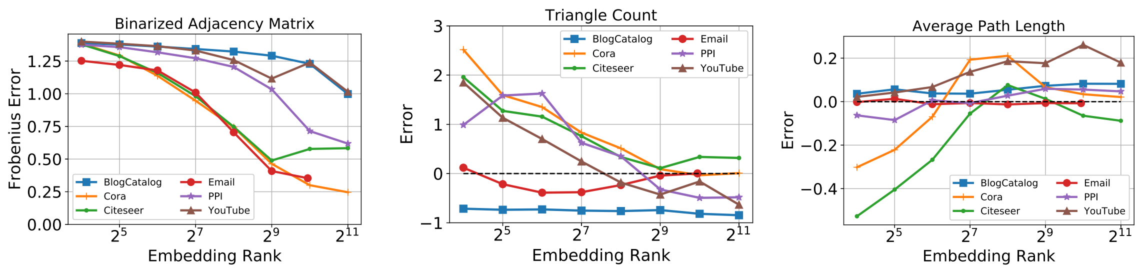


Figure 2.9. From left to right: Relative Frobenius error for the binarized adjacency matrix; relative error for the number of triangles; and relative error for the average path length. All plots are versus the embedding rank.

2.2.3.2 Analytical vs. Optimization Based Inversion

Figure 2.8 reports the relative Frobenius error of the analytical method (Alg. 2) and the optimization approach (Alg. 3) in embedding inversion as we range k. We can see that Alg. 3 *significantly* outperforms Alg. 2. While Alg. 2 comes with strong theoretical guarantees (Theorem 2.2.1) 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. 2 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 2.8 that the embedding error stays low across different values of k when using Alg. 3, indicating that performance is insensitive to the dimension parameter.

2.2.3.3 Evaluating Graph Recovery

Adjacency matrix reconstruction. We next examine how closely the output of Alg. 3, the binarized adjacency matrix $\tilde{\boldsymbol{A}}_b$, matches the original adjacency matrix $\boldsymbol{A}$, especially as we vary the embedding dimensionality k. As can be seen in Figure 2.9, at low ranks, the relative Frobenius error is often quite high – near 1. In combination with Figure 2.8 (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 2.2.1 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 2.9 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 constrast, for another important global property, the average path length, the reconstruction error is always relatively low (also shown in Figure 2.9).

In Figure 2.10, 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

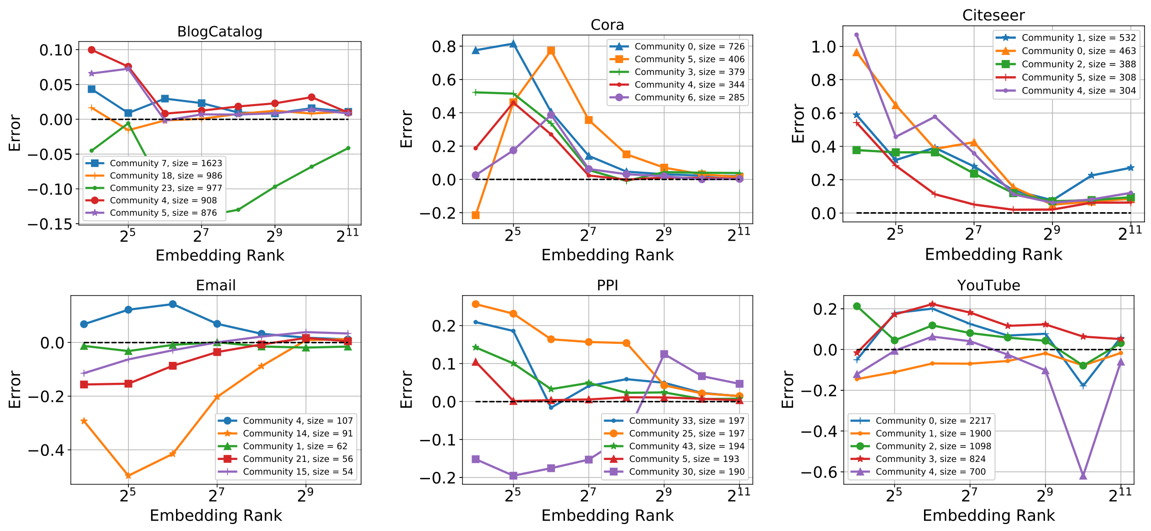


Figure 2.10. Relative error for the conductances of the five largest communities for each of the selected networks.

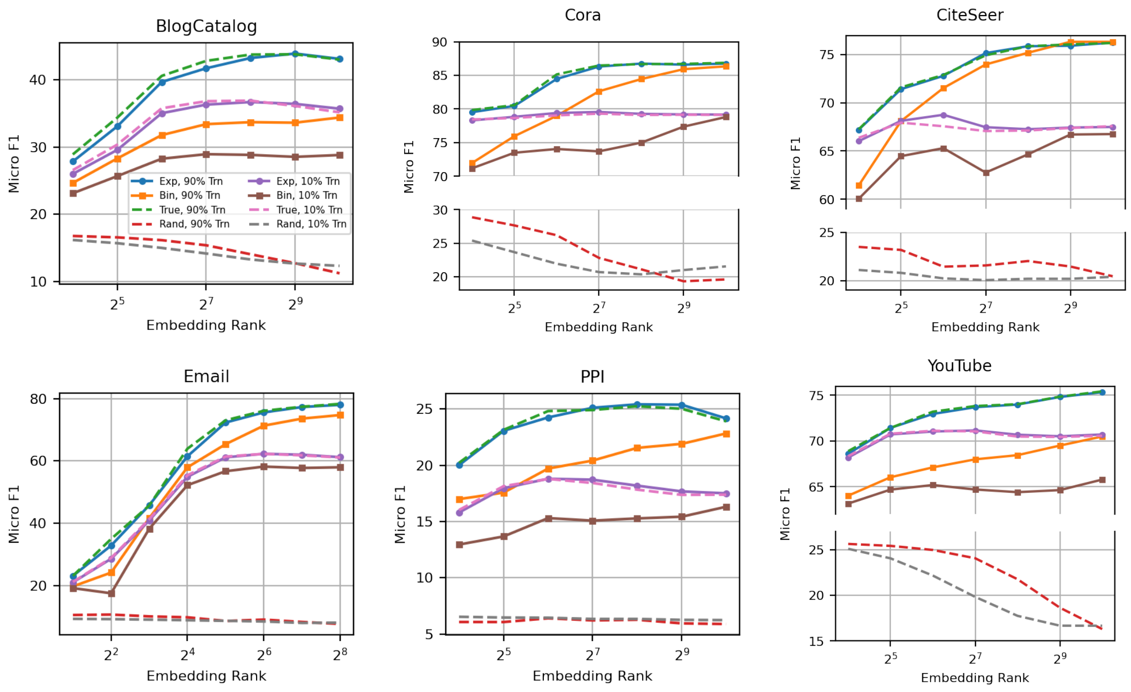


Figure 2.11. Multi-label classification using embeddings from reconstructed networks. Performance when using embeddings from a random graph is included as a baseline.

are used for node classification and community detection, it is to be expected that they preserve community 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 $\mathcal{E}(\tilde{G})$ 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 2.2.2.2.

Our classification setting is the same as that of [147]: 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 2.11, 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 2.8 (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.

Name	**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

Table 2.4. Configuration of SBM networks; all networks have 1000 nodes.

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) [2]. 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 2.4. All networks have 1000 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 and the binarized adjacency matrix (Figure 2.12); the error of triangles count, and average path length (Figure 2.13); the error of the conductances of the top communities (Figure 2.14); and the node classification performance using embeddings made from the reconstructed networks (Figure 2.15). For the node classification task, each node is a member of a single ground-truth community which corresponds to its cluster in the SBM.

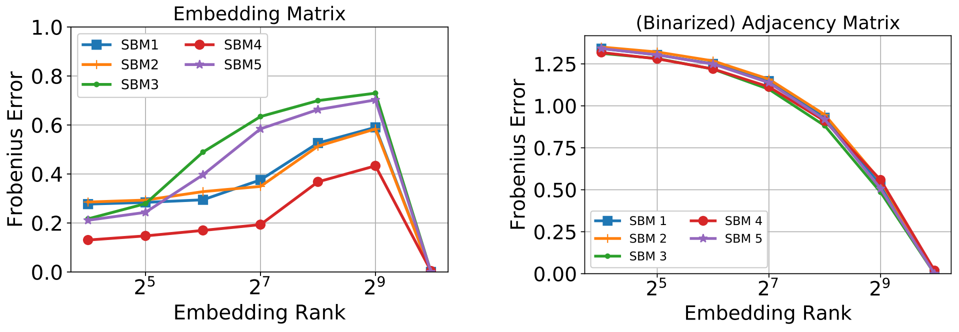


Figure 2.12. Relative Frobenius error for the low-rank PPMI matrices of reconstructions of the synthetic SBM networks (left) and the binarized adjacency matrix (right).

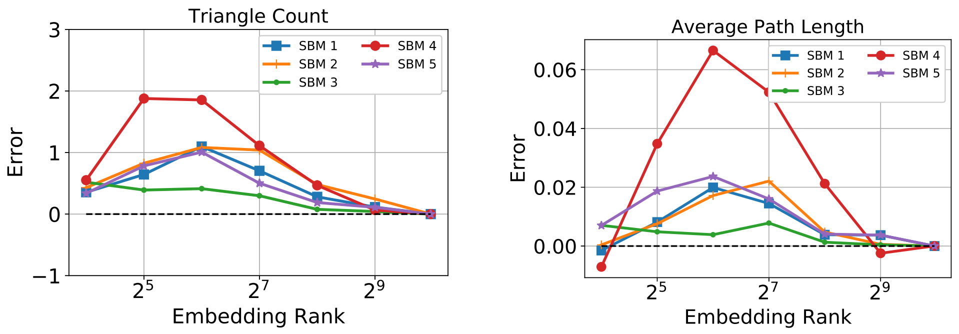


Figure 2.13. Graph reconstruction errors for synthetic SBM networks. Relative error for the number of triangles (left) and for the average path length (right).

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 2.7 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 2.2.1 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.

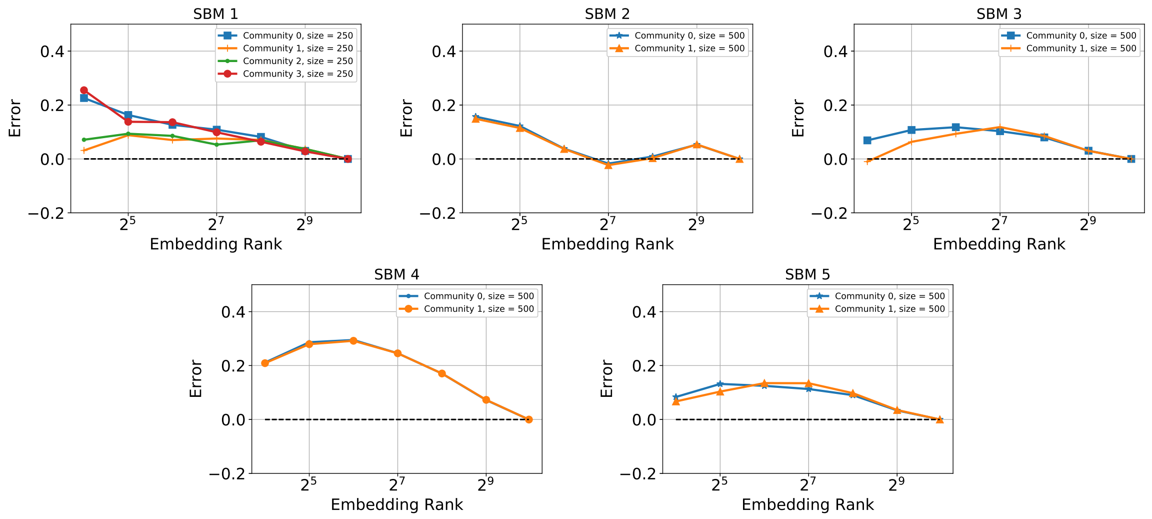


Figure 2.14. Relative error for the conductances of the five most populous communities for each synthetic SBM network.

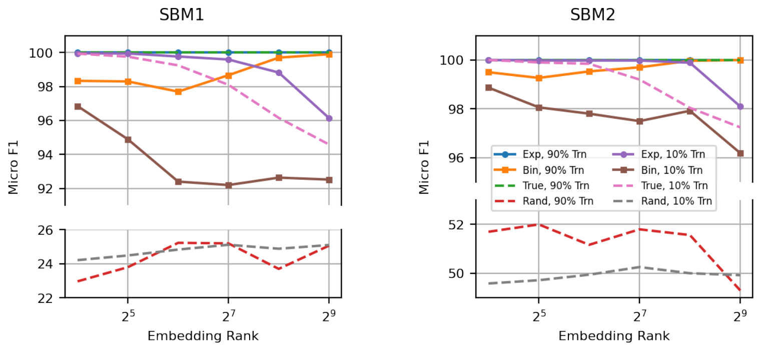


Figure 2.15. Multi-label classification using embeddings from reconstructions of two of the synthetic SBM networks.

2.2.4 Conclusion

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 this framework can be extended beyond the NetMF method, and whether we can formalize these empirical findings mathematically.

CHAPTER 3
POWER AND LIMITATIONS OF EMBEDDINGS

In this chapter, we examine upper and lower bounds on the capabilities of node embeddings at representing graph structure.

3.1 Exact Low-Rank Representations of Complex Networks

This recent explosion of novel node embedding methods, which has been discussed previously in this thesis, has already proved valuable for numerous graph mining tasks. But what are the limitations of these methods?

This question was recently posed by Seshadhri, Sharma, Stolman, and Goel in [167]. Seshadhri et al. remark that (i) regardless of the node embedding method, the goal is to produce a low-dimensional embedding that captures as much structure in G as possible, and that (ii) it is well-known that real-world networks are sparse in edges, and rich in triangles. They ask the following intriguing question: can low-dimensional node embeddings represent triangle-rich complex networks? Their key conclusion is that graphs generated from low-dimensional embeddings cannot contain many triangles on low-degree vertices, and thus the answer to the aforementioned question is negative. See Theorem 3.1.4 in Section 3.1.1 for a formal statement of this result.

In this section, we prove that the results in [167] are a consequence of the model they use, rather than a general property of low-dimensional embeddings. We state our contributions as informal results; for the formal statements, see Section 3.1.2. Our first main result is:

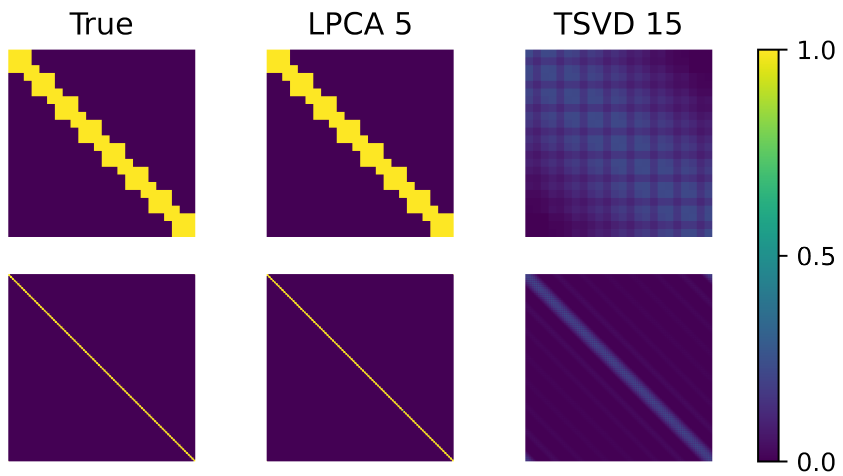


Figure 3.1. Reconstructions of a toy graph with 100 triangles connected in a loop and a self-loops on each vertex. Top: Zoomed in to the first 24 vertices, i.e., the first 8 triangles; Bottom: Whole graph. Left: True adjacency matrix; Middle: rank-5 approximation produced by our logistic PCA variant (LPCA); Right: rank-15 approximation with truncated SVD (TSVD) method [167].

Result 3.1.1. Low-dimensional node embeddings are able to represent triangle-rich graphs.

Figure 3.1 gives an illustrative example of Result 3.1.1. Consider the family of graphs consisting of a set of triangles connected in a cycle. This family is a hard instance according to result of [167] since it has near maximum triangle density given its low maximum degree. Indeed, we can observe that an optimal 15-dimensional representation using the proposed method of [167] preserves very little structure in the graph. However, as Figure 3.1 shows, there exists a rank-5 representation which nearly fully captures the graph structure. We discuss the details in Section 3.1.3.

Our second key result is a low-dimensional model that can perfectly capture *all bounded degree graphs*, regardless of structure. An important corollary of this result is that preferential attachment graphs [16] admit a $\Theta(\sqrt{n})$-rank factorization with high probability without losing any information about their structure. Furthermore, our result is constructive.

Dataset	# Nodes	Mean Degree	**Exact Factorization Dimension**
Pubmed	19 581	4.48	48
ca-HepPh	11 204	21.0	32
BlogCatalog	10 312	64.8	128
Citeseer	3 327	2.74	16
Cora	2 708	3.90	16

Table 3.1. Preview of our results in Section 3.1.3; real-world graphs admit *exact* low-rank factorizations.

Result 3.1.2. There exists a low-rank factorization algorithm that provably produces exact low-dimensional embeddings for bounded degree graphs.

We believe such *exact embeddings* are of independent interest to researchers working in the interplay between privacy and node embeddings, e.g., [54, 209] and on graph autoencoders [185, 192, 139, 162].

We complement our results with several experiments on real-world networks. We observe that a simple algorithm can produce *very low-dimensional* exact representations, that go below the theoretical bounds we prove in Result 3.1.2, and still preserve all local structure. See Table 3.1 for a preview of our results on some popular datasets. We show that even lower dimensional factorizations, while not exact, suffice to capture important structure such as degree and triangle density.

Result 3.1.3. Empirically we observe that our proposed algorithm produces *very low-dimensional* embeddings that preserve the local structure of large real-world networks.

3.1.1 Background: Representations of Triangle-rich Networks

Recently, Seshadhri et al. [167] asked a crucial question: are there any inherent limitations on the ability of low-dimensional embeddings to capture relevant structure in complex networks? They argue that low-dimensional embeddings provably cannot capture important properties of real-world complex networks. In particular, it is well-known that real-world networks are sparse and contain many triangles, see e.g., [57, 105]. They argue that a graph generated from a natural low-dimensional embedding cannot have this property. In particular they consider a truncated dot product model, where each node v_i is associated with an embedding $\boldsymbol{x}_i \in \mathbb{R}^k$ and nodes v_i, v_j connect with probability proportional to the dot product $\boldsymbol{x}_i^\top \boldsymbol{x}_j$, truncated to lie in $[0, 1]$. Formally:

Theorem 3.1.4 (Theorem 1, [167])**.** *Let $\boldsymbol{A} = \sigma(\boldsymbol{X}\boldsymbol{X}^\top)$ where $\boldsymbol{X} \in \mathbb{R}^{n \times k}$ and $\sigma(x) = \max(0, \min(1, x))$ is a thresholding function which is applied entry-wise to $\boldsymbol{X}\boldsymbol{X}^\top$. For any $c \geq 4$ and $\Delta \geq 0$, if a graph G is generated by adding edge (i, j) independently with probability $\boldsymbol{A}_{ij}$ and the expected number of triangles in G that only involve vertices of expected degree $\leq c$ is $\geq \Delta n$, then the embedding dimension $k \geq \min(1, \Delta^4/c^9) \cdot n/\log^2 n$.*

If the triangle density Δ and the maximum degree c are fixed, Theorem 3.1.4 implies that X must have near-linear dimension $k = \mathcal{O}n/\log^2 n)$. That is, no low-dimensional embedding can capture the important feature of high triangle density on low-degree nodes. This result contrasts with the well-known fact that low-rank approximations can be used to approximate global triangle counts [186], showing that counts restricted to a subgraph of bounded degree nodes cannot be preserved.

Seshadhri et al. conjecture that Theorem 3.1.4 generalizes to models where $\boldsymbol{A}_{ij}$ is generated by natural functions of the embeddings $\boldsymbol{x}_i, \boldsymbol{x}_j$ other than the truncated dot product. In the next section we argue that Theorem 3.1.4 is in fact brittle and a consequence of the specific matrix factorization model used.

3.1.2 Theoretical Results

We start by showing the impossibility result of Theorem 3.1.4 depends critically on the fact that each node is associated with just a single embedding $\boldsymbol{x}_i \in \mathbb{R}^k$. This ensures that the low-rank matrix $\boldsymbol{X}\boldsymbol{X}^\top$ is positive semidefinite (PSD), which is key in proving Theorem 3.1.4. Many network embeddings, such as DeepWalk and Node2Vec produce two embeddings $\boldsymbol{x}_i, \boldsymbol{y}_i \in \mathbb{R}^k$ for each node – sometimes called "word" and "context" embeddings due to their use in the word embedding literature [126]. This leads to a factorization of the form $\boldsymbol{X}\boldsymbol{Y}^\top$ for $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n \times k}$ which is not necessarily PSD. Further, as discussed in [167], other methods [83] base connection probability on the Euclidean distance between k-dimensional points. The underlying squared Euclidean distance matrix $\boldsymbol{D} \in \mathbb{R}^{n \times n}$ is known to be exactly factorized as $\boldsymbol{D} = \boldsymbol{X}\boldsymbol{Y}^\top$ for $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n \times (k+2)}$.

We show that this simple relaxation to allow for a non-PSD factorization allows extremely low-dimensional embeddings to capture sparse, triangle dense graphs.

Theorem 3.1.5 (Low-Dimensional Embeddings Capture Triangles)**.** *Let $\boldsymbol{A} = \sigma(\boldsymbol{X}\boldsymbol{Y}^\top)$ where $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n \times 3}$ and $\sigma(x) = \max(0, \min(1, x))$ is applied entrywise to $\boldsymbol{X}\boldsymbol{Y}^\top$. For any integer $c > 0$, there exist $\boldsymbol{X}, \boldsymbol{Y}$ such that if a graph G is generated by adding edge (i, j) independently with probability $\boldsymbol{A}_{ij}$ then G has maximum degree $< c$ and G contains $\Omega(c^2 n)$ triangles.*

We note that G generated in Theorem 3.1.5 is just a union of $\frac{n}{c}$ c-cliques and in fact has the maximum triangle density possible for a graph with max degree c. G's adjacency matrix $\boldsymbol{A}$ is block diagonal with blocks of size c. This matrix is very far from low-rank and cannot be well approximated by a low-rank factorization $\boldsymbol{X}\boldsymbol{Y}^\top$. However, as we will see, the simple $\sigma(\cdot)$ non-linearity is quite powerful here, allowing an exact factorization of the form $\sigma(\boldsymbol{X}\boldsymbol{Y}^\top)$ for $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n \times 3}$.

Proof of Theorem 3.1.5. We show how to form $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n \times 3}$ such that $\boldsymbol{A} = \sigma(\boldsymbol{X}\boldsymbol{Y}^\top)$ is the adjacency matrix for a union of n/c cliques. Each clique contains $\binom{c}{3}$ triangles. Thus, there are $\frac{n}{c} \cdot \binom{c}{3} = \Omega(c^2 n)$ triangles in the graph, with maximum degree $c - 1$, giving the theorem.

We place n points along a line in $^n/_c$ clusters of c nodes each. All points in a cluster are very close to each other, and clusters are spaced far apart. We then consider a constant matrix minus the squared distance matrix between these points. We can observe that this matrix has rank at most 3: consider $\boldsymbol{x} \in \mathbb{R}^{n\times 1}$ which represents the n positions on the line. Let $\boldsymbol{x_2}$ contain the entry-wise squares of the values in $\boldsymbol{x}$. Let $\boldsymbol{D} = \boldsymbol{x_2}\boldsymbol{1}^\top + \boldsymbol{1}\boldsymbol{x_2}^\top - 2\boldsymbol{x}\boldsymbol{x}^\top$ be the matrix whose entries are the squared Euclidean distances between the points in $\boldsymbol{x}$. Let $\bar{\boldsymbol{D}} = 2\boldsymbol{J} - \boldsymbol{D}$, where $\boldsymbol{J}$ is the all-ones matrix. Note that $\bar{\boldsymbol{D}}$ has rank $\leq \operatorname{rank}(2\boldsymbol{J} - \boldsymbol{x_2}\boldsymbol{1}^\top) + \operatorname{rank}(-\boldsymbol{1}\boldsymbol{x_2}^\top) + \operatorname{rank}(-2\boldsymbol{x}\boldsymbol{x}^\top) = 3$ and so can be written as $\boldsymbol{X}\boldsymbol{Y}^\top$ for $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n\times 3}$. We set $\boldsymbol{A} = \sigma(\boldsymbol{X}\boldsymbol{Y}^\top) = \sigma(\bar{\boldsymbol{D}})$.

Choose the points in $\boldsymbol{u}$ such that the clusters are separated by distance > 2 and within each cluster the c points are arbitrarily close. For i, j in the same cluster, $\boldsymbol{A}_{ij} = \bar{\boldsymbol{D}}_{ij} = 2 - \|\boldsymbol{u}_i - \boldsymbol{u}_j\|_2^2 > 1$ and so we have an edge in G with probability 1. For i, j in different clusters, $\boldsymbol{A}_{ij} = \bar{\boldsymbol{D}}_{ij} = 2 - \|\boldsymbol{u}_i - \boldsymbol{u}_j\|_2^2 \leq 0$, and so they do not have an edge in G. Thus, G consists of a union of n/c disjoint c-cliques. $\square$

Exact Embeddings of Bounded-Degree Graphs. Observe that in the proof of Theorem 3.1.5 we use the thresholded dot product model of [167] in a very restricted way: all entries of $\boldsymbol{X}\boldsymbol{Y}^\top$ are either > 1 or < 0 and thus all large entries are thresholded to 1 in $\sigma(\boldsymbol{X}\boldsymbol{Y}^\top)$ and all small entries to 0. Thus, the same example would hold if we replaced σ with the sign function s with $s(x) = 0$ for $x < 0$ and $s(x) = 1$ otherwise. In other words, our example relies on the fact that the adjacency matrix of G has low *sign-rank*. It can be written as $\boldsymbol{A} = s(\boldsymbol{X}\boldsymbol{Y}^\top)$ for $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n\times 3}$. The sign-rank is widely studied due to its connections to circuit complexity [149, 27], communication complexity [10, 114], and learning theory [12]. It is known via a polynomial interpolation argument [10] that any matrix with sparse rows or columns has low sign-rank, depending linearly on the sparsity. This yields the following theorem, as well as a proof following the approach of of [10]:

Theorem 3.1.6 (Exact Embeddings for Bounded-Degree Graphs)**.** *Let $\boldsymbol{A} \in \{0,1\}^{n\times n}$ be the adjacency matrix of a graph G with maximum degree c. Then there exist embeddings $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n\times(2c+1)}$ such that $\boldsymbol{A} = \sigma(\boldsymbol{X}\boldsymbol{Y}^\top)$ where $\sigma(x) = \max(0, \min(1, x))$ is applied entry-wise to $\boldsymbol{X}\boldsymbol{Y}^\top$.*

Proof. Let $\boldsymbol{V} \in \mathbb{R}^{n\times 2c+1}$ be the Vandermonde matrix with $\boldsymbol{V}_{t,j} = t^{j-1}$. For any $\boldsymbol{x} \in \mathbb{R}^{2c+1}$, $[\boldsymbol{V}\boldsymbol{x}](t) = \sum_{j=1}^{2c+1} \boldsymbol{x}(j) \cdot t^{j-1}$. That is: $\boldsymbol{V}\boldsymbol{x} \in \mathbb{R}^n$ is a degree $2c$ polynomial evaluated at the integers $t = 1, \ldots, n$.

Let $\boldsymbol{a_i}$ be the i^{th} row of $\boldsymbol{A}$. $\boldsymbol{a_i}$ has at most c nonzeros since G has maximum degree c. We seek to find $\boldsymbol{x_i}$ so that $s(\boldsymbol{V}\boldsymbol{x_i}) = \boldsymbol{a_i}$, and thus, letting $\boldsymbol{X} \in \mathbb{R}^{n\times 2c+1}$ have $\boldsymbol{x_i}$ as its i^{th} row, will have $\boldsymbol{A} = s(\boldsymbol{V}\boldsymbol{X}^\top)$. This yields the theorem since, if we scale $\boldsymbol{V}\boldsymbol{X}^\top$ by a large enough constant (which does not change its rank), all its positive entries will be larger than 1 and thus we will have $\sigma(\boldsymbol{V}\boldsymbol{X}^\top) = A$.

To give $\boldsymbol{x_i}$ with $s(\boldsymbol{V}\boldsymbol{x_i}) = \boldsymbol{a_i}$, we equivalently must find a degree $2c$ polynomial which is positive at all integers t with $\boldsymbol{a_i}(t) = 1$ and negative at all t with $\boldsymbol{a_i}(t) = 0$. Let $t_1, t_2, \ldots, t_c$ denote the indices where $\boldsymbol{a_i}$ is 1. Let $r_{i,L}$ and $r_{i,U}$ be any values with $t_{i-1} < r_{i,L} < t_i$ and $t_i < r_{i,U} < t_{i+1}$. If we chose the polynomial with roots at each $r_{i,L}$ and $r_{i,U}$, it will have $2c$ roots and so degree $2c$. Further, this polynomial will switch signs just at each root $r_{i,L}$ and $r_{i,U}$. We can observe then that the polynomial will have the same sign at $t_1, t_2, \ldots, t_c$ (either positive or negative). Flipping the sign to be positive, we have the result. □

Theorem 3.1.6 stands in sharp contrast to the impossibility result of [167] (Theorem 3.1.4). Not only can low-rank models capture complex network structure, but they can capture the structure of *any bounded-degree graph* with rank depending only on the max degree. We remark that the technique used to prove Theorem 3.1.6 applies also when each row of $\boldsymbol{A}$ is block sparse – with a few contiguous blocks of ones. Considering the union of cliques example in Theorem 3.1.5, if we set the diagonal of $\boldsymbol{A}$ to one, we have a block diagonal matrix – each row has a single contiguous block of c ones. This matrix thus has sign rank at most $2 \cdot 1 + 1 = 3$, giving an alternative proof of Theorem 3.1.5.

An interesting corollary of Theorem 3.1.6 is that even *random graphs* admit exact low-dimensional factorizations if they have bounded degree. For example, preferential attachment graphs [16], which bear certain similarities with real-world networks, are sparse graphs with maximum degree bounded by $\mathcal{O}(\sqrt{n})$ with high probability [21, 60]. We thus have:

Corollary 3.1.7. *A random preferential attachment graph with n nodes generated according to the Barabási-Albert-Bollobás-Riordan [16, 21] model admits an exact $\Theta(\sqrt{n})$ factorization.*

Corollary 3.1.7 applies to numerous other random graph models with power law degree distributions as long as the maximum degree produced is sublinear, e.g., [50, 26, 64].

We can interpret Theorem 3.1.6 and Corollary 3.1.7 in multiple ways: they illustrate the power of low-dimensional models to exactly represent local structure in sparse graphs. At the same time, they show that the goal of finding a low-dimensional embedding to reconstruct a graph may be misleading, since a sufficiently optimized embedding can interpolate and maybe 'over-fit' any bounded degree graph. This emphasizes that obtaining low or even zero approximation error graph embedding may simply be due to capturing the fact that the given graph has low maximum degree.

We will see that in practice, the bound of Theorem 3.1.6 is not tight. Via a simple logistic PCA method, we can construct very low-dimensional exact factorizations of many real-world graphs, even when they have high max degree. The precise description of our algorithm follows in Section 3.1.3.

3.1.3 Empirical Results

We now empirically evaluate the effectiveness of low-dimensional embeddings in capturing graph structure, showing that a simple approach can find exact embeddings that match and in fact out perform our theoretical bounds. Code is available at `https://github.com/schariya/exact-embeddings`.

Datasets. Our evaluations are based on 11 popular real-network datasets, detailed below. Table 3.2 lists and shows some statistics of these datasets. For all networks, we ignore weights (setting non-zero weights to 1) and remove self-loops where applicable.

Protein-Protein Interaction (PPI) [177] is a subgraph of the PPI network for Homo Sapiens. Vertices represent proteins and edges represent interactions between them.

Wikipedia [75] is a co-occurrence network of words from a subset of the Wikipedia dump. Nodes represent words and edges represent co-occurrences within windows of length 2 in the corpus.

BlogCatalog [4] is a social network of bloggers. Edges represent friendships.

Facebook [107] is a subset of the Facebook social network collected from survey participants.

ca-HepPh and **ca-GrQc** [105] are collaboration networks from the "High Energy Physics - Phenomenology" and "General Relativity and Quantum Cosmology" categories of arXiv, respectively. Nodes represent authors, and two authors are connected if they have coauthored a paper.

Pubmed [130] consists of scientific publications from the PubMed database pertaining to diabetes. Nodes are publications, and edges represent citations among them.

p2p-Gnutella04 [105] is a snapshot of the Gnutella peer-to-peer network from August 4, 2002. Nodes are hosts in Gnutella, and directed edges are connections between hosts.

Wiki-Vote [104] represents voting on Wikipedia till January 2008. In particular, nodes are users that either request adminship or vote for/against such a promotion. A directed edge from node i to node j represents that user i voted on user j.

Citeseer [166] represents papers from six scientific categories as nodes and the citations among them as directed edges.

Cora [166] contains machine learning papers. Each node is a publication, and there is a directed edge from node i to node j when paper i cites paper j.

Reconstruction Algorithms. The empirical results of Seshadhri et al. [167] focus on the Truncated SVD (TSVD) algorithm. Let $\boldsymbol{Z} \in \mathbb{R}^{n\times k}$ be the orthonormal matrix whose columns comprise the eigenvectors of the adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n\times n}$ corresponding to the k largest magnitude eigenvalues. Let $\boldsymbol{W} \in \mathbb{R}^{k\times k}$ be diagonal, with entries corresponding to the top k eigenvalues. The TSVD embeddings are given by $\boldsymbol{X} = \boldsymbol{Z}\mathrm{s}(\boldsymbol{W})|\boldsymbol{W}|^{1/2}$ and $\boldsymbol{Y} = \boldsymbol{Z}|\boldsymbol{W}|^{1/2}$, where $s(\cdot)$ denotes the sign function and all functions are applied entry-wise to $\boldsymbol{W}$. To form an expected adjacency matrix, we compute $\sigma(\boldsymbol{X}\boldsymbol{Y}^\top)$ where $\sigma(x) = \max(0, \min(1, x))$ is applied element-wise.

Note that $\boldsymbol{X}\boldsymbol{Y}^\top$ produced by TSVD is the rank-k matrix that is closest to $\boldsymbol{A}$ in terms of Frobenius norm. That is, it would be an optimal low-rank factorization *if the threshold $\sigma(\cdot)$ were not applied.* As discussed in Section 3.1.2, many natural adjacency matrices, especially triangle dense ones such as the example of Theorem 3.1.5, are very far from low-rank and thus $\boldsymbol{X}\boldsymbol{Y}^\top$ does not well approximate $\boldsymbol{A}$ either both before and after the threshold.

This motivates our proposed embedding method, which is based on Logistic PCA (LPCA). Rather than minimizing the error between $\boldsymbol{XY}^\top$ and $\boldsymbol{A}$, we attempt to directly minimize the error between $\sigma(\boldsymbol{XY}^\top)$ and $\boldsymbol{A}$. For efficiency, we replace σ with a natural smooth surrogate: the logistic function (the sigmoid). Specifically, given $\boldsymbol{A} \in \{0,1\}^{n\times n}$ and embeddings $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n\times k}$, we let $\tilde{A} = 2\boldsymbol{A} - 1$ denote the shifted adjacency matrix with -1's in place of 0's and use the loss function:

$$L = \sum_{i=1}^{n}\sum_{j=1}^{n} -\log \ell\left(\tilde{\boldsymbol{A}}_{ij}[\boldsymbol{XY}^\top]_{ij}\right), \tag{3.1}$$

where $\ell(x) = (1+e^{-x})^{-1}$ is the logistic function.We initialize elements of the factors $\boldsymbol{X}, \boldsymbol{Y}$ independently and uniformly at random on $[-1, +1]$. We find factors that approximately minimize the loss using the SciPy [89] implementation of the L-BFGS [115, 210] algorithm with default hyper-parameters and up to a maximum of 2000 iterations. We check for exact factorization by comparing $\boldsymbol{A}$ to $\sigma(\boldsymbol{XY}^\top)$. If these are not equal, the factorization is inexact; in that case, to reconstruct an expected adjacency matrix, we apply the logistic function ℓ entry-wise to $\boldsymbol{XY}^\top$.

Toy Graph. We return to the initial demonstrative example from the start of this section, where we considered the family of graphs consisting of a set of t triangles connected in a cycle (Figure 3.1). This family is interesting as it has near maximum triangle density given its sparsity. It starkly illustrates the difference in the capacities of LPCA and TSVD. With embeddings of rank 5, our LPCA method reconstructs a graph with 100 triangles with only minor errors. By contrast, elements of the reconstruction from TSVD at rank 5 are too small to visualize effectively on the same scale, with a maximum below 0.08; even at rank 15, TSVD struggles to capture this graph, significantly diffusing the mass of the adjacency matrix away from the diagonal. In particular, the relative Frobenius errors of the reconstructions (i.e. the Frobenius norm of the difference between the true and reconstructed expected adjacency matrices, divided by the norm of the true adjacency matrix) are 0.031 and 0.894, respectively; for a direct comparison, with a rank 5 embedding, the error of TSVD is 0.966.

Exact Factorization of Real Networks. In Table 3.2 we report the exact factorization dimension (EFD) for 11 real-world networks, the rank at which LPCA exactly recovers the network within 2000 iterations (i.e., returns $\boldsymbol{X}, \boldsymbol{Y}$ with $\sigma(\boldsymbol{X}\boldsymbol{Y}^\top) = \boldsymbol{A}$.). We only compute factorizations at ranks which are multiples of 16, and thus the EFDs are calculated up to a multiple of 16. The values for EFDs are remarkably low – for 3 of the tested networks we achieve exact factorization even at our minimum attempted rank of 16; for these networks, we attempted rank 8 LPCA, but did not achieve exact factorization within 2000 iterations. Moreover, for the rank that we achieved perfect reconstruction (EFD), we report the relative Frobenius error of the TSVD approach; we observe the error is quite high in all cases. For all networks, the EFD is significantly lower than the upper bound presented in Theorem 3.1.6, of twice the max degree plus one. With the exception of PUBMED, all network are factored exactly at or below ranks that are twice just the 95$^{\text{th}}$ percentile degree; the max degree for PUBMED is 171, so it, too, is factored within the theoretical bound.

As a baseline, we generate for each network a set of random graphs with the same expected degree sequence using the algorithm of [187]. We report the EFD at which three random networks generated can be perfectly reconstructed; we note that for a lower rank than the one reported, it was not possible to reconstruct the networks in any of the runs. In general, results are well concentrated and show that EFD is consistently higher for the random networks. In other words, the embeddings capture structure inherent to real-world networks outside just the degree sequence. Understanding this structure more precisely is an interesting direction for future work. For completeness, we repeat the previous experiment generating Erdős-Rényi random graphs with the same expected number of edges. We observe that reconstructing these

Table 3.2. Real world graphs for which we find exact adjacency matrix factorizations of the form $\boldsymbol{A} = \sigma(\boldsymbol{X}\boldsymbol{Y}^\top)$ where $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n \times k}$ and $\sigma(x) = \max(0, \min(1, x))$ is a thresholding function applied entrywise to $\boldsymbol{X}\boldsymbol{Y}^\top$. EFD is the exact factorization dimension for LPCA. We report the 95^{th} percentile degree as a more robust and informative alternative to the maximum degree. TSVD Error is the relative Frobenius error of TSVD at the EFD for LPCA. The final two columns give the EFDs of the random graphs related to these networks described above.

Dataset	# of Nodes	Mean Deg.	95th% Deg.	**EFD**	TSVD Error	**EFD** (Exp. Degree)	**EFD** (Erdős–Rényi)
Pubmed	19 581	4.48	18	48	0.95	48	32
ca-HepPh	11 204	21.0	90	32	0.63	96	64
p2p-Gnutella04	10 876	3.68	32	32	0.97	32	16
BlogCatalog	10 312	64.8	239	128	0.71	160	128
Wiki-Vote	7 115	14.6	75	48	0.77	80	48
ca-GrQc	5 242	5.53	20	16	0.85	32	32
Wikipedia	4 777	38.7	99	64	0.69	80	80
Facebook	4 039	43.7	153	32	0.66	96	80
PPI	3 890	19.7	72	48	0.81	64	48
Citeseer	3 327	2.74	8	16	0.94	16	16
Cora	2 708	3.90	9	16	0.93	16	16

networks is in fact easier, due to the absence of high degree nodes. This justifies the choice of random networks with the same expected degree sequence as the true networks as a more suitable baseline.

Recovery of Degree and Triangle Count Sequences. We next assess the accuracy of very low-dimensional embeddings with respect to reconstructing fundamental network information, namely the sequence of (i) degrees and (ii) participating triangles per node. See our detailed findings in Figures 3.2 and 3.3. We see that the LPCA based embeddings are able to capture both sequences near exactly, even with rank much smaller than the EFD. As we range the rank, LPCA's reconstruction quality for both sequences is a monotone function of the rank; TSVD's performance is not monotone as can be seen e.g., in the PPI plots in Figures 3.2 and 3.3 for ranks 32 and 128 respectively. Generally, the TSVD method performs significantly worse than LPCA.

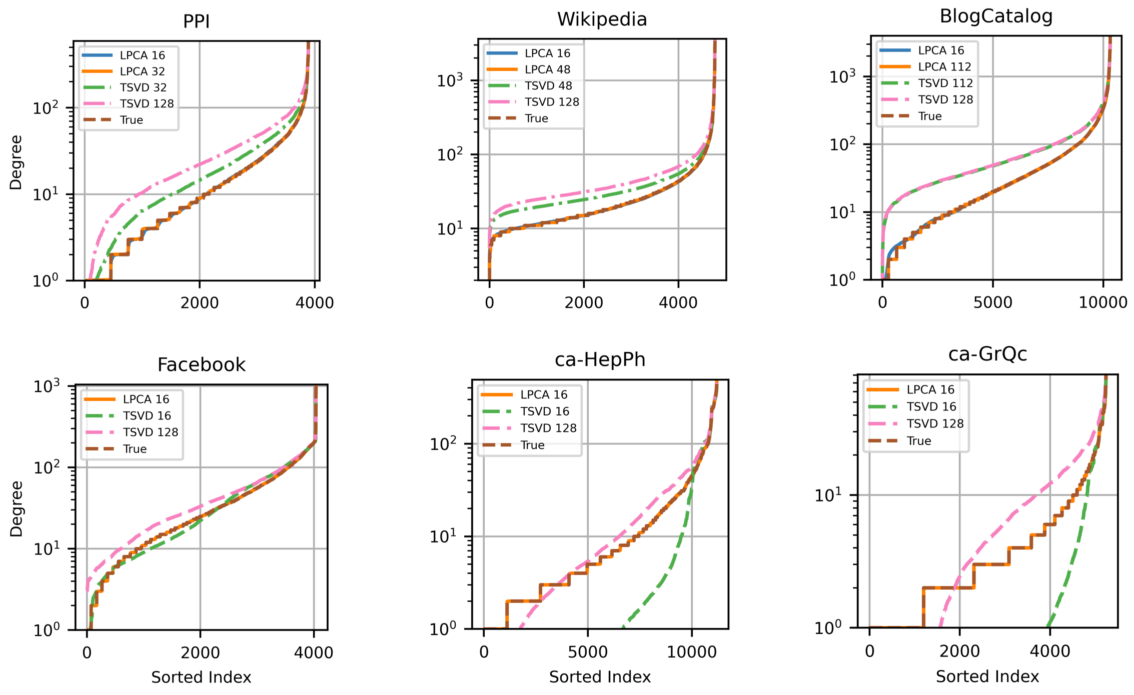


Figure 3.2. Sorted expected degrees of reconstructed networks.

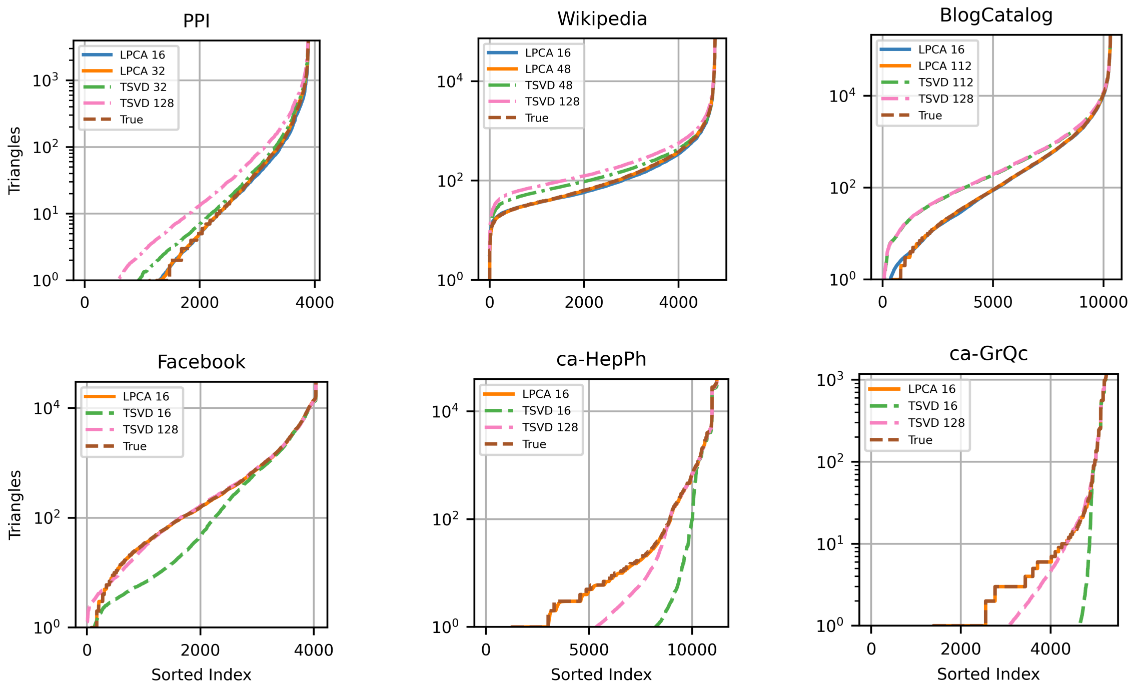


Figure 3.3. Sorted expected count of triangles involving each vertex in reconstructed networks.

Recovery of Low-Degree Triangles. We next turn to the challenge of reconstructing triangles on low-degree nodes, which was the focus of [167]. We assess the recovery of low-degree triangles in real networks when the embedding rank is not sufficient for exact factorization. Our results are shown in Figure 3.4 for six networks. The results are representative of what we observe across all our experiments. In each figure, we plot using different factorization ranks the reconstructed normalized number of triangles (y-axis) among all nodes whose degree does not exceed a specific upper bound (x-axis). We normalize the counts by the number of nodes n to have a consistent measure across all six networks. Notice that the minimum possible non-zero value is $\frac{1}{n}$ and corresponds to exactly one triangle. We plot the performance of LPCA using rank 16, and for TSVD using rank 128. We also plot the performance of both methods for rank equal to the EFD minus 16. Note that, for ca-GrQc, which is reconstructed exactly at our minimum rank of 16, we simply plot rank 16 itself. In addition to the reconstruction results, we also plot the true normalized triangle counts.

In agreement with [167], we find that the TSVD method consistently underestimates low-degree triangles: whereas the true network begins producing triangles with fairly low-degree vertices, TSVD requires much higher-degree vertices to recover a single expected triangle. This holds at both ranks across the surveyed networks. By contrast, across all of these networks, the just-below-exact rank LPCA tightly matches the true triangle-degree curve. With the exception of BlogCatalog, even rank 16 LPCA closely matches the true curve, especially at low degrees. Interestingly, in BlogCatalog, while rank 16 LPCA has a higher reconstruction Frobenius error (.82) than either rank 112 TSVD (.73) or rank 128 TSVD (.71), the former still achieves a single expected triangle with lower-degree vertices and overall matches the true triangle-degree curve more closely than the TSVD methods. This seems to suggest an implicit bias of the LPCA method towards capturing local structure

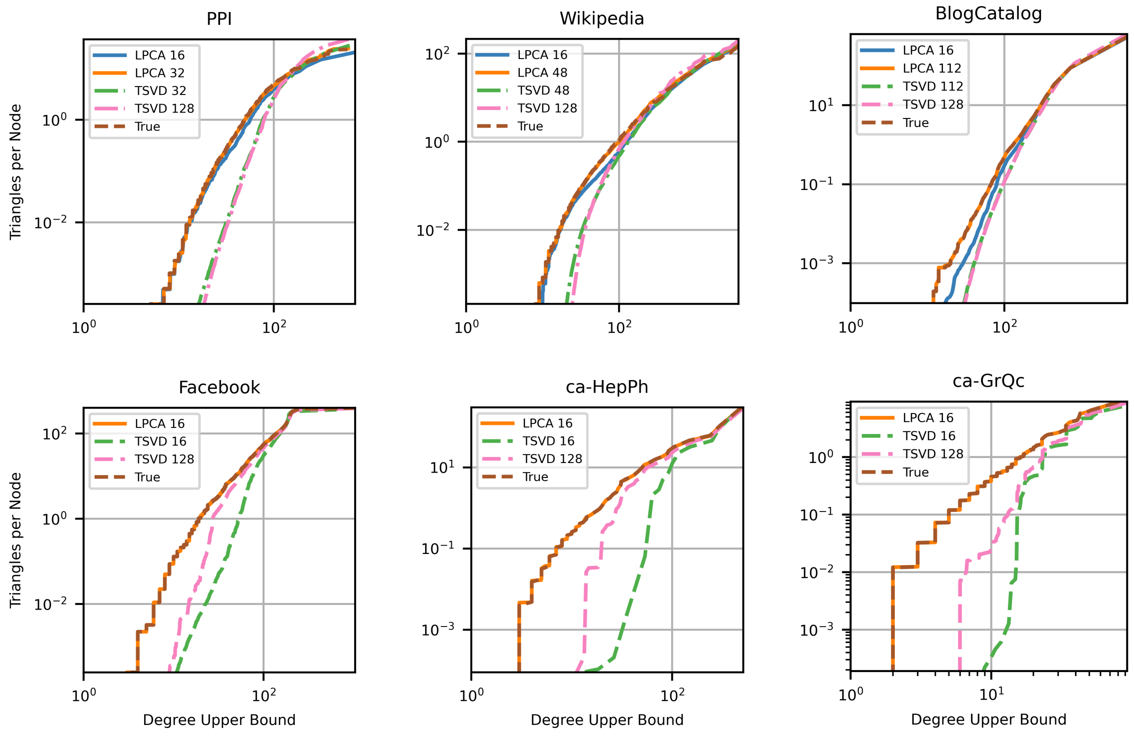


Figure 3.4. True and recovered counts of triangles in subgraphs induced by nodes whose degree is upper bounded by c (y-axis) vs. the degree upper bound c (x-axis) for six networks. The recovered triangles have been counted in reconstructed networks using TSVD and LPCA for different ranks.

in real-world graphs even when factorization is inexact. Understanding this bias more precisely would be an interesting direction for future work. Overall we confirm that LPCA not only outperforms TSVD, but more importantly, illustrates that embeddings can capture the triangle-rich structure of real networks with remarkably accuracy, even at very low ranks where exact factorization is impossible.

3.1.4 Conclusion

In this work we show that low-dimensional node embeddings can capture accurately the graph structure. Specifically, we prove that the results of Seshadhri et al. [167] are intimately connected to the positive semidefinite constraint of the factorizations they consider. While their results are remarkable, and also open an interesting research direction for better understanding node embeddings, our work clarifies that low-rank factorizations *do* capture triangle-rich graphs. Furthermore, we show that a simple algorithm that combines a non-linearity with logistic PCA can recover the entire graph structure from bounded-degree graphs. Our empirical findings indicate that our algorithm is able to produce even lower rank matrix factorizations on real-data by exploiting other structural characteristics; indeed, by using the configuration model we have verified that random graphs with the same degree sequence require a rank close to the theoretical rank bound we provide.

Our work in this section leaves several open questions. A key question is if we can strengthen our theoretical results, and better explain the empirical performance of our algorithm on real-world graphs. Answering this would help us understand the type of structure that our embeddings, and perhaps modern node embeddings more broadly, leverage to compress complex networks. From a practical perspective, understanding the connection between the ability of an embedding to reconstruct a graph and performance in downstream classification tasks is an important related question, key to work on graph auto-encoders and the privacy of node embeddings. In initial experiments, we find that our LPCA embeddings do not give good performance in downstream classification tasks. Are there embeddings that simultaneously yield exact or near exact factorizations and good performance in downstream applications? We also leave open the question of giving even stronger and more general theoretical results that help explain explain our empirical findings. We return to this question in Section 3.2.

3.2 Nonnegative Symmetric Representations of Sparse Networks

To choose amongst graph models for some downstream task, one must generally consider two criteria: 1) whether the model can express structures of interest in the graph, 2) whether the model expresses these structure in an interpretable way.

Expressiveness of low-dimensional embeddings As real-world graphs are high-dimensional objects, graph models generally compress information about the graph. Such models are exemplified by the family of dot product models, which associate each node with a real-valued "embedding" vector; the predicted probability of a link between two nodes increases with the similarity of their embedding vectors. These models can alternatively be seen as factorizing the graph's adjacency matrix to approximate it with a low-rank matrix. Recent work of [167] has shown that dot product models are limited in their ability to model common structures in real-world graphs, such as triangles incident only on low-degree nodes. In contrast, in Section 3.1 we show that with the logistic principal components analysis (LPCA) model, which has two embeddings per node (i.e., using the dot product of the 'left' embedding of one node and the 'right' embedding of another), not only can such structures be represented, but further, any graph can be exactly represented with embedding vectors whose lengths are linear in the max degree of the graph. There are two keys to this result. First is the presence of a nonlinear linking function in the LPCA model; since adjacency matrices are generally not low-rank, exact low-rank factorization is generally impossible without a linking function. Second is that having two embeddings rather than one allows for expression of non-positive semidefinite (PSD) matrices. As discussed in [145], that the single-embedding models can only represent PSD matrices precludes representation of 'heterophilous' structures in graphs; heterophilous structures are those wherein dissimilar nodes are linked, in contrast to more intuitive 'homophilous' linking between similar nodes.

Interpretability and node clustering Beyond being able to capture a given network accurately, it is often desirable for a graph model to form interpretable representations of nodes and to produce edge probabilities in an interpretable fashion. Dot product models can achieve this by restricting the node embeddings to be nonnegative. Nonnegative factorization has long been used to decompose data into parts [49]. In the context of graphs, this entails decomposing the set of nodes of the network into clusters or communities. In particular, each entry of the nonnegative embedding vector of a node represents the intensity with which the node participates in a community. This allows the edge probabilities output by dot product models to be interpretable in terms of coparticipation in communities. Depending on the model, these vectors may have restrictions such as a sum-to-one requirement, meaning the node is assigned a categorical distribution over communities. The least restrictive and most expressive case is that of soft assignments to overlapping communities, where the entries can vary totally independently. In such models, which include the BigClam model of [202], the output of the dot product may be mapped through a nonlinear link function (as in LPCA) to produce a probability for each edge, i.e., to ensure the values lie in $[0, 1]$.

Heterophily: Motivating example To demonstrate how heterophily can manifest in networks, as well as how models which assume homophily can fail to represent such networks, we provide a simple synthetic example. Suppose we have a graph of matches between users of a mostly heterosexual dating app, and the users each come from one of ten cities. Members from the same city are likely to match with each other; this typifies homophily, wherein links occur between similar nodes. Furthermore, users having the same gender are are unlikely to match with each other; this typifies heterophily. Figure 3.5 shows an instantiation of such an adjacency matrix with 1000 nodes, which are randomly assigned to man or woman and to one of the ten cities. We recreate this network with our proposed embedding model and with BigClam, which explicitly assumes homophily. We also compare with the SVD of the adjacency matrix, which outputs the best (lowest Frobenius error) low-rank approximation that is possible without a nonlinear linking function. Since SVD lacks nonnegativity constraints on the factors, we do not expect intepretability. In Figure 3.5, we show how BigClam captures only the ten communities based on city, i.e., only the homophilous structure, and fails to capture the heterophilous distinction between men and women. We also plot the error of the reconstructions as the embedding length increases. There are $10 \cdot 2 = 20$ different kinds of nodes, meaning the expected adjacency matrix is rank-20, and our model maintains the lowest error up to this embedding length; by contrast, BigClam is unable to decrease error after capturing city information with length-10 embeddings. In Figure 3.7, we visualize the features generated by the three methods, i.e., the factors returned by each factorization. Our model's factors captures the relevant latent structure in an interpretable way. By contrast, SVD's factors are harder to interpret, and BigClam does not represent the heterophilous structure.

Summary of main contributions The key contributions of this work are as follows:

- We prove that the LPCA model admits exact low-rank factorizations of graphs with bounded *arboricity*, which is the minimum number of forests into which a graph's edges can be partitioned. By the Nash-Williams theorem, arboricity is a measure of a graph's density in that, letting S denote an induced subgraph and n_S and m_S denote the number of nodes and edges in S, arboricity is the maximum over all subgraphs S of $\lceil \frac{m_S}{n_S - 1} \rceil$. Our result is more applicable to real-world graphs than the prior one for graphs with bounded max degree, since sparsity is a common feature of real networks, whereas low max degree is not.

- We introduce a graph model which is both highly expressive and interpretable. Our model incorporates two embeddings per node and a nonlinear linking function, and hence is able to express both heterophily and overlapping communities. At the same time, our model is based on symmetric nonnegative matrix factorization, so it outputs link probabilities which are interpretable in terms of the communities it detects.

- We show how any graph with a low-rank factorization in the LPCA model also admits a low-rank factorization in our community-based model. This means that the guarantees on low-rank representation for bounded max degree and arboricity also apply to our model.

- In experiments, we show that our method is competitive with and often outperforms other comparable models on real-world graphs in terms of representing the network, doing interpretable link prediction, and detecting communities that align with ground-truth.

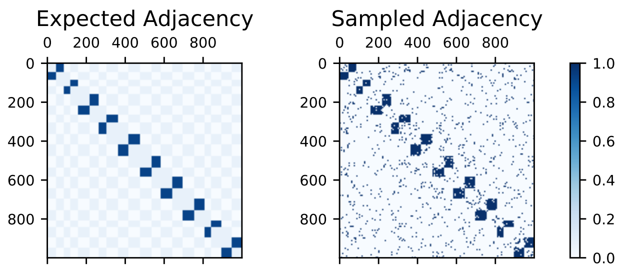


Figure 3.5. The motivating synthetic graph. The expected adjacency matrix (left) and the sampled matrix (right); the latter is passed to the training algorithms. The network is approximately a union of ten bipartite graphs, each of which correspond to men and women in one of the ten cities.

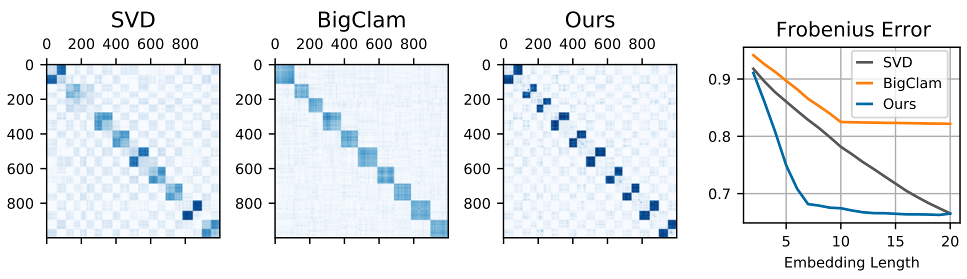


Figure 3.6. Left: Reconstructions of the motivating synthetic graph of Figure 3.5 with SVD, BigClam, and our model, using 12 communities or singular vectors. Note the lack of the small diagonal structure in BigClam's reconstruction; this corresponds to its inability to capture the heterophilous interaction between men and women. Right: Frobenius error when reconstructing the motivating synthetic graph of Figure 3.5 with SVD, BigClam, and our model, as the embedding length is varied. The error is normalized by the sum of the true adjacency matrix (i.e., twice the number of edges).

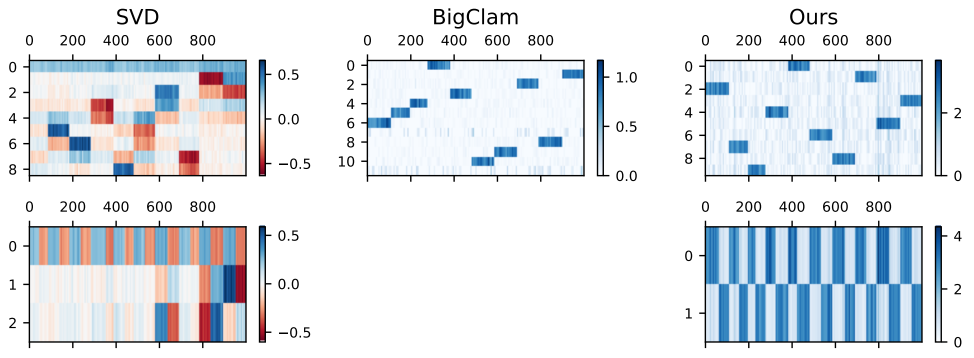


Figure 3.7. Factors resulting from decomposition of the motivating synthetic graph of Figure 3.5 with the three models, using 12 communities or singular vectors. The top/bottom rows represent the positive/negative eigenvalues corresponding to homophilous/heterophilous communities (note that BigClam does not include the latter). The homophilous factors from BigClam and our model reflect the 10 cities, and the heterophilous factor from our model reflect men and women. The factors from SVD are harder to interpret. Note that the order of the communities in the factors is arbitrary.

3.2.1 Community-Based Graph Factorization Model

Consider the set of undirected, unweighted graphs on n nodes, i.e., the set of graphs with symmetric adjacency matrices in $\{0,1\}^{n\times n}$. We propose an edge-independent generative model for such graphs. Given nonnegative parameter matrices $\boldsymbol{B} \in \mathbb{R}_+^{n\times k_B}$

and $\boldsymbol{C} \in \mathbb{R}_+^{n \times k_C}$, we set the probability of an edge existing between nodes i and j to be the (i,j)-th entry of matrix $\tilde{\boldsymbol{A}}$:

$$\tilde{\boldsymbol{A}} := \sigma(\boldsymbol{B}\boldsymbol{B}^\top - \boldsymbol{C}\boldsymbol{C}^\top), \tag{3.2}$$

where σ is the logistic function. Here k_B, k_C are the number of homophilous/heterophilous clusters. Intuitively, if $\boldsymbol{b}_i \in \mathbb{R}_+^{k_B}$ is the i-th row of matrix $\boldsymbol{B}$, then $\boldsymbol{b}_i$ is the affinity of node i to each of the k_B homophilous communities. Similarly, $\boldsymbol{c}_i \in \mathbb{R}_+^{k_C}$ is the affinity of node i to the k_C heterophilous communities. As an equivalent statement, for each pair of nodes i and j, $\tilde{\boldsymbol{A}}_{i,j} := \sigma(\boldsymbol{b}_i\boldsymbol{b}_j^\top - \boldsymbol{c}_i\boldsymbol{c}_j^\top)$. We will soon discuss the precise interpretation of this model, but the idea is roughly similar to the attract-repel framework of [145]. When nodes i and j have similar 'attractive' $\boldsymbol{b}$ embeddings, i.e., when $\boldsymbol{b}_i\boldsymbol{b}_j^\top$ is high, the likelihood of an edge between them increases, hence why the $\boldsymbol{B}$ factor is homophilous. By contrast, the $\boldsymbol{C}$ factor is 'repulsive'/heterophilous since, when $\boldsymbol{c}_i\boldsymbol{c}_j^\top$ is high, the likelihood of an edge between i and j decreases.

Alternate expression We note that the model above can also be expressed in a form which normalizes cluster assignments and is more compact, in that it combines the homophilous and heterophilous cluster assignments. Instead of $\boldsymbol{B}$ and $\boldsymbol{C}$, this form uses a matrix $\boldsymbol{V} \in [0,1]^{n \times k}$ and a diagonal matrix $\boldsymbol{W} \in \mathbb{R}^{k \times k}$, where $k = k_B + k_C$ is the total number of clusters. In particular, let $\boldsymbol{m}_B$ and $\boldsymbol{m}_C$ be the vectors containing the maximums of each column of $\boldsymbol{B}$ and $\boldsymbol{C}$. By setting

$$\begin{aligned} \boldsymbol{V} &= \left(\boldsymbol{B} \times \operatorname{diag}\left(\boldsymbol{m}_B^{-1}\right);\quad \boldsymbol{C} \times \operatorname{diag}\left(\boldsymbol{m}_C^{-1}\right)\right) \\ \boldsymbol{W} &= \operatorname{diag}\left(\left(+\boldsymbol{m}_B^2;\quad -\boldsymbol{m}_C^2\right)\right), \end{aligned} \tag{3.3}$$

the constraint on $\boldsymbol{V}$ is satisfied. Further, $\boldsymbol{V}\boldsymbol{W}\boldsymbol{V}^\top = \boldsymbol{B}\boldsymbol{B}^\top - \boldsymbol{C}\boldsymbol{C}^\top$, so

$$\tilde{\boldsymbol{A}} := \sigma(\boldsymbol{B}\boldsymbol{B}^\top - \boldsymbol{C}\boldsymbol{C}^\top) = \sigma(\boldsymbol{V}\boldsymbol{W}\boldsymbol{V}^\top). \tag{3.4}$$

Here, if $\boldsymbol{v}_i \in [0,1]^k$ is the i-th row of matrix $\boldsymbol{V}$, then $\boldsymbol{v}_i$ is the soft (normalized) assignment of node i to the k communities. The diagonal entries of $\boldsymbol{W}$ represent the strength of the homophily (if positive) or heterophily (if negative) of the communities. For each entry, $\tilde{\boldsymbol{A}}_{i,j} = \sigma(\boldsymbol{v}_i \boldsymbol{W} \boldsymbol{v}_j^\top)$. We use these two forms interchangeably throughout this work.

Interpretation The edge probabilities output by this model have an intuitive interpretation. Recall that there are bijections between probability $p \in [0,1]$, odds $o = \frac{p}{1-p} \in [0,\infty)$, and logit $\ell = \log(o) \in (-\infty,+\infty)$. The logit of the link probability between nodes i and j is $\boldsymbol{v}_i^\top \boldsymbol{W} \boldsymbol{v}_j$, which is a summation of terms $\boldsymbol{v}_{ic}\boldsymbol{v}_{jc}\boldsymbol{W}_{cc}$ over all communities $c \in [k]$. If the nodes both fully participate in community c, that is, $\boldsymbol{v}_{ic} = \boldsymbol{v}_{jc} = 1$, then the edge logit is changed by $\boldsymbol{W}_{cc}$ starting from a baseline of 0, or equivalently, the odds of an edge is multiplied by $\exp(\boldsymbol{W}_{cc})$ starting from a baseline odds of 1; if either of the nodes participates only partially in community c, then the change in logit and odds is accordingly prorated. Homophily and heterophily also have a clear interpretation in this model: homophilous communities, which are expressed in $\boldsymbol{B}$, are those with $\boldsymbol{W}_{cc} > 0$, where two nodes both participating in the community increases the odds of a link, whereas communities with $\boldsymbol{W}_{cc} < 0$, which are expressed in $\boldsymbol{C}$, are heterophilous, and coparticipation decreases the odds of a link.

3.2.2 Related Work

Community detection via interpretable factorizations There is extensive prior work on the community detection / node clustering problem [164, 5, 131], perhaps the most well-known being the normalized cuts algorithm of [169], which produces a clustering based on the entrywise signs of an eigenvector of the graph Laplacian matrix. However, the clustering algorithms which are most relevant to our work are those based on non-negative matrix factorization (NMF) [100, 18, 194, 66].

One such algorithm is that of [207], which approximately factors a graph's adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n \times n}$ into two positive matrices $\boldsymbol{H}$ and $\boldsymbol{\Lambda}$, where $\boldsymbol{H} \in \mathbb{R}_{+}^{n \times k}$ is left-stochastic (i.e. each of its columns sums to 1) and $\boldsymbol{\Lambda} \in \mathbb{R}_{+}^{k \times k}$ is diagonal, such that $\boldsymbol{H}\boldsymbol{\Lambda}\boldsymbol{H}^{\top} \approx \boldsymbol{A}$. Here $\boldsymbol{H}$ represents a soft clustering of the n nodes into k clusters, while the diagonal entries of $\boldsymbol{\Lambda}$ represent the prevalence of edges within clusters. Note the similarity of the factorization to our model, save for the lack of a nonlinearity. Other NMF approaches include those of [48], [204], [97], and [98] (SymNMF).

Modeling heterophily Much of the existing work on graph models has an underlying assumption of network homophily [88, 134]. There has been significant recent interest in the limitations of graph neural network (GNN) models [53, 94, 78] at addressing network heterophily [135, 212], as well as proposed solutions [142, 201], but relatively less work for more fundamental models such as those for clustering. Some existing NMF approaches to clustering do naturally model heterophilous structure in networks. For example, the model of [127] is similar to ours and also allows for heterophily, though it restricts the cluster assignment matrix $\boldsymbol{V}$ to be binary; additionally, their training algorithm is not based on gradient descent as ours is, and it does not scale to larger networks. More recently, [145] propose a decomposition of the form $\boldsymbol{A} \approx \boldsymbol{D} + \boldsymbol{B}\boldsymbol{B}^{\top} - \boldsymbol{C}\boldsymbol{C}^{\top}$, where $\boldsymbol{D} \in \mathbb{R}^{n \times n}$ is diagonal and $\boldsymbol{B}, \boldsymbol{C} \in \mathbb{R}^{n \times k}$ are low-rank. Note that their decomposition does not include a nonlinear linking function, and their work does not pursue a clustering interpretation or investigate setting the factors $\boldsymbol{B}$ and $\boldsymbol{C}$ to be nonnegative.

Overlapping communities and exact embeddings Many models discussed above focus on the single-label clustering task and thus involve highly-constrained factorizations (e.g., sum-to-one conditions). We are interested in the closely related but distinct task of multi-label clustering, also known as overlapping community detection [198, 87], which involves less constrained, more expressive factorizations. The

BigClam algorithm of [202] uses the following generative model for this task: the probability of a link between two nodes i and j is given by $1 - \exp(-\boldsymbol{f}_i \cdot \boldsymbol{f}_j)$, where $\boldsymbol{f}_i, \boldsymbol{f}_j \in \mathbb{R}_+^k$ represent the intensities with which the nodes participate in each of the k communities. Note that BigClam assumes strict homophily of the communities: two nodes participating in the same community always increases the probability of a link. However, this model allows for expression of very dense intersections of communities, which the authors observe is generally a characteristic of real-world networks. To ensure that output entries are probabilities, BigClam's factorization includes a nonlinear linking function (namely, $f(x) = 1 - e^x$), like our model and LPCA. Recent work outside clustering and community detection on graph generative models [152] suggests that incorporating a linking function can greatly increase the expressiveness of factorization-based graph models, to the point of being able to exactly represent a graph, as we showed in Section 3.1. This adds to a growing body of literature on expressiveness guarantees for embeddings on relational data [161, 19, 23]. As previously discussed, in Section 3.1, we provide a guarantee for exact low-rank representation of graphs with bounded max degree when using the LPCA factorization model. In this section, we provide a new such guarantee, except for bounded arboricity, which is more applicable to real-world networks, and extend these guarantees to our community-based factorization.

3.2.3 Theoretical Results

We first restate the main result from Section 3.1 on exact representation of graphs with bounded max degree using the logistic principal components analysis (LPCA) model, which reconstructs a graph $\boldsymbol{A} \in \{0,1\}^{n \times n}$ using logit factors $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n \times k}$ via

$$\boldsymbol{A} \approx \sigma(\boldsymbol{X}\boldsymbol{Y}^\top). \tag{3.5}$$

Note that unlike our community-based factorization, the factors of the LPCA model are not nonnegative, and the factorization does not reflect the symmetry of the undirected graph's adjacency matrix. Regardless of the model's interpretability, the following theorem provides a significant guarantee on its expressiveness. We use the following notation: given a matrix $\boldsymbol{M}$, let $H(\boldsymbol{M})$ denote the matrix resulting from entrywise application of the Heaviside step function to $\boldsymbol{M}$, that is, setting all positive entries to 1, negative entries to 0, and zero entries to $1/2$.

Theorem 3.2.1 (Exact LPCA Factorization for Bounded-Degree Graphs)**.** *Let $\boldsymbol{A} \in \{0,1\}^{n\times n}$ be the adjacency matrix of a graph G with maximum degree c. Then there exist matrices $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n\times(2c+1)}$ such that $\boldsymbol{A} = H(\boldsymbol{X}\boldsymbol{Y}^\top)$.*

This corresponds to arbitrarily small approximation error in the LPCA model (Equation 3.5) because, provided such factors $\boldsymbol{X}, \boldsymbol{Y}$ for some graph $\boldsymbol{A}$, we have that $\lim_{s\to\infty} \sigma\left(s\boldsymbol{X}\boldsymbol{Y}^\top\right) = H(\boldsymbol{X}\boldsymbol{Y}^\top) = \boldsymbol{A}$. That is, we can scale the factors larger to reduce the error to an arbitrary extent.

We expand on this result in two ways. First, give a new bound for exact embedding in terms of arboricity, rather than max degree. This significantly increases the applicability to real-world networks, which often are sparse (i.e., low arboricity) and have right-skewed degree distributions (i.e., high max degree). Second, we show that any rank-k LPCA factorization can be converted to our model's symmetric nonnegative factorization with $\mathcal{O}(k)$ communities. This extends the guarantees on the LPCA model's power for exact representation of graphs, both the prior guarantee in terms of max degree and our new one in terms of arboricity, to our community-based model as well. After this, we also introduce an example of a natural family of graphs - Community Overlap Threshold (COT) graphs - for which our model's community-based factorization not only exactly represents the graph, but also must capture some latent structure to do so with sufficiently low embedding dimensionality.

Arboricity bound for exact representation We will use the following well-known fact: the rank of the entrywise product of two matrices is at most the product of their individual ranks, that is,

$$\mathrm{rank}(\boldsymbol{X} \circ \boldsymbol{Y}) \leq \mathrm{rank}(\boldsymbol{X}) \cdot \mathrm{rank}(\boldsymbol{Y}).$$

Theorem 3.2.2 (Exact LPCA Factorization for Bounded-Arboricity Graphs)**.** *Let $\boldsymbol{A} \in \{0,1\}^{n \times n}$ be the adjacency matrix of an undirected graph G with arboricity α. Then there exist embeddings $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n \times (4\alpha^2+1)}$ such that $\boldsymbol{A} = H(\boldsymbol{X}\boldsymbol{Y}^\top)$.*

Proof. Let the undirected graph $\boldsymbol{A}$ have arboricity α, i.e., the edges can be partitioned into α forests. We produce a directed graph $\boldsymbol{B}$ from $\boldsymbol{A}$ by orienting the edges in these forests so that each node's edges point towards its children. Now $\boldsymbol{A} = \boldsymbol{B} + \boldsymbol{B}^\top$, and every node in $\boldsymbol{B}$ has in-degree at most α.

Let $\boldsymbol{V} \in \mathbb{R}^{n \times 2\alpha}$ be the Vandermonde matrix with $\boldsymbol{V}_{t,j} = t^{j-1}$. For any $\boldsymbol{c} \in \mathbb{R}^{2\alpha}$, $[\boldsymbol{V}\boldsymbol{c}](t) = \sum_{j=1}^{2\alpha} \boldsymbol{c}(j) \cdot t^{j-1}$, that is, $\boldsymbol{V}\boldsymbol{c} \in \mathbb{R}^n$ is a degree-(2α) polynomial with coefficients $\boldsymbol{c}$ evaluated at the integers $t \in [n] = \{1, \ldots, n\}$. Let $\boldsymbol{b}_i$ be the i^{th} column of $\boldsymbol{B}$. We seek to construct a polynomial such that for t with $\boldsymbol{b}_i(t) = 1$, $[\boldsymbol{V}\boldsymbol{c}_i](t) = 0$, and $[\boldsymbol{V}\boldsymbol{c}_i](t) < 0$ elsewhere; that is, when inputting an index $t \in [n]$ such that the t^{th} node is an in-neighbor of the i^{th} node, we want the polynomial to output 0, and for all other indices in $[n]$, we want it to have a negative output. Letting $N(i)$ denote the in-neighbors of the i^{th} node, a simple instantiation of such a polynomial in t is $-1 \cdot \prod_{j \in N(i)} (t-j)^2$. Note that since all nodes have in-degree at most α, this polynomial's degree is at most 2α, and hence there exists a coefficient vector $\boldsymbol{c}_i \in \mathbb{R}^{2\alpha}$ encoding this polynomial.

Let $\boldsymbol{C} \in \mathbb{R}^{n \times 2\alpha}$ be the matrix resulting from stacking such coefficient vectors for each of the n nodes. Consider $\boldsymbol{P} = \boldsymbol{V}\boldsymbol{C} \in \mathbb{R}^{n \times n}$: $\boldsymbol{P}_{i,j}$ is 0 if $\boldsymbol{B}_{i,j} = 1$ and negative otherwise. Then $(\boldsymbol{P} \circ \boldsymbol{P}^\top)_{i,j}$ is 0 when either $\boldsymbol{B}_{i,j} = 1$ or $(\boldsymbol{B}^\top)_{i,j} = 1$ and positive otherwise; equivalently, since $\boldsymbol{A} = \boldsymbol{B} + \boldsymbol{B}^\top$, $(\boldsymbol{P} \circ \boldsymbol{P}^\top)_{i,j} = 0$ iff $\boldsymbol{A}_{i,j} = 1$. Take any positive ϵ less than the smallest positive entry of $\boldsymbol{P} \circ \boldsymbol{P}^\top$. Letting $\boldsymbol{J}$ be an all-ones matrix, define $\boldsymbol{M} = \epsilon \boldsymbol{J} - (\boldsymbol{P} \circ \boldsymbol{P}^\top)$. Note that $\boldsymbol{M}_{i,j} > 0$ if $\boldsymbol{A} = 1$ and $\boldsymbol{M}_{i,j} < 0$ if $\boldsymbol{A} = 0$, that is, $\boldsymbol{M} = H(\boldsymbol{A})$ as desired. Since $\text{rank}(\boldsymbol{J}) = 1$ and $\text{rank}(\boldsymbol{P}) \leq 2\alpha$, by the bound on the rank of entrywise products of matrices, the rank of $\boldsymbol{M}$ is at most $(2\alpha)^2 + 1$. □

Exact representation with community factorization LPCA factors $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n \times k}$ can be processed into nonnegative factors $\boldsymbol{B} \in \mathbb{R}_+^{n \times k_B}$ and $\boldsymbol{C} \in \mathbb{R}_+^{n \times k_C}$ such that $k_B + k_C = 6k$ and

$$\boldsymbol{B}\boldsymbol{B}^\top - \boldsymbol{C}\boldsymbol{C}^\top = \tfrac{1}{2}\left(\boldsymbol{X}\boldsymbol{Y}^\top + \boldsymbol{Y}\boldsymbol{X}^\top\right). \tag{3.6}$$

Observe that the left-hand side can only represent symmetric matrices, but $\boldsymbol{X}\boldsymbol{Y}^\top$ is not necessarily symmetric even if $H(\boldsymbol{X}\boldsymbol{Y}^\top) = \boldsymbol{A}$ for a symmetric $\boldsymbol{A}$. For this reason, we use a symmetrization: let $\boldsymbol{L} = \frac{1}{2}\left(\boldsymbol{X}\boldsymbol{Y}^\top + \boldsymbol{Y}\boldsymbol{X}^\top\right)$. Note that $H(\boldsymbol{L}) = H(\boldsymbol{X}\boldsymbol{Y}^\top)$, so if $\boldsymbol{X}\boldsymbol{Y}^\top$ constitutes an exact representation of $\boldsymbol{A}$ in that $H(\boldsymbol{X}\boldsymbol{Y}^\top) = \boldsymbol{A}$, so too do both expressions for $\boldsymbol{L}$ in Equation 3.6. Pseudocode for the procedure of constructing $\boldsymbol{B}, \boldsymbol{C}$ given $\boldsymbol{X}, \boldsymbol{Y}$ is given in Algorithm 5. The concept of this algorithm is to first separate the logit matrix $\boldsymbol{L}$ into a sum and difference of rank-1 components via eigendecomposition. Each of these components can be written as $+\boldsymbol{v}\boldsymbol{v}^\top$ or $-\boldsymbol{v}\boldsymbol{v}^\top$ with $\boldsymbol{v} \in \mathbb{R}^n$, where the sign depends on the sign of the eigenvalue. Each component is then separated into a sum and difference of three outer products of nonnegative vectors, via Lemma 3.2.3 below.

Lemma 3.2.3. *Let $\phi : \mathbb{R} \to \mathbb{R}$ denote the ReLU function, i.e., $\phi(z) = \max\{z, 0\}$. For any vector $\boldsymbol{v}$,*

$$\boldsymbol{v}\boldsymbol{v}^\top = 2\phi(\boldsymbol{v})\phi(\boldsymbol{v})^\top + 2\phi(-\boldsymbol{v})\phi(-\boldsymbol{v})^\top - |\boldsymbol{v}||\boldsymbol{v}|^\top.$$

Proof. Take any $\boldsymbol{v} \in \mathbb{R}^k$. Then

$$\begin{aligned}
\boldsymbol{v}\boldsymbol{v}^\top &= (\phi(\boldsymbol{v}) - \phi(-\boldsymbol{v})) \cdot (\phi(\boldsymbol{v})^\top - \phi(-\boldsymbol{v})^\top) \\
&= \phi(\boldsymbol{v})\phi(\boldsymbol{v})^\top + \phi(-\boldsymbol{v})\phi(-\boldsymbol{v})^\top - \phi(\boldsymbol{v})\phi(-\boldsymbol{v})^\top - \phi(-\boldsymbol{v})\phi(\boldsymbol{v})^\top \\
&= 2\phi(\boldsymbol{v})\phi(\boldsymbol{v})^\top + 2\phi(-\boldsymbol{v})\phi(-\boldsymbol{v})^\top - (\phi(\boldsymbol{v}) + \phi(-\boldsymbol{v})) \cdot (\phi(\boldsymbol{v}) + \phi(-\boldsymbol{v}))^\top \\
&= 2\phi(\boldsymbol{v})\phi(\boldsymbol{v})^\top + 2\phi(-\boldsymbol{v})\phi(-\boldsymbol{v})^\top - |\boldsymbol{v}||\boldsymbol{v}|^\top,
\end{aligned}$$

where the first step follows from $\boldsymbol{v} = \phi(\boldsymbol{v}) - \phi(-\boldsymbol{v})$, and the last step from $|\boldsymbol{v}| = \phi(\boldsymbol{v}) + \phi(-\boldsymbol{v})$. □

Algorithm 5 follows from Lemma 3.2.3 and constitutes a constructive proof of the following theorem:

Theorem 3.2.4 (Exact Community Factorization from Exact LPCA Factorization)**.** *Given a symmetric matrix $\boldsymbol{A} \in \{0, 1\}$ and $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n\times k}$ such that $\boldsymbol{A} = H(\boldsymbol{X}\boldsymbol{Y}^\top)$, there exist nonnegative matrices $\boldsymbol{B} \in \mathbb{R}_+^{n\times k_B}$ and $\boldsymbol{C} \in \mathbb{R}_+^{n\times k_C}$ such that $k_B + k_C = 6k$ and $\boldsymbol{A} = H(\boldsymbol{B}\boldsymbol{B}^\top - \boldsymbol{C}\boldsymbol{C}^\top)$.*

Algorithm 5 Converting LPCA Factorization to Community Factorization

```
input logit factors $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n\times k}$
output $\boldsymbol{B} \in \mathbb{R}_+^{n\times k_B}$ and $\boldsymbol{C} \in \mathbb{R}_+^{n\times k_C}$ such that $k_B + k_C = 6k$ and
        $\boldsymbol{B}\boldsymbol{B}^\top - \boldsymbol{C}\boldsymbol{C}^\top = \frac{1}{2}\left(\boldsymbol{X}\boldsymbol{Y}^\top + \boldsymbol{Y}\boldsymbol{X}^\top\right)$
1: Set $\boldsymbol{Q} \in \mathbb{R}^{n\times 2k}$ and $\boldsymbol{\lambda} \in \mathbb{R}^{2k}$ by truncated eigendecomposition such that
        $\boldsymbol{Q} \times \mathrm{diag}(\boldsymbol{\lambda}) \times \boldsymbol{Q}^\top = \frac{1}{2}(\boldsymbol{X}\boldsymbol{Y}^\top + \boldsymbol{Y}\boldsymbol{X}^\top)$
2: $\boldsymbol{B}^* \leftarrow \boldsymbol{Q}^+ \times \mathrm{diag}(\sqrt{+\boldsymbol{\lambda}^+})$, where $\boldsymbol{\lambda}^+$, $\boldsymbol{Q}^+$ are the positive eigenvalues/vectors
3: $\boldsymbol{C}^* \leftarrow \boldsymbol{Q}^- \times \mathrm{diag}(\sqrt{-\boldsymbol{\lambda}^-})$, where $\boldsymbol{\lambda}^-$, $\boldsymbol{Q}^-$ are the negative eigenvalues/vectors
4: $\boldsymbol{B} \leftarrow \left(\sqrt{2}\phi(\boldsymbol{B}^*);\ \ \sqrt{2}\phi(-\boldsymbol{B}^*);\ \ |\boldsymbol{C}^*|\right)$ ▷ $\phi$ / $|\cdot|$ are entrywise ReLU / abs. value
5: $\boldsymbol{C} \leftarrow \left(\sqrt{2}\phi(\boldsymbol{C}^*);\ \ \sqrt{2}\phi(-\boldsymbol{C}^*);\ \ |\boldsymbol{B}^*|\right)$
6: return $\boldsymbol{B}, \boldsymbol{C}$
```

As stated in the introduction to this section, Theorem 3.2.4 extends any upper bound on the exact factorization dimensionality from the LPCA model to our community-based model. That is, up to a constant factor, the bound in terms of max degree from Theorem 3.2.1 and the bound in terms of arboricity from Theorem 3.2.2 also apply to our model; for brevity, we state just the latter here.

Corollary 3.2.5 (Exact Community Factorization for Bounded-Arboricity Graphs)**.** *Let $\boldsymbol{A} \in \{0,1\}^{n\times n}$ be the adjacency matrix of an undirected graph G with arboricity α. Then there exist nonnegative embeddings $\boldsymbol{B} \in \mathbb{R}_+^{n\times k_B}$ and $\boldsymbol{C} \in \mathbb{R}_+^{n\times k_C}$ such that $k_B + k_C = 6(4\alpha^2+1)$ and $\boldsymbol{A} = H(\boldsymbol{B}\boldsymbol{B}^\top - \boldsymbol{C}\boldsymbol{C}^\top)$.*

Note that Corollary 3.2.5 is purely a statement about the capacity of our model; Theorem 3.2.2 stems from a constructive proof based on polynomial interpolation, and therefore so too does this corollary. We do not expect this factorization to be informative about the graph's latent structure. In the following Section 3.2.4, we will fit the model with an entirely different algorithm for downstream applications.

Exact representation of COT Graphs As a theoretical demonstration of the capability of our model to learn latent structure, we additionally show that our model can exactly represent a natural family of graphs, which exhibits both homophily and heterophily, with small k and interpretably. The family of graphs is specified below in Definition 3; roughly speaking, nodes in such graphs share an edge iff they coparticipate in some number of homophilous communities and don't coparticipate in a number of heterophilous communities. For example, the motivating graph described at the start of this section would be an instance of such a graph if an edge occurs between two users iff the two users are from the same city and have different genders.

Definition 3 (Community Overlap Threshold (COT) Graph)**.** *An unweighted, undirected graph whose edges are determined by an overlapping clustering and a "thresholding" integer $t \in \mathbb{Z}$ as follows: for each vertex i, there are two latent binary vectors $\boldsymbol{b}_i \in \{0,1\}^{k_b}$ and $\boldsymbol{c}_i \in \{0,1\}^{k_c}$, and there is an edge between vertices i and j iff $\boldsymbol{b}_i \cdot \boldsymbol{b}_j - \boldsymbol{c}_i \cdot \boldsymbol{c}_j \geq t$.*

Theorem 3.2.6 (Compact Representation of COT Graphs)**.** *Suppose* $\boldsymbol{A}$ *is the adjacency matrix of a COT graph on* n *nodes with latent vectors* $\boldsymbol{b}_i \in \{0,1\}^{k_b}$ *and* $\boldsymbol{c}_i \in \{0,1\}^{k_c}$ *for* $i \in \{1, 2, \ldots, n\}$*. Let* $k = k_b + k_c$*. Then, for any* $\epsilon > 0$*, there exist* $\boldsymbol{V} \in [0,1]^{n\times(k+1)}$ *and diagonal* $\boldsymbol{W} \in \mathbb{R}^{(k+1)\times(k+1)}$ *such that* $\left\|\sigma(\boldsymbol{V}\boldsymbol{W}\boldsymbol{V}^\top) - \boldsymbol{A}\right\|_F < \epsilon$*.*

Proof. Let t be the thresholding integer of the graph, and let the rows of $\boldsymbol{B} \in \{0,1\}^{n\times k_b}$ and $\boldsymbol{C} \in \{0,1\}^{n\times k_c}$ contain the vectors $\boldsymbol{b}$ and $\boldsymbol{c}$ of all nodes. Via Equation 3.3, we can find $\boldsymbol{V}^* \in [0,1]^{n\times k}$ and diagonal $\boldsymbol{W}^* \in \mathbb{R}^{k\times k}$ such that $\boldsymbol{V}^*\boldsymbol{W}^*\boldsymbol{V}^{*\top} = \boldsymbol{B}\boldsymbol{B}^\top - \boldsymbol{C}\boldsymbol{C}^\top$. Now let

$$\boldsymbol{V} = \begin{pmatrix} \boldsymbol{V}^* & \boldsymbol{1} \end{pmatrix} \qquad \boldsymbol{W} = \begin{pmatrix} \boldsymbol{W}^* & 0 \\ 0 & \frac{1}{2} - t \end{pmatrix}.$$

Then $(\boldsymbol{V}\boldsymbol{W}\boldsymbol{V}^\top)_{ij} = \boldsymbol{b}_i\cdot\boldsymbol{b}_j - \boldsymbol{c}_i\cdot\boldsymbol{c}_j + \frac{1}{2} - t$. Hence $(\boldsymbol{V}\boldsymbol{W}\boldsymbol{V}^\top)_{ij} > 0$ iff $\boldsymbol{b}_i\cdot\boldsymbol{b}_j - \boldsymbol{c}_i\cdot\boldsymbol{c}_j > t - \frac{1}{2}$, which is true iff $\boldsymbol{A}_{ij} = 1$ by the assumption on the graph. Similarly, $(\boldsymbol{V}\boldsymbol{W}\boldsymbol{V}^\top)_{ij} < 0$ iff $\boldsymbol{A}_{ij} = 0$. It follows that

$$\lim_{s\to\infty} \sigma\left(\boldsymbol{V}(s\boldsymbol{W})\boldsymbol{V}^\top\right) = \lim_{s\to\infty} \sigma\left(s\boldsymbol{V}\boldsymbol{W}\boldsymbol{V}^\top\right) = \boldsymbol{A}. \qquad \square$$

3.2.4 Experiments

We now present a training algorithm to fit our model, then evaluate our method on a benchmark of five real-world networks.

3.2.4.1 Dataset Descriptions

We first briefly describe the five real-world datasets that are employed in this section, including discussion how some of them exhibit heterophily. These are fairly common small to mid-size datasets ranging from around 1K to 10K nodes. Statistics for these datasets are given in Table 3.3.

Table 3.3. Network statistics for experiments in Section 3.2. As in [178], for YouTube and Amazon, we take only nodes which participate in at least one of the largest 5 ground-truth communities. Note that degeneracy is an upper bound on arboricity.

Name	**Reference**	**Nodes**	**Edges**	**Labels**	**Max Degree**	**Degeneracy**
Blog	[183]	10,312	333,983	39	3992	114
YouTube	[203]	5,346	24,121	5	628	19
POS	[147]	4,777	92,406	40	3644	49
PPI	[25]	3,852	76,546	50	593	29
Amazon	[203]	794	2,109	5	29	6

Blog is a social network of relationships between online bloggers; the node labels represent interests of the bloggers. Similarly, **YouTube** is a social network of YouTube users, and the labels represent groups that the users joined.

POS is a word co-occurrence network: nodes represent words, and there are edges between words which are frequently adjacent in a section of the Wikipedia corpus. Each node label represents the part-of-speech of the word. **PPI** is a subgraph of the protein-protein interaction network for Homo Sapiens. Labels represent biological states. Finally, **Amazon** is a co-purchasing network: nodes represent products, and there are edges between products which are frequently purchased together. Labels represent categories of products.

While social networks like the former two in this list are generally dominated by homophily [123], the latter three should exhibit significant heterophily. For co-purchasing networks like Amazon, depending on the product, two of the same kind of product are generally not co-purchased, e.g., Pepsi and Coke, as discussed in [145]. Though less intuitively accessible, there is also prior discussion of disassortativity in word adjacencies [61, 213], as well as in PPI networks [132, 80].

3.2.4.2 Training Algorithm

Given an input graph $\boldsymbol{A} \in \{0,1\}^{n\times n}$, we find low-rank nonnegative matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ such that the model produces $\tilde{\boldsymbol{A}} = \sigma(\boldsymbol{B}\boldsymbol{B}^\top - \boldsymbol{C}\boldsymbol{C}^\top) \in (0,1)^{n\times n}$ as in Equation 3.2

which approximately matches $\boldsymbol{A}$. In particular, we train the model to minimize the sum of binary cross-entropies of the link predictions over all pairs of nodes:

$$R = -\sum \Big(\boldsymbol{A} \log(\tilde{\boldsymbol{A}}) + (1 - \boldsymbol{A}) \log(1 - \tilde{\boldsymbol{A}}) \Big) , \tag{3.7}$$

where $\sum$ denotes the scalar summation of all entries in the matrix. We fit the parameters by gradient descent over this loss, as well as L_2 regularization of the factors $\boldsymbol{B}$ and $\boldsymbol{C}$, subject to the nonnegativity of $\boldsymbol{B}$ and $\boldsymbol{C}$. This algorithm is fairly straightforward; pseudocode is given in Algorithm 6. This is quite similar to the training algorithm from Section 3.1, but in contrast to that section, which only targets an exact fit, here we explore the expression of graph structure in the factors and their utility in downstream tasks. Regularization of the factors is implemented to this end to avoid overfitting. Though we outline a non-stochastic version of the training algorithm, it generalizes straightforwardly to a stochastic version, i.e., by sampling links and non-links for the loss function.

Algorithm 6 Fitting the Constrained Model

```
input adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n \times n}$, regularization weight $\lambda \geq 0$, number of iterations $I$, number of homophilous/heterophilous communities $k_B/k_C$
output fitted factors $\boldsymbol{B} \in \mathbb{R}_+^{n \times k_B}$ and $\boldsymbol{C} \in \mathbb{R}_+^{n \times k_C}$ such that $\sigma(\boldsymbol{B}\boldsymbol{B}^\top - \boldsymbol{C}\boldsymbol{C}^\top) \approx \boldsymbol{A}$
1: Initialize $\boldsymbol{B}, \boldsymbol{C}$ by setting entries to independent samples of $\mathrm{Unif}(0, 1/\sqrt{k_B}), \mathrm{Unif}(0, 1/\sqrt{k_C})$
2: for $i \leftarrow 1$ to $I$ do
3:     $\tilde{\boldsymbol{A}} \leftarrow \sigma(\boldsymbol{B}\boldsymbol{B}^\top - \boldsymbol{C}\boldsymbol{C}^\top)$
4:     $R \leftarrow -\sum \Big( \boldsymbol{A} \log(\tilde{\boldsymbol{A}}) + (1 - \boldsymbol{A}) \log(1 - \tilde{\boldsymbol{A}}) \Big)$
5:     $R \leftarrow R + \lambda \left( \|\boldsymbol{B}\|_F^2 + \|\boldsymbol{C}\|_F^2 \right)$
6:     Calculate $\partial_{\boldsymbol{B},\boldsymbol{C}} R$ via differentiation through Steps 2 to 4
7:     Update $\boldsymbol{B}, \boldsymbol{C}$ to minimize $R$ using $\partial_{\boldsymbol{B},\boldsymbol{C}} R$, subject to $\boldsymbol{B}, \boldsymbol{C} \geq 0$
8: return $\boldsymbol{B}, \boldsymbol{C}$
```

Implementation details Our implementation uses PyTorch [140] for automatic differentiation and minimizes loss using the SciPy [89] implementation of the L-

BFGS [115, 210] algorithm with default hyperparameters and up to a max of 200 iterations of optimization. We set regularization weight $\lambda = 10$ as in [202].

3.2.4.3 Results

Expressiveness First, we investigate the expressiveness of our generative model, that is, the fidelity with which it can reproduce an input network. At the start of this section, we used a simple synthetic network to show that our model is more expressive than others due to its ability to represent heterophilous structures in addition to homophilous ones. We now evaluate the expressiveness of our model on real-world networks. As with the synthetic graph, we fix the number of communities or singular vectors, fit the model, then evaluate the reconstruction error. In Figure 3.8, we compare the results of our model with those of SVD, BigClam (which is discussed in detail in Section 3.2.2), and SymNMF [98]. SymNMF simply factors the adjacency matrix as $\boldsymbol{A} \approx \boldsymbol{H}\boldsymbol{H}^\top$, where $\boldsymbol{H} \in \mathbb{R}_+^{n \times k}$; note that, like SVD, SymNMF does not necessarily output a matrix whose entries are probabilities (i.e., bounded in $[0, 1]$), and hence it is not a graph generative model like ours and BigClam.

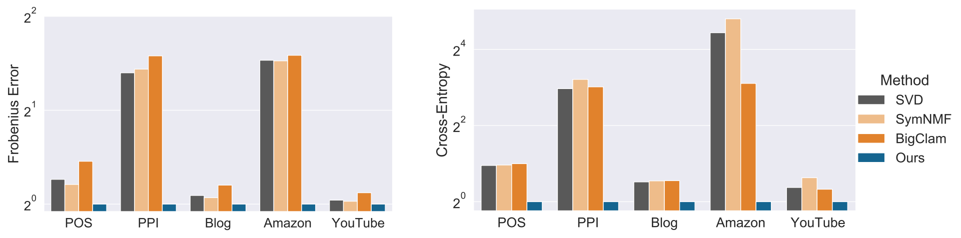


Figure 3.8. Reconstruction error on real-world networks, relative to our model's error.

For each method, we fix the number of communities or singular vectors at the ground-truth number. For this experiment only, we are not concerned with learning the latent structure of the graph; the only goal is accurate representation of the network with limited parameters. So, for a fair comparison with SVD, we do not regularize the training of the other methods. Our method consistently has the lowest reconstruction error, both in terms of Frobenius error and entrywise cross-entropy (Equation 3.7). Interestingly, we find the most significant improvement exactly on the three datasets which have been noted to exhibit significant heterophily: POS, PPI, and Amazon.

Similarity to ground-truth communities To assess the interpretability of clusters generated by our method, we evaluate the similarity of these clusters to ground-truth communities (i.e., class labels), and we compare other methods for overlapping clustering. We additionally compare to another recent but non-generative approach, the vGraph method of [178], which is based on link clustering; the authors found their method to generally achieve state-of-the-art results in this task. For all methods, we set the number of communities to be detected as the number of ground-truth communities. We report F1-Score as computed in [202]. See Figure 3.9 (left): the performance of our method is competitive with SymNMF, BigClam, and vGraph.

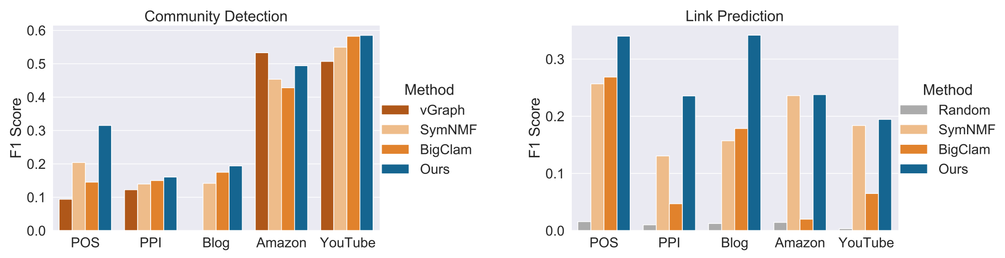


Figure 3.9. Left: Similarity of recovered communities to ground-truth labels of real-world datasets. We are unable to run the authors' implementation of vGraph on Blog with limited memory. Right: Accuracy of link prediction on real-world datasets.

Interpretable link prediction We assess the predictive power of our generative model on the link prediction task. As discussed in Section 3.2.1, the link probabilities output by our model are interpretable in terms of a clustering of nodes that it generates; we compare results with our method to those with other models which permit similar interpretation, namely BigCLAM and SymNMF. We randomly select 10%

of node pairs to hold out, fit the models on the remaining 90%, then use the trained models to predict links between node pairs in the held out 10%. As a baseline, we also show results for randomly predicting link or no link with equal probability. See Figure 3.9 (right). The performance of our method is competitive with or exceeds that of the other methods in terms of F1 Score.

3.2.5 Conclusion

We introduce a community-based graph generative model based on symmetric nonnegative matrix factorization which is capable of representing both homophily and heterophily. We expand on our prior guarantee of exact representation for bounded *max degree* graphs from Section 3.1 with a new, more applicable guarantee for bounded *arboricity* graphs, and we show that both of these bounds apply to our more interpretable graph model. We illustrate our model's capabilities with experiments on a synthetic motivating example. Experiments on real-world networks show its effectiveness on several key tasks. More broadly, our results suggest that incorporating heterophily into models and methods for networks can improve both theoretical grounding and overall empirical performance, while maintaining simplicity and interpretability. A deeper understanding of the expressiveness of both nonnegative and arbitrary low-rank logit models for graphs is an interesting future direction.

CHAPTER 4

POWER AND LIMITATIONS OF RANDOM GRAPH MODELS

In this chapter, we abstract away from node embeddings and examine upper and lower bounds on the capabilities of different random graph models at representing graph structure.

4.1 Inherent Limitations of Edge Independent Models

This section centers on *edge independent graph models*, in which each edge (i, j) is added to the graph independently with some probability $\boldsymbol{P}_{ij} \in [0, 1]$. Formally,

Definition 4 (Edge Independent Graph Model)**.** *For any symmetric matrix $\boldsymbol{P} \in [0,1]^{n\times n}$ let $\mathcal{G}(\boldsymbol{P})$ be the distribution over undirected unweighted graphs where $G \sim \mathcal{G}(\boldsymbol{P})$ contains edge (i, j) independently, with probability $\boldsymbol{P}_{ij}$. That is, $p(G) = \prod_{(i,j)\in E(G)} \boldsymbol{P}_{ij} \cdot \prod_{(i,j)\notin E(G)}(1 - \boldsymbol{P}_{ij})$.*

Edge independent models encompass many classic random graph models. This includes the Erdös-Rényi model, where for all $i \neq j$, $\boldsymbol{P}_{ij} = p$ for some fixed $p \in [0, 1]$ [56]. It also includes the stochastic block model where $\boldsymbol{P}_{ij} = p$ if two nodes are in the same community and $\boldsymbol{P}_{ij} = q$ if two nodes are in different communities for some fixed $p, q \in [0, 1]$ with $q < p$ [172]. Other examples include e.g., the Chung-Lu configuration model [39], stochastic Kronecker graphs [103].

Recently, significant attention has focused on *graph generative models*, which seek to learn a distribution over graphs that share similar properties to a given training graph, or set of graphs. Many algorithms parameterize this distribution as an edge independent model or closely related distribution. E.g., NetGAN and the closely related CELL model both produce $\boldsymbol{P} \in [0,1]^{n \times n}$ and then sample edges independently without replacement with probabilities proportional to its entries, ensuring that at least one edge is sampled adjacent to each node [20, 151]. Variational Graph Autoencoders (VGAE), GraphVAE, Graphite, and MolGAN are also all based on edge independent models [93, 171, 45, 76].

Given their popularity in both classical and modern graph generative models, it is natural to ask:

> *How suited are edge independent models to modeling real-world networks. Are they able to capture features such as power-law degree distributions, small-world properties, and high clustering coefficients (triangle densities)?*

4.1.1 Impossibility Results for Edge Independent Models

In this work we focus on the ability of edge independent models to generate graphs with high triangle, or other small subgraph densities. High triangle density (equivalently, a high clustering coefficient) is a well-known hallmark of real-work networks [195, 160, 52] and has been the focus of recent work exploring the power and limitations of edge-independent graph models [167, 34].

It is clear that edge independent models can generate triangle dense graphs. In particular, $\boldsymbol{P} \in [0,1]^{n \times n}$ in Definition 4 can be set to the binary adjacency matrix of any undirected graph, and $\mathcal{G}(\boldsymbol{P})$ will generate that graph with probability 1, no matter how triangle dense it is. However, this would not be a particularly interesting generative model – ideally $\mathcal{G}(\boldsymbol{P})$ should generate a wide range of graphs. To capture this intuitive notion, we define the *overlap* of an edge-independent model, which is closely related to the overlap stopping criterion for training used in training graph generative models [20, 151].

Definition 5 (Expected Overlap)**.** *For symmetric* $\boldsymbol{P} \in [0,1]^{n\times n}$ *let* $\mathrm{Vol}(\boldsymbol{P}) := \mathbb{E}_{G\sim\mathcal{G}(\boldsymbol{P})}|E(G)|$ *and*

$$\mathrm{Ov}(\boldsymbol{P}) := \frac{\mathbb{E}_{G_1,G_2\sim\mathcal{G}(\boldsymbol{P})}|E(G_1)\cap E(G_2)|}{\mathrm{Vol}(\boldsymbol{P})}.$$

That is, for any $\boldsymbol{P} \in [0,1]^{n\times n}$, $\mathrm{Ov}(\boldsymbol{P}) \in [0,1]$ is the ratio of the expected number of edges shared by two graphs drawn independently from $\mathcal{G}(\boldsymbol{P})$ to the expected number of edges in a graph drawn from $\mathcal{G}(\boldsymbol{P})$. In one extreme, when $\boldsymbol{P}$ is a binary adjacency matrix, $\mathrm{Ov}(\boldsymbol{P}) = 1$, and our generative model has simply memorized a single graph. In the other, if $\boldsymbol{P}_{ij} = p$ for all $i \neq j$ (i.e., $\mathcal{G}(\boldsymbol{P})$ is Erdös-Rényi), $\mathrm{Ov}(\boldsymbol{P}) = p$. This is the minimum possible overlap when $\mathrm{Vol}(\boldsymbol{P}) = p \cdot \binom{n}{2}$.

Our main result is that for any edge independent model with bounded overlap, $G \sim \mathcal{G}(\boldsymbol{P})$ cannot have too many triangles in expectation. In particular:

Theorem 4.1.1 (Main Result – Expected Triangles)**.** *For a graph* G*, let* $\Delta(G)$ *denote the number of triangles in* G*. Consider symmetric* $\boldsymbol{P} \in [0,1]^{n\times n}$*.*

$$\mathbb{E}_{G\sim\mathcal{G}(\boldsymbol{P})}\left[\Delta(G)\right] \leq \frac{\sqrt{2}}{3} \cdot \mathrm{Ov}(\boldsymbol{P})^{3/2} \cdot \mathrm{Vol}(\boldsymbol{P})^{3/2}.$$

As an example, consider the setting where the distribution generates sparse graphs, with $\mathrm{Vol}(\boldsymbol{P}) = \Theta(n)$. Theorem 4.1.1 shows that whenever $\mathrm{Ov}(\boldsymbol{P}) = o(1/n^{1/3})$, we have $\mathbb{E}_{G\sim\mathcal{G}(\boldsymbol{P})}\Delta(G) = o(n)$. That is, the graph is very triangle sparse with the number of triangles sublinear in the number of nodes. This verifies that an Erdös-Rényi graph cannot achieve simultaneously linear number of edges (i.e., $\mathrm{Ov}(\boldsymbol{P}) = \mathcal{O}(1/n)$) and super-linear number of triangles (i.e., $\mathrm{Ov}(\boldsymbol{P}) = \Omega(1/n^{1/3})$) under our proposed lens of viewing generative models.

We extend Theorem 4.1.1 to give similar bounds for the density of squares and other k-cycles (Theorem 4.1.4), as well as for the global clustering coefficient (Theorem 4.1.6). In all cases we show that our bounds are tight – e.g., in the triangle case, there is indeed an edge independent model with

$$\mathbb{E}_{G \sim \mathcal{G}(\boldsymbol{P})} [\Delta(G)] = \Theta \left(\mathrm{Ov}(\boldsymbol{P})^{3/2} \cdot \mathrm{Vol}(\boldsymbol{P})^{3/2} \right),$$

matching the lower bound in Theorem 4.1.1.

4.1.2 Empirical Findings

Our theoretical results help explain why, despite performing well in a variety of other metrics, edge independent graph generative models have been reported to generate graphs with many fewer triangles and squares on average than the real-world graphs that they are trained on. Rendsburg et al. [151] test a suite of these models, including their own CELL model and the related NetGAN model [20]. Of all these models, when trained on the Cora-ML graph with 2,802 triangles and 14,268 squares, none is able to generate graphs with more than 1,461 triangles and 6,880 squares on average. Similar gaps are observed for a number of other graphs. Rendsburg et al. also report that the triangle count increases as their notion of overlap (closely related to Definition 5) increases. Theorem 4.1.1 demonstrates that this underestimation of triangle count, and its connection to overlap is *inherent to all edge independent models, no matter how refined a method used to learn the underlying probability matrix* $\boldsymbol{P}$.

While our theoretical results bound the performance of any edge independent model, there may still be variation in how specific models trade-off overlap and realistic graph generation. To better understand this trade-off, we introduce two simple models with easily tunable overlap as baselines. One is based on reproducing the degree sequence of the original graph; the other, which is even simpler, is based on reproducing the volume. In both models, $\boldsymbol{P}$ is a weighted average of the input graph adjacency matrix and a probability matrix of minimal complexity which matches either the input degrees or the volume. In the latter case, to match just the volume, we simply use an Erdös-Rényi graph. In the former case, to match the degree sequence, we introduce our own model, the *odds product model*; this model is similar to the Chung-Lu configuration model [39], but, unlike Chung-Lu, is able to match degree sequences of real-world graphs with high maximum degree. We find that these simple baselines are often competitive with more complex models like CELL in terms of matching key graph statistics, like triangle count and clustering coefficient, at similar levels of overlap.

4.1.3 Related Work

Existing impossibility results. Our work is inspired by that of Seshadhri et al. [167], which also proves limitations on the ability of edge independent models to represent triangle dense graphs. They show that if $\boldsymbol{P} = \max(0, \min(1, \boldsymbol{X}\boldsymbol{X}^\top))$ where $\boldsymbol{X} \in \mathbb{R}^{n\times k}$ for $k \ll n$ and the max and min are applied entrywise, then $G \sim \mathcal{G}(\boldsymbol{P})$ cannot have many triangles adjacent to low-degree nodes in expectation. This setting arises commonly when $\boldsymbol{P}$ is generated using low-dimensional node embeddings – represented by the rows of $\boldsymbol{X}$. In Section 3.1, we show that in a slightly more general model, where $\boldsymbol{P} = \max(0, \min(1, \boldsymbol{X}\boldsymbol{Y}^\top))$, this lower bound no longer holds – $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n\times k}$ can be chosen so that $\boldsymbol{P}$ is the binary adjacency matrix of any graph with maximum degree upper bounded by $\mathcal{O}(k)$ – no matter how triangle dense that graph is. Thus, even such low-rank edge independent models can represent triangle dense graphs – by memorizing a single one.

Our results show that this trade-off between the ability to capture triangle density and memorization is inherent – even without any low-rank constraint, edge independent models with low overlap simply cannot represent graphs with high triangle or other small subgraph density.

It is well understood that specific edge independent models, e.g., Erdös-Rényi graphs, the Chung-Lu model, and stochastic Kronecker graphs, do not capture many properties of real-world networks, including high triangle density [195, 146]. Our results can be viewed as a generalization of these observations, to all edge independent models with low overlap. Despite the limitations of classic models, edge independent models are still very prevalent in today's literature on graph generative models. Our more general results make clear the limitations of this approach.

Non-independent models. While edge independent models are very prevalent in the literature, many important models do not fit into this framework. Classic models include the Barabási–Albert and other preferential attachment models [16], Watts–Strogatz small-world graphs [195], and random geometric graphs [43]. Many of these models were introduced directly in response to shortcomings of classic edge independent models, including their inability to produce high triangle densities.

More recent graph generative models include GraphRNN [205] and a number of other works [111, 112]. Our impossibility results do not apply to such models, and in fact suggest that perhaps they may be preferable to edge independent models, if a distribution over graphs with high triangle density is desired. An interesting direction that we return to in Section 4.2 is proving limitations on broad classes of non-independent models, and perhaps to understand exactly what type of correlation amongst edges is needed to generate graphs with both low overlap and hallmark features of real-world networks.

4.1.4 Impossibility Results for Edge Independent Models

We now prove our main results on the limitations of edge independent models with bounded overlap. We start with a simple lemma that will be central in all our proofs.

Lemma 4.1.2. *For any symmetric* $\boldsymbol{P} \in [0,1]^{n \times n}$, $\frac{\|\boldsymbol{P}\|_F^2}{2} \leq \mathrm{Ov}(\boldsymbol{P}) \cdot \mathrm{Vol}(\boldsymbol{P}) \leq \|\boldsymbol{P}\|_F^2$.

Proof. Let $I[(i,j) \in G]$ be the $0,1$ indicator random variable that an edge (i,j) appears in the graph G. $\mathrm{Ov}(\boldsymbol{P}) \cdot \mathrm{Vol}(\boldsymbol{P}) = \mathbb{E}_{G_1,G_2 \sim \mathcal{G}(\boldsymbol{P})} |E(G_1) \cap E(G_2)|$. By linearity of expectation and the independence of G_1 and G_2 we have,

$$\mathrm{Ov}(\boldsymbol{P}) \cdot \mathrm{Vol}(\boldsymbol{P}) = \mathbb{E}_{G_1,G_2 \sim \mathcal{G}(\boldsymbol{P})} \sum_{i \le j} I[(i,j) \in G_1] \cdot I[(i,j) \in G_2] = \sum_{i \le j} \boldsymbol{P}_{ij}^2.$$

The bound follows since $\boldsymbol{P}$ is symmetric. Note that the lower bound $\frac{\|\boldsymbol{P}\|_F^2}{2} \le \mathrm{Ov}(\boldsymbol{P}) \cdot \mathrm{Vol}(\boldsymbol{P})$ is an equality if $\boldsymbol{P}$ is 0 on the diagonal – i.e., there is no probability of self loops. $\square$

4.1.4.1 Triangles

Lemma 4.1.2 connects $\mathrm{Ov}(\boldsymbol{P}) \cdot \mathrm{Vol}(\boldsymbol{P})$ to $\|\boldsymbol{P}\|_F^2$ and in turn the eigenvalue spectrum of $\boldsymbol{P}$ since $\|\boldsymbol{P}\|_F^2 = \sum_{i=1}^n \lambda_i(\boldsymbol{P})^2$, where $\lambda_1(\boldsymbol{P}), \ldots, \lambda_n(\boldsymbol{P}) \in \mathbb{R}$ are the eigenvalues of $\boldsymbol{P}$. The expected number of triangles in $G \sim \mathcal{G}(\boldsymbol{P})$ can be written in terms of this spectrum as well, allowing us to relate overlap to this expected triangle count, and prove our main theorem (Theorem 4.1.1), restated below.

Theorem 4.1.1. *For a graph G, let $\Delta(G)$ denote the number of triangles in G. Consider symmetric $\boldsymbol{P} \in [0,1]^{n \times n}$.*

$$\mathbb{E}_{G \sim \mathcal{G}(\boldsymbol{P})}\left[\Delta(G)\right] \le \frac{\sqrt{2}}{3} \cdot \mathrm{Ov}(\boldsymbol{P})^{3/2} \cdot \mathrm{Vol}(\boldsymbol{P})^{3/2}.$$

Proof. By linearity of expectation,

$$\begin{aligned} \mathbb{E}_{G \sim \mathcal{G}(\boldsymbol{P})}\left[\Delta(G)\right] &= \frac{1}{6} \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n \Pr\left[(i,j) \in E(G) \cap (j,k) \in E(G) \cap (k,i) \in E(G)\right] \\ &= \frac{1}{6} \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n \boldsymbol{P}_{ij} \boldsymbol{P}_{jk} \boldsymbol{P}_{ki} = \frac{1}{6} \mathrm{tr}(\boldsymbol{P}^3) = \frac{1}{6} \sum_{i=1}^n \lambda_i(\boldsymbol{P})^3. \qquad (4.1) \end{aligned}$$

Letting $\lambda_1(\boldsymbol{P})$ denote the largest magnitude eigenvalue of $\boldsymbol{P}$, we can in turn bound

$$\operatorname{tr}(\boldsymbol{P}^3) \leq |\lambda_1(\boldsymbol{P})| \cdot \sum_{i=1}^{n} \lambda_i(\boldsymbol{P})^2 = |\lambda_1(\boldsymbol{P})| \cdot \|\boldsymbol{P}\|_F^2 .$$

Since $|\lambda_1(\boldsymbol{P})| \leq \|\boldsymbol{P}\|_F$, this gives via Lemma 4.1.2

$$\operatorname{tr}(\boldsymbol{P}^3) \leq \|\boldsymbol{P}\|_F^3 \leq 2\sqrt{2} \cdot \operatorname{Ov}(\boldsymbol{P})^{3/2} \cdot \operatorname{Vol}(\boldsymbol{P})^{3/2}.$$

Combining this bound with Equation 4.1 completes the theorem. $\square$

The bound of Theorem 4.1.1 is tight up to constants, for any possible value of $\operatorname{Ov}(\boldsymbol{P})$. The tight example is when $\boldsymbol{P}$ is simply an Erdös-Rényi graph.

Theorem 4.1.3 (Tightness of Expected Triangle Bound)**.** *For any $\gamma \in (0,1]$, there exists a symmetric $\boldsymbol{P} \in [0,1]^{n \times n}$ with $\operatorname{Ov}(\boldsymbol{P}) = \gamma$ and $\mathbb{E}_{G \sim \mathcal{G}(\boldsymbol{P})}[\Delta(G)] = \Theta(\gamma^{3/2} \cdot \operatorname{Vol}(\boldsymbol{P})^{3/2})$.*

Proof. Let $\boldsymbol{P}_{ij} = \gamma$ for all $i \neq j$. We have $\operatorname{Vol}(\boldsymbol{P}) = \gamma \cdot \binom{n}{2}$ and $\operatorname{Ov}(\boldsymbol{P}) \cdot \operatorname{Vol}(\boldsymbol{P}) = \gamma^2 \cdot \binom{n}{2}$ Thus, $\operatorname{Ov}(\boldsymbol{P}) = \gamma$. Further, by linearity of expectation,

$$\mathbb{E}_{G \sim \mathcal{G}(\boldsymbol{P})}[\Delta(G)] = \gamma^3 \cdot \binom{n}{3} = \Theta(\gamma^3 \cdot n^3) = \Theta(\gamma^{3/2} \cdot \operatorname{Vol}(\boldsymbol{P})^{3/2}). \qquad \square$$

We note that another example when Theorem 4.1.1 is tight is when $\boldsymbol{P}$ is a union of a fixed clique on $\Theta(\gamma \cdot n)$ nodes and an Erdös-Rényi graph with connection probability $1/n$ on the rest of the nodes.

4.1.4.2 Squares and Other k-cycles

We can extend Theorem 4.1.1 to bound the expected number of k-cycles in $G \sim \mathcal{G}(\boldsymbol{P})$ in terms of $\operatorname{Ov}(\boldsymbol{P})$.

Theorem 4.1.4 (Bound on Expected k-cycles)**.** *For a graph G, let $C_k(G)$ denote the number of k-cycles in G. Consider symmetric $\boldsymbol{P} \in [0,1]^{n\times n}$.*

$$\mathbb{E}_{G\sim\mathcal{G}(\boldsymbol{P})}\left[C_k(G)\right] \le \frac{2^{k/2}}{2k}\cdot \mathrm{Ov}(\boldsymbol{P})^{k/2}\cdot \mathrm{Vol}(\boldsymbol{P})^{k/2}.$$

Proof. For notational simplicity, we focus on $k = 4$. The proof directly extends to general k. $C_4(G)$ is the number of non-backtracking 4-cycles in G (i.e. squares), which can be written as

$$\mathbb{E}_{G\sim\mathcal{G}(\boldsymbol{P})}\left[C_4(G)\right] = \frac{1}{8}\cdot\sum_{i=1}^{n}\sum_{j\in[n]\setminus i}\sum_{k\in[n]\setminus\{i,j\}}\sum_{\ell\in[n]\setminus\{i,j,k\}} \boldsymbol{P}_{ij}\boldsymbol{P}_{jk}P_{k\ell}P_{\ell i}.$$

The $1/8$ factor accounts for the fact that in the sum, each square is counted 8 times – once for each potential starting vector i and once of each direction it may be traversed. For general k-cycles this factor would be $\frac{1}{2k}$. We then can bound

$$\mathbb{E}_{G\sim\mathcal{G}(\boldsymbol{P})}\left[C_4(G)\right] \le \frac{1}{8}\cdot\sum_{i\in[n]}\sum_{j\in[n]}\sum_{k\in[n]}\sum_{\ell\in[n]} \boldsymbol{P}_{ij}\boldsymbol{P}_{jk}P_{k\ell}P_{\ell i} = \frac{1}{8}\cdot\mathrm{tr}(P^4).$$

For general k-cycles this bound would be $\mathbb{E}_{G\sim\mathcal{G}(\boldsymbol{P})}\left[C_k(G)\right] \le \frac{1}{2k}\mathrm{tr}(P^k)$. This in turn gives

$$\mathbb{E}_{G\sim\mathcal{G}(\boldsymbol{P})}\left[C_k(G)\right] \le \frac{1}{2k}\cdot|\lambda_1(\boldsymbol{P})|^{k-2}\cdot\|\boldsymbol{P}\|_F^2 \le \frac{1}{2k}\,\|\boldsymbol{P}\|_F^k \le \frac{2^{k/2}}{2k}\mathrm{Ov}(\boldsymbol{P})^{k/2}\cdot\mathrm{Vol}(\boldsymbol{P})^{k/2},$$

where the last bound follows from Lemma 4.1.2. This completes the theorem. □

It is not hard to see that Theorem 4.1.4 is also tight up to a constant depending on k for any overlap $\gamma \in (0,1]$, also for an Erdös-Rényi graph with connection probability γ.

Theorem 4.1.5 (Tightness of Expected k-cycle Bound)**.** *For any $\gamma \in (0,1]$, there exists $\boldsymbol{P} \in [0,1]^{n\times n}$ with $\mathrm{Ov}(\boldsymbol{P}) = \gamma$ and $\mathbb{E}_{G\sim\mathcal{G}(\boldsymbol{P})}[C_k(G)] = \Theta\left(\frac{\gamma^{k/2}\cdot\mathrm{Vol}(\boldsymbol{P})^{k/2}}{k!}\right)$.*

4.1.4.3 Clustering Coefficient

Theorem 4.1.1 shows that the expected number of triangles generated by an edge independent model is bounded in terms of the model's overlap. Intuitively, we thus expect that graphs generated by the edge independent model will have low global clustering coefficient, which is the fraction of wedges in the graph that are closed into triangles [195].

Definition 6 (Global Clustering Coefficient). *For a graph G with $\Delta(G)$ triangles, no self-loops, and node degrees $d_1, d_2, \ldots, d_n$, the global clustering coefficient is given by*

$$C(G) = \frac{3\Delta(G)}{\sum_{i=1}^{n} d_i(d_i - 1)}.$$

We extend Theorem 4.1.1 to give a bound on $E_{G\sim\mathcal{G}(\boldsymbol{P})}[C(G)]$ in terms of $\mathrm{Ov}(\boldsymbol{P})$. The proof is related, but more complex due to the $\sum_{i=1}^{n} d_i(d_i - 1)$ in the denominator of $C(G)$.

Theorem 4.1.6 (Bound on Expected Clustering Coefficient). *Consider symmetric $\boldsymbol{P} \in [0,1]^{n\times n}$ with zeros on the diagonal and with $\mathrm{Vol}(\boldsymbol{P}) \geq 2n$.*

$$E_{G\sim\mathcal{G}(\boldsymbol{P})}[C(G)] = O\left(\frac{\mathrm{Ov}(\boldsymbol{P})^{3/2}\cdot n}{\mathrm{Vol}(\boldsymbol{P})^{1/2}}\right).$$

Proof. By Theorem 4.1.1 we have $\mathbb{E}_{G\sim\mathcal{G}(\boldsymbol{P})}[3\Delta(G)] \leq \sqrt{2}\cdot\mathrm{Ov}(\boldsymbol{P})^{3/2}\cdot\mathrm{Vol}(\boldsymbol{P})^{3/2}$. We will show that with high probability, $\sum_{i=1}^{n} d_i(d_i-1) = \Omega(\mathrm{Vol}(\boldsymbol{P})^2/n)$, which will give the theorem. Note that $\mathbb{E}_{G\sim\mathcal{G}(\boldsymbol{P})}[\sum_{i=1}^{n} d_i] = \mathbb{E}_{G\sim\mathcal{G}(\boldsymbol{P})}[2|E(G)|] = 2\cdot\mathrm{Vol}(\boldsymbol{P})$. Thus, by a Bernstein bound, for large enough n since $\mathrm{Vol}(\boldsymbol{P}) \geq 2n$.

$$\Pr\left[\left|\sum_{i=1}^{n} d_i - 2\mathrm{Vol}(\boldsymbol{P})\right| \geq \mathrm{Vol}(\boldsymbol{P})/5\right] \leq 2\exp\left(-\frac{\mathrm{Vol}(\boldsymbol{P})^2/50}{\mathrm{Vol}(\boldsymbol{P}) + \mathrm{Vol}(\boldsymbol{P})/15}\right) \ll \frac{1}{n^2},$$

We can bound $\sum_{i=1}^n d_i^2 \geq \frac{\left(\sum_{i=1}^n d_i\right)^2}{n}$. Thus, with probability $\geq 1 - 1/n^2$,

$$\sum_{i=1}^n d_i(d_i - 1) \geq \frac{(8/5)^2 \cdot \text{Vol}(\boldsymbol{P})^2}{n} - \frac{12}{5}\text{Vol}(\boldsymbol{P}) \geq \frac{\text{Vol}(\boldsymbol{P})^2}{n},$$

where in the last step we use that $\text{Vol}(\boldsymbol{P}) \geq 2n$ and so $\frac{12}{5} \cdot \text{Vol}(\boldsymbol{P}) \leq \frac{6}{5} \cdot \frac{\text{Vol}(\boldsymbol{P})^2}{n}$. Combined with our bound on $\mathbb{E}_{G\sim\mathcal{G}(\boldsymbol{P})}[3\Delta(G)]$, and the fact that $C(G) \leq 1$ always, we have

$$E_{G\sim\mathcal{G}(\boldsymbol{P})}[C(G)] = O\left(\frac{\text{Ov}(\boldsymbol{P})^{3/2}\text{Vol}(\boldsymbol{P})^{3/2}}{\frac{\text{Vol}(\boldsymbol{P})^2}{n}} + \frac{1}{n^2}\right) = O\left(\frac{\text{Ov}(\boldsymbol{P})^{3/2} \cdot n}{\text{Vol}(\boldsymbol{P})^{1/2}}\right). \qquad \square$$

Thus, to have a constant clustering coefficient for a graph with $\mathcal{O}(n)$ edges in expectation, we need $\text{Ov}(\boldsymbol{P}) = \Omega(1/n^{1/3})$. Note that the requirement of $\text{Vol}(\boldsymbol{P}) \geq 2n$ is very mild – it means that the expected average degree is at least 1.

As with our triangle bound, Theorem 4.1.6 is tight when $\mathcal{G}(\boldsymbol{P})$ is just an Erdös-Rényi distribution.

Theorem 4.1.7 (Tightness of Expected Clustering Coefficient Bound)**.** *For any* $\gamma \in (0, 1]$*, there exists* $\boldsymbol{P} \in [0,1]^{n\times n}$ *with zeros on the diagonal,* $\text{Ov}(\boldsymbol{P}) \leq \gamma$ *and* $\mathbb{E}_{G\sim\mathcal{G}(\boldsymbol{P})}[C(G)] = \Theta\left(\frac{\gamma^{3/2}\cdot n}{\text{Vol}(\boldsymbol{P})^{1/2}}\right)$.

Proof. Let $\boldsymbol{P}_{ij} = \gamma$ for all $i \neq j$. We have $\text{Vol}(\boldsymbol{P}) = \gamma \cdot \binom{n}{2} = \Theta(\gamma n^2)$ and $\text{Ov}(\boldsymbol{P}) = \gamma$. Additionally, $\mathbb{E}[\Delta(G)] = \Theta(\gamma^3 \cdot n^3)$, and, if n is large enough with respect to γ, with very high probability, $\sum_{i=1}^n d_i(d_i - 1) \leq \sum_{i=1}^n d_i^2 = \mathcal{O}(\gamma^2 n^3)$. This gives:

$$\mathbb{E}_{G\sim\mathcal{G}(\boldsymbol{P})}[C(G)] = \Theta(\gamma) = \Theta\left(\frac{\gamma^{3/2} \cdot n}{\gamma^{1/2} \cdot n}\right) = \Theta\left(\frac{\gamma^{3/2} \cdot n}{\text{Vol}(\boldsymbol{P})^{1/2}}\right). \qquad \square$$

4.1.5 Baseline Edge Independent Models

We now shift from proving theoretical limitations of edge independent models to empirically evaluating the tradeoff between overlap and performance for a number of particular models. Given an input adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n\times n}$, these generative models produce a $\boldsymbol{P} \in [0,1]^{n\times n}$, samples from which should match various graph statistics of $\boldsymbol{A}$, such as the triangle count, clustering coefficient, and assortativity. At the same time, $\boldsymbol{P}$ should ideally have lower overlap so that the model does not just memorize the original graph. We propose two simple generative models as baselines to more complicated existing models – in both the level of overlap is easily tuned. Our first baseline, the *odds product model*, is based on just matching the degree sequence of $\boldsymbol{A}$; more simple still, the second baseline computes $\boldsymbol{P}$ as a linear function of $\boldsymbol{A}$, just matching its volume.

Odds product model. In this model, each node is assigned a logit $\ell \in \mathbb{R}$, and the probability of adding an edge between nodes i and j is $\boldsymbol{P}_{ij} = \sigma(\ell_i + \ell_j)$, where σ is the logistic function. We fit the model by finding a vector $\boldsymbol{\ell} \in \mathbb{R}^n$ of logits, with one logit for each node, such that the reconstructed network has the same expected degrees (i.e., row and column sums) as the original graph. We note that this model can be seen as a special case of the MaxEnt [44] and inner-product [118, 81, 82] models. In the context of directed graphs, $\boldsymbol{\ell}_i$ has been called the expansiveness or popularity of node i [72].

For adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n\times n}$, we denote its degree sequence by $\boldsymbol{d} = A\mathbf{1} \in \mathbb{R}^n$, where $\mathbf{1}$ is the all-ones vector of length n. Similarly, the degree sequence of the model is $\hat{\boldsymbol{d}} = P\mathbf{1}$. We pose fitting the model as a root-finding problem: we seek $\boldsymbol{\ell} \in \mathbb{R}^n$ such that the degree errors are zero, that is, $\hat{\boldsymbol{d}} - \boldsymbol{d} = \mathbf{0}$. We use the multivariate Newton-Raphson method to solve this root-finding problem. To apply Newton-Raphson, we need the Jacobian matrix J of derivatives of the degree errors with respect to the entries of $\boldsymbol{\ell}$. Since $\boldsymbol{d}$ does not vary with $\boldsymbol{\ell}$, these derivatives are exactly $\frac{\partial \hat{\boldsymbol{d}}_i}{\partial \boldsymbol{\ell}_j}$. Letting δ_{ij} be 1 if $i = j$ and 0 otherwise (i.e. the Kronecker delta),

$$
\begin{aligned}
\tfrac{\partial \hat{\boldsymbol{d}}_i}{\partial \boldsymbol{\ell}_j} &= \tfrac{\partial}{\partial \boldsymbol{\ell}_j} \sum\nolimits_{k\in[n]} \boldsymbol{P}_{ik} \\
&= \tfrac{\partial}{\partial \boldsymbol{\ell}_j} \sum\nolimits_{k\in[n]} \sigma(\boldsymbol{\ell}_i + \boldsymbol{\ell}_k) \\
&= \tfrac{\partial}{\partial \boldsymbol{\ell}_j} \sigma(\boldsymbol{\ell}_i + \boldsymbol{\ell}_j) + \delta_{ij} \sum\nolimits_{k\in[n]} \tfrac{\partial}{\partial \boldsymbol{\ell}_i} \sigma(\boldsymbol{\ell}_i + \boldsymbol{\ell}_k) \\
&= \sigma(\boldsymbol{\ell}_i + \boldsymbol{\ell}_j)\,(1 - \sigma(\boldsymbol{\ell}_i + \boldsymbol{\ell}_j)) + \delta_{ij} \sum\nolimits_{k\in[n]} \sigma(\boldsymbol{\ell}_i + \boldsymbol{\ell}_k)\,(1 - \sigma(\boldsymbol{\ell}_i + \boldsymbol{\ell}_k)) \\
&= \boldsymbol{P}_{ij}\,(1 - \boldsymbol{P}_{ij}) + \delta_{ij} \sum\nolimits_{k\in[n]} \boldsymbol{P}_{ik}\,(1 - \boldsymbol{P}_{ik})\,.
\end{aligned}
$$

In Algorithm 7, we provide pseudocode for computing the Jacobian matrix J and for implementing Newton-Raphson method to compute $\boldsymbol{P}$. We do not have a proof that Algorithm 7 always converges and produces $\boldsymbol{\ell}$ which exactly reproduces in the input degree sequence. However, the algorithm converged on all test cases, and proving that it always converges would be an interesting future direction.

Algorithm 7 Fitting the odds product model

```
input graphical degree sequence $\boldsymbol{d} \in \mathbb{R}^n$, error threshold $\epsilon$
output symmetric matrix $\boldsymbol{P} \in (0,1)^{n\times n}$ with row/column sums approximately $\boldsymbol{d}$
 1: $\boldsymbol{\ell} \leftarrow \mathbf{0}$        ▷ $\boldsymbol{\ell} \in \mathbb{R}^n$ is the vector of logits, initialized to all zeros
 2: $\boldsymbol{P} \leftarrow \sigma\left(\boldsymbol{\ell}\mathbf{1}^\top + \mathbf{1}\boldsymbol{\ell}^\top\right)$        ▷ $\sigma$ is the logistic function applied entrywise, and
                                          $\mathbf{1}$ is the all-ones column vector of length $n$
 3: $\tilde{\boldsymbol{d}} \leftarrow P\mathbf{1}$        ▷ degree sequence of $\boldsymbol{P}$
 4: while $\left\|\tilde{\boldsymbol{d}} - \boldsymbol{d}\right\|_2 > \epsilon$ do
 5:     $\boldsymbol{B} \leftarrow \boldsymbol{P} \circ \left(\mathbf{1}\mathbf{1}^\top - \boldsymbol{P}\right)$        ▷ $\circ$ is an entrywise product
 6:     $\boldsymbol{J} \leftarrow \boldsymbol{B} + \operatorname{diag}(\boldsymbol{B}\mathbf{1})$
 7:     $\boldsymbol{\ell} \leftarrow \boldsymbol{\ell} - J^{-1}\left(\tilde{\boldsymbol{d}} - \boldsymbol{d}\right)$        ▷ rather than inverting $\boldsymbol{J}$, we solve this linear system
 8:     $\boldsymbol{P} \leftarrow \sigma\left(\boldsymbol{\ell}\mathbf{1}^\top + \mathbf{1}\boldsymbol{\ell}^\top\right)$
 9:     $\tilde{\boldsymbol{d}} \leftarrow \boldsymbol{P}\mathbf{1}$
10: return $\boldsymbol{P}$
```

Our odds product model can be viewed as a variant of the Chung-Lu configuration model [39], which is also based on degree sequence matching. However, our model comes without a certain restriction on the maximum degree: in Chung-Lu, it is assumed that the degrees of all nodes are bounded above by the square root of the volume of the graph, that is, $\boldsymbol{d}_i \leq \sqrt{\mathrm{Vol}(\boldsymbol{A})}$ for all nodes i. Given this restriction, each node is assigned a weight $\boldsymbol{w}_i = \boldsymbol{d}_i/\sqrt{\mathrm{Vol}(\boldsymbol{A})}$, and the probability of adding edge (i, j) is $\boldsymbol{P}_{ij} = \boldsymbol{w}_i \boldsymbol{w}_j$. Since the weights are all in $[0, 1]$, they can be interpreted as probabilities, and the probability of adding an edge between two nodes is the product of the two nodes' probabilities.

Our odds product model works similarly, but instead of a probability, for each node, there is an associated odds, that is, a value in $(0, \infty)$, and the odds of adding an edge between two nodes is the product of the two nodes' odds. There is a one-to-one-to-one relationship between probability $p \in [0, 1]$, odds $o = \frac{p}{1-p} \in [0, \infty)$, and logit $\ell = \ln(o) \in (-\infty, +\infty)$. We outlined above how our model is based on adding logits associated with each node; since the odds is the exponentiation of the logit, the model can equally be viewed as multiplying odds associated with nodes.

Varying overlap in the odds product model. We propose a simple method to control the trade-off between overlap and accuracy in matching the input graph statistics in the odds product model. Given the original adjacency matrix $\boldsymbol{A}$ and the $\boldsymbol{P}$ generated by the odds product model to match the degree sequence of $\boldsymbol{A}$, we use a convex combination of $\boldsymbol{P}$ and $\boldsymbol{A}$. That is, we use $\tilde{\boldsymbol{P}} = (1 - \omega)\boldsymbol{P} + \omega\boldsymbol{A}$, where $0 \leq \omega \leq 1$. As ω increases to 1, $\tilde{\boldsymbol{P}}$ approaches a model which returns the original graph with high certainty; hence high ω produce $\tilde{\boldsymbol{P}}$ with high overlap which closely match graph statistics, while low ω produce $\tilde{\boldsymbol{P}}$ with lower overlap which may diverge from $\boldsymbol{A}$ in some statistics. Note that since $\tilde{\boldsymbol{P}}$ is a convex combination of adjacency matrices with the expected degree sequence of $\boldsymbol{A}$, $\tilde{\boldsymbol{P}}$ also has the same expected degree sequence regardless of the value of ω.

Linear model. As an even simpler baseline, we also propose and evaluate the following model: we produce an Erdös-Rényi model $\boldsymbol{P}$ with the same expected volume as the original graph $\boldsymbol{A}$, then return a convex combination $\tilde{\boldsymbol{P}}$ of $\boldsymbol{P}$ and $\boldsymbol{A}$. In particular, each entry of $\boldsymbol{P}$ is $\mathrm{Vol}(\boldsymbol{A})/n^2$, and, as with the odds product model, $\tilde{\boldsymbol{P}} = (1 - \omega)\boldsymbol{P} + \omega\boldsymbol{A}$, where $0 \leq \omega \leq 1$. This model can alternatively be seen as producing a $\tilde{\boldsymbol{P}}$ by lowering each entry of $\boldsymbol{A}$ which is 1 to some probability α, and raising each entry of $\boldsymbol{A}$ which is 0 to a probability β, with $\alpha \geq \beta$, such that the volume is conserved.

4.1.6 Experimental Results

We now present our evaluations of different edge independent graph generative models in terms of the tradeoff achieved between overlap and performance in generating graphs with similar key statistics to an input network. These experiments highlight the strengths and limitations of each model, as well as the overall limitations of this class, as established by our theoretical bounds.

4.1.6.1 Methods

We compare our proposed models from Section 4.1.5 with a number of existing models described below

1. **CELL [151]** (Cross-Entropy Low-rank Logits) An alternative to the popular NetGAN method [20] which strips the proposed architecture of deep leaning components and achieves comparable performance in significantly less time, via a low-rank approximation approach. To control overlap, we follow the approach of the original paper, halting training once the generated graph exceeds a specified overlap threshold with the input graph. We set the rank parameter to a value that allows us to get up to 75% overlap (typical values are 16 and 32).

2. **TSVD** (Truncated Singular Value Decomposition) A classic spectral method which computes a rank-k approximation of the adjacency matrix using truncated SVD. As in [167], the resulting matrix is clipped to [0,1] to yield $\boldsymbol{P}$. Overlap is controlled by varying k.

3. **CCOP** (Convex Combination Odds Product) The odds product model as of Sec. 4.1.5 with overlap controlled by taking a convex combination of $\boldsymbol{P}$ and the input adjacency matrix $\boldsymbol{A}$.

4. **HDOP** (Highest Degree Odds Product) The odds product model, but with overlap controlled by fixing the edges adjacency to a certain number of the highest degree nodes.

5. **Linear** The convex combination between the input adjacency matrix and an Erdös-Rényi graph, as described in Sec. 4.1.5, with overlap controlled by varying the ω parameter.

CCOP, HDOP, and Linear all produce edge probability matrices $\boldsymbol{P}$ with the same volume, $\mathrm{Vol}(G)$, in expectation as the original adjacency matrix. For TSVD, letting $\boldsymbol{L}$ be the low-rank approximation of the adjacency matrix, we learn a scalar shift parameter σ using Newton's method such that $\boldsymbol{P} = \max(0, \min(1, \boldsymbol{L}+\sigma))$ has volume $\mathrm{Vol}(G)$. We then generate new networks from the edge independent distribution $\mathcal{G}(\boldsymbol{P})$ (Definition 4). For CELL, we follow the authors' approach of generating $\mathrm{Vol}(G)$ edges without replacement - an edge (i, j) is added with probability proportional to $\boldsymbol{P}_{ij}$).

We sample 5 networks from each distribution and report the average for every statistic.

4.1.6.2 Datasets and network statistics

For evaluation, we use the following seven popular datasets with varied structure, from triangle-rich social networks to planar road networks:

1. PolBlogs: A collection of political blogs and the links between them.

2. Citeseer: A collection of papers from six scientific categories and the citations among them.

Table 4.1. Network statistics for experiments in Section 4.1.

Dataset	Nodes	Edges	Triangles
PolBlogs [3]	1,222	33,428	101,043
Citeseer [166]	2,110	7,336	1,083
Cora [166]	2,485	10,138	1,558
Road-Minnesota [155]	2,640	6,604	53
Web-Edu [69]	3,031	12,948	10,058
PPI [177]	3,852	75,682	91,461
Facebook [107]	4,039	176,468	1,612,010

3. Cora: A collection of scientific publications and the citations among them.

4. Road-Minnesota: A road network from the state of Minnesota. Each intersection is a node.

5. Web-Edu: A web-graph drawn from educational institutions.

6. PPI: A subgraph of the PPI network for Homo Sapiens. Vertices represent proteins and edges represent interactions.

7. Facebook: A union of ego networks of Facebook users.

See Table 4.1 for statistics about the networks. We treat all networks as binary, in that we set all non-zero weights to 1, and undirected, in that if edge (i, j) appears in the network, we also include edge (j, i) . Also, we keep only the largest connected component of each network.

We evaluate performance in matching the following key network statistics:

1. Pearson correlation of the degree sequences of the input and the generated network.

2. Maximum degree over all nodes.

3. Exponent of a power-law distribution fit to the degree sequence.

4. Assortativity, a measure that captures the preference of nodes to attach to others with similar degree (ranging from -1 to 1).

5. Pearson correlation of the triangle sequence (number of triangles a node participates in).

6. Total triangle count (analyzed theoretically in Theorem 4.1.1).

7. Global clustering coefficient (defined in Definition 6 and analyzed theoretically in Theorem 4.1.6).

8. Characteristic path length (average path length between any two nodes).

4.1.6.3 Results

The theoretical results from Section 4.1.4 highlight a key weakness of edge independent generative models: they cannot generate many triangles (or other higher-order locally dense areas), without having high overlap and thus not generating a diversity of graphs. We observe that these theoretical findings hold in practice – generally speaking, all models tested tend to significantly underestimate triangle count and global clustering coefficient, as well as inaccurately match the triangle degree sequence, when overlap is low. See Figures 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, and 4.7 for results on the tested networks. As overlap increases, performance in reconstructing these metrics does as well, as expected.

All methods are able to capture certain network characteristics accurately, even at low overlap. Even for a relatively small overlap (less than 0.2), the CCOP and HDOP methods accurately capture the degree sequences of the true networks (as they are designed to do). These methods, especially HDOP which fixes edges from high degree nodes, often outperform more sophisticated methods like CELL in terms of triangle density and triangle degree sequence correlation. On the other hand, CELL seems to do a somewhat better job capturing global features, like the characteristic

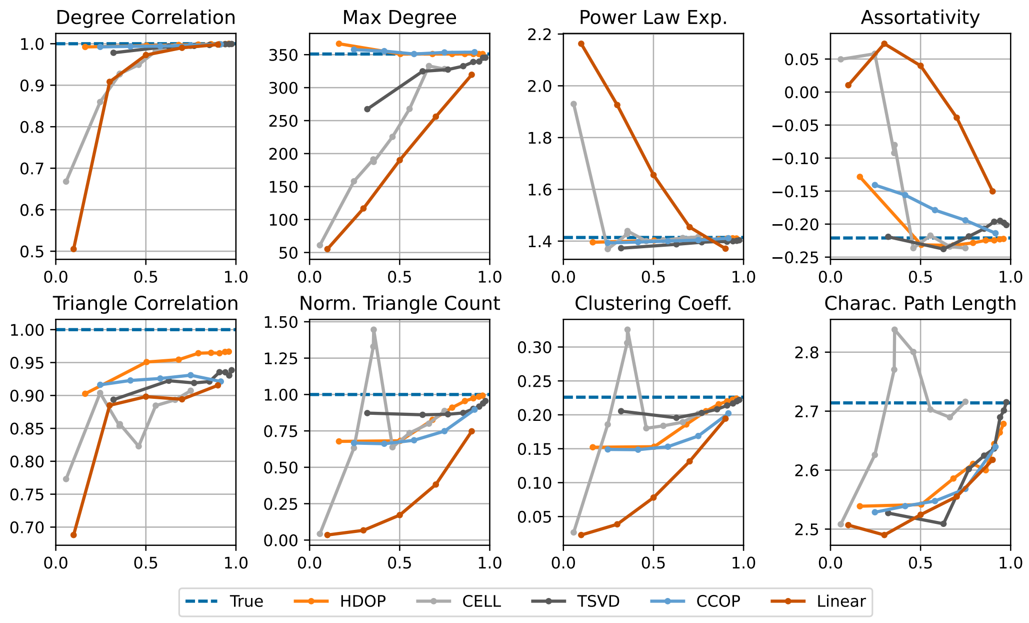


Figure 4.1. Graph statistics fidelity vs overlap with edge independent models on PolBlogs.

path length. TSVD provides a fair compromise – it performs better than CELL in terms of degree sequence and triangle counts, but worse in terms of characteristic path length. In general, it is the method that gives the best results when the overlap is extremely small, appearing to be less sensitive to the variation in overlap.

Broadly speaking, all methods do reasonably well in matching the power-law degree distribution of the networks, even when they do not match the actual degree sequence closely. With the exception of Web-Edu, they tend to underestimate the characteristic path length. This is perhaps not surprising due to the independent random edge connections, but it would be interesting to understand more theoretically.

Code for reproducing results. Code is available at `https://github.com/konsotirop/edge_independent_models`. Our implementation of the methods we introduce is written in Python and uses the NumPy [79] and SciPy [191] packages. Additionally, to calculate the various graph metrics, we use the following packages: powerlaw [14] and MACE (MAximal Clique Enumerator) [180].

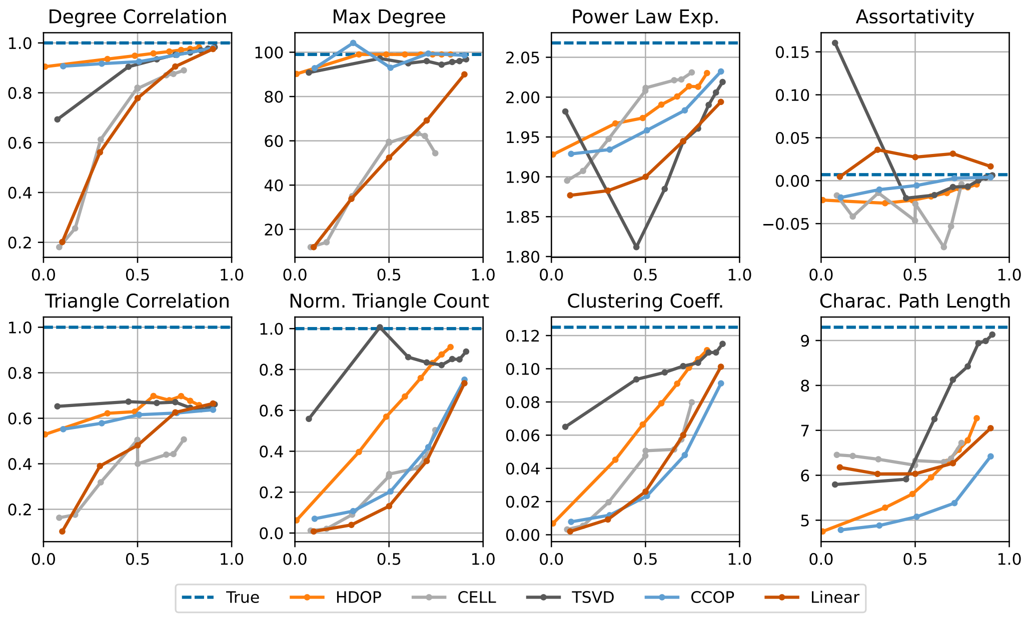


Figure 4.2. Graph statistics fidelity vs overlap with edge independent models on Citeseer.

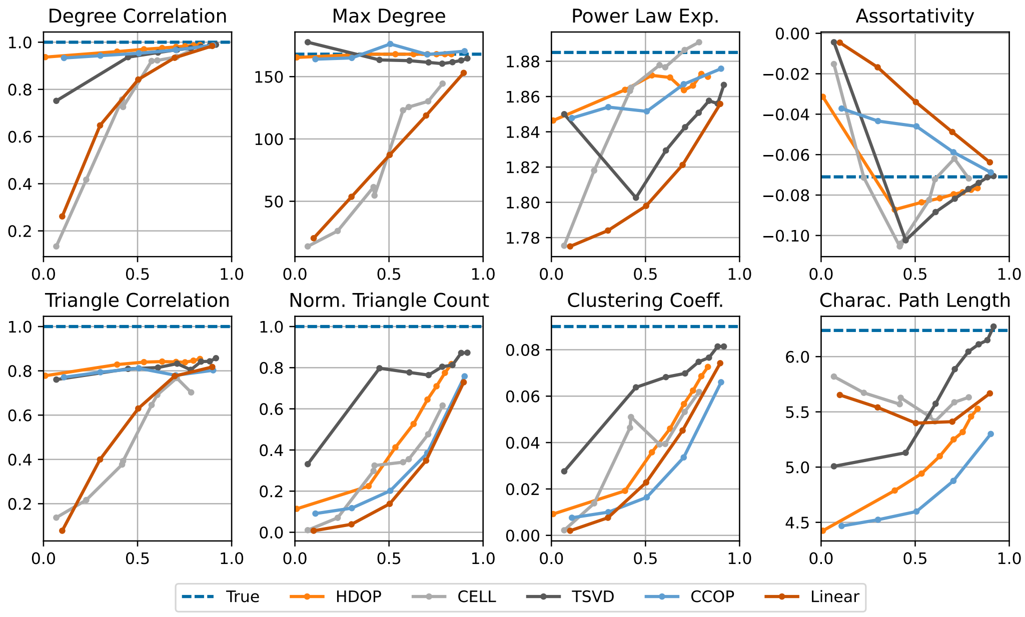


Figure 4.3. Graph statistics fidelity vs overlap with edge independent models on Cora.

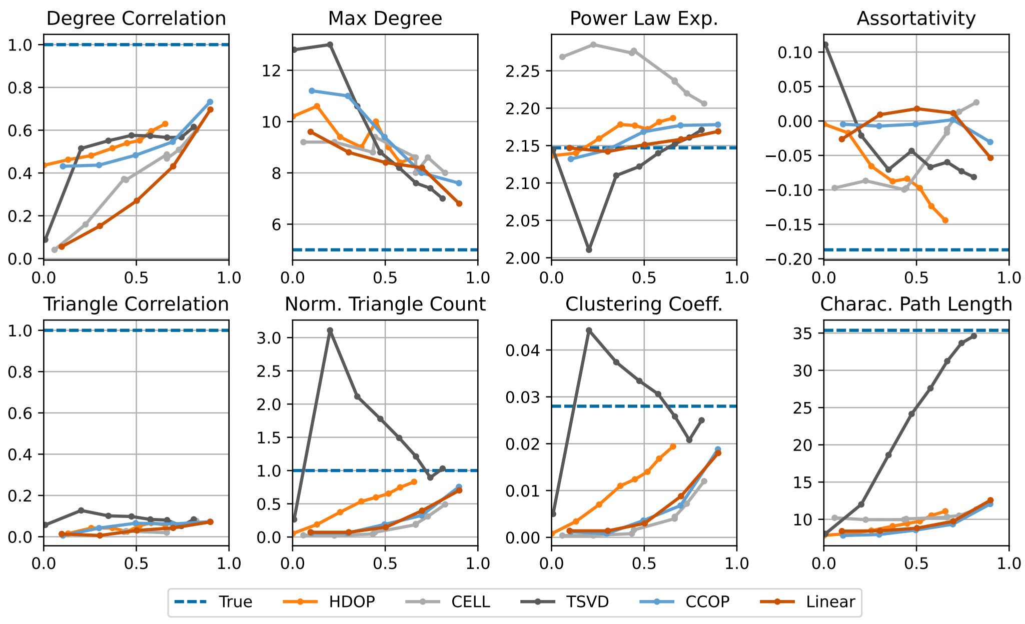


Figure 4.4. Graph statistics fidelity vs overlap with edge independent models on Road-Minnesota.

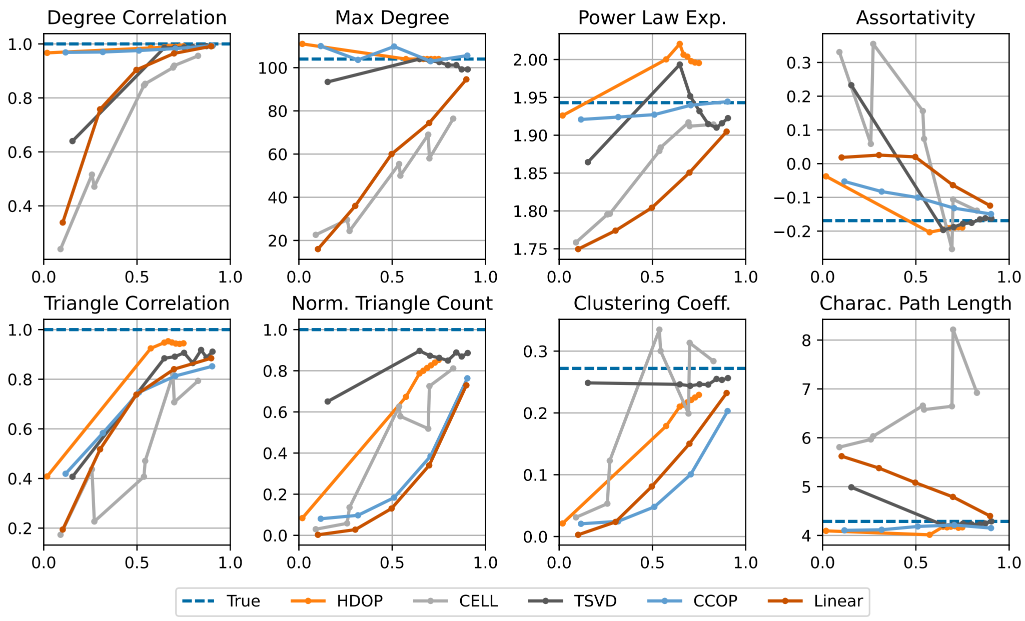


Figure 4.5. Graph statistics fidelity vs overlap with edge independent models on Web-Edu.

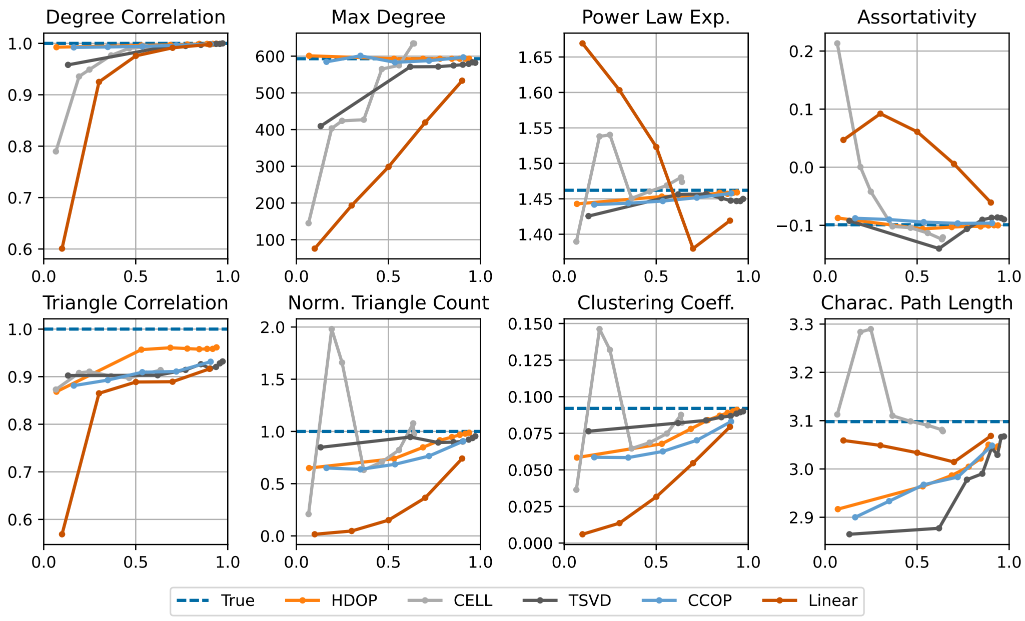


Figure 4.6. Graph statistics fidelity vs overlap with edge independent models on PPI

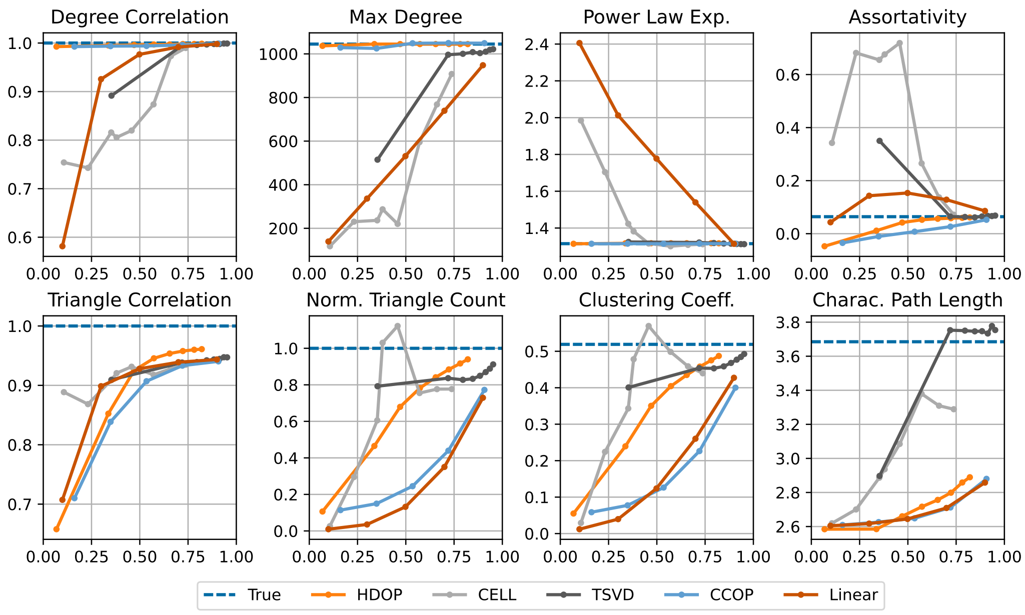


Figure 4.7. Graph statistics fidelity vs overlap with edge independent models on Facebook.

4.1.7 Conclusion

Our theoretical results prove limitations on the ability of any edge independent graph generative model to produce networks that match the high triangle densities of real-world graphs, while still generating a diverse set of networks, with low model overlap. These results match empirical findings that popular edge independent models indeed systematically underestimate triangle density, clustering coefficient, and related measures. Despite the popularity of edge independent models, many non-independent models, such as graph RNNs [205] have been proposed. This section leaves open the study of the representative power and limitations of such models, giving general theoretical results that provide a foundation for the study of graph generative models. We return to this direction in Section 4.2.

4.2 On the Role of Edge Dependency in Random Graph Models

In Section 4.1, we introduced the notion of *overlap*, to quantify the diversity of the graphs generated by a given model. Intuitively, useful models that produce diverse output graphs have bounded overlap. Further, without the requirement of bounded overlap, it is trivial to generate graphs that have the same statistical properties as a given input graph, by simply memorizing that graph and outputting it with probability 1. Formally, we showed a trade-off between the number of triangles that an edge independent model can generate and the model overlap.

The notion of overlap will play a central role in this section. In particular, we introduce the following definition, which generalizes the concept of overlap beyond edge independent models. For simplicity, we focus on node-labeled, unweighted, and undirected graphs.

Definition 7 (Overlap). *Suppose $\mathcal{A}$ is a distribution over binary adjacency matrices of undirected unweighted graphs on n nodes without self-loops. Let the 'volume' of $\mathcal{A}$ be the expected number of edges in a sample:*

$$\mathrm{Vol}(\mathcal{A}) \coloneqq \mathbb{E}_{\mathbf{A}\sim\mathcal{A}}\left[\sum\nolimits_{i<j} \mathbf{A}_{ij}\right].$$

The overlap of $\mathcal{A}$ is the expected number of shared edges between two samples, normalized by volume:

$$\mathrm{Ov}(\mathcal{A}) \coloneqq \frac{\mathbb{E}_{\mathbf{A},\mathbf{A}'\sim\mathcal{A}}\left[\sum_{i<j} \mathbf{A}_{ij}\cdot \mathbf{A'}_{ij}\right]}{\mathrm{Vol}(\mathcal{A})}.$$

Our Contributions. Our main contributions in this section are summarized as follows:

- We advocate for a new evaluation framework of generative models that includes the *overlap*, to require that models generate graphs that are both accurate and edge-diverse.
- We propose a hierarchy of graph generative models with three levels of complexity: edge independent, node independent, and fully dependent models. These levels capture many of today's most popular methods — see Table 4.2.
- We prove theoretical bounds on the number of triangles and other short-length cycles that can be generated by each level of our hierarchy, as a function of the model overlap. Table 4.2 summarizes our bounds. Furthermore, we provide instances that show that our bounds are (asymptotically) optimal. Intuitively, models with more dependencies can achieve higher triangle/motif counts for a given level of overlap. Our bounds here extend those of Section 4.1, which considered only edge independent models (the lowest level of our hierarchy).
- We propose new generative models for each of the three levels that are based on dense subgraph discovery via maximal clique detection [67].

- We evaluate our models and other popular models on real-world datasets focusing on the output quality and the overlap. We see that while several models can succeed in memorizing the input graph, they fail to match graph statistics at a low overlap.

Table 4.2. Summary of results for Section 4.2. The level of edge dependency in graph generative models inherently limits the range of graph statistics, such as triangle counts, that they can produce. Note that $\text{Ov}(\mathcal{A}) \in [0, 1]$, so a higher power on $\text{Ov}(\mathcal{A})$ means a tighter bound on the number of triangles.

Model	Upper Bound on Δ/n^3	Examples
Edge Independent	$\text{Ov}(\mathcal{A})^3$	• Erdős–Rényi • SBM • NetGAN [20, 152]
Node Independent	$\text{Ov}(\mathcal{A})^{3/2}$	• Variational Graph Auto-Encoder (VGAE) [94]
Fully Dependent	$\text{Ov}(\mathcal{A})$	• GraphVAE [170] • GraphVAE-MM [208] • Exponential Random Graph Models (ERGMs) [62]

4.2.1 Hierarchy of Graph Generative Models

Our main conceptual contribution is a hierarchical categorization of graph generative models into three nested levels: edge independent, node independent, and fully dependent. For each level, we prove asymptotic upper bounds on triangle count with respect to overlap; we defer the proof of these bounds, as well as generalizations to bounds on k-cycles, to Section 4.1.4. Also for each level, we provide a simple characteristic family of graphs that achieves this upper bound on triangle counts, showing that the bound is asymptotically tight. We also discuss examples of models from prior work that fit into each level.

4.2.1.1 Edge Independent Model

We begin with the well-established class of edge independent models. This class of models includes many important models. Classical models that are edge independent include the Erdős-Rényi model and the stochastic block model (SBM) [1] as well as modern deep-learning based models such as NetGAN [20, 151], Graphite [76], and MolGAN [45] as they ultimately output a $\boldsymbol{P}$ matrix based on which edges are sampled independently.

Definition 8 (Edge Independent Graph Model). *An edge independent (EI) graph generative model is a distribution $\mathcal{A}$ over symmetric adjacency matrices $\boldsymbol{A} \in \{0,1\}^{n\times n}$, such that, for some symmetric $\boldsymbol{P} \in [0,1]^{n\times n}$, $\boldsymbol{A}_{ij} = \boldsymbol{A}_{ji} = 1$ with probability $\boldsymbol{P}_{ij}$ for all $i, j \in [n]$, otherwise $\boldsymbol{A}_{ij} = \boldsymbol{A}_{ji} = 0$. Furthermore, all entries (i.e., in the upper diagonal) of $\boldsymbol{A}$ are mutually independent.*

Any EI graph generative model satisfies the following theorem:

Theorem 4.2.1. *Any edge independent graph model $\mathcal{A}$ with overlap $\mathrm{Ov}(\mathcal{A})$ has $\mathcal{O}(n^3 \cdot \mathrm{Ov}(\mathcal{A})^3)$ triangles in expectation. That is, for a sample $\boldsymbol{A}$ drawn from $\mathcal{A}$, the number of triangles $\Delta(\boldsymbol{A})$ in $\boldsymbol{A}$ in expectation is*

$$\mathbb{E}[\Delta(\boldsymbol{A})] = \mathcal{O}(n^3 \cdot \mathrm{Ov}(\mathcal{A})^3) \tag{4.2}$$

In Section 4.1, we give a spectral proof of the upper bound Equation 4.2. Here we provide an alternative proof based on an elegant generalization of Shearer's Entropy Lemma [13, 41] due to [63]. Furthermore, we show that the upper bound is tight. Consider the $G(n,p)$ model. The expected volume is $\mathrm{Vol}(\mathcal{A}) = p \cdot \binom{n}{2}$, whereas the expected number of shared edges between two samples is $p^2 \cdot \binom{n}{2}$, yielding an overlap of p (see Definition 7). Furthermore, since any triangle is materialized with probability p^3, by the linearity of expectation there are $\mathcal{O}(n^3 \cdot p^3)$ triangles, which shows that inequality Equation 4.2 is tight.

4.2.1.2 Fully Dependent Model

We now move to the other end of the edge dependency spectrum, namely to models that allow for arbitrary edge dependencies. A classic fully dependent model is the ERGM [62, 35] as it can model arbitrary higher order dependencies.

Definition 9 (Fully Dependent Graph Model)**.** *A fully dependent graph generative model allows for any possible distribution $\mathcal{A}$ over symmetric adjacency matrices $\boldsymbol{A} \in \{0,1\}^{n\times n}$.*

Allowing for arbitrary edge dependencies enables us to have models with significantly more triangles as a function of the overlap:

Theorem 4.2.2. *For any fully-dependent model $\mathcal{A}$ with overlap $\mathrm{Ov}(\mathcal{A})$, the expected number of triangles is $\mathcal{O}(n^3 \cdot \mathrm{Ov}(\mathcal{A}))$.*

As in the case of EI models, there is a simple instance that shows that the bound is tight. Specifically, consider a model that outputs a complete graph with probability p and an empty graph otherwise. Each edge occurs with probability p, so by the same computation as for EI models, the overlap is p. As for the triangle count, there are no triangles when the graph is empty, but when it is complete, there are all $\binom{n}{3}$ possible triangles. Thus, the expected number of triangles is $\mathcal{O}(n^3 \cdot p) = \mathcal{O}(n^3 \cdot \mathrm{Ov}(\mathcal{A}))$, again showing that our bound is tight.

At a high level, fully dependent graph generative models often arise when methods sample a graph-level embedding, then produce a graph sample from this embedding, allowing for arbitrary dependencies between edges in the sample. A specific example is the graph variational auto-encoder (GraphVAE) [170], in which decoding involves sampling a single graph-level embedding $\boldsymbol{x_G} \in \mathbb{R}^k$, and the presence of each edge is an independent Bernoulli random variable with a parameter that is some function f_{ij} of $\boldsymbol{x}$: $\boldsymbol{A}_{ij} = \mathrm{Bernoulli}(f_{ij}(\boldsymbol{x_G}))$. In particular, these functions are encoded by a fully-connected neural network. Assuming these f_{ij} can closely approximate the sign function, with a 1-dimensional graph embedding ($k = 1$), this model can in fact match the triangle bound of Theorem 4.2.2 by simulating the tight instance described above (outputting a complete graph with probability p and the empty graph otherwise).

4.2.1.3 Node Independent Model

We have managed to identify a natural middle layer in the hierarchy, between the two extremes of EI and FD generative models. This level is built upon the common concept of node embeddings that are stochastic and generated independently for each node.

Definition 10 (Node Independent Graph Model)**.** *A node independent (NI) graph generative model is a distribution $\mathcal{A}$ over symmetric adjacency matrices $\boldsymbol{A} \in \{0,1\}^{n\times n}$, where, for some embedding space $\mathcal{E}$, some mutually independent random variables $\boldsymbol{x_1}, \ldots, \boldsymbol{x_n} \in \mathcal{E}$, and some symmetric function $e : \mathcal{E} \times \mathcal{E} \to [0,1]$, the entries of $\boldsymbol{A}$ are Bernoulli random variables $\boldsymbol{A}_{ij} = \text{Bernoulli}(e(\boldsymbol{x_i}, \boldsymbol{x_j}))$ that are mutually independent conditioned on $\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}$.*

Interestingly the triangle bound for this class of generative models lies in the middle of the EI and FD triangle bounds.

Theorem 4.2.3. *Any node independent graph model $\mathcal{A}$ with overlap $\mathrm{Ov}(\mathcal{A})$ has $\mathcal{O}(n^3 \cdot \mathrm{Ov}(\mathcal{A})^{3/2})$ triangles in expectation.*

The proof of Theorem 4.2.3 requires expressing the probability of edges and triangles appearing as integrals in the space of node embeddings. Then we apply a continuous version of Shearer's inequality. We show that the bound of Theorem 4.2.3 is tight. Generate a random graph by initially starting with an empty graph comprising n nodes. Subsequently, each node independently becomes "active" with a probability of $\sqrt{p}$, and any edges connecting active nodes are subsequently added into the graph. Note that for a given edge to be added, both of its endpoint nodes must be active, which occurs with probability p; this again yields an overlap of p as with the prior tight examples. For a triangle to be added, all three of its endpoint nodes must be active, which occurs with probability $p^{3/2}$. Hence the expected number of triangles is $\mathcal{O}(n^3 \cdot p^{3/2}) = \mathcal{O}(n^3 \cdot \mathrm{Ov}(\mathcal{A})^{3/2})$.

The random graph can be represented using embeddings based on the provided definition. Consider a 1-dimensional embedding $\boldsymbol{x} \in \mathbb{R}^{n\times 1}$. Let $\boldsymbol{x}_i = 1$ with probability $\sqrt{p}$ and otherwise let $\boldsymbol{x}_i = 0$, independently for each $i \in [n]$. These coin tosses are made independently for each $i \in [n]$. To capture the edges between nodes, we define $e(\boldsymbol{x}_i, \boldsymbol{x}_j)$ as 1 when both arguments are 1, and 0 otherwise. This embedding-based representation precisely implements the described random graph.

Node independent graph models most commonly arise when methods independently sample n node-level embeddings, then determine the presence of edges between two nodes based on some compatibility function of their embeddings. One notable example is the variational graph auto-encoder (VGAE) model [93]. In the decoding step of this model, a Gaussian-distributed node embedding is sampled independently for each node, and the presence of each possible edge is an independent Bernoulli random variable with a parameter that varies with the dot product of the corresponding embeddings as follows: $\boldsymbol{A}_{ij} = \mathrm{Bernoulli}(\sigma(\boldsymbol{x_i} \cdot \boldsymbol{x_j}))$. Thus, the VGAE model seamlessly fits into our node independent category.

4.2.2 Impossibility Results for Random Graph Models

We now prove the bounds on triangle count that were discussed in Section 4.2.1, then state generalized bounds for k-cycles. We start with some notions that will be useful for the proofs.

4.2.3 Theoretical Preliminaries

Inner product space formulation of overlap. We now state equivalent definitions for volume and overlap that may be more illuminating, and will be useful for proofs. We first make the following observation:

Observation 1 (Inner Product Space of Distributions over Vectors)**.** *Suppose $\mathcal{U}, \mathcal{V}$ are distributions over real-valued d-dimensional vectors. Then the following operation defines an inner product:*

$$\langle \mathcal{U}, \mathcal{V} \rangle \coloneqq \mathop{\mathbb{E}}_{\boldsymbol{u} \sim \mathcal{U}, \boldsymbol{v} \sim \mathcal{V}} [\boldsymbol{u} \cdot \boldsymbol{v}],$$

where $\boldsymbol{u} \cdot \boldsymbol{v}$ is the standard vector dot product.

For the remainder of this paper, we deal with adjacency matrices of undirected graphs on n nodes without self-loops, and distributions over such matrices. Dot products and inner products of these objects are taken over $\binom{n}{2}$-dimensional vectorizations of entries below the diagonal. Let $\mathcal{F}$ denote the distribution that returns the adjacency matrix $\boldsymbol{F}$ of a such a graph which is fully connected (i.e., has all possible edges) with probability 1. Then

$$\mathrm{Vol}(\mathcal{A}) = \langle \mathcal{A}, \mathcal{F} \rangle \quad \text{and} \quad \mathrm{Ov}(\mathcal{A}) = \frac{\langle \mathcal{A}, \mathcal{A} \rangle}{\mathrm{Vol}(\mathcal{A})}.$$

In addition to these expressions for overlap and volume, we will also continue to use $\mathcal{F}$ and $\boldsymbol{F}$ as defined above through the rest of this paper.

These expressions allow us to derive the following upper bound on volume in terms of overlap, which applies to any graph generative model:

Lemma 4.2.4. *For a graph generative model $\mathcal{A}$ with overlap $\mathrm{Ov}(\mathcal{A})$, the expected number of edges $\mathrm{Vol}(\mathcal{A})$ is at most $\binom{n}{2} \cdot \mathrm{Ov}(\mathcal{A})$.*

Proof. By the definition of overlap and Cauchy-Schwarz,

$$\mathrm{Ov}(\mathcal{A}) = \frac{\langle \mathcal{A}, \mathcal{A}\rangle}{\langle \mathcal{A}, \mathcal{F}\rangle} \geq \frac{\langle \mathcal{A}, \mathcal{F}\rangle^2}{\langle \mathcal{A}, \mathcal{F}\rangle \cdot \langle \mathcal{F}, \mathcal{F}\rangle} = \frac{\langle \mathcal{A}, \mathcal{F}\rangle}{\langle \mathcal{F}, \mathcal{F}\rangle} = \frac{\mathrm{Vol}(\mathcal{A})}{\langle \mathcal{F}, \mathcal{F}\rangle}.$$

Since $\langle \mathcal{F}, \mathcal{F}\rangle = \langle \boldsymbol{F}, \boldsymbol{F}\rangle = \binom{n}{2}$, rearranging yields

$$\mathrm{Vol}(\mathcal{A}) \leq \tbinom{n}{2} \cdot \mathrm{Ov}(\mathcal{A}). \qquad \square$$

4.2.4 Triangle Count

We now prove the upper bounds on expected triangle count for edge independent, node independent, and fully dependent models of $\mathcal{O}(n^3 \cdot \mathrm{Ov}(\mathcal{A})^3)$, $\mathcal{O}(n^3 \cdot \mathrm{Ov}(\mathcal{A})^{3/2})$, and $\mathcal{O}(n^3 \cdot \mathrm{Ov}(\mathcal{A}))$, respectively.

Edge independent. A proof for Theorem 4.2.1 based on expressing triangle count in terms of eigenvalues of $\boldsymbol{P}$ follows directly from Lemma 4.2.4 and results from Section 4.1, but we present a different proof based on the following variant of Cauchy-Schwarz from [63]:

$$\sum\nolimits_{ijk} a_{ij} b_{jk} c_{ki} \leq \sqrt{\sum\nolimits_{ij} a_{ij}^2 \sum\nolimits_{ij} b_{ij}^2 \sum\nolimits_{ij} c_{ij}^2}.$$

Proof of Theorem 4.2.1. Let $\boldsymbol{P}$ be the edge probability matrix of the edge independent model $\mathcal{A}$. Then, by the above inequality, for a sample $\boldsymbol{A}$ from $\mathcal{A}$,

$$\begin{aligned}\mathbb{E}[\Delta(\boldsymbol{A})] = \tfrac{1}{6} \sum\nolimits_{ijk} \boldsymbol{P}_{ij} \boldsymbol{P}_{jk} \boldsymbol{P}_{ki} \leq \tfrac{1}{6} \sqrt{\left(\sum\nolimits_{ij} \boldsymbol{P}_{ij}^2\right)^3} = \tfrac{1}{6} \left(2 \sum\nolimits_{i<j} \boldsymbol{P}_{ij}^2\right)^{3/2} \\ = \tfrac{1}{6} \left(2 \cdot \mathrm{Ov}(\mathcal{A}) \mathrm{Vol}(\mathcal{A})\right)^{3/2} = \tfrac{\sqrt{2}}{3} \cdot \mathrm{Ov}(\mathcal{A})^{3/2} \mathrm{Vol}(\mathcal{A})^{3/2}.\end{aligned}$$

Now, applying Lemma 4.2.4,

$$\mathbb{E}[\Delta(\boldsymbol{A})] \leq \tfrac{\sqrt{2}}{3} \tbinom{n}{2}^{3/2} \cdot \mathrm{Ov}(\mathcal{A})^3. \qquad \square$$

Fully dependent. We now prove the triangle bound for fully dependent (arbitrary) graph generative models, which follows from Lemma 4.2.4. Note that, given a random adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n\times n}$, the product $\boldsymbol{A}_{ij}\boldsymbol{A}_{jk}\boldsymbol{A}_{ik}$ is an indicator random variable for the existence of a triangle between nodes $i, j, k \in [n]$.

Proof of Theorem 4.2.2. Let $\boldsymbol{A}$ be a sample from $\mathcal{A}$. From Lemma 4.2.4, we have

$$\sum\nolimits_{i<j} \mathbb{E}[\boldsymbol{A}_{ij}] = \mathrm{Vol}(\mathcal{A}) \leq \tbinom{n}{2} \cdot \mathrm{Ov}(\mathcal{A}).$$

So, for the expected number of triangles in a sample, we have:

$$\begin{aligned}\mathbb{E}[\Delta(\boldsymbol{A})] &= \mathbb{E}\left[\sum\nolimits_{i<j<k} \boldsymbol{A}_{ij}\boldsymbol{A}_{jk}\boldsymbol{A}_{ik}\right] = \sum\nolimits_{i<j<k} \mathbb{E}\left[\boldsymbol{A}_{ij}\boldsymbol{A}_{jk}\boldsymbol{A}_{ik}\right] \\ &\leq \sum\nolimits_{i<j<k} \mathbb{E}[\boldsymbol{A}_{ij}] = \mathcal{O}(n) \sum\nolimits_{i<j} \mathbb{E}[\boldsymbol{A}_{ij}] \leq \mathcal{O}(n^3) \cdot \mathrm{Ov}(\mathcal{A}). \qquad \square\end{aligned}$$

Node independent. Before we prove Theorem 4.2.3, which is more involved than proofs for the prior theorems, we first establish the following lemma:

Lemma 4.2.5. *For a sample $\boldsymbol{A}$ from any node independent model on n nodes and any three nodes $i, j, k \in [n]$, the probability that i, j, k form a triangle is upper-bounded as follows:*

$$\mathbb{E}[\boldsymbol{A}_{ij}\boldsymbol{A}_{jk}\boldsymbol{A}_{ik}] \leq \sqrt{\mathbb{E}[\boldsymbol{A}_{ij}]\mathbb{E}[\boldsymbol{A}_{jk}]\mathbb{E}[\boldsymbol{A}_{ik}]}.$$

The concept for how we will prove this is essentially to express the probability of edges and triangles appearing as integrals in the space of node embeddings. After this, we can apply the following theorem from [63] (also proven by [59]), which can be seen as a continuous version of Shearer's inequality.

Theorem 4.2.6 ([63]). *Let X, Y, Z be three independent probability spaces and let*

$$f : X \times Y \to \mathbb{R}, \quad g : Y \times Z \to \mathbb{R}, \quad \textit{and} \quad h : Z \times X \to \mathbb{R}$$

be functions that are square-integrable with respect to the relevant product measures. Then

$$\int f(x,y)g(y,z)h(z,x)\,dx\,dy\,dz \leq \sqrt{\int f^2(x,y)\,dx\,dy \int g^2(y,z)\,dy\,dz \int h^2(z,x)\,dz\,dx}.$$

Given this theorem, we proceed with the proof of the lemma:

Proof. Fix a triplet of distinct nodes (i, j, k). Then, by assumption, the embeddings of these nodes Z_i, Z_j, Z_k are independent random variables. Let $\rho_{Z_i}, \rho_{Z_j}, \rho_{Z_k}$ be the corresponding PDFs of the respective nodes' embeddings. Recall that there exists a symmetric function $e : \mathbb{R}^k \times \mathbb{R}^k \to [0, 1]$ of two nodes' embeddings that determines the probability of an edge between them. Based on this, the probability that nodes i, j, k form a triangle is:

$$\mathbb{E}[\boldsymbol{A}_{ij}\boldsymbol{A}_{jk}\boldsymbol{A}_{ik}] = \int \rho_{Z_i}(z_i)\rho_{Z_j}(z_j)\rho_{Z_k}(z_k)e(z_i,z_j)e(z_j,z_k)e(z_i,z_k)\,dz_i\,dz_j\,dz_k$$

Now, define $f(z_i, z_j) = \sqrt{\rho_{Z_i}(z_i) \cdot \rho_{Z_j}(z_j)} \cdot e(z_i, z_j)$, $g(z_j, z_k) = \sqrt{\rho_{Z_j}(z_j) \cdot \rho_{Z_k}(z_k)} \cdot e(z_j, z_k)$, and $h(z_i, z_k) = \sqrt{\rho_{Z_i}(z_i) \cdot \rho_{Z_k}(z_k)} \cdot e(z_i, z_k)$, so that

$$\mathbb{E}[\boldsymbol{A}_{ij}\boldsymbol{A}_{jk}\boldsymbol{A}_{ik}] = \int f(z_i,z_j)g(z_j,z_k)h(z_i,z_k)\,dz_i\,dz_j\,dz_k$$

Now, we can apply Theorem 4.2.6, yielding:

$$\mathbb{E}[\boldsymbol{A}_{ij}\boldsymbol{A}_{jk}\boldsymbol{A}_{ik}] \leq \sqrt{\int f^2(z_i, z_j)\, dz_i\, dz_j \int g^2(z_j, z_k)\, dz_j\, dz_k \int h^2(z_i, z_k)\, dz_i\, dz_k}$$
$$= \sqrt{\int \rho_{Z_i}(z_i)\rho_{Z_j}(z_j)e^2(z_i, z_j)\, dz_i\, dz_j \int \rho_{Z_j}(z_j)\rho_{Z_k}(z_k)e^2(z_j, z_k)\, dz_j\, dz_k \int \rho_{Z_i}(z_i)\rho_{Z_k}(z_k)e^2(z_i, z_k)\, dz_i\, dz_k}$$
$$\leq \sqrt{\int \rho_{Z_i}(z_i)\rho_{Z_j}(z_j)e(z_i, z_j)\, dz_i\, z_j \int \rho_{Z_j}(z_j)\rho_{Z_k}(z_k)e(z_j, z_k)\, dz_j\, dz_k \int \rho_{Z_i}(z_i)\rho_{Z_k}(z_k)e(z_i, z_k)\, dz_i\, dz_k}.$$

where the last inequality simply uses the fact that since the image of the function e is $[0, 1]$, it must be that $e^2(x, y) \leq e(x, y)$. Finally, one can observe that

$$\int \rho_{Z_i}(z_i)\rho_{Z_j}(z_j)e(z_i, z_j)\, dz_i\, dz_j = \mathbb{E}[\boldsymbol{A}_{ij}],$$

from which the lemma follows. $\square$

Provided this lemma, it is straightforward to prove Theorem 4.2.3:

Proof of Theorem 4.2.3. Let $\boldsymbol{A}$ be a sample from a node independent model. For the expected number of triangles $\mathbb{E}[\Delta(\boldsymbol{A})]$ in the sample, we have that

$$\mathbb{E}[\Delta(\boldsymbol{A})] = \sum\nolimits_{i<j<k} \mathbb{E}[\boldsymbol{A}_{ij}\boldsymbol{A}_{ik}\boldsymbol{A}_{jk}] \leq \sum\nolimits_{i<j<k} \sqrt{\mathbb{E}[\boldsymbol{A}_{ij}] \cdot \mathbb{E}[\boldsymbol{A}_{ik}] \cdot \mathbb{E}[\boldsymbol{A}_{jk}]}.$$

Let T denote the last expression. By Cauchy-Schwarz and the identity $(\sum_i \sqrt{a_i})^2 \leq n \sum_i a_i$, we get

$$T^2 \leq \tbinom{n}{3} \cdot \sum\nolimits_{i<j<k} \mathbb{E}[\boldsymbol{A}_{ij}]\mathbb{E}[\cdot \boldsymbol{A}_{ik}] \cdot \mathbb{E}[\boldsymbol{A}_{jk}] \leq \tbinom{n}{3}^2 \mathrm{Ov}(\mathcal{A})^3,$$

where the last inequality follows from Theorem 4.2.1 for edge independent models. Finally, rearranging yields

$$\mathbb{E}[\Delta(\boldsymbol{A})] \leq T \leq \tbinom{n}{3} \cdot \mathrm{Ov}(\mathcal{A})^{3/2}.$$ $\square$

4.2.5 Squares and Other k-cycles

We can extend the prior bounds on triangles to bounds on the expected number of k-cycles in graphs sampled from the generative model $\mathcal{A}$ in terms of $\mathrm{Ov}(\mathcal{A})$. For the adjacency matrix $\boldsymbol{A}$ of a graph G, let $C_k(\boldsymbol{A})$ denote the number of k-cycles in G.

Theorem 4.2.7 (Bound on Expected k-cycles)**.** *Let $\boldsymbol{A}$ be an adjacency matrix sampled from a graph generative model $\mathcal{A}$, and let $C_k(\boldsymbol{A})$ denote the number of k-cycles in the graph corresponding to $\boldsymbol{A}$. If $\mathcal{A}$ is edge independent, node independent, or fully dependent then $\mathbb{E}[C_k(\boldsymbol{A})]$ is bounded above asymptotically by $n^k \cdot \mathrm{Ov}(\mathcal{A})^k$, $n^k \cdot \mathrm{Ov}(\mathcal{A})^{k/2}$, and $n^k \cdot \mathrm{Ov}(\mathcal{A})$, respectively.*

Proof. For notational simplicity, we focus on $k = 4$. The proof directly extends to general k. Let $C_4(G)$ be the number of non-backtracking 4-cycles in G (i.e. squares), which can be written as

$$\mathbb{E}_{\boldsymbol{A}\sim\mathcal{A}}\left[C_4(\boldsymbol{A})\right] = \frac{1}{8} \cdot \sum_{i=1}^{n} \sum_{j\in[n]\setminus\{i\}} \sum_{k\in[n]\setminus\{i,j\}} \sum_{\ell\in[n]\setminus\{i,j,k\}} \boldsymbol{A}_{ij}\boldsymbol{A}_{jk}\boldsymbol{A}_{k\ell}\boldsymbol{A}_{\ell i}.$$

The $1/8$ factor accounts for the fact that in the sum, each square is counted 8 times – once for each potential starting vector i and once of each direction it may be traversed. For general k-cycles this factor would be $\frac{1}{2k}$. We then can bound

$$\mathbb{E}_{\boldsymbol{A}\sim\mathcal{A}}\left[C_4(\boldsymbol{A})\right] \leq \frac{1}{8} \cdot \sum_{i\in[n]} \sum_{j\in[n]} \sum_{k\in[n]} \sum_{\ell\in[n]} \boldsymbol{A}_{ij}\boldsymbol{A}_{jk}\boldsymbol{A}_{k\ell}\boldsymbol{A}_{\ell i}.$$

If $\mathcal{A}$ is an edge independent model, the bound on the expected number of 4-cycles proceeds like the one for triangles, except using the following variant of Cauchy-Schwarz:

$$\sum\nolimits_{ijkl} a_{ij}b_{jk}c_{kl}d_{li} \leq \sqrt{\sum\nolimits_{ij} a_{ij}^2 \sum\nolimits_{ij} b_{ij}^2 \sum\nolimits_{ij} c_{ij}^2 \sum\nolimits_{ij} d_{ij}^2},$$

which, letting $\boldsymbol{P} = \mathbb{E}[\boldsymbol{A}]$ be the edge probability matrix for $\boldsymbol{A} \sim \mathcal{A}$, yields

$$\begin{aligned}\mathbb{E}[C_4(\boldsymbol{A})] &= \tfrac{1}{8}\sum\nolimits_{ijkl} \boldsymbol{P}_{ij}\boldsymbol{P}_{jk}\boldsymbol{P}_{kl}\boldsymbol{P}_{li} \le \tfrac{1}{8}\sqrt{\left(\sum\nolimits_{ij}\boldsymbol{P}_{ij}^2\right)^4} \\ &= \tfrac{1}{8}\left(2\cdot \mathrm{Ov}(\mathcal{A})\mathrm{Vol}(\mathcal{A})\right)^{4/2} = O\left(n^4 \mathrm{Ov}(\mathcal{A})^4\right),\end{aligned}$$

where again the last step is applying Lemma 4.2.4.

If $\mathcal{A}$ is a fully dependent model, the proof of the bound carries through almost exactly:

$$\begin{aligned}\mathbb{E}[C_4(\boldsymbol{A})] &= \sum_{i<j<k<l} \mathbb{E}\left[\boldsymbol{A}_{ij}\boldsymbol{A}_{jk}\boldsymbol{A}_{kl}\boldsymbol{A}_{li}\right] \le \sum_{i<j<k<l} \mathbb{E}[\boldsymbol{A}_{ij}] \\ &= \mathcal{O}(n^2)\sum\nolimits_{i<j}\mathbb{E}[\boldsymbol{A}_{ij}] = \mathcal{O}(n^2)\cdot \mathrm{Vol}(\mathcal{A}) \le \mathcal{O}(n^4)\cdot \mathrm{Ov}(\mathcal{A}). \qquad \square\end{aligned}$$

If $\mathcal{A}$ is a node independent model, the bound on the expected number of 4-cycles follows as before, except using the following variant of Theorem 4.2.6:

$$\begin{aligned}&\int f(x,y)g(y,z)h(z,w)l(w,x)\,dx\,dy\,dz\,dw \\ &\le \sqrt{\int f^2(x,y)\,dx\,dy \int g^2(y,z)\,dy\,dz \int h^2(z,w)\,dz\,dw \int l^2(w,x)\,dw\,dx}.\end{aligned}$$

Applying this equation as before yields

$$\mathbb{E}[\boldsymbol{A}_{ij}\boldsymbol{A}_{jk}\boldsymbol{A}_{kl}\boldsymbol{A}_{li}] \le \sqrt{\mathbb{E}[\boldsymbol{A}_{ij}]\mathbb{E}[\boldsymbol{A}_{jk}]\mathbb{E}[\boldsymbol{A}_{kl}]\mathbb{E}[\boldsymbol{A}_{li}]},$$

which allows us to apply the edge independent bound for squares:

$$\begin{aligned}\mathbb{E}[C_4(\boldsymbol{A})] &= \sum_{i<j<k<l} \mathbb{E}[\boldsymbol{A}_{ij}\boldsymbol{A}_{jk}\boldsymbol{A}_{kl}\boldsymbol{A}_{li}] \le \sum_{i<j<k<l} \sqrt{\mathbb{E}[\boldsymbol{A}_{ij}]\mathbb{E}[\boldsymbol{A}_{jk}]\mathbb{E}[\boldsymbol{A}_{kl}]\mathbb{E}[\boldsymbol{A}_{li}]} \\ &\le \sqrt{\tbinom{n}{4}\sum\nolimits_{i<j<k<l}\mathbb{E}[\boldsymbol{A}_{ij}]\mathbb{E}[\boldsymbol{A}_{jk}]\mathbb{E}[\boldsymbol{A}_{kl}]\mathbb{E}[\boldsymbol{A}_{li}]} \\ &\le \sqrt{O\left((n^4)^2\cdot \mathrm{Ov}(\mathcal{A})^4\right)} = \mathcal{O}(n^4\mathrm{Ov}(\mathcal{A})^{4/2}).\end{aligned}$$

4.2.6 Experimental Methodology and Baselines

We turn our attention to evaluating the real-world trade-off between overlap and performance for several specific models empirically on real world networks. In this work, we focus on graph generative models that, given an input adjacency matrix $\mathbf{A} \in \{0,1\}^{n\times n}$, produce a distribution $\mathcal{A}$ over adjacency matrices in $\{0,1\}^{n\times n}$. Broadly, it is desirable for these distributions to have two properties:

1. Samples from $\mathcal{A}$ should approximately match various graph statistics of the input graph $\mathbf{A}$, such as the degree distribution and triangle count.

2. $\mathcal{A}$ should have low overlap, to prevent the model from just memorizing and outputting $\mathbf{A}$.

Because there is generally some inherent tension between these two objectives for $\mathcal{A}$, it is desirable for the model to allow for **easily tuning overlap**.

Recent graph generative models, especially those that incorporate edge dependency, often involve complex deep architectures with a large number of parameters that are trained with (stochastic) gradient descent. The abundance of parameters implies that these models may have the capacity to simply memorize the input graph. At the same time, the complexity of the approaches obscures the roles of each component in yielding performance (in particular, the role of edge dependency is unclear), and specifying a desired overlap with the input graph is generally not possible, short of heuristics like early stopping. Altogether, the preceding discussion inspires the following research questions:

1. Are the looser theoretical limitations on triangle counts and other graph statistics for graph generative models with edge dependency reflected in modeling of real-world networks?

2. Can edge dependency be achieved with simple baselines that allow for overlap to be easily tuned?

3. Can such simple models match graph statistics comparably well to deep models, at given levels of overlap?

As a preview of our results in Section 4.2.7, we find overall positive answers to each of these three questions.

4.2.6.1 Graph Generative Models based on Dense Subgraph Discovery

As interpretable baselines to compare to deep-learning models, we introduce three graph generative models, one for each of the categories of our framework: edge independent (EI), node independent (NI), and fully dependent (FD). These baseline models mainly exploit the dense subgraph structure of the input graph, specifically the set of maximal cliques (MCs) [55]. We refer to our models as MCEI, MCNI, and MCFD, respectively. The high-level concept of these models is to start with an empty graph and plant edges from each of the input graph $\boldsymbol{A}$'s max cliques with some fixed probability hyperparameter p. How the edges are planted depends on the desired type of edge dependency and reflects the characteristic 'tight' examples for each category of the hierarchy from Section 4.2.1 – see Algorithm 8 for details. Note that, the lower p, the fewer the expected edges in the final sampled graph G_p, and hence the greater the 'residual' with respect to the input graph. To compensate, we also sample a second graph G_r and return the union of G_p and G_r.

Specifically, we produce G_r by sampling from a simple edge independent model, the odds product model, which we also leveraged in Section 4.1. Recall that this is as a variant of the well-known Chung-Lu configuration model [6, 39, 40], which produces graphs that match an input degree sequence in expectation. The odds product model does the same, except without constraints on the input degree sequence. In this section, we generalize this model to produce a sampled graph G_r such that its union with a G_p matches the input graph's degree distribution in expectation. Essentially

Algorithm 8 Sampling G_p for our max clique-based graph generative baselines

```
input input graph $G_i$, planting probability $p$, model type (MCEI, MCFD, or MCNI)
output sampled graph $G_p$
1: initialize the sampled graph $G_p$ to be empty
2: for each max clique $M \in$ input graph $G_i$ do
3:     if MCEI then
4:         for each edge $e \in M$ do
5:             with probability $p$, add $e$ to $G_p$
6:     else if MCFD then
7:         with probability $p$, add all edges in $M$ to $G_p$
8:     else if MCNI then
9:         for each node $v \in M$ do
10:            with probability $\sqrt{p}$, set $v$ to be ‘active’ in $M$
11:        add edges between all pairs of nodes active in $M$ to $G_p$
12: return $G_p$
```

the same derivation yields the modified fitting method, pseudocode for which is given in Algorithm 9.

Overall, this generative process gives rise to two desirable properties shared by our baseline models:

1. Each model has a single hyperparameter, the planting probability p, and ranging this probability from 0 to 1 ranges the overlap of the resulting distribution from very low to 1. In particular, the distribution $\mathcal{A}$ goes from being nearly agnostic to the input graph $\mathbf{A}$ (depending only on its degree sequence), to exactly returning $\mathbf{A}$ with probability 1.

2. Given input graph $\mathbf{A}$, samples from the distribution $\mathcal{A}$ that these models output will exactly match the node degrees of $\mathbf{A}$ in expectation.

The first of these properties encapsulates the two desiderata for graph generative models that were listed at the start of this section. As for the second, matching the degree sequence is an especially beneficial starting point for a generative model since doing so also results in matching some other statistics of interest, such as max degree and best-fit power-law exponent, as a byproduct.

Algorithm 9 Fitting the modified odds product model

```
input target degrees $\boldsymbol{d} \in \mathbb{R}^n$, primary expected adjacency matrix $\mathbb{E}[\boldsymbol{A_p}] \in [0,1]^{n \times n}$,
        error threshold $\epsilon$
output symmetric probability matrix $\mathbb{E}[\boldsymbol{A_r}] \in [0,1]^{n \times n}$
 1: $\boldsymbol{\ell} \leftarrow \mathbf{0}$    ▷ $\boldsymbol{\ell} \in \mathbb{R}^n$ is the vector of logits, initialized to all zeros
 2: $\mathbb{E}[\boldsymbol{A_r}] \leftarrow \sigma\left(\boldsymbol{\ell}\mathbf{1}^\top + \mathbf{1}\boldsymbol{\ell}^\top\right)$    ▷ $\sigma$ is the logistic function applied entrywise, and
                                     $\mathbf{1}$ is the all-ones column vector of length $n$
 3: $\mathbb{E}[\boldsymbol{A_u}] \leftarrow \mathbf{1}\mathbf{1}^\top - \left(\mathbf{1}\mathbf{1}^\top - \mathbb{E}[\boldsymbol{A_p}]\right) \circ \left(\mathbf{1}\mathbf{1}^\top - \mathbb{E}[\boldsymbol{A_r}]\right)$    ▷ $\circ$ is an entrywise product
 4: $\tilde{\boldsymbol{d}} \leftarrow \mathbb{E}[\boldsymbol{A_u}]\mathbf{1}$    ▷ expected degree sequence of $\mathbb{E}[\boldsymbol{A_u}]$
 5: while $\left\|\tilde{\boldsymbol{d}} - \boldsymbol{d}\right\|_2 > \epsilon$ do
 6:     $B \leftarrow \mathbb{E}[\boldsymbol{A_r}] \circ \left(\mathbf{1}\mathbf{1}^\top - \mathbb{E}[\boldsymbol{A_u}]\right)$
 7:     $J \leftarrow B + \operatorname{diag}(B\mathbf{1})$
 8:     $\boldsymbol{\ell} \leftarrow \boldsymbol{\ell} - J^{-1}\left(\tilde{\boldsymbol{d}} - \boldsymbol{d}\right)$    ▷ rather than inverting $J$, we solve this linear system
 9:     $\mathbb{E}[\boldsymbol{A_r}] \leftarrow \sigma\left(\boldsymbol{\ell}\mathbf{1}^\top + \mathbf{1}\boldsymbol{\ell}^\top\right)$
10:     $\mathbb{E}[\boldsymbol{A_u}] \leftarrow \mathbf{1}\mathbf{1}^\top - \left(\mathbf{1}\mathbf{1}^\top - \mathbb{E}[\boldsymbol{A_p}]\right) \circ \left(\mathbf{1}\mathbf{1}^\top - \mathbb{E}[\boldsymbol{A_r}]\right)$
11:     $\tilde{\boldsymbol{d}} \leftarrow \mathbb{E}[\boldsymbol{A_u}]\mathbf{1}$
12: return $\mathbb{E}[\boldsymbol{A_r}]$
```

4.2.7 Experimental Results

In this section we perform an evaluation of different graph generative models under the perspective of the *overlap* criterion.We choose 8 statistics that capture both the connectivity of a graph, as well local patterns within it. We look at the following network statistics:

- Pearson correlation coefficient (PCC) of the input and generated degree sequences (number of nodes incident to each node), as well as max degree.
- PCC of the input and generated triangle sequences (number of triangles each node belongs to).
- Normalized triangle, 4-clique, and 4-cycle counts.
- Characteristic (average) path length and fraction of node pairs which are connected. Letting $|C_i|$ be the size of i-th connected component, the latter quantity is $\sum_i \binom{|C_i|}{2} / \binom{n}{2})$.

We evaluate the following methods. It is worth outlining that we also explored ERGMs but they do not scale to our datasets.

- The three models we introduce in Section 4.2.6: MCEI, MCNI and MCFD.
- CELL [151], an efficient variant of the edge independent NetGAN method [20].
- VGAE [93], a node independent autoencoder-based model.
- GraphVAE [170], a fully dependent deep generative model. As this model is designed for small-sized graphs, we only evaluate it on a subset of smaller datasets.

Summary of experimental setting. We evaluate the above models on 8 publicly available graph datasets. In order to tune the overlap for our models (MCEI, MCNI, and MCFD), we simply increase the probability parameter p. For CELL and GraphVAE, we follow an early stopping criterion as in [20, 151]. More precisely, we pause the training at certain intervals and sample graphs from the model trained so far. For the VGAE model, we increase the dimension of the hidden layers and train for $5,000$ epochs. Each point in the following figures is an average over 10 samples.

Results. Our findings highlight the importance of *overlap* as a third dimension in the evaluation of graph generative models. See our plots of graph statistics vs overlap in Figures 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, and 4.15. We see that deep-learning based models, like GraphVAE and CELL, can almost fit the input graph as we allow the training to be performed for a sufficient number of epochs. However, when one wants to generate a diverse set of graphs, these models fail to match certain statistics of the input graph. For example, we see in Figure 4.8 that the CELL method generates graphs with a low number of triangles for the CiteSeer dataset when the overlap between the generated graphs is small. We observe similar results for almost all datasets and other statistics, especially those pertaining to small dense subgraphs (like 4-cliques and 4-cycles). Similarly, the GraphVAE method almost always fits

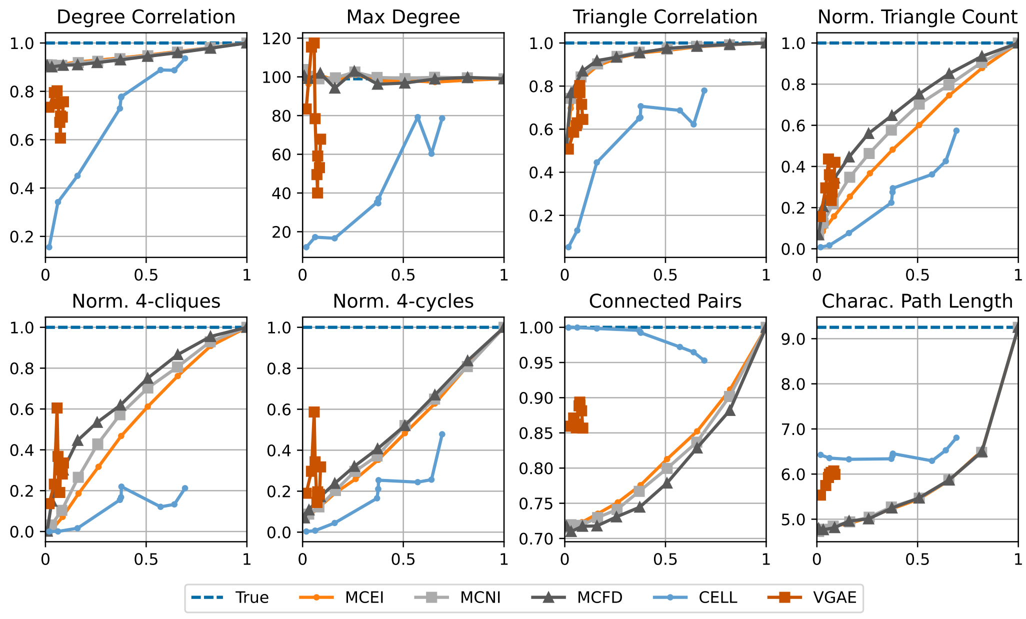


Figure 4.8. Graph statistics fidelity vs overlap with edge dependent models on CiteSeer.

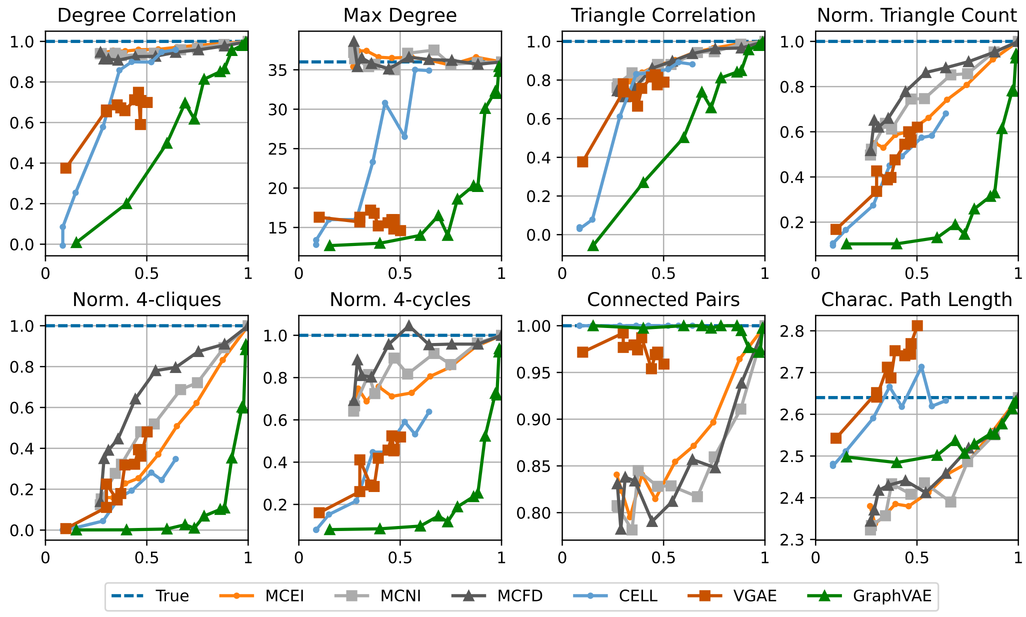


Figure 4.9. Graph statistics fidelity vs overlap with edge dependent models on Les Miserables.

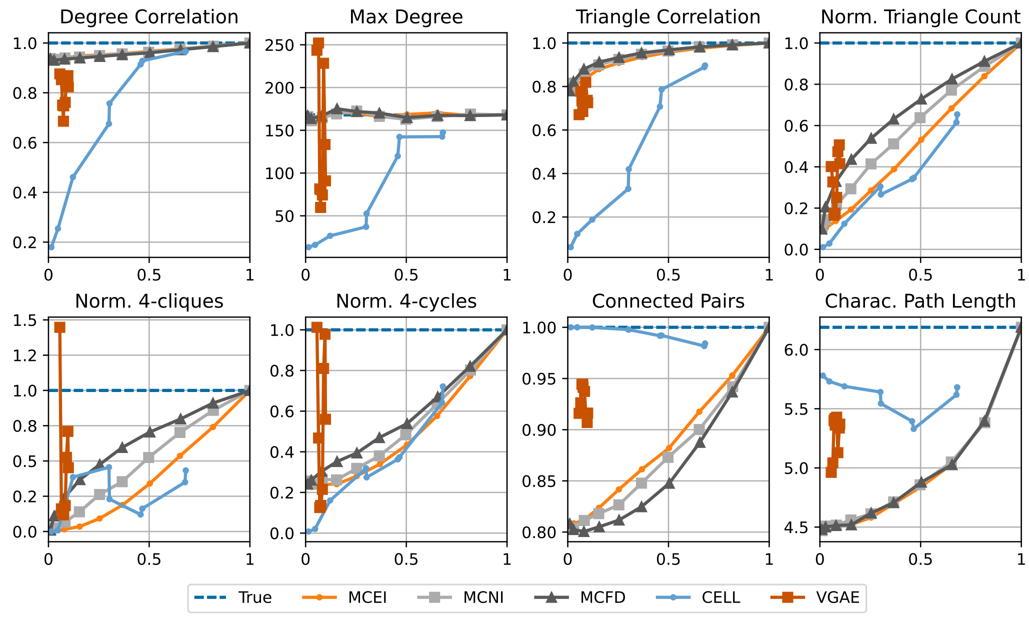


Figure 4.10. Graph statistics fidelity vs overlap with edge dependent models on Cora.

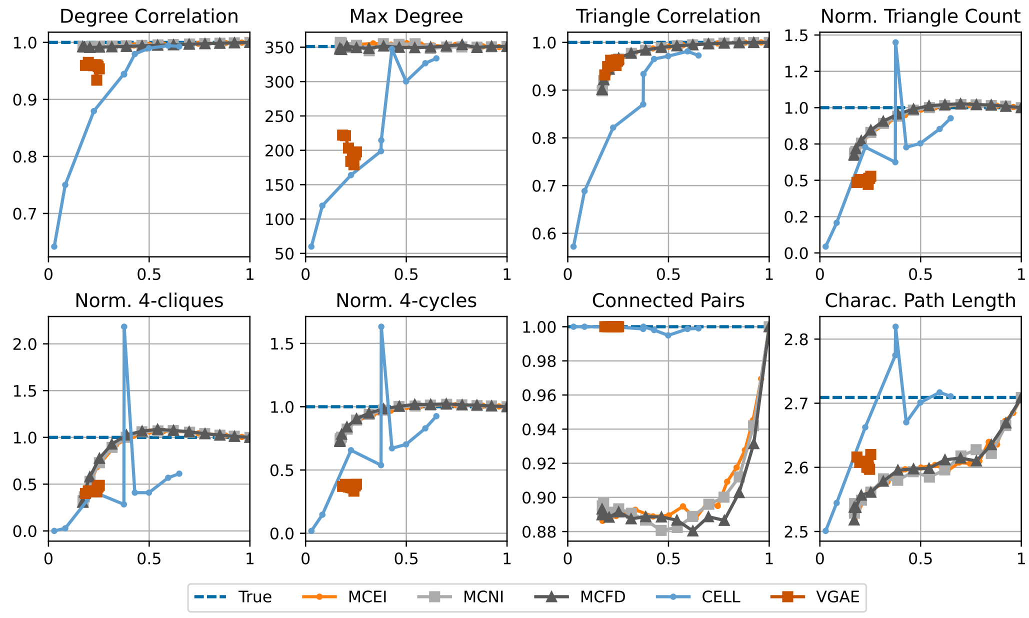


Figure 4.11. Graph statistics fidelity vs overlap with edge dependent models on PolBlogs.

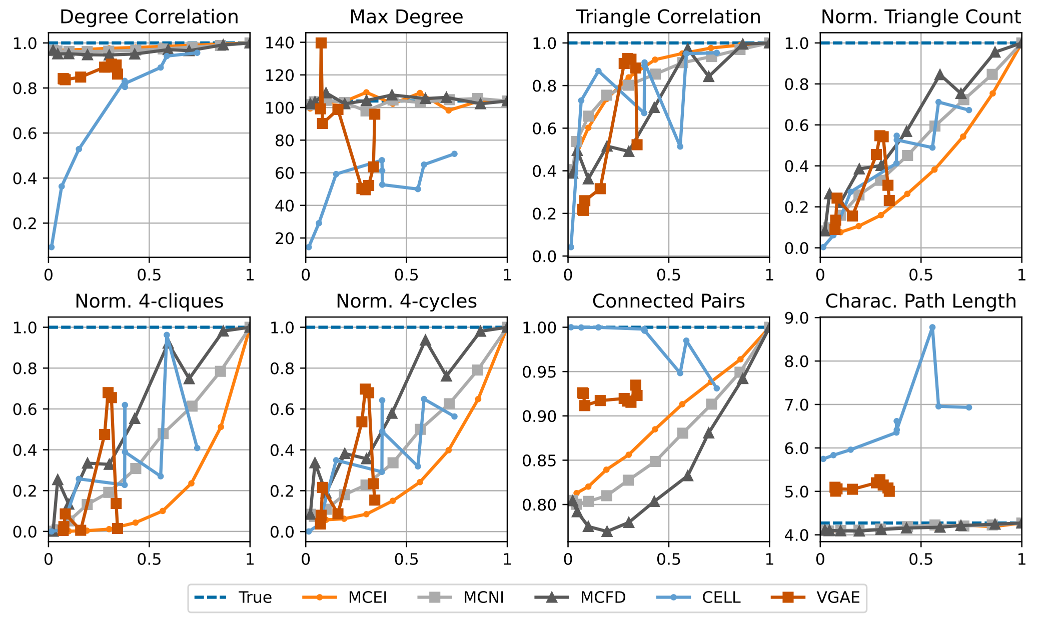


Figure 4.12. Graph statistics fidelity vs overlap with edge dependent models on Web-Edu.

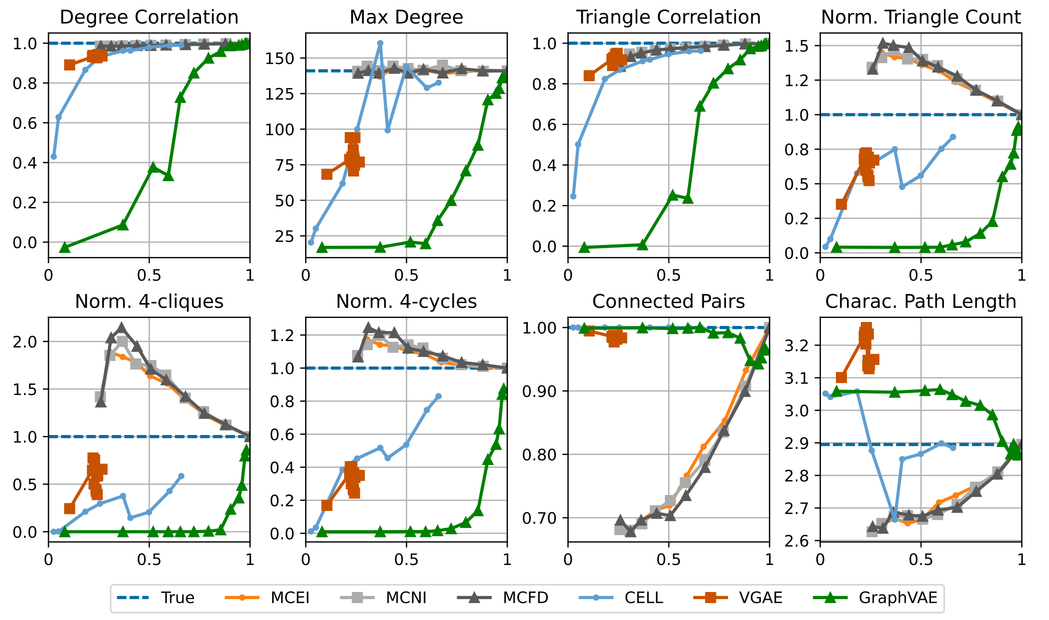


Figure 4.13. Graph statistics fidelity vs overlap with edge dependent models on WikiElect.

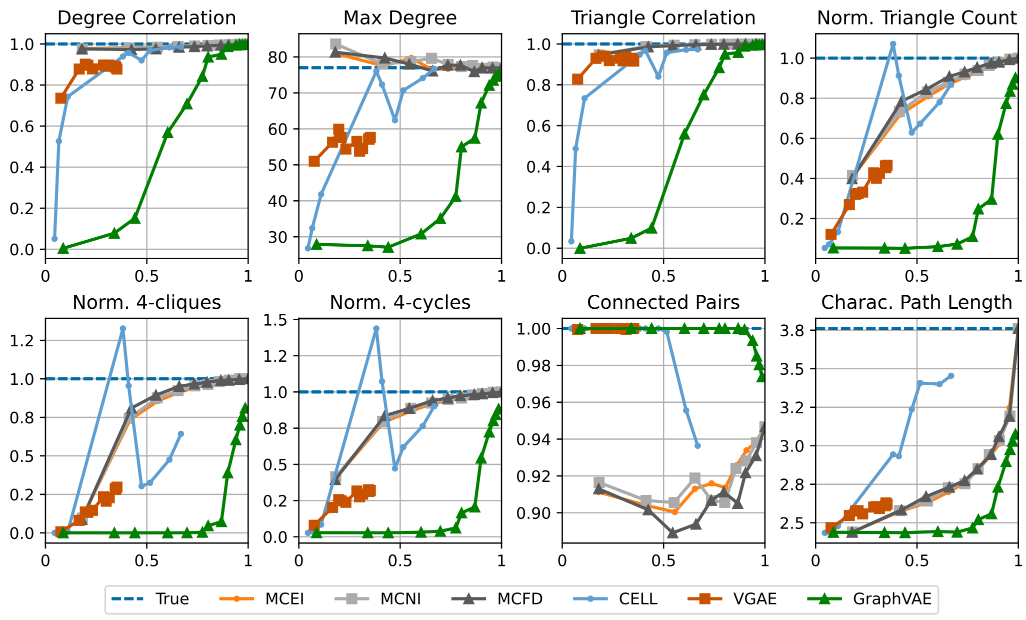


Figure 4.14. Graph statistics fidelity vs overlap with edge dependent models on FACEBOOK-EGO.

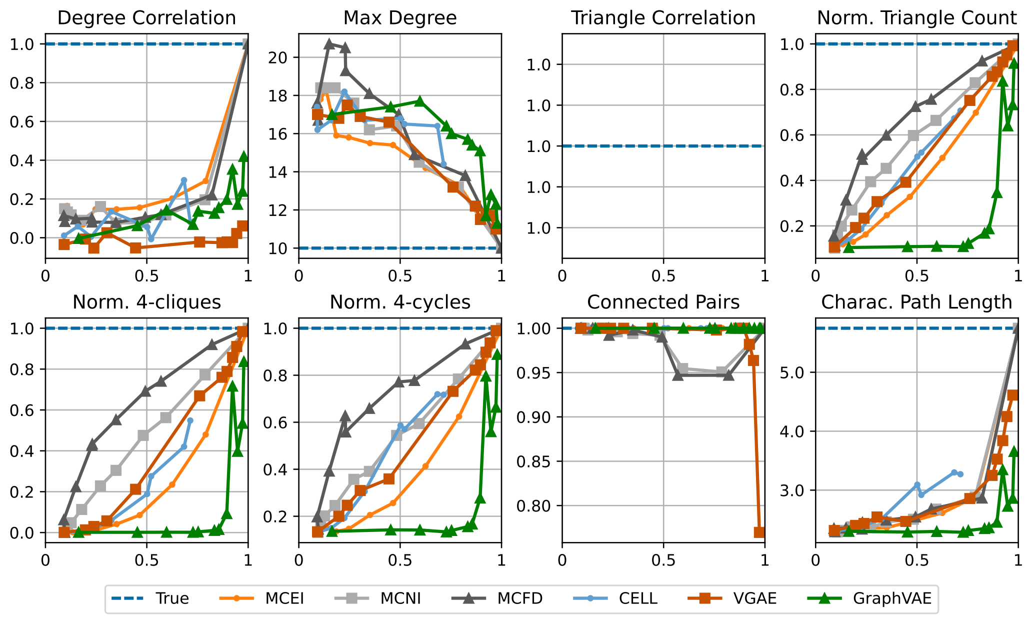


Figure 4.15. Graph statistics fidelity vs overlap with edge dependent models on RINGOFCLIQUES. As each node belongs to the same number of triangles, triangle correlation is not defined here. Thus, we replace it with transitivity.

the input graph in a high overlap regime. Although GraphVAE possesses greater theoretical capacity as an FD generative model, we observe that instances with low overlap deviate significantly from the statistical characteristics of the input graph, as illustrated in Figure 4.9. Consequently, this approach fails to achieve generalization when trained on a single graph instance. the VGAE model encounters additional limitations due to its dot-product kernel, which has been demonstrated to have certain constraints in generating graphs with sparsity and triangle density [167]. As depicted in Figures 4.8 and 4.9, we observe that this model fails to adequately capture the characteristics of the input graph, even when trained for an extensive number of epochs.

Surprisingly, despite their simplicity, the three introduced models exhibit desirable characteristics. Firstly, the overlap can be easily adjusted by increasing the probability hyperparameter p, providing more predictability compared to other models. Secondly, these models generally generate a higher number of triangles, which closely aligns with the input graph, in comparison to methods such as CELL. For instance, in Figures 4.8 and 4.9, these baselines produce graphs with a greater number of triangles, 4-cliques, and 4-cycles than CELL and GraphVAE, particularly when the overlap is low.

Furthermore, as the level of dependency rises (from edge independent to fully dependent), we observe a higher triangle count for a fixed overlap. This finding supports our theoretical assertions from Section 4.2.1 regarding the efficacy of different models within the introduced hierarchy. However, a drawback of these two-stage methods is their inability to capture the connectivity patterns of the input graph, as evident from the fraction of connected pairs and the characteristic path length statistics.

4.2.8 Conclusion

We have proved tight trade-offs for graph generative models between their ability to produce networks that match the high triangle densities of real-world graphs, and their ability to achieve low *overlap* and generate a diverse set of networks. We show that as the models are allowed higher levels of edge dependency, they are able to achieve higher triangle counts with lower overlap, and we formalize this finding by introducing a three-level hierarchy of edge dependency. An interesting future direction is to refine this hierarchy to be finer-grained, and also to investigate the roles of embedding length and complexity of the embedding distributions. We also emphasize our introduction of *overlap* as a third dimension along which to evaluate graph generative models, together with output quality and efficiency. We believe these directions can provide a solid groundwork for the systematic theoretical and empirical study of graph generative models.

CHAPTER 5
SIMPLIFYING DEEP GRAPH NETWORKS

In this chapter, we investigate simpler alternatives to deep representation learning for network data. Note that in this chapter only, the graphs we work with are also associated with node feature vectors, that is, the graph data comprise not only an $n \times n$ adjacency matrix $\boldsymbol{A}$, but also a node feature matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$.

5.1 Adaptive Simple Graph Convolution

Deep learning [99] has enjoyed great success in modeling image and text data, and graph convolutional networks (GCNs) [94] attempt to extend this success to graph data. Various deep models branch off from GCNs, offering additional speed, accuracy, or other features. However, like other deep models, these algorithms involve repeated nonlinear transformations of inputs and are therefore time and memory intensive to train. Recent work has shown that a much simpler model, simple graph convolution (SGC) [197], is competitive with GCNs in common graph machine learning benchmarks. SGC is much faster than GCNs, partly because the role of the graph in SGC is restricted to a feature extraction step in which node features are smoothed across the graph; by contrast, GCNs include graph convolution steps as part of an end-to-end model, resulting in expensive backpropagation calculations. However, the feature smoothing operation in SGC implicity assumes the common but not universal graph characteristic of homophily, wherein nodes mostly link to similar nodes; indeed, recent work [135] suggests that many GCNs may assume such structure. In this section, we ask whether a feature extraction approach, that, like SGC, is free of deep learning, can tackle heterophilous (i.e., non-homophilous) graph structure. We see this work as extending the following broader research question introduced by the SGC paper to a wider range of graphs:

Are nonlinearities, end-to-end backpropagation, and other characteristics of deep learning essential to effective learning on graphs?

Contributions. We confirm that SGC, which uses a fixed smoothing filter, can indeed be ineffective for heterophilous features via experiments on synthetic and real-world datasets. We propose adaptive simple graph convolution (ASGC), which fits a different filter for each feature. These filters can be smoothing or non-smoothing, and thus can adapt to both homophilous and heterophilous structures. Like SGC, ASGC is not a deep model, instead being based on linear least squares, and hence is fast, scalable, and interpretable. We propose a natural synthetic model for networks with a node feature, Featured Stochastic Block Models (FSBMs), and prove that ASGC denoises the network feature regardless of whether the model is set to produce homophilous or heterophilous networks, in contrast to SGC, which we show is inappropriate for the heterophilous setting. Finally, we show that the performance of ASGC is superior to that of SGC at node classification on real-world heterophilous networks, and generally competitive with recent deep methods on a benchmark of both heterophilous and homophilous networks. The SGC paper suggested that deep learning is not necessarily required for good performance on common graph learning tasks and benchmarks involving homophilous networks; our results suggest that simple methods can also be competitive for heterophilous networks.

5.1.1 Background

Preliminaries. We first establish common notation for working with graphs. An undirected, possibly weighted graph on n nodes can be represented by its symmetric adjacency matrix $\boldsymbol{A} \in \mathbb{R}_{\geq 0}^{n \times n}$. Letting $\mathbf{1}$ denote an all-ones column vector, $\boldsymbol{d} = \boldsymbol{A}\mathbf{1}$ is the vector of degrees of the nodes; the degree matrix $\boldsymbol{D}$ is the diagonal matrix with $\boldsymbol{d}$

along the diagonal. The normalized adjacency matrix is given by $\boldsymbol{S} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$, while the symmetric normalized graph Laplacian is $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{S}$, where $\boldsymbol{I}$ is an identity matrix. Note that the eigenvectors of $\boldsymbol{L}$ are exactly the eigenvectors of $\boldsymbol{S}$. It is a well known fact that the eigenvalues of $\boldsymbol{S}$ are contained within $[-1, +1]$. It follows that the eigenvalues of $\boldsymbol{L}$ are within $[0, 2]$, so $\boldsymbol{L}$ is positive semidefinite.

Consider a feature $\boldsymbol{x} \in \mathbb{R}^n$ on the graph, that is, a real-valued vector where each entry is associated with a node. The quadratic form over $\boldsymbol{L}$ is known to have the following equivalency:

$$\boldsymbol{x}^\top \boldsymbol{L}\boldsymbol{x} = \tfrac{1}{2} \sum\nolimits_{(i,j)\in[n]^2} A_{ij} \left(\tfrac{1}{\sqrt{d_i}} x_i - \tfrac{1}{\sqrt{d_j}} x_j \right)^2 .$$

Up to a reweighting based on the nodes' degrees, this expression is the sum of squared differences of the feature's values between adjacent nodes. Hence, the quadratic form has a low value, near 0, if the feature $\boldsymbol{x}$ is 'smooth' across the graph, that is, if adjacent nodes have generally similar values of the feature. Similarly, the quadratic has a high value if the feature is 'rough' across the graph, if adjacent nodes have generally differing values of the feature. When $\boldsymbol{x}$ is an eigenvector of $\boldsymbol{L}$, these 'smooth' and 'rough' cases correspond to the eigenvalue being low or high, respectively. (In terms of eigenvalues of $\boldsymbol{S}$, the opposite is true, with positive eigenvalues being smooth and negative eigenvalues being rough.) More generally, decomposition of an arbitrary feature vector $\boldsymbol{x}$ as a linear combination of the eigenvectors of $\boldsymbol{L}$ separates $\boldsymbol{x}$ into components ranging from 'smooth' to 'rough' across the graph. In the graph signal processing literature [137, 86, 176], these smooth and rough components are also called low and high frequency 'modes,' respectively, based on the eigenvalues of $\boldsymbol{L}$.

Simple graph convolution. Our work is primarily inspired by the simple graph convolution (SGC) algorithm [197]. SGC comprises logistic regression after the following feature extraction step, which can be interpreted as smoothing the nodes' features across the graph's edges:

$$\boldsymbol{x}_{\text{SGC}} = \tilde{\boldsymbol{S}}^{K}\boldsymbol{x}. \tag{5.1}$$

This equation shows how a single raw feature $\mathbf{x}$ is filtered to produce the smoothed feature $\boldsymbol{x}_{\text{SGC}}$. Here $\tilde{\boldsymbol{S}} \in \mathbb{R}^{n\times n}$ is the normalized adjacency matrix after addition of a self-loop to each node (that is, addition of the identity matrix to the adjacency matrix), and $K \in \mathbb{Z}_{+}$ is a hyperparameter; K determines the radius of the filter, that is, the maximum number of hops between two nodes whose features can directly influence others' features in the filtering process. [197] show that the addition of the self-loop increases the minimum eigenvalue of $\boldsymbol{S}$; intuitively, the self-loop limits the extent to which a feature can be 'rough' across the graph. This results in the highest magnitude eigenvalues of the normalized adjacency $\tilde{\boldsymbol{S}}$ tending to be positive (smooth). Because the eigenvalues of $\tilde{\boldsymbol{S}}$ all have magnitude at most 1, powering up $\tilde{\boldsymbol{S}}$ results in a filter which generally attenuates the feature $\boldsymbol{x}$, but does so least along these high magnitude, smooth eigenvectors. Hence the SGC filter smooths out the feature locally along the edges. Since it attenuates the high-frequency modes more than the low-frequency ones, SGC is described as a 'low-pass' filter.

Heterophily. If node features are used for node classification or regression, smoothing the features of nodes along edges encourages similar predictions along locally connected nodes. This seems sensible when locally connected nodes should be generally similar in terms of features and labels; if the variance in features and labels between connected nodes is generally attributable to noise, then this smoothing procedure acts as a useful denoising step. Graphs in which connected nodes tend to be similar are called homophilous or assortative. An example would be a citation network of papers on various topics: papers concerning the same topic tend to cite each other. Much of the existing work on graph models has an underlying assumption of network homophily, and there has been significant recent interest (discussed further in Section 5.1.5) on the limitations of graph models at addressing network heterophily/disassortativity, wherein connections tend to be between dissimilar nodes. An example would be a network based on adjacencies of words in text, where the labels are based on the part of speech: adjacent words tend to be of different parts of speech. For disassortative networks, smoothing a feature across connections as in SGC may not be sensible, since encouraging predictions of connected nodes to be similar is contrary to disassortativity.

5.1.2 Methodology

Adaptive SGC. In our method, we replace the fixed feature propagation step of SGC (Equation 5.1) with an adaptive one, which may or may not be smoothing based on the feature and graph. We produce a filtered version $\boldsymbol{x}_{\text{ASGC}}$ of the raw feature $\boldsymbol{x}$ by multiplying $\boldsymbol{x}$ with a learned polynomial of the normalized adjacency matrix $\boldsymbol{S}$; this polynomial is set so that $\boldsymbol{x}_{\text{ASGC}} \approx \boldsymbol{x}$:

$$\boldsymbol{x}_{\text{ASGC}} = \left(\sum_{k=0}^{K} \beta_k \boldsymbol{S}^k\right) \boldsymbol{x} \approx \boldsymbol{x}, \tag{5.2}$$

where K is a hyperparameter which, as in SGC, controls the radius of feature propagation, and the coefficients $\boldsymbol{\beta} \in \mathbb{R}^{K+1}$ are learned by minimizing the approximation error in a least squares sense. Note that, unlike SGC, we do not add self-loops to $\boldsymbol{S}$. This approximation error would be trivially minimized with $\beta_0 = 1$ and all other coefficients set to zero, resulting in $\boldsymbol{x}_{\text{ASGC}} = \boldsymbol{S}^0\boldsymbol{x} = \boldsymbol{x}$, so we regularize the magnitude of β_0.

More concretely, let $\boldsymbol{T} \in \mathbb{R}^{n \times (K+1)}$ denote the Krylov matrix generated by multiplying a feature vector $\boldsymbol{x} \in \mathbb{R}^n$ by the normalized adjacency matrix $\boldsymbol{S}$ up to some $K \in \mathbb{Z}_{>0}$ times:

$$\boldsymbol{T} = \left(\boldsymbol{S}^0\boldsymbol{x}; \boldsymbol{S}^1\boldsymbol{x}; \boldsymbol{S}^2\boldsymbol{x}; \ldots; \boldsymbol{S}^K\boldsymbol{x} \right). \tag{5.3}$$

Here the leftmost column of $\boldsymbol{T}$ is just the raw feature $\boldsymbol{x}$, and each column represents the feature generated by propagating the feature vector to its left across the graph, i.e., by multiplying once by $\boldsymbol{S}$. We produce a filtered version $\boldsymbol{x}_{\text{ASGC}}$ of the raw feature $\boldsymbol{x}$ by linear combination of the columns of $\boldsymbol{T}$. That is, $\boldsymbol{x}_{\text{ASGC}} = \boldsymbol{T}\boldsymbol{\beta}$, and we set the combination coefficients $\boldsymbol{\beta} \in \mathbb{R}^{k+1}$ by minimizing a loss function as follows:

$$\min_{\boldsymbol{\beta}} \left(\|\boldsymbol{T}\boldsymbol{\beta} - \boldsymbol{x}\|_2^2 + (R\beta_0)^2 \right). \tag{5.4}$$

The term on the right is L_2 regularization applied to the zeroth combination coefficient β_0 (which multiplies the raw feature $\boldsymbol{S}^0\boldsymbol{x}$), and $R \in \mathbb{R}_{\geq 0}$ if a hyperparameter which controls the strength of this regularization. Equation 5.4 is solved for the optimal coefficents $\boldsymbol{\beta}$ using linear least squares.

Algorithm 10 Adaptive Simple Graph Convolution (ASGC) Filter

```
**Input:** undirected graph $\boldsymbol{A} \in \{0,1\}^{n \times n}$, node feature $\boldsymbol{x} \in \mathbb{R}^n$, number of hops $K \in \mathbb{Z}_{>0}$, regularization strength $R \in \mathbb{R}_{\geq 0}$
**Output:** ASGC-filtered feature $\boldsymbol{x}_{\text{ASGC}}$
$\boldsymbol{D} \leftarrow \operatorname{diag}(\boldsymbol{A}\boldsymbol{1})$                ▷ degree matrix
$\boldsymbol{S} \leftarrow \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$                ▷ normalized adjacency matrix
$\boldsymbol{T} \in \mathbb{R}^{n \times (K+1)} \leftarrow \boldsymbol{0}$                ▷ $k$-step propagated features
$\boldsymbol{T}_0 \leftarrow \boldsymbol{x}$
**for** $k = 1$ **to** $K$ **do**
    $\boldsymbol{T}_k \leftarrow \boldsymbol{S}\boldsymbol{T}_{k-1}$
$\boldsymbol{R} \in \mathbb{R}^{K+1} \leftarrow (R, 0, 0, \ldots, 0)$
$\boldsymbol{\beta} \leftarrow$ least squares solution of $\binom{\boldsymbol{T}}{\boldsymbol{R}}\boldsymbol{\beta} \approx \binom{\boldsymbol{x}}{0}$
**Return:** $\boldsymbol{T}\boldsymbol{\beta}$
```

Intuition. As noted above, if we set the regularization $R = 0$, or more generally in the limit as $R \to 0$, the approximation error in Equation 5.4 is trivially minimized and results in $\boldsymbol{x}_{\text{ASGC}} = \boldsymbol{x}$; that is, the learned filter just ignores the graph structure and maps the input feature through unchanged. Nonzero regularization forces the least squares reconstruction to use the graph structure as it approximates the raw, unpropagated feature. Ideally, this results in a denoising effect that is able to extend beyond SGC's fixed smoothing along edges. For example, when a graph $\boldsymbol{S}$ is homophilous with respect to a node feature $\boldsymbol{x}$, in that neighbors tend to have similar feature values, the raw feature is correlated positively with the propagated version $\boldsymbol{Sx}$ of the feature. By contrast, when a graph is heterophilous with respect to a feature, the correlation is negative. The least squares in ASGC is able to adapt to both cases and exploit this correlation, as well as correlations that occur when repeatedly propagating features (i.e., correlations of $\boldsymbol{x}$ with $\boldsymbol{S}^k\boldsymbol{x}$ for $k > 1$).

Further remarks. In theory, as K is raised to higher values, $\boldsymbol{T}$ will provide a sufficiently high-rank basis that $\boldsymbol{x}_{\text{ASGC}}$ will be arbitrarily close to $\boldsymbol{x}$, even if R is very high. Then ASGC would have essentially the same performance as using the raw feature. While this issue could be resolved by introducing a smaller regularization term for the remaining coefficients, we find that this is generally not a problem over reasonable values of K on real-world graphs; hence, for simplicity, we do not introduce this further regularization.

Pseudocode to filter a single feature is given in Algorithm 10. This algorithm is applied independently to all features; note that this is trivially parallelizable across features. After this, as in SGC, the filtered features are passed as input to a logistic regression classifier for node classification. The core computations in Algorithm 10 are 1) creation of the matrix $\boldsymbol{T}$ by multiplying $\boldsymbol{x}$ by $\boldsymbol{S}$ up to K times, for which the time complexity is $\mathcal{O}(mK)$, where m is the number of edges; and 2) linear least squares with a matrix of dimensionality $(n+1) \times (K+1)$, for which the complexity is $\mathcal{O}(nK^2)$.

Spectral Interpretation of ASGC. ASGC admits an interesting alternate interpretation based on a spectral view of Equation 5.4. Let $\boldsymbol{S} = \boldsymbol{Q}\,\mathrm{diag}(\boldsymbol{\lambda})\boldsymbol{Q}^\top$ be an eigendecomposition of $\boldsymbol{S}$, and let $\boldsymbol{\gamma} = \boldsymbol{Q}^{-1}\boldsymbol{x}$, that is, $\boldsymbol{\gamma}$ is the graph Fourier transform of the feature $\boldsymbol{x}$. The central objective in ASGC is the norm of the residual of the least squares in Equation 5.4. As in Parseval's Theorem, due to the orthogonality of $\boldsymbol{Q}$, this norm is invariant under the graph Fourier transform:

$$\|\boldsymbol{x}_{\mathrm{ASGC}} - \boldsymbol{x}\|^2 = \|\boldsymbol{T}\boldsymbol{\beta} - \boldsymbol{x}\|^2 = \left\|\boldsymbol{Q}^\top\left(\boldsymbol{T}\boldsymbol{\beta} - \boldsymbol{x}\right)\right\|^2 = \left\|\boldsymbol{Q}^{-1}\left(\boldsymbol{T}\boldsymbol{\beta} - \boldsymbol{x}\right)\right\|^2.$$

Recall that each column of $\boldsymbol{T}$ is of the form $\boldsymbol{S}^i\boldsymbol{x}$ for some nonnegative power i. Then

$$\boldsymbol{Q}^{-1}\boldsymbol{S}^i\boldsymbol{x} = \boldsymbol{Q}^{-1}\left(\boldsymbol{Q}\,\mathrm{diag}(\boldsymbol{\lambda})^i\boldsymbol{Q}^\top\right)(\boldsymbol{Q}\boldsymbol{\gamma}) = \mathrm{diag}(\boldsymbol{\lambda})^i\boldsymbol{\gamma},$$

and the minimization objective can be written as

$$\|\boldsymbol{x}_{\mathrm{ASGC}} - \boldsymbol{x}\|^2 = \left\|\boldsymbol{Q}^{-1}\boldsymbol{T}\boldsymbol{c} - \boldsymbol{Q}^{-1}\boldsymbol{x}\right\|^2 = \|\mathrm{diag}(\boldsymbol{\gamma})\boldsymbol{V}_{\boldsymbol{\lambda}}\boldsymbol{\beta} - \boldsymbol{\gamma}\|^2 = \|\mathrm{diag}(\boldsymbol{\gamma})\left(\boldsymbol{V}_{\boldsymbol{\lambda}}\boldsymbol{\beta} - \boldsymbol{1}\right)\|^2,$$

where, letting superscript $\circ i$ denote the entrywise i^{th} power of a vector,

$$\boldsymbol{V}_{\boldsymbol{\lambda}} = \left(\boldsymbol{\lambda}^{\circ 0}; \boldsymbol{\lambda}^{\circ 1}; \boldsymbol{\lambda}^{\circ 2}; \ldots; \boldsymbol{\lambda}^{\circ K}\right)$$

is the Vandermonde matrix of powers 0 to K of the eigenvalues of $\boldsymbol{S}$. Note that multiplying $\boldsymbol{V}_{\boldsymbol{\lambda}}$ by the vector $\boldsymbol{\beta}$ yields the values of the polynomial with coefficients $\boldsymbol{\beta}$, evaluated at the eigenvalues $\boldsymbol{\lambda}$. Hence ASGC can be interpreted as fitting a K-degree polynomial over the graph's eigenvalues, with the target being all-ones. The value the polynomial takes over each eigenvalue represents how the learned filter scales the component of the feature $\boldsymbol{x}$ which is along the direction of the corresponding eigenvector of $\boldsymbol{S}$; the all-ones target corresponds to a do-nothing filter. The least squares loss is weighed proportionately to the magnitude of this component at each eigenvalue. That K is small, and the use of regularization on the zeroth coefficient, precludes the learned filter actually being the do-nothing filter, and instead results in a simple polynomial which adapts to the feature.

5.1.3 Motivating Example

We now use a synthetic network to demonstrate the capability of ASGC, and the potential deficiencies of SGC, at denoising a single heterophilous feature. We propose Featured SBMs, which augment stochastic block models (SBMs) [84] with a single feature; we note that our FSBMs can be seen as a simplified variant of recently studied Contextual SBMs [47].

Definition 11 (Featured SBM)**.** *An SBM graph G has n nodes partitioned into r communities $C_1, C_2, \ldots, C_r$, with intra- and inter- community edge probabilities p and q. Let $\boldsymbol{c_1}, \boldsymbol{c_2}, \ldots, \boldsymbol{c_r} \in \{0,1\}^n$ be indicator vectors for membership in each community, i.e., the j^{th} entry of $\boldsymbol{c_i}$ is 1 if the j^{th} node is in C_i and 0 otherwise. A Featured SBM (FSBM) is such a graph model G, plus a feature vector $\boldsymbol{x} = \boldsymbol{f} + \boldsymbol{\eta} \in \mathbb{R}^n$, where $\boldsymbol{\eta} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$ is zero-centered, isotropic Gaussian noise and $\boldsymbol{f} = \sum_i \mu_i \boldsymbol{c_i}$ for some $\mu_1, \mu_2, \ldots, \mu_r \in \mathbb{R}$, which are the expected feature values of each community.*

We consider FSBMs with $n = 1000$, 2 equally-sized communities C_+ and C_-, feature means $\mu_+ = +1$, and $\mu_- = -1$, and noise variance $\sigma = 1$. Thus, there are 500 nodes in each community; calling these communities 'plus' and 'minus,' the feature mean is $+1$ for nodes in the former and -1 for the latter, to which standard normal noise is added. We generate different graphs by setting the expected degree of all nodes to 10 (that is, $\frac{1}{2}(p+q) \cdot n = 10$) , then varying the intra- and inter-community

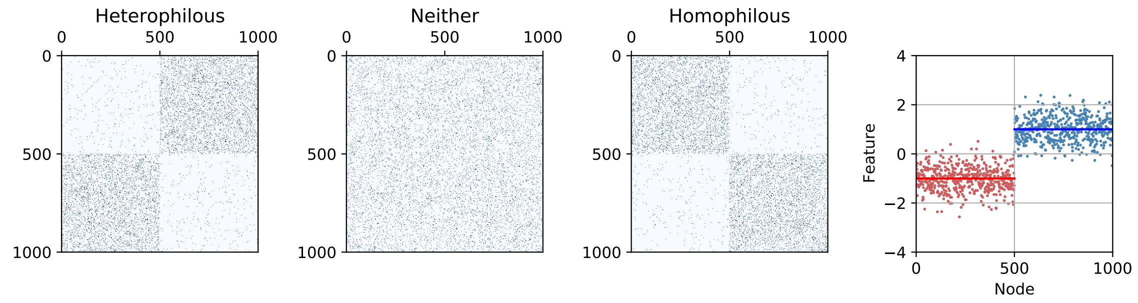


Figure 5.1. Synthetic dataset visualization. Left: 3 sample adjacency matrices, from the highly heterophilous ($q \ll p$) to the highly homophilous ($p \gg q$); for visual clarity, these graphs are 10 times denser the description in Section 5.1.3. Right: Feature values by node; note the feature means.

edge probabilities p and q from $p \gg q$ (highly homophilous, in that 'plus' nodes are much more likely to connect to other 'plus' nodes than to 'minus' nodes) to $q \gg p$ (highly heterophilous, in that 'plus' nodes tend to connect to 'minus' nodes). See Figure 5.1 for illustration.

We seek to denoise the feature by exploiting the graph, which should result in the feature values moving towards the means of the respective communities. We employ SGC and our ASGC, both with number of hops $K = 2$. Figure 5.2 (left) shows the deviation from the feature means after denoising. It also shows the proportion of nodes whose filtered feature differs from the community mean in sign, that is, the error when classifying nodes into C_+ and C_-. By both metrics, ASGC outperforms SGC on heterophilous graphs, while SGC outperforms ASGC on homophilous graphs. Both methods can lose accuracy relative to just using the raw feature when the graph is neither homophilous nor heterophilous, that is, when the graph is not informative about the communities. However, the performance of ASGC increases similarly as the graph becomes either more heterophilous or homophilous, whereas SGC's performance improves significantly less in the heterophilous direction. Finally, the performance gap between the two is smaller on homophilous graphs, particularly on sign accuracy, suggesting that ASGC can better adapt to varying degrees of ho-

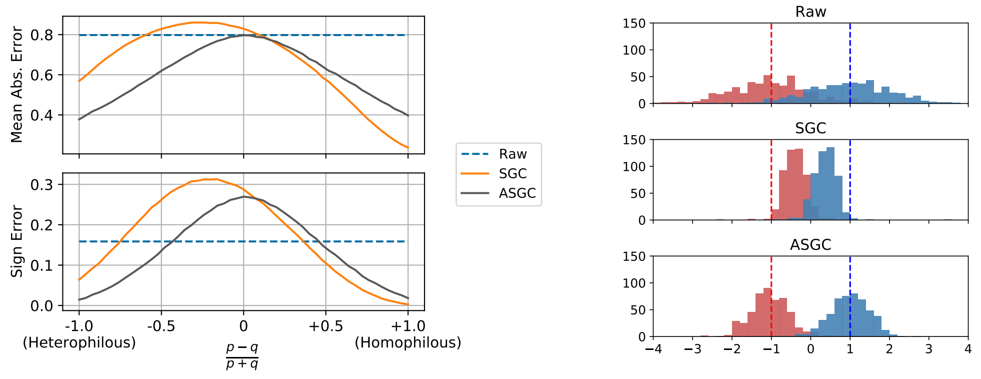


Figure 5.2. Left: Denoising results on the synthetic graphs using SGC and ASGC with number of hops $K = 2$. 'Raw' shows the error when no filtering method is applied. ASGC and SGC are more effective at denoising on heterophilous and homophilous networks, respectively, and ASGC is more effective overall. Right: Distribution of the feature values before and after applying each of the filtering methods on a very heterophilous synthetic graph ($\frac{p-q}{p+q} = -{}^{9}\!/_{10}$), separated by ground-truth communities. SGC tends to merge the two communities, while ASGC is able to keep them separated.

mophily/heterophily. We examine the highly heterophilous case in more detail in Figure 5.2 (right), which shows the distributions of the feature before and after filtering. The fixed propagation of SGC tends towards merging the two communities' feature distributions; by contrast, ASGC is able to keep them separated, preserves the feature means, and pulls the feature distributions towards the respective community means.

5.1.4 Theoretical Guarantees

To support our empirical investigation in Section 5.1.3, we now theoretically verify the limitations and capabilities of SGC and ASGC at denoising FSBM networks. For simplicity, we analyze SGC without the addition of a self-loop (that is, using $\boldsymbol{S}$ in Equation 5.1 rather than $\tilde{\boldsymbol{S}}$); the distinction between the two in the analysis vanishes as the number of intra-community edges grows, i.e., if $n \cdot p$ is high. Further, we assume that the regularization hyperparameter R for ASGC is high, in which case the coefficient β_0 is fixed to zero, or equivalently, the column $\boldsymbol{S}^0\boldsymbol{x}$ is removed from the Krylov matrix $\boldsymbol{T}$ in Equations 5.3 and 5.4. Finally, we analyze using expected adjacency matrices from the model, though we conjecture that via concentration bounds one could extend the following results to the sampled setting [173].

Two Community Case. We begin by analyzing FSBMs with 2 equally-sized communities. Unless otherwise specified, we use the same notation as Definition 11.

Theorem 5.1.1 (Effect of SGC on Two-Community FSBM Networks)**.** *Consider FSBMs having 2 equally-sized communities with indicator vectors $\boldsymbol{c_u}$ and $\boldsymbol{c_v}$, expected adjacency matrix $\boldsymbol{A}$, and feature vector $\boldsymbol{x} = \boldsymbol{f} + \boldsymbol{\eta}$. Let $\boldsymbol{x}_{SGC}$ be the feature vector returned by applying SGC, with number of hops K, to $\boldsymbol{A}$ and $\boldsymbol{x}$. Further, let $\bar{\mu} = \frac{1}{2}(\mu_u + \mu_v)$ be the average of the feature means. Then, $\boldsymbol{x}_{SGC} = \boldsymbol{f'} + \theta_u \boldsymbol{c_u} + \theta_v \boldsymbol{c_v}$, where $\boldsymbol{f'} = \lambda_2^K \boldsymbol{f} + (1 - \lambda_2^K)(\bar{\mu}\boldsymbol{1})$, and θ_u and θ_v are both distributed according to $\mathcal{N}\left(0, \frac{1}{n}(1 + \lambda_2^{2K})\sigma^2\right)$.*

Proof. In expectation, an entry A_{ij} of the adjacency matrix of the graph is p if both $i, j \in C_u$ or both $i, j \in C_v$, and it is q otherwise. The eigendecomposition $\boldsymbol{Q}\,\mathrm{diag}(\boldsymbol{\lambda})\boldsymbol{Q}^\top$ of the associated normalized adjacency matrix $\boldsymbol{S}$ has two nonzero eigenvalues: $\lambda_1 = 1$, with eigenvector $\boldsymbol{q_1} = (1/\sqrt{n})\boldsymbol{1} = (1/\sqrt{n})(\boldsymbol{c_u} + \boldsymbol{c_v})$, and $\lambda_2 = \frac{p-q}{p+q}$, with $\boldsymbol{q_2} = (1/\sqrt{n})(\boldsymbol{c_u} - \boldsymbol{c_v})$. In the following analysis, we use the fact that zero-centered, isotropic Gaussian distributions are invariant to rotation, meaning $\boldsymbol{Q}^\top\boldsymbol{\eta} = \boldsymbol{\eta'} \sim \mathcal{N}(0, \sigma^2\boldsymbol{I})$ for any orthonormal matrix $\boldsymbol{Q}$.

$$\begin{aligned}
\boldsymbol{x}_{\mathrm{SGC}} = \boldsymbol{S}^K\boldsymbol{x} &= \boldsymbol{Q}\,\mathrm{diag}(\boldsymbol{\lambda})^K\boldsymbol{Q}^\top(\mu_u\boldsymbol{c_u} + \mu_v\boldsymbol{c_v} + \boldsymbol{\eta}) \\
&= \boldsymbol{q_1}\boldsymbol{q_1}^\top(\mu_u\boldsymbol{c_u} + \mu_v\boldsymbol{c_v} + \boldsymbol{\eta}) + \lambda_2^K\boldsymbol{q_2}\boldsymbol{q_2}^\top(\mu_u\boldsymbol{c_u} + \mu_v\boldsymbol{c_v} + \boldsymbol{\eta}) \\
&= \boldsymbol{q_1}\left(\tfrac{\sqrt{n}}{2}(\mu_u + \mu_v) + \eta_1'\right) + \lambda_2^K\boldsymbol{q_2}\left(\tfrac{\sqrt{n}}{2}(\mu_u - \mu_v) + \eta_2'\right) \\
&= \left(\tfrac{1}{2}(\mu_u + \mu_v) + \tfrac{1}{\sqrt{n}}\eta_1'\right)(\boldsymbol{c_u} + \boldsymbol{c_v}) + \lambda_2^K\left(\tfrac{1}{2}(\mu_u - \mu_v) + \tfrac{1}{\sqrt{n}}\eta_2'\right)(\boldsymbol{c_u} - \boldsymbol{c_v}) \\
&= \quad \lambda_2^K(\mu_u\boldsymbol{c_u} + \mu_v\boldsymbol{c_v}) + (1 - \lambda_2^K)\cdot\tfrac{1}{2}(\mu_u + \mu_v)(\boldsymbol{c_u} + \boldsymbol{c_v}) \\
&\quad + \tfrac{1}{\sqrt{n}}\left(\eta_1' + \lambda_2^K\eta_2'\right)\boldsymbol{c_u} + \tfrac{1}{\sqrt{n}}\left(\eta_1' - \lambda_2^K\eta_2'\right)\boldsymbol{c_v} \\
&= \lambda_2^K\boldsymbol{f} + (1 - \lambda_2^K)(\bar{\mu}\mathbf{1}) + \theta_+\boldsymbol{c_u} + \theta_-\boldsymbol{c_v},
\end{aligned}$$

where $\theta_\pm = \frac{1}{\sqrt{n}}(\eta_1' \pm \lambda_2^K\eta_2')$, which has the specified distribution. □

Note that $\lambda_2 = \frac{p-q}{p+q} \in [-1, +1]$, with negative values indicating heterophily ($p < q$) and positive values indicating homophily ($p > q$). SGC only preserves the feature means in certain limiting cases. In particular, this occurs as $\lambda_2 \to +1$, or as $\lambda_2 \to -1$ if K is even; then $\lambda_2^K \to 1$, so the expected filtered feature vector $\boldsymbol{f'} \to \boldsymbol{f}$. On the other hand, if $\lambda_2 \to 0$, then $\lambda_2^K \to 0$ and $\boldsymbol{f'} \to \bar{\mu}\mathbf{1}$, that is, the feature value means are averaged between the communities. Finally, if $\lambda_2 \to -1$ and K is odd, then $\lambda_2^K \to -1$ and $\boldsymbol{f'} = \bar{\mu}\mathbf{1} + (\bar{\mu}\mathbf{1} - \boldsymbol{f}) = \mu_v\boldsymbol{c_u} + \mu_u\boldsymbol{c_v}$: the feature value means are entirely exchanged across the communities. By contrast, ASGC preserves the means, while similarly reducing noise by an $\Omega(n)$ factor, with much looser restrictions on λ_2 and K:

Theorem 5.1.2 (Effect of ASGC on Two-Community FSBM Networks)**.** *Consider FSBMs with $p \neq q$ having 2 equally-sized communities with community indicator vectors $\boldsymbol{c_u}$ and $\boldsymbol{c_v}$, expected adjacency matrix $\boldsymbol{A}$, and feature vector $\boldsymbol{x} = \boldsymbol{f} + \boldsymbol{\eta}$. Let $\boldsymbol{x}_{ASGC}$ be the feature vector returned by applying ASGC, with number of hops $K \geq 2$, to $\boldsymbol{A}$ and $\boldsymbol{x}$. Then $\boldsymbol{x}_{ASGC} = \boldsymbol{f} + \theta_+'\boldsymbol{c_u} + \theta_-'\boldsymbol{c_v}$, where θ_+' and θ_-' are both distributed according to $\mathcal{N}\left(0, \frac{2}{n}\sigma^2\right)$.*

Proof. The least squares in ASGC is equivalent to projecting the feature $\boldsymbol{x}$ onto the column span of the Krylov matrix $\boldsymbol{T} = \left(\boldsymbol{S}^1\boldsymbol{x}; \boldsymbol{S}^2\boldsymbol{x}; \ldots; \boldsymbol{S}^K\boldsymbol{x}\right)$. Observe that the column span of $\boldsymbol{T}$ is contained in the column span of $\boldsymbol{S}$. Further, $\boldsymbol{S}$ is rank-2 (by the assumption that $p \neq q$), so with probability 1 over the distribution of $\boldsymbol{\eta}$, as long as $K \geq 2$, the column span of $\boldsymbol{T}$ equals that of $\boldsymbol{S}$. Thus ASGC projects $\boldsymbol{x}$ onto the column span of $\boldsymbol{S}$, i.e., the span of $\boldsymbol{q_1}, \boldsymbol{q_2}$, the eigenvectors of $\boldsymbol{S}$. The following analysis proceeds exactly like the one for SGC, just without the terms for the eigenvalue λ_2, so we use the same notation and abbreviate the steps:

$$\begin{aligned}
\boldsymbol{x}_{\text{ASGC}} &= \boldsymbol{Q}\boldsymbol{Q}^\top \boldsymbol{x} \\
&= \boldsymbol{q_1}\boldsymbol{q_1}^\top(\mu_u \boldsymbol{c_u} + \mu_v \boldsymbol{c_v} + \boldsymbol{\eta}) + \boldsymbol{q_2}\boldsymbol{q_2}^\top(\mu_u \boldsymbol{c_u} + \mu_v \boldsymbol{c_v} + \boldsymbol{\eta}) \\
&= \boldsymbol{f} + \theta'_+ \boldsymbol{c_u} + \theta'_- \boldsymbol{c_v},
\end{aligned}$$

where $\theta'_\pm = \frac{1}{\sqrt{n}}(\eta'_1 \pm \eta'_2)$, which has the specified distribution. □

Observe that in the sampled setting, standard matrix concentration results can be used to show that, while $\boldsymbol{S}$ will be full rank with high probability, it will have two outlying eigenvalues, corresponding to eigenvectors close to $\boldsymbol{q_1}$ and $\boldsymbol{q_2}$ [173]. It is well known that the span of the Krylov matrix $\boldsymbol{T}$ will align well with these outlying eigendirections [158]. Thus, we expect the projection $\boldsymbol{Q}\boldsymbol{Q}^\top\boldsymbol{x} = \boldsymbol{Q}\boldsymbol{Q}^\top\boldsymbol{f} + \boldsymbol{Q}\boldsymbol{Q}^\top\boldsymbol{\eta}$ to still approximately preserve $\boldsymbol{f}$. At the same time, $\boldsymbol{Q}\boldsymbol{Q}^T\boldsymbol{\eta}$ is the projection of a random Gaussian vector $\boldsymbol{\eta}$ onto a fixed K-dimensional subspace. Thus, we will have $\left\|\boldsymbol{Q}\boldsymbol{Q}^\top\boldsymbol{\eta}\right\|_2^2 \approx \frac{K}{n}\left\|\boldsymbol{\eta}\right\|_2^2$, so ASGC will still perform significant denoising when K is small.

Multi-Community Case. We now prove generalizations of the preceding theorems for FSBMs that may have more than 2 communities, i.e., with $r \geq 2$ from Definition 11, and otherwise the same assumptions. The theorem statements and their implications are essentially the same. The proofs are also similar in concept, just using a different projection matrix $\boldsymbol{Q}$, though the proof for the generalized Theorem 5.1.3 is significantly more involved.

Theorem 5.1.3 (Effect of SGC on FSBM Networks)**.** *Consider FSBMs having r equally-sized communities with indicator vectors $\boldsymbol{c_1}, \boldsymbol{c_2}, \ldots, \boldsymbol{c_k}$, expected adjacency matrix $\boldsymbol{A}$, and feature vector $\boldsymbol{x} = \boldsymbol{f} + \boldsymbol{\eta}$. Let $\boldsymbol{x}_{SGC}$ be the feature vector returned by applying SGC, with number of hops K, to $\boldsymbol{A}$ and $\boldsymbol{x}$. Further, let $\bar{\mu} = \frac{1}{r}\sum_i \mu_i$ be the average of the feature means. Then, $\boldsymbol{x}_{SGC} = \boldsymbol{f'} + \sum_i \theta_i \boldsymbol{c_i}$, where $\boldsymbol{f'} = \lambda_2^K \boldsymbol{f} + (1 - \lambda_2^K)(\bar{\mu}\boldsymbol{1})$, and each θ_i is distributed according to $\mathcal{N}\left(0, \frac{1}{n}\left(1 + \lambda_2^{2K}(r-1)\right)\sigma^2\right)$.*

Proof. Let $\hat{\boldsymbol{1}} = (1/\sqrt{n})\boldsymbol{1}$, where $\boldsymbol{1}$ is the n-length all-ones vector. Also let $\boldsymbol{C} = \Big(\boldsymbol{c_1}; \boldsymbol{c_2}; \ldots; \boldsymbol{c_r}\Big)$ and $\boldsymbol{Q} = \left(\sqrt{r/n}\right)\boldsymbol{C}$. Note that $\hat{\boldsymbol{1}}$ has norm 1, and the columns of $\boldsymbol{Q}$ are orthonormal. Finally, let $\lambda_2 = \frac{p-q}{p+(r-1)q}$. We we will not make use of this fact, but, as in the two-community case, this is still the second largest eigenvalue of $\boldsymbol{S}$, and it now has multiplicity $r-1$.

The expected adjacency matrix of the graph is

$$\boldsymbol{A} = (p-q)\boldsymbol{C}\boldsymbol{C}^\top + q\boldsymbol{1}\boldsymbol{1}^\top = (p-q)(n/r)\boldsymbol{Q}\boldsymbol{Q}^\top + qn\hat{\boldsymbol{1}}\hat{\boldsymbol{1}}^\top,$$

and the expected degree vector is

$$\boldsymbol{d} = \boldsymbol{A}\boldsymbol{1} = (p-q)(n/r)\boldsymbol{1} + qn\boldsymbol{1} = (p + (r-1)q)\,(n/r)\boldsymbol{1},$$

yielding the expected normalized adjacency matrix

$$\begin{aligned}\boldsymbol{S} &= \frac{(p-q)(n/r)\boldsymbol{Q}\boldsymbol{Q}^\top + qn\hat{\boldsymbol{1}}\hat{\boldsymbol{1}}^\top}{(p+(r-1)q)\,(n/r)} \\ &= \lambda_2\boldsymbol{Q}\boldsymbol{Q}^\top + \frac{qr}{p+(r-1)q}\hat{\boldsymbol{1}}\hat{\boldsymbol{1}}^\top \\ &= \lambda_2\boldsymbol{Q}\boldsymbol{Q}^\top + (1-\lambda_2)\hat{\boldsymbol{1}}\hat{\boldsymbol{1}}^\top \\ &= \left(\boldsymbol{Q}\boldsymbol{Q}^\top\right)\left(\lambda_2\boldsymbol{I} + (1-\lambda_2)\hat{\boldsymbol{1}}\hat{\boldsymbol{1}}^\top\right).\end{aligned}$$

Note that $\left(\boldsymbol{Q}\boldsymbol{Q}^\top\right)^2 = \boldsymbol{Q}\boldsymbol{Q}^\top$ and $\left(\hat{\mathbf{1}}\hat{\mathbf{1}}^\top\right)^2 = \hat{\mathbf{1}}\hat{\mathbf{1}}^\top$ since these are projection matrices. We show that

$$\left(\lambda_2 \boldsymbol{I} + (1-\lambda_2)\hat{\mathbf{1}}\hat{\mathbf{1}}^\top\right)^K = \lambda_2^K \boldsymbol{I} + \left(1-\lambda_2^K\right)\hat{\mathbf{1}}\hat{\mathbf{1}}^\top$$

by induction as follows:

$$\begin{aligned}
\left(\lambda_2 \boldsymbol{I} + (1-\lambda_2)\hat{\mathbf{1}}\hat{\mathbf{1}}^\top\right)^K &= \left(\lambda_2 \boldsymbol{I} + (1-\lambda_2)\hat{\mathbf{1}}\hat{\mathbf{1}}^\top\right)\left(\lambda_2 \boldsymbol{I} + (1-\lambda_2)\hat{\mathbf{1}}\hat{\mathbf{1}}^\top\right)^{K-1} \\
&= \left(\lambda_2 \boldsymbol{I} + (1-\lambda_2)\hat{\mathbf{1}}\hat{\mathbf{1}}^\top\right)\left(\lambda_2^{K-1} \boldsymbol{I} + \left(1-\lambda_2^{K-1}\right)\hat{\mathbf{1}}\hat{\mathbf{1}}^\top\right) \\
&= \lambda_2^K \boldsymbol{I} + \left(\lambda_2\left(1-\lambda_2^{K-1}\right) + (1-\lambda_2)\lambda_2^{K-1} + (1-\lambda_2)\left(1-\lambda_2^{K-1}\right)\right)\hat{\mathbf{1}}\hat{\mathbf{1}}^\top \\
&= \lambda_2^K \boldsymbol{I} + \left(1-\lambda_2^K\right)\hat{\mathbf{1}}\hat{\mathbf{1}}^\top.
\end{aligned}$$

Using this result and the fact that $\left(\boldsymbol{Q}\boldsymbol{Q}^\top\right)\left(\hat{\mathbf{1}}\hat{\mathbf{1}}^\top\right) = \left(\hat{\mathbf{1}}\hat{\mathbf{1}}^\top\right)\left(\boldsymbol{Q}\boldsymbol{Q}^\top\right) = \hat{\mathbf{1}}\hat{\mathbf{1}}^\top$, we have

$$\begin{aligned}
\boldsymbol{S}^K &= \left(\left(\boldsymbol{Q}\boldsymbol{Q}^\top\right)\left(\lambda_2 \boldsymbol{I} + (1-\lambda_2)\hat{\mathbf{1}}\hat{\mathbf{1}}^\top\right)\right)^K \\
&= \left(\boldsymbol{Q}\boldsymbol{Q}^\top\right)^K\left(\lambda_2 \boldsymbol{I} + (1-\lambda_2)\hat{\mathbf{1}}\hat{\mathbf{1}}^\top\right)^K \\
&= \left(\boldsymbol{Q}\boldsymbol{Q}^\top\right)\left(\lambda_2^K \boldsymbol{I} + \left(1-\lambda_2^K\right)\hat{\mathbf{1}}\hat{\mathbf{1}}^\top\right). \qquad (5.5)
\end{aligned}$$

Now, as in the two-community case, $\boldsymbol{Q}^\top\boldsymbol{\eta} = \boldsymbol{\eta}' \sim \mathcal{N}(0, \sigma^2\boldsymbol{I})$, yielding

$$\begin{aligned}
\boldsymbol{Q}^\top\boldsymbol{x} &= \boldsymbol{Q}^\top\left(\textstyle\sum_i \mu_i \boldsymbol{c_i} + \boldsymbol{\eta}\right) \\
&= \left(\sqrt{r/n}\right)\left(\mu_1, \mu_2, \ldots, \mu_r\right)^\top + \left(\eta_1', \eta_2', \ldots, \eta_r'\right)^\top, \\
\boldsymbol{Q}\boldsymbol{Q}^\top\boldsymbol{x} &= \textstyle\sum_i \left(\mu_i + \left(\sqrt{r/n}\right)\eta_i'\right)\boldsymbol{c_i}, \text{ and} \qquad (5.6) \\
\left(\hat{\mathbf{1}}\hat{\mathbf{1}}^\top\right)\left(\boldsymbol{Q}\boldsymbol{Q}^\top\right)\boldsymbol{x} &= \left(\tfrac{1}{r}\textstyle\sum_i \left(\mu_i + \left(\sqrt{r/n}\right)\eta_i'\right)\right)\mathbf{1}.
\end{aligned}$$

Finally, combining these equations with the expression for $\boldsymbol{S}^K$, we have

$$\begin{aligned}\boldsymbol{x}_{\text{SGC}} = \boldsymbol{S}^K\boldsymbol{x} &= \lambda_2^K \sum\nolimits_i \left(\mu_i + (\sqrt{r/n})\eta_i'\right) \boldsymbol{c_i} + \left(1-\lambda_2^K\right)\left(\tfrac{1}{r}\sum\nolimits_j \left(\mu_j + (\sqrt{r/n})\eta_j'\right)\right)\boldsymbol{1} \\ &= \sum\nolimits_i \left(\lambda_2^K\left(\mu_i + (\sqrt{r/n})\eta_i'\right) + \left(1-\lambda_2^K\right)\cdot\tfrac{1}{r}\sum\nolimits_j \left(\mu_j + (\sqrt{r/n})\eta_j'\right)\right)\boldsymbol{c_i} \\ &= \sum\nolimits_i \left(\lambda_2^K\mu_i + \left(1-\lambda_2^K\right)\bar{\mu} + \lambda_2^K(\sqrt{r/n})\eta_i' + \left(1-\lambda_2^K\right)\cdot\tfrac{1}{r}\sum\nolimits_j (\sqrt{r/n})\eta_j'\right)\boldsymbol{c_i} \\ &= \boldsymbol{f}' + \sum\nolimits_i \left(\lambda_2^K(\sqrt{r/n})\eta_i' + \left(1-\lambda_2^K\right)\cdot\tfrac{1}{r}\sum\nolimits_j (\sqrt{r/n})\eta_j'\right)\boldsymbol{c_i},\end{aligned}$$

so the expectation $\boldsymbol{f}'$ of the filtered feature is as desired. Further, letting the noise term summands be $\theta_i\boldsymbol{c_i}$, we have

$$\begin{aligned}\theta_i &= \lambda_2^K(\sqrt{r/n})\eta_i' + \left(1-\lambda_2^K\right)\cdot\tfrac{1}{r}\sum\nolimits_j \left((\sqrt{r/n})\eta_j'\right) \\ &= \left(\lambda_2^K + \tfrac{1}{r}\left(1-\lambda_2^K\right)\right)(\sqrt{r/n})\eta_i' + \left(1-\lambda_2^K\right)\left(\tfrac{1}{r}\sum\nolimits_{j\neq i}(\sqrt{r/n})\eta_j'\right),\end{aligned}$$

which is normally distributed with mean 0 and variance

$$\begin{aligned}&\left(\lambda_2^K + \tfrac{1}{r}\left(1-\lambda_2^K\right)\right)^2\cdot\tfrac{r}{n}\sigma^2 + \left(1-\lambda_2^K\right)^2\cdot\tfrac{1}{r^2}(r-1)\cdot\tfrac{r}{n}\sigma^2 \\ &= \tfrac{\sigma^2}{n}\left(\left(r\lambda_2^K + \left(1-\lambda_2^K\right)\right)^2\cdot\tfrac{1}{r} + \left(1-\lambda_2^K\right)^2\left(1-\tfrac{1}{r}\right)\right) \\ &= \tfrac{\sigma^2}{n}\left(1 + \lambda_2^{2K}(r-1)\right),\end{aligned}$$

so the noise variance is also as desired. □

Theorem 5.1.4 (Effect of ASGC on FSBM Networks)**.** *Consider FSBMs with $p \neq q$ having r equally-sized communities with indicator vectors $\boldsymbol{c_1}, \boldsymbol{c_2}, \ldots, \boldsymbol{c_k}$, expected adjacency matrix $\boldsymbol{A}$, and feature vector $\boldsymbol{x} = \boldsymbol{f} + \boldsymbol{\eta}$. Let $\boldsymbol{x}_{ASGC}$ be the feature vector returned by applying ASGC, with number of hops $K \geq r$, to $\boldsymbol{A}$ and $\boldsymbol{x}$. Then, $\boldsymbol{x}_{ASGC} = \boldsymbol{f} + \sum_i \theta_i'\boldsymbol{c_i}$, where each θ_i' is distributed according to $\mathcal{N}\left(0, \frac{r}{n}\sigma^2\right)$.*

Proof. Following the same argument as for the two-community case, the least squares in ASGC is equivalent to projecting the feature $\boldsymbol{x}$ onto the column span of the Krylov matrix $\boldsymbol{T} = \left(\boldsymbol{S}^1\boldsymbol{x}; \boldsymbol{S}^2\boldsymbol{x}; \ldots; \boldsymbol{S}^K\boldsymbol{x}\right)$. The column span of $\boldsymbol{T}$ is contained in the column span of $\boldsymbol{S}$, and since $\boldsymbol{S}$ is rank-r (by the assumption that $p \neq q$), with probability 1 over the distribution of $\boldsymbol{\eta}$, as long as $K \geq r$, the column span of $\boldsymbol{T}$ equals that of $\boldsymbol{S}$; by Equation 5.5 for $\boldsymbol{S}$, this span is exactly that of the community indicator matrix $\boldsymbol{Q}$. Thus, ASGC is equivalent to multiplication of the feature $\boldsymbol{x}$ by the projection matrix $\boldsymbol{Q}\boldsymbol{Q}^\top$, for which we use Equation 5.6 as follows:

$$\begin{aligned}
\boldsymbol{x}_{\text{ASGC}} &= \boldsymbol{Q}\boldsymbol{Q}^\top\boldsymbol{x} \\
&= \sum\nolimits_i \left(\mu_i + \left(\sqrt{r/n}\right)\eta_i'\right)\boldsymbol{c_i} \\
&= \boldsymbol{f} + \sum\nolimits_i \left(\sqrt{r/n}\right)\eta_i'\boldsymbol{c_i} \\
&= \boldsymbol{f} + \sum\nolimits_i \theta_i'\boldsymbol{c_i},
\end{aligned}$$

where $\theta_i' = \left(\sqrt{r/n}\right)\eta_i'$, which has the specified distribution. $\square$

5.1.5 Related Work

Deep graph models. As discussed previously, the SGC algorithm is a drastic simplification of the graph convolutional network (GCN) model [94]. GCNs learn a sequence of node representations that evolve via repeated propagation through the graph and nonlinear transformations. The starting node representations $\boldsymbol{H}^{(0)}$ are set to the input feature matrix $\boldsymbol{X} \in \mathbb{R}^{n\times f}$, where f is the number of features. The k^{th}-step representations are $\boldsymbol{H}^{(k)} = \sigma\left(\boldsymbol{S}\boldsymbol{H}^{(k-1)}\boldsymbol{\Theta}^{(k)}\right)$, where $\boldsymbol{\Theta}^{(k)}$ is the learned linear transformation matrix for the k^{th} layer and σ is a nonlinearity like ReLU. After K such steps, the representations are used to classify the nodes via a softmax layer, and the whole model is trained end-to-end via gradient descent. [197] observe that if the nonlinearities are ignored, all of the linear transformations collapse into a single one, while the repeated multiplications by $\boldsymbol{S}$ collapse into a single one by $\boldsymbol{S}^K$; this yields their algorithm of the SGC filter (Equation 5.1) followed by logistic regression. GCNs have spawned streamlined versions like FastGCN [36], as well as more complicated variants like graph isomorphism networks (GINs) [199] and graph attention networks (GATs) [188]; despite being much simpler and faster than these competitors, SGC manages similar performance on common benchmarks, though, based on the analysis of [135], this may be due in part to the simplicity of the benchmark datasets in that they mainly exhibit homophily/assortativity.

Addressing heterophily. Like our work, some other recent methods attempt to address node heterophily. [65] and [211] augment classical feature propagation and GCNs, respectively, to accommodate heterophily by modifying feature propagation based on node classes. [212] and [201] analyze common structures in heterophilous graphs and the failure points of GCNs, then propose GCN variants based on their analyses. The Geom-GCN paper [142] introduces several of the real-world heterophilous networks which are commonly used in related papers, including this one. Their method allows for long-range feature propagation based on similarity of pre-trained node embeddings. The preceding is just a sample of recent works in this area, which has seen a surge of activity [116, 117, 179]. We note that, like the GNN method of [94] but unlike SGC and our ASGC, almost all of these methods are based on deep learning and are trained via backpropagation through repeated feature propagation and linear transformation steps, and hence incur an associated speed and memory requirement. Understanding and implementing these methods is also more complicated relative to our method, which just constitutes a learned feature filter and logistic regression. We mainly compare our results with the deep method which is most similar to ours, Generalized PageRank GNN (GPR-GNN) [38]. Like ASGC, GPR-GNN produces node representations by linear combination of propagated versions of node features; unlike ASGC, the raw features are first transformed by a neural network, and parameters for this network, as well as the linear combination coefficients, are learned by backpropagation using the known node labels. To our knowledge, our work is the first to show that heterophily can be handled using just feature pre-processing.

5.1.6 Empirical Performance

We test the performance of ASGC for the node classification task on a benchmark of real-world datasets given in Table 5.1, and compare with SGC and several deep methods.

Real-world datasets. We experiment on 10 commonly-used datasets, the same collection of datasets as [38]. Cora, Citeseer, and Pubmed are citation networks which are common benchmarks for node classification [166, 130]; these have been used for evaluation on the GCN [94] and GAT [188] papers, in addition to SGC itself. The features are bag-of-words representations of papers, and the node labels give the topics of the paper. Computers and Photo are segments of the Amazon co-purchase graph [121, 168]; features are derived from product reviews, and labels are product categories. These first 5 datasets are considered assortative/homophilous; the remaining 5 datasets, which are disassortative/heterophilous, come from the Geom-GCN paper [142], which also introduces the following measure of of a network's homophily: $H(\mathcal{G}) = \frac{1}{|V|}\sum_{v\in V} \frac{\#\ v\text{'s neighbors with the same label as } v}{\#\ \text{neighbors of } v} \in [0,1]$. We include this statistic in Table 5.1. The latter 5 datasets have much lower values of $H(\mathcal{G})$. Chameleon and Squirrel are hyperlink networks of pages in Wikipedia which concern the two topics [157]. Features derive from text in the pages, and labels correspond to the amount of web traffic to the page, split into 5 categories. Actor is the actor-only induced subgraph of the film-director-actor-writer network of [183]. Nodes and edges represent actors and co-occurrence on a Wikipedia page. Features are based on keywords on the webpage, and labels derive from the number of words on the page, split into 5 categories. Finally, Texas and Cornell are hyperlink networks from university

Table 5.1. Network statistics for experiments in Section 5.1, separated by homophilous vs heterophilous.

Dataset	Cora	Cite.	PubM.	Comp.	Photo	Cham.	Squi.	Actor	Texas	Corn.
Nodes	2708	3327	19717	13752	7650	2277	5201	7600	183	183
Edges	5278	4552	44324	245861	119081	31421	198493	26752	295	280
Features	1433	3703	500	767	745	2325	2089	932	1703	1703
Classes	7	6	3	10	8	5	5	5	5	5
$H(\mathcal{G})$	0.825	0.718	0.792	0.802	0.849	0.247	0.217	0.215	0.057	0.301

websites [42]; features derive from webpage text, and the labels represent the type of page: student, project, course, staff, or faculty.

Implementation. The SGC and ASGC algorithms are implemented in Python using NumPy [79] for least squares regression and other linear algebraic computations. We use scikit-learn [141] for logistic regression with 1,000 maximum iterations and otherwise default settings. For our implementations of SGC and ASGC, we treat each network as undirected, in that if edge (i, j) appears, we also include edge (j, i). Like [38], we use random 60%/20%/20% splits as training/validation/test data for the 5 heterophilous datasets, as in [142], and use random 2.5%/2.5%/95% splits for the homophilous datasets, which is closer to the original setting from [94] and [168]. We release code in the form of a Jupyter notebook [143] demo which is available at github.com/schariya/adaptive-simple-convolution.

Hyperparameter settings. We tune the number of hops over $K \in \{1, 2, 4, 8\}$, roughly covering the range analyzed in [197], and the regularization strength $R = \sqrt{n \cdot R'}$ over $\log_{10}(R') \in \{-4, -3, -2, -1, 0\}$. This dependency on the number of nodes n allows the regularization loss to scale with the least squares loss, which generally grows linearly with n.

Classification results. We apply our implementations of SGC and ASGC to these datasets and report the mean test accuracy across 10 random splits of the data. As a baseline, we also report the accuracy of logistic regression on the 'raw,' unfiltered

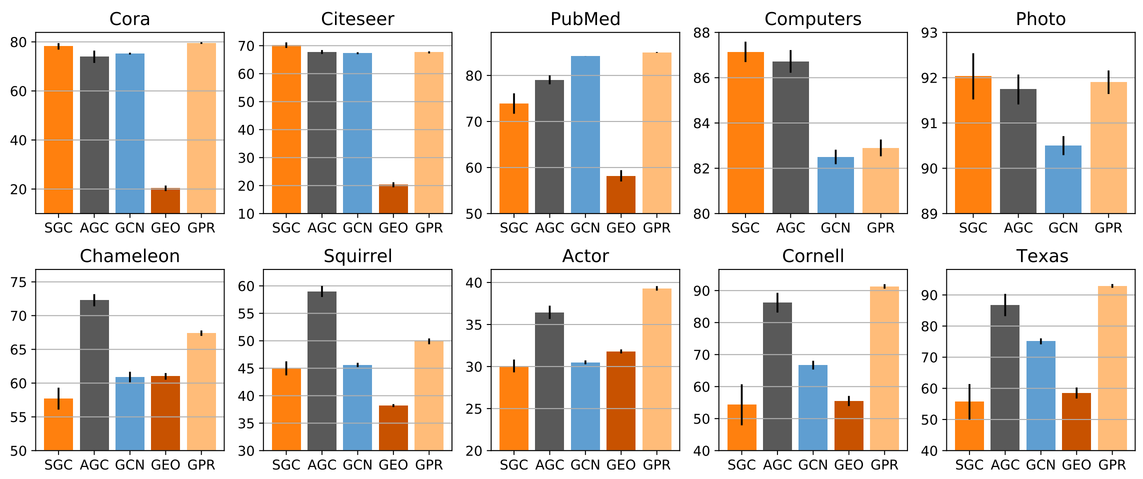


Figure 5.3. Test classification accuracy on the benchmark of datasets from Table 5.1 for selected methods: 2 non-deep, SGC and ASGC; and 3 deep, GCN, Geom-GCN, and GPR-GNN. Error bars show the 95% confidence intervals. SGC is generally competitive with the deep methods on the homophilous datasets (top row), but not so on the heterophilous ones, whereas ASGC is competitive throughout.

features, ignoring the graph. We compare to results from [38] for 9 deep methods applied to these datasets. These methods are 1) a multi-layer perceptron which ignores the graph; 2) GCN; 3) GAT; 4) SAGE [78]; 5) JKNet [200]; 6) GCN-Cheby [46]; 7) Geom-GCN; 8) APPNP [95]; and 9) GPR-GNN.

We plot accuracies for selected methods in Figure 5.3. In addition to SGC and ASGC, we include 3 deep methods: 'vanilla' GCN; Geom-GCN, which originated the heterophilous datasets; and GPR-GNN, a recent method claiming state-of-the-art performance. We find that SGC is generally competitive with the deep methods on the homophilous datasets, but not so on the heterophilous ones, whereas ASGC is generally competitive throughout. Interestingly, the datasets on which GPR-GNN significantly outperforms ASGC (PubMed, Actor, Texas, Cornell) are exactly those on which a multi-layer perceptron significantly outperforms logistic regression; note that the latter two methods both ignore the graph. This suggests that the some nonlinear processing of the node features may be key to performance on these

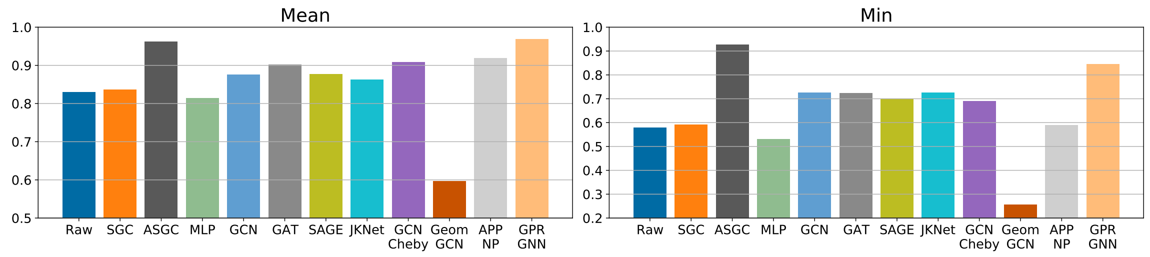


Figure 5.4. Test accuracy as a proportion of the best method's accuracy; mean and minimum performance over the 10 networks. The 3 non-deep (logistic regression) methods are on the left. ASGC achieves the highest minimum performance, and is competitive with GPR-GNN on the mean.

networks, separate from how the graph is exploited. To compactly compare all 12 of the methods across these 10 datasets, we aggregate the performance across the datasets as follows. For each dataset, we calculate the accuracy of each method as a proportion of the accuracy of the best method. We plot the mean and the minimum across the 10 datasets of each method's proportional accuracies. See Figure 5.4. GPR-GNN and ASGC achieve the highest mean performance. Further, ASGC achieves the highest minimum performance: at worst, it achieves over 90% of the best method's test accuracy on each of the datasets.

5.1.7 Conclusion

Building on SGC, we propose a feature filtering technique, ASGC, based on feature propagation and least-squares regression. We propose a natural class of synthetic featured networks, FSBMs, and show both empirically and with theoretical guarantees that ASGC can denoise both homophilous and heterophilous FSBMs, whereas SGC is inappropriate for the latter. Further, we find that ASGC is generally competitive with recent deep learning-based methods on a benchmark of real-world datasets, covering both homophilous and heterophilous networks. Our results suggest that deep learning is not strictly necessary for handling heterophily, and that even a simple feature pre-processing method can be competitive. We hope that, like SGC, ASGC can serve as a good first method to try, especially for node classification on heterophilous networks, and a baseline for future works.

CHAPTER 6
CONCLUSION

To characterize the power and limitations of modern node embedding methods, especially in contrast to classical spectral embedding, we have explored the roles of three central factors: linear algebraic constraints, nonlinearities, and randomness. We find that the constraint of positive semi-definiteness imposed by some embedding methods can be restrictive: perhaps surprisingly, even for undirected graphs, factorizations of the adjacency matrix of the form $\boldsymbol{A} \approx \boldsymbol{X}\boldsymbol{Y}^\top$ can be significantly more expressive than those of the form $\boldsymbol{A} \approx \boldsymbol{X}\boldsymbol{X}^\top$. In Sections 3.1 and 3.2, we show that, when coupled with a nonlinearity, the former is provably highly expressive: we provide several guarantees for exact embedding in the $\boldsymbol{A} \approx \sigma(\boldsymbol{X}\boldsymbol{Y}^\top)$ and $\boldsymbol{A} \approx \sigma(\boldsymbol{B}\boldsymbol{B}^\top - \boldsymbol{C}\boldsymbol{C}^\top)$ models. In Section 2.1, we also find empirically that much of the performance gain of modern methods, e.g., in node classification tasks, can be attributed to the (possibly implicit) addition of a nonlinearity, and we explore possible explanations in that section and in Section 2.2. Finally, we look beyond matching a single graph to expressing distributions over graphs, which can be seen as arising from graph models with random node embeddings rather than deterministic ones. In Section 4.1, we establish an inherent trade-off between a model's ability to produce a diverse set of graphs, and its ability to produce graphs with certain (often realistic) properties such as high triangle count. We also show, in Section 4.2, how more complex dependency structures amongst the random node embeddings' distributions can yield provably more expressive graph models.

Future directions include extensions of the expressiveness guarantees, which could come in several varieties. First, the guarantees in this thesis all assume that the embeddings can be represented with infinite precision. Considering limited precision would better reflect real-world usage, and may also inspire new constructions of exact embeddings, similar to the polynomial interpolation-based ones in Sections 3.1 and 3.2, but with better performance in downstream applications. Further, we only look here at guarantees of *capacity* of certain graph models, but we have not provided any guarantees for *fitting* the models. The possibility of such guarantees raises several questions. For example, provided a k-dimensional exact embedding of a graph exists, under what circumstances is gradient descent guaranteed to find an exact embedding? Also, more broadly, how do embeddings produced by gradient descent differ from the ones produced by our polynomial-based constructions? Finally, as perhaps the most straightforward kind of extension, here we focus on guarantees for embedding the simplest kind of undirected graphs, but there are other settings to explore, including directed graphs, temporal graphs, multiplex graphs, etc.

Shifting to longer-term implications and directions from this thesis, much of the work here explores what is gained by nonlinearity, e.g., the power of $\boldsymbol{A} \approx \sigma(\boldsymbol{X}\boldsymbol{Y}^\top)$ vs $\boldsymbol{A} \approx \boldsymbol{X}\boldsymbol{Y}^\top$. While we can prove that the former model is significantly more expressive than the latter, adding nonlinearity is less fruitful in other settings: as we show in Section 5.1, for filtering of node features, a linear algebraic method can compete with deep methods. Given that nonlinearity can add computational complexity, and can make theoretical analysis more difficult, a potentially fruitful goal is achieving a *predictive* understanding of when and what kind of nonlinearity is necessary for different graph tasks. Besides this, focusing on an important case where we have shown that nonlinearity is fruitful – exact low-rank representation of graphs with the $\boldsymbol{A} \approx \sigma(\boldsymbol{X}\boldsymbol{Y}^\top)$ model – another direction is treating such an exact embedding as a way of representing the graph, and developing mathematical tools to exploit this representation for computational gains. For example, multiplying vectors by the adjacency matrix of a graph ("matvecs" with $\boldsymbol{A}$), are a key primitive in various spectral graph methods – can these matvecs be sped up if provided an exact embedding of the graph? Specifically, given a vector $\boldsymbol{v} \in \mathbb{R}^n$ and an exact embedding $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n \times k}$ with $k \ll n$, is it possible to exploit the low-rank structure and approximate the matvec $\boldsymbol{A}\boldsymbol{v} \approx \sigma(\boldsymbol{X}\boldsymbol{Y}^\top)\, \boldsymbol{v}$ in sub-quadratic, that is, $o(n^2)$, time? If so, the guarantees we provide here on the *capacity* of nonlinear low-rank factorizations of graphs may be channeled into new *runtime* guarantees for graph algorithms. Finally, we look towards applications of the results in this thesis beyond machine learning on graph data. In particular, graph methods are closely related to kernel machines and self-attention, as each of these three areas centers on some similarity matrix. For this reason, perhaps our approaches here can yield some fresh insights for these methods as well.

BIBLIOGRAPHY

[1] Abbe, Emmanuel. Community detection and stochastic block models: recent developments. *The Journal of Machine Learning Research 18*, 1 (2017), 6446–6531.

[2] Abbe, Emmanuel, Bandeira, Afonso S, and Hall, Georgina. Exact recovery in the stochastic block model. *IEEE Transactions on Information Theory 62*, 1 (2015), 471–487.

[3] Adamic, Lada A, and Glance, Natalie. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd International Workshop on Link Discovery* (2005), pp. 36–43.

[4] Agarwal, Nitin, Liu, Huan, Murthy, Sudheendra, Sen, Arunabha, and Wang, Xufei. A social identity approach to identify familiar strangers in a social network. In *Third International AAAI Conference on Weblogs and Social Media* (2009).

[5] Aggarwal, Charu C, and Wang, Haixun. A survey of clustering algorithms for graph data. In *Managing and Mining Graph Data*. Springer, 2010, pp. 275–301.

[6] Aiello, William, Chung, Fan, and Lu, Linyuan. A random graph model for power law graphs. *Experimental mathematics 10*, 1 (2001), 53–66.

[7] Airoldi, Edoardo M, Blei, David M, Fienberg, Stephen E, and Xing, Eric P. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research 9*, Sep (2008), 1981–2014.

[8] Allen, Carl, Balazevic, Ivana, and Hospedales, Timothy. What the vec? towards probabilistically grounded embeddings. In *Advances in Neural Information Processing Systems* (2019), pp. 7465–7475.

[9] Allen, Carl, and Hospedales, Timothy. Analogies explained: Towards understanding word embeddings. In *International Conference on Machine Learning* (2019), pp. 223–231.

[10] Alon, Noga, Frankl, Peter, and Rodl, V. Geometrical realization of set systems and probabilistic communication complexity. In *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science (FOCS)* (1985), pp. 277–280.

[11] Alon, Noga, and Milman, Vitali D. λ_1, isoperimetric inequalities for graphs, and superconcentrators. *Journal of Combinatorial Theory, Series B 38*, 1 (1985), 73–88.

[12] Alon, Noga, Moran, Shay, and Yehudayoff, Amir. Sign rank versus VC dimension. In *Proceedings of the 29th Annual Conference on Computational Learning Theory (COLT)* (2016), pp. 47–80.

[13] Alon, Noga, and Spencer, Joel H. *The probabilistic method.* John Wiley & Sons, 2016.

[14] Alstott, Jeff, Bullmore, Ed, and Plenz, Dietmar. powerlaw: a python package for analysis of heavy-tailed distributions. *PloS one 9*, 1 (2014), e85777.

[15] Arora, Sanjeev, Li, Yuanzhi, Liang, Yingyu, Ma, Tengyu, and Risteski, Andrej. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics 4* (2016), 385–399.

[16] Barabási, Albert-László, and Albert, Réka. Emergence of scaling in random networks. *Science 286*, 5439 (1999), 509–512.

[17] Belkin, Mikhail, and Niyogi, Partha. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation 15*, 6 (2003), 1373–1396.

[18] Berry, Michael W, Browne, Murray, Langville, Amy N, Pauca, V Paul, and Plemmons, Robert J. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis 52*, 1 (2007), 155–173.

[19] Bhattacharjee, Robi, and Dasgupta, Sanjoy. What relations are reliably embeddable in euclidean space? In *Algorithmic Learning Theory* (2020), PMLR, pp. 174–195.

[20] Bojchevski, Aleksandar, Shchur, Oleksandr, Zügner, Daniel, and Günnemann, Stephan. NetGAN: Generating graphs via random walks. *Proceedings of the 35th International Conference on Machine Learning (ICML)* (2018).

[21] Bollobás, Béla, Riordan, Oliver, Spencer, Joel, and Tusnády, Gábor. The degree sequence of a scale-free random graph process. *Random Structures & Algorithms 18*, 3 (2001), 279–290.

[22] Bonato, Anthony. A survey of models of the web graph. In *Workshop on Combinatorial and Algorithmic Aspects of Networking* (2004), Springer, pp. 159–172.

[23] Boratko, Michael, Zhang, Dongxu, Monath, Nicholas, Vilnis, Luke, Clarkson, Kenneth L, and McCallum, Andrew. Capacity and bias of learned geometric embeddings for directed graphs. *Advances in Neural Information Processing Systems 34* (2021), 16423–16436.

[24] Bourgain, Jean. On Lipschitz embedding of finite metric spaces in Hilbert space. *Israel Journal of Mathematics 52*, 1-2 (1985), 46–52.

[25] Breitkreutz, Bobby-Joe, Stark, Chris, Reguly, Teresa, Boucher, Lorrie, Breitkreutz, Ashton, Livstone, Michael, Oughtred, Rose, Lackner, Daniel H, Bähler, Jürg, Wood, Valerie, et al. The biogrid interaction database: 2008 update. *Nucleic Acids Research 36*, suppl_1 (2007), D637–D640.

[26] Buckley, Pierce G, and Osthus, Deryk. Popularity based random graph models leading to a scale-free degree sequence. *Discrete Mathematics 282*, 1-3 (2004), 53–68.

[27] Bun, Mark, and Thaler, Justin. Improved bounds on the sign-rank of AC^0. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP)* (2016).

[28] Cao, Shaosheng, Lu, Wei, and Xu, Qiongkai. GraRep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management* (2015), pp. 891–900.

[29] Cao, Shaosheng, Lu, Wei, and Xu, Qiongkai. Deep neural networks for learning graph representations. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)* (2016).

[30] Chanpuriya, Sudhanshu, and Musco, Cameron. 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).

[31] Chanpuriya, Sudhanshu, and Musco, Cameron. Simplified graph convolution with heterophily. In *Advances in Neural Information Processing Systems 35 (NeurIPS)* (2022).

[32] Chanpuriya, Sudhanshu, Musco, Cameron, Sotiropoulos, Konstantinos, and Tsourakakis, Charalampos. Deepwalking backwards: from embeddings back to graphs. In *International Conference on Machine Learning* (2021), PMLR, pp. 1473–1483.

[33] Chanpuriya, Sudhanshu, Musco, Cameron, Sotiropoulos, Konstantinos, and Tsourakakis, Charalampos. On the power of edge independent graph models. *Advances in Neural Information Processing Systems 34* (2021), 24418–24429.

[34] Chanpuriya, Sudhanshu, Musco, Cameron, Sotiropoulos, Konstantinos, and Tsourakakis, Charalampos E. Node embeddings and exact low-rank representations of complex networks. In *Advances in Neural Information Processing Systems 33 (NeurIPS)* (2020).

[35] Chatterjee, Sourav, and Diaconis, Persi. Estimating and understanding exponential random graph models. *The Annals of Statistics* (2013), 2428–2461.

[36] Chen, Jie, Ma, Tengfei, and Xiao, Cao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *International Conference on Learning Representations* (2018).

[37] Chen, Wei, Fang, Wenjie, Hu, Guangda, and Mahoney, Michael W. On the hyperbolicity of small-world and treelike random graphs. *Internet Mathematics 9*, 4 (2013), 434–491.

[38] Chien, Eli, Peng, Jianhao, Li, Pan, and Milenkovic, Olgica. Adaptive universal generalized pagerank graph neural network. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021* (2021).

[39] Chung, Fan, and Lu, Linyuan. The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences 99*, 25 (2002), 15879–15882.

[40] Chung, Fan, and Lu, Linyuan. Connected components in random graphs with given expected degree sequences. *Annals of combinatorics 6*, 2 (2002), 125–145.

[41] Chung, Fan RK, Graham, Ronald L, Frankl, Peter, and Shearer, James B. Some intersection theorems for ordered sets and graphs. *Journal of Combinatorial Theory, Series A 43*, 1 (1986), 23–37.

[42] Craven, Mark, DiPasquo, Dan, Freitag, Dayne, McCallum, Andrew, Mitchell, Tom, Nigam, Kamal, and Slattery, Seán. Learning to construct knowledge bases from the world wide web. *Artificial intelligence 118*, 1-2 (2000), 69–113.

[43] Dall, Jesper, and Christensen, Michael. Random geometric graphs. *Physical Review E 66*, 1 (2002), 016121.

[44] De Bie, Tijl. Maximum entropy models and subjective interestingness: an application to tiles in binary databases. *Data Mining and Knowledge Discovery 23*, 3 (2011), 407–446.

[45] De Cao, Nicola, and Kipf, Thomas. MolGAN: An implicit generative model for small molecular graphs. *ICML Deep Generative Models Workshop* (2018).

[46] Defferrard, Michaël, Bresson, Xavier, and Vandergheynst, Pierre. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain* (2016), pp. 3837–3845.

[47] Deshpande, Yash, Sen, Subhabrata, Montanari, Andrea, and Mossel, Elchanan. Contextual stochastic block models. In *Advances in Neural Information Processing Systems* (2018).

[48] Ding, Chris, Li, Tao, and Jordan, Michael I. Nonnegative matrix factorization for combinatorial optimization: Spectral clustering, graph matching, and clique finding. In *2008 Eighth IEEE International Conference on Data Mining* (2008), IEEE, pp. 183–192.

[49] Donoho, David L., and Stodden, Victoria. When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in Neural Information Processing Systems 16* (2003), MIT Press, pp. 1141–1148.

[50] Drinea, Eleni, Enachescu, Mihaela, and Mitzenmacher, Michael. Variations on random graph models for the web. In *SODA* (2001), Citeseer.

[51] Duddu, Vasisht, Boutet, Antoine, and Shejwalkar, Virat. Quantifying privacy leakage in graph embedding. *arXiv:2010.00906* (2020).

[52] Durak, Nurcan, Pinar, Ali, Kolda, Tamara G, and Seshadhri, C. Degree relations of triangles in real-world networks and graph models. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management* (2012).

[53] Duvenaud, David, Maclaurin, Dougal, Aguilera-Iparraguirre, Jorge, Gómez-Bombarelli, Rafael, Hirzel, Timothy, Aspuru-Guzik, Alán, and Adams, Ryan P. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems 28* (2015), pp. 2224–2232.

[54] Ellers, Michael, Cochez, Michael, Schumacher, Tobias, Strohmaier, Markus, and Lemmerich, Florian. Privacy attacks on network embeddings. *arXiv:1912.10979* (2019).

[55] Eppstein, David, Löffler, Maarten, and Strash, Darren. Listing all maximal cliques in sparse graphs in near-optimal time. *arXiv preprint arXiv:1006.5440* (2010).

[56] Erdös, Paul, and Rényi, Alfréd. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences 5*, 1 (1960), 17–60.

[57] Faloutsos, Christos, Miller, Gary, and Tsourakakis, Charalampos Babis. Large graph-mining: Power tools and a practitioner's guide. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)* (2009), Citeseer.

[58] Fan, Rong-En, Chang, Kai-Wei, Hsieh, Cho-Jui, Wang, Xiang-Rui, and Lin, Chih-Jen. Liblinear: A library for large linear classification. *Journal of machine learning research 9*, Aug (2008), 1871–1874.

[59] Finner, Helmut. A generalization of holder's inequality and some probability inequalities. *The Annals of Probability* (1992), 1893–1901.

[60] Flaxman, Abraham, Frieze, Alan, and Fenner, Trevor. High degree vertices and eigenvalues in the preferential attachment graph. *Internet Mathematics 2*, 1 (2005), 1–19.

[61] Foster, Jacob G, Foster, David V, Grassberger, Peter, and Paczuski, Maya. Edge direction and the structure of networks. *Proceedings of the National Academy of Sciences 107*, 24 (2010), 10815–10820.

[62] Frank, Ove, and Strauss, David. Markov graphs. *Journal of the american Statistical association 81*, 395 (1986), 832–842.

[63] Friedgut, Ehud. Hypergraphs, entropy, and inequalities. *The American Mathematical Monthly 111*, 9 (2004), 749–760.

[64] Frieze, Alan, and Tsourakakis, Charalampos E. On certain properties of random apollonian networks. In *International Workshop on Algorithms and Models for the Web-Graph* (2012), Springer, pp. 93–112.

[65] Gatterbauer, Wolfgang. Semi-supervised learning with heterophily. *arXiv preprint arXiv:1412.3100* (2014).

[66] Gillis, Nicolas. *Nonnegative Matrix Factorization*. SIAM, 2020.

[67] Gionis, Aristides, and Tsourakakis, Charalampos E. Dense subgraph discovery: Kdd 2015 tutorial. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2015), pp. 2313–2314.

[68] Gittens, Alex, Achlioptas, Dimitris, and Mahoney, Michael W. Skip-gram- zipf+ uniform= vector additivity. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2017), pp. 69–76.

[69] Gleich, David, Zhukov, Leonid, and Berkhin, Pavel. Fast parallel pagerank: A linear system approach. *Yahoo! Research Technical Report 13* (2004), 22.

[70] Goemans, Michel X, and Williamson, David P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM) 42*, 6 (1995), 1115–1145.

[71] Goldberg, Yoav, and Levy, Omer. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722* (2014).

[72] Goldenberg, Anna, Zheng, Alice X, Fienberg, Stephen E, Airoldi, Edoardo M, et al. A survey of statistical network models. *Foundations and Trends® in Machine Learning 2*, 2 (2010), 129–233.

[73] Gopalan, Prem K, and Blei, David M. Efficient discovery of overlapping communities in massive networks. *Proceedings of the National Academy of Sciences 110*, 36 (2013), 14534–14539.

[74] Grohe, Martin. *Descriptive complexity, canonisation, and definable graph structure theory*, vol. 47. Cambridge University Press, 2017.

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

[76] Grover, Aditya, Zweig, Aaron, and Ermon, Stefano. Graphite: Iterative generative modeling of graphs. In *Proceedings of the 36th International Conference on Machine Learning (ICML)* (2019).

[77] Gupta, Anupam. Embedding tree metrics into low dimensional euclidean spaces. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing* (1999), pp. 694–700.

[78] Hamilton, Will, Ying, Zhitao, and Leskovec, Jure. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30 (NeurIPS)* (2017), pp. 1024–1034.

[79] Harris, Charles R., Millman, K. Jarrod, van der Walt, Stéfan J., Gommers, Ralf, Virtanen, Pauli, Cournapeau, David, Wieser, Eric, Taylor, Julian, Berg, Sebastian, Smith, Nathaniel J., Kern, Robert, Picus, Matti, Hoyer, Stephan, van Kerkwijk, Marten H., Brett, Matthew, Haldane, Allan, del Río, Jaime Fernández, Wiebe, Mark, Peterson, Pearu, Gérard-Marchant, Pierre, Sheppard, Kevin, Reddy, Tyler, Weckesser, Warren, Abbasi, Hameer, Gohlke, Christoph, and Oliphant, Travis E. Array programming with NumPy. *Nature 585*, 7825 (Sept. 2020), 357–362.

[80] Hase, Takeshi, Niimura, Yoshihito, and Tanaka, Hiroshi. Difference in gene duplicability may explain the difference in overall structure of protein-protein interaction networks among eukaryotes. *BMC Evolutionary Biology 10*, 1 (2010), 1–15.

[81] Hoff, Peter D. Random effects models for network data. In *Dynamic Social Network Modeling and Analysis: Workshop Summary and Papers* (2003), Citeseer.

[82] Hoff, Peter D. Bilinear mixed-effects models for dyadic data. *Journal of the American Statistical Association 100*, 469 (2005), 286–295.

[83] Hoff, Peter D, Raftery, Adrian E, and Handcock, Mark S. Latent space approaches to social network analysis. *Journal of the American Statistical association 97*, 460 (2002), 1090–1098.

[84] Holland, Paul W, Laskey, Kathryn Blackmond, and Leinhardt, Samuel. Stochastic blockmodels: First steps. *Social Networks 5*, 2 (1983), 109–137.

[85] Hoskins, Jeremy G, Musco, Cameron, Musco, Christopher, and Tsourakakis, Charalampos E. Learning networks from random walk-based node similarities. In *Advances in Neural Information Processing Systems 31 (NeurIPS)* (2018).

[86] Huang, Weiyu, Goldsberry, Leah, Wymbs, Nicholas F, Grafton, Scott T, Bassett, Danielle S, and Ribeiro, Alejandro. Graph frequency analysis of brain signals. *IEEE Journal of Selected Topics in Signal Processing 10*, 7 (2016), 1189–1203.

[87] Javed, Muhammad Aqib, Younis, Muhammad Shahzad, Latif, Siddique, Qadir, Junaid, and Baig, Adeel. Community detection in networks: A multidisciplinary review. *Journal of Network and Computer Applications 108* (2018), 87–111.

[88] Johnson, Samuel, Torres, Joaquín J, Marro, J, and Munoz, Miguel A. Entropic origin of disassortativity in complex networks. *Physical Review Letters 104*, 10 (2010), 108702.

[89] Jones, Eric, Oliphant, Travis, Peterson, Pearu, et al. SciPy: Open source scientific tools for Python, 2001.

[90] Kang, Ross J, and Müller, Tobias. Sphere and dot product representations of graphs. In *Proceedings of the Twenty-Seventh Annual Symposium on Computational Geometry* (2011), pp. 308–314.

[91] Kelner, Jonathan A, Orecchia, Lorenzo, Sidford, Aaron, and Zhu, Zeyuan Allen. A simple, combinatorial algorithm for solving sdd systems in nearly-linear time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing* (2013), pp. 911–920.

[92] Kemp, Charles, Tenenbaum, Joshua B, Griffiths, Thomas L, Yamada, Takeshi, and Ueda, Naonori. Learning systems of concepts with an infinite relational model. In *Proceedings of the 20th AAAI Conference on Artificial Intelligence (AAAI)* (2006).

[93] Kipf, Thomas N, and Welling, Max. Variational graph auto-encoders. *NeurIPS Bayesian Deep Learning Workshop* (2016).

[94] Kipf, Thomas N, and Welling, Max. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations* (2017).

[95] Klicpera, Johannes, Bojchevski, Aleksandar, and Günnemann, Stephan. Predict then propagate: Graph neural networks meet personalized pagerank. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019* (2019).

[96] Koutis, Ioannis, Miller, Gary L, and Peng, Richard. Approaching optimality for solving sdd linear systems. *SIAM Journal on Computing 43*, 1 (2014), 337–354.

[97] Kuang, Da, Ding, Chris, and Park, Haesun. Symmetric nonnegative matrix factorization for graph clustering. In *Proceedings of the 2012 SIAM International Conference on Data Mining* (2012), SIAM, pp. 106–117.

[98] Kuang, Da, Yun, Sangwoon, and Park, Haesun. Symnmf: nonnegative low-rank approximation of a similarity matrix for graph clustering. *Journal of Global Optimization 62*, 3 (2015), 545–574.

[99] LeCun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. Deep learning. *Nature 521*, 7553 (2015), 436–444.

[100] Lee, Daniel D, and Seung, H Sebastian. Learning the parts of objects by non-negative matrix factorization. *Nature 401*, 6755 (1999), 788–791.

[101] Lehoucq, Richard B, Sorensen, Danny C, and Yang, Chao. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*, vol. 6. Siam, 1998.

[102] Leighton, Tom, and Rao, Satish. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM (JACM) 46*, 6 (1999), 787–832.

[103] Leskovec, Jure, Chakrabarti, Deepayan, Kleinberg, Jon, Faloutsos, Christos, and Ghahramani, Zoubin. Kronecker graphs: an approach to modeling networks. *Journal of Machine Learning Research 11*, 2 (2010).

[104] Leskovec, Jure, Huttenlocher, Daniel, and Kleinberg, Jon. Signed networks in social media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2010), pp. 1361–1370.

[105] Leskovec, Jure, Kleinberg, Jon, and Faloutsos, Christos. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD) 1*, 1 (2007), 2–es.

[106] Leskovec, Jure, and Krevl, Andrej. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, 2014.

[107] Leskovec, Jure, and Mcauley, Julian J. Learning to discover social circles in ego networks. In *Advances in Neural Information Processing Systems 25 (NeurIPS)* (2012), pp. 539–547.

[108] Levinson, Howard. An eigenvalue representation for random walk hitting times and its application to the rook graph.

[109] Levy, Omer, and Goldberg, Yoav. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27 (NeurIPS)* (2014), pp. 2177–2185.

[110] Li, Yitan, Xu, Linli, Tian, Fei, Jiang, Liang, Zhong, Xiaowei, and Chen, Enhong. Word embedding revisited: A new representation learning and explicit matrix factorization perspective. In *Twenty-Fourth International Joint Conference on Artificial Intelligence* (2015).

[111] Li, Yujia, Vinyals, Oriol, Dyer, Chris, Pascanu, Razvan, and Battaglia, Peter. Learning deep generative models of graphs. *arXiv:1803.03324* (2018).

[112] Liao, Renjie, Li, Yujia, Song, Yang, Wang, Shenlong, Nash, Charlie, Hamilton, William L, Duvenaud, David, Urtasun, Raquel, and Zemel, Richard S. Efficient graph generation with graph recurrent attention networks. *arXiv:1910.00760* (2019).

[113] Linial, Nathan, London, Eran, and Rabinovich, Yuri. The geometry of graphs and some of its algorithmic applications. *Combinatorica 15*, 2 (1995), 215–245.

[114] Linial, Nati, Mendelson, Shahar, Schechtman, Gideon, and Shraibman, Adi. Complexity measures of sign matrices. *Combinatorica 27*, 4 (2007), 439–463.

[115] Liu, Dong C, and Nocedal, Jorge. On the limited memory BFGS method for large scale optimization. *Mathematical Programming 45*, 1-3 (1989), 503–528.

[116] Liu, Meng, Wang, Zhengyang, and Ji, Shuiwang. Non-local graph neural networks. *arXiv preprint arXiv:2005.14612* (2020).

[117] Luan, Sitao, Hua, Chenqing, Lu, Qincheng, Zhu, Jiaqi, Zhao, Mingde, Zhang, Shuyuan, Chang, Xiao-Wen, and Precup, Doina. Is heterophily a real nightmare for graph neural networks to do node classification? *arXiv preprint arXiv:2109.05641* (2021).

[118] Ma, Zhuang, Ma, Zongming, and Yuan, Hongsong. Universal latent space model fitting for large networks with edge covariates. *J. Mach. Learn. Res. 21* (2020), 4–1.

[119] Maehara, Hiroshi. Space graphs and sphericity. *Discrete Applied Mathematics 7*, 1 (1984), 55–64.

[120] Mason, Oliver, and Verwoerd, Mark. Graph theory and networks in biology. *IET Systems Biology 1*, 2 (2007), 89–119.

[121] McAuley, Julian, Targett, Christopher, Shi, Qinfeng, and Van Den Hengel, Anton. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information retrieval* (2015), pp. 43–52.

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

[123] McPherson, Miller, Smith-Lovin, Lynn, and Cook, James M. Birds of a feather: Homophily in social networks. *Annual review of sociology 27*, 1 (2001), 415–444.

[124] McSherry, Frank. Spectral partitioning of random graphs. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)* (2001), pp. 529–537.

[125] Meyer, Jr, Carl D. Generalized inversion of modified matrices. *SIAM Journal on Applied Mathematics 24*, 3 (1973), 315–323.

[126] Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S, and Dean, Jeff. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26 (NeurIPS)* (2013), pp. 3111–3119.

[127] Miller, Kurt, Jordan, Michael I, and Griffiths, Thomas L. Nonparametric latent feature models for link prediction. In *Advances in Neural Information Processing Systems 22 (NeurIPS)* (2009), pp. 1276–1284.

[128] Morris, Christopher, Ritzert, Martin, Fey, Matthias, Hamilton, William L, Lenssen, Jan Eric, Rattan, Gaurav, and Grohe, Martin. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the −1953th AAAI Conference on Artificial Intelligence (AAAI)* (2019), vol. 33, pp. 4602–4609.

[129] Musco, Cameron, Musco, Christopher, and Tsourakakis, Charalampos E. Minimizing polarization and disagreement in social networks. In *Proceedings of the 27th International World Wide Web Conference (WWW)* (2018), pp. 369–378.

[130] Namata, Galileo, London, Ben, Getoor, Lise, Huang, Bert, and EDU, UMD. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs* (2012), vol. 8, p. 1.

[131] Nascimento, Maria CV, and De Carvalho, Andre CPLF. Spectral methods for graph clustering–a survey. *European Journal of Operational Research 211*, 2 (2011), 221–231.

[132] Newman, Mark EJ. Assortative mixing in networks. *Physical Review Letters 89*, 20 (2002), 208701.

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

[134] Noldus, Rogier, and Van Mieghem, Piet. Assortativity in complex networks. *Journal of Complex Networks 3*, 4 (2015), 507–542.

[135] NT, Hoang, and Maehara, Takanori. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550* (2019).

[136] Oliphant, Travis E. *A guide to NumPy*, vol. 1. Trelgol Publishing USA, 2006.

[137] Ortega, Antonio, Frossard, Pascal, Kovačević, Jelena, Moura, José MF, and Vandergheynst, Pierre. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE 106*, 5 (2018), 808–828.

[138] Palla, Konstantina, Knowles, David, and Ghahramani, Zoubin. An infinite latent attribute model for network data. *arXiv:1206.6416* (2012).

[139] Pan, Shirui, Hu, Ruiqi, Long, Guodong, Jiang, Jing, Yao, Lina, and Zhang, Chengqi. Adversarially regularized graph autoencoder for graph embedding. *arXiv:1802.04407* (2018).

[140] Paszke, Adam, Gross, Sam, Massa, Francisco, Lerer, Adam, Bradbury, James, Chanan, Gregory, Killeen, Trevor, Lin, Zeming, Gimelshein, Natalia, Antiga, Luca, Desmaison, Alban, Kopf, Andreas, Yang, Edward, DeVito, Zachary, Raison, Martin, Tejani, Alykhan, Chilamkurthy, Sasank, Steiner, Benoit, Fang, Lu, Bai, Junjie, and Chintala, Soumith. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32 (NeurIPS)* (2019), pp. 8024–8035.

[141] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[142] Pei, Hongbin, Wei, Bingzhe, Chang, Kevin Chen-Chuan, Lei, Yu, and Yang, Bo. Geom-gcn: Geometric graph convolutional networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020* (2020).

[143] Pérez, Fernando, and Granger, Brian E. IPython: a system for interactive scientific computing. *Computing in Science and Engineering 9*, 3 (May 2007), 21–29.

[144] Perozzi, Bryan, Al-Rfou, Rami, and Skiena, Steven. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014), ACM, pp. 701–710.

[145] Peysakhovich, Alexander, and Bottou, Leon. An attract-repel decomposition of undirected networks. *arXiv preprint arXiv:2106.09671* (2021).

[146] Pinar, Ali, Seshadhri, Comandur, and Kolda, Tamara G. The similarity between stochastic Kronecker and Chung-Lu graph models. In *Proceedings of the 2012 SIAM International Conference on Data Mining* (2012).

[147] Qiu, Jiezhong, Dong, Yuxiao, Ma, Hao, Li, Jian, Wang, Kuansan, and Tang, Jie. 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* (2018), ACM, pp. 459–467.

[148] Ranjan, Gyan, Zhang, Zhi-Li, and Boley, Daniel. Incremental computation of pseudo-inverse of laplacian. In *International Conference on Combinatorial Optimization and Applications* (2014), Springer, pp. 729–749.

[149] Razborov, Alexander A, and Sherstov, Alexander A. The sign-rank of AC^0. *SIAM Journal on Computing 39*, 5 (2010), 1833–1855.

[150] Reiterman, Jan, Rödl, Vojtech, and Šinajová, E. Geometrical embeddings of graphs. *Discrete Mathematics 74*, 3 (1989), 291–319.

[151] Rendsburg, Luca, Heidrich, Holger, and von Luxburg, Ulrike. NetGAN without GAN: From random walks to low-rank approximations. In *Proceedings of the 37th International Conference on Machine Learning (ICML)* (2020).

[152] Rendsburg, Luca, Heidrich, Holger, and Von Luxburg, Ulrike. Netgan without gan: From random walks to low-rank approximations. In *International Conference on Machine Learning* (2020), PMLR, pp. 8073–8082.

[153] Roberts, Fred S. On the boxicity and cubicity of a graph. *Recent progress in combinatorics 1*, 1 (1969), 301–310.

[154] Rohe, Karl, Chatterjee, Sourav, Yu, Bin, et al. Spectral clustering and the high-dimensional stochastic blockmodel. *The Annals of Statistics 39*, 4 (2011), 1878–1915.

[155] Rossi, Ryan A., and Ahmed, Nesreen K. The network data repository with interactive graph analytics and visualization. In *AAAI* (2015).

[156] Roweis, Sam T, and Saul, Lawrence K. Nonlinear dimensionality reduction by locally linear embedding. *Science 290*, 5500 (2000), 2323–2326.

[157] Rozemberczki, Benedek, Allen, Carl, and Sarkar, Rik. Multi-scale attributed node embedding. *J. Complex Networks 9*, 2 (2021).

[158] Saad, Yousef. *Numerical methods for large eigenvalue problems: revised edition.* SIAM, 2011.

[159] Sala, Alessandra, Cao, Lili, Wilson, Christo, Zablit, Robert, Zheng, Haitao, and Zhao, Ben Y. Measurement-calibrated graph models for social network experiments. In *Proceedings of the 19th International World Wide Web Conference (WWW)* (2010), pp. 861–870.

[160] Sala, Alessandra, Cao, Lili, Wilson, Christo, Zablit, Robert, Zheng, Haitao, and Zhao, Ben Y. Measurement-calibrated graph models for social network experiments. In *Proceedings of the 19th International World Wide Web Conference (WWW)* (2010).

[161] Sala, Frederic, De Sa, Chris, Gu, Albert, and Ré, Christopher. Representation tradeoffs for hyperbolic embeddings. In *International Conference on Machine Learning* (2018), PMLR, pp. 4460–4469.

[162] Salha, Guillaume, Hennequin, Romain, and Vazirgiannis, Michalis. Keep it simple: Graph autoencoders without graph convolutional networks. *arXiv:1910.00942* (2019).

[163] Sarkar, Rik. Low distortion delaunay embedding of trees in hyperbolic plane. In *International Symposium on Graph Drawing* (2011), Springer, pp. 355–366.

[164] Schaeffer, Satu Elisa. Graph clustering. *Computer Science Review 1*, 1 (2007), 27–64.

[165] Scott, John. Social network analysis. *Sociology 22*, 1 (1988), 109–127.

[166] Sen, Prithviraj, Namata, Galileo, Bilgic, Mustafa, Getoor, Lise, Gallagher, Brian, and Eliassi-Rad, Tina. Collective classification in network data. *AI Mag. 29*, 3 (2008), 93–106.

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

[168] Shchur, Oleksandr, Mumme, Maximilian, Bojchevski, Aleksandar, and Günnemann, Stephan. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).

[169] Shi, Jianbo, and Malik, Jitendra. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 8 (2000), 888–905.

[170] Simonovsky, Martin, and Komodakis, Nikos. Graphvae: Towards generation of small graphs using variational autoencoders. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I 27* (2018), Springer, pp. 412–422.

[171] Simonovsky, Martin, and Komodakis, Nikos. GraphVAE: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks* (2018), Springer, pp. 412–422.

[172] Snijders, Tom AB, and Nowicki, Krzysztof. Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of Classification 14*, 1 (1997), 75–100.

[173] Spielman, Daniel. Spectral graph theory. *Combinatorial Scientific Computing 18* (2012).

[174] Spielman, Daniel A, and Srivastava, Nikhil. Graph sparsification by effective resistances. *SIAM Journal on Computing 40*, 6 (2011), 1913–1926.

[175] Spielman, Daniel A, and Teng, Shang-Hua. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing* (2004), pp. 81–90.

[176] Stanković, Ljubiša, Daković, Miloš, and Sejdić, Ervin. Introduction to graph signal processing. In *Vertex-Frequency Analysis of Graph Signals*. Springer, 2019, pp. 3–108.

[177] Stark, Chris, Breitkreutz, Bobby-Joe, Chatr-Aryamontri, Andrew, Boucher, Lorrie, Oughtred, Rose, Livstone, Michael S, Nixon, Julie, Van Auken, Kimberly, Wang, Xiaodong, Shi, Xiaoqi, et al. The BioGRID interaction database: 2011 update. *Nucleic Acids Research 39* (2010), D698–D704.

[178] Sun, Fan-Yun, Qu, Meng, Hoffmann, Jordan, Huang, Chin-Wei, and Tang, Jian. vgraph: A generative model for joint community detection and node representation learning. In *Advances in Neural Information Processing Systems 32* (2019), pp. 512–522.

[179] Suresh, Susheel, Budde, Vinith, Neville, Jennifer, Li, Pan, and Ma, Jianzhu. Breaking the limit of graph neural networks by improving the assortativity of graphs with local mixing patterns. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021* (2021), ACM, pp. 1541–1551.

[180] Takeaki, UNO. Implementation issues of clique enumeration algorithm. *Special issue: Theoretical computer science and discrete mathematics, Progress in Informatics 9* (2012), 25–30.

[181] Tang, Jian, Qu, Meng, and Mei, Qiaozhu. PTE: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)* (2015), pp. 1165–1174.

[182] Tang, Jian, Qu, Meng, Wang, Mingzhe, Zhang, Ming, Yan, Jun, and Mei, Qiaozhu. LINE: Large-scale information network embedding. In *Proceedings of the 24th International World Wide Web Conference (WWW)* (2015), pp. 1067–1077.

[183] Tang, Lei, and Liu, Huan. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2009), ACM, pp. 817–826.

[184] Tenenbaum, Joshua B, De Silva, Vin, and Langford, John C. A global geometric framework for nonlinear dimensionality reduction. *Science 290*, 5500 (2000), 2319–2323.

[185] Tian, Fei, Gao, Bin, Cui, Qing, Chen, Enhong, and Liu, Tie-Yan. Learning deep representations for graph clustering. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)* (2014).

[186] Tsourakakis, Charalampos E. Fast counting of triangles in large real networks without counting: Algorithms and laws. In *2008 Eighth IEEE International Conference on Data Mining* (2008), IEEE, pp. 608–617.

[187] Van Der Hofstad, Remco. *Random graphs and complex networks*, vol. 1. Cambridge University Press, 2016.

[188] Veličković, Petar, Cucurull, Guillem, Casanova, Arantxa, Romero, Adriana, Lio, Pietro, and Bengio, Yoshua. Graph attention networks. *International Conference on Learning Representations* (2018).

[189] Veličković, Petar, Fedus, William, Hamilton, William L, Liò, Pietro, Bengio, Yoshua, and Hjelm, R Devon. Deep graph infomax. *arXiv preprint arXiv:1809.10341* (2018).

[190] Verbeek, Kevin, and Suri, Subhash. Metric embedding, hyperbolic space, and social networks. In *Proceedings of the thirtieth annual symposium on Computational geometry* (2014), pp. 501–510.

[191] Virtanen, Pauli, Gommers, Ralf, Oliphant, Travis E., Haberland, Matt, Reddy, Tyler, Cournapeau, David, Burovski, Evgeni, Peterson, Pearu, Weckesser, Warren, Bright, Jonathan, van der Walt, Stéfan J., Brett, Matthew, Wilson, Joshua, Millman, K. Jarrod, Mayorov, Nikolay, Nelson, Andrew R. J., Jones, Eric, Kern, Robert, Larson, Eric, Carey, C J, Polat, İlhan, Feng, Yu, Moore, Eric W., VanderPlas, Jake, Laxalde, Denis, Perktold, Josef, Cimrman, Robert, Henriksen, Ian, Quintero, E. A., Harris, Charles R., Archibald, Anne M., Ribeiro, Antônio H., Pedregosa, Fabian, van Mulbregt, Paul, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods 17* (2020), 261–272.

[192] Wang, Daixin, Cui, Peng, and Zhu, Wenwu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)* (2016), pp. 1225–1234.

[193] Wang, Hongwei, Wang, Jia, Wang, Jialin, Zhao, Miao, Zhang, Weinan, Zhang, Fuzheng, Xie, Xing, and Guo, Minyi. GraphGAN: Graph representation learning with generative adversarial nets. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)* (2018).

[194] Wang, Yu-Xiong, and Zhang, Yu-Jin. Nonnegative matrix factorization: A comprehensive review. *IEEE Transactions on Knowledge and Data Engineering 25*, 6 (2012), 1336–1353.

[195] Watts, Duncan J, and Strogatz, Steven H. Collective dynamics of 'small-world'networks. *Nature 393*, 6684 (1998), 440–442.

[196] Weber, Melanie. Neighborhood growth determines geometric priors for relational representation learning. In *International Conference on Artificial Intelligence and Statistics* (2020), PMLR, pp. 266–276.

[197] Wu, Felix, Souza, Amauri, Zhang, Tianyi, Fifty, Christopher, Yu, Tao, and Weinberger, Kilian. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning* (2019), PMLR, pp. 6861–6871.

[198] Xie, Jierui, Kelley, Stephen, and Szymanski, Boleslaw K. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Computing Surveys 45*, 4 (2013), 43:1–43:35.

[199] Xu, Keyulu, Hu, Weihua, Leskovec, Jure, and Jegelka, Stefanie. How powerful are graph neural networks? *International Conference on Learning Representations* (2019).

[200] Xu, Keyulu, Li, Chengtao, Tian, Yonglong, Sonobe, Tomohiro, Kawarabayashi, Ken-ichi, and Jegelka, Stefanie. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning* (2018), PMLR, pp. 5453–5462.

[201] Yan, Yujun, Hashemi, Milad, Swersky, Kevin, Yang, Yaoqing, and Koutra, Danai. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. *arXiv preprint arXiv:2102.06462* (2021).

[202] Yang, Jaewon, and Leskovec, Jure. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining* (2013), pp. 587–596.

[203] Yang, Jaewon, and Leskovec, Jure. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems 42*, 1 (2015), 181–213.

[204] Yang, Zhirong, Hao, Tele, Dikmen, Onur, Chen, Xi, and Oja, Erkki. Clustering by nonnegative matrix factorization using graph random walk. In *Advances in Neural Information Processing Systems* (2012), pp. 1079–1087.

[205] You, Jiaxuan, Ying, Rex, Ren, Xiang, Hamilton, William, and Leskovec, Jure. GraphRNN: Generating realistic graphs with deep auto-regressive models. In *Proceedings of the 35th International Conference on Machine Learning (ICML)* (2018), pp. 5708–5717.

[206] Young, Stephen J, and Scheinerman, Edward R. Random dot product graph models for social networks. In *International Workshop on Algorithms and Models for the Web-Graph* (2007), Springer, pp. 138–149.

[207] Yu, Kai, Yu, Shipeng, and Tresp, Volker. Soft clustering on graphs. In *Advances in Neural Information Processing Systems* (2005), pp. 1553–1560.

[208] Zahirnia, Kiarash, Schulte, Oliver, Naddaf, Parmis, and Li, Ke. Micro and macro level graph modeling for graph variational auto-encoders. In *Advances in Neural Information Processing Systems* (2022), S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35, Curran Associates, Inc., pp. 30347–30361.

[209] Zhou, Jun, Chen, Chaochao, Zheng, Longfei, Zheng, Xiaolin, Wu, Bingzhe, Liu, Ziqi, and Wang, Li. Privacy-preserving graph neural network for node classification. *arXiv:2005.11903* (2020).

[210] Zhu, Ciyou, Byrd, Richard H, Lu, Peihuang, and Nocedal, Jorge. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS) 23*, 4 (1997), 550–560.

[211] Zhu, Jiong, Rossi, Ryan A, Rao, Anup, Mai, Tung, Lipka, Nedim, Ahmed, Nesreen K, and Koutra, Danai. Graph neural networks with heterophily. *AAAI Conference on Artificial Intelligence* (2020).

[212] Zhu, Jiong, Yan, Yujun, Zhao, Lingxiao, Heimann, Mark, Akoglu, Leman, and Koutra, Danai. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems 33* (2020).

[213] Zweig, Katharina Anna. Are word-adjacency networks networks? In *Towards a theoretical framework for analyzing complex linguistic networks*. Springer, 2016, pp. 153–163.