Research Article

Methods for Analyzing Large Bipartite Graphs with Applications to Link Prediction

Christopher W. Curtis[1], **Maryam Sabagh**[1]

1. San Diego State University, United States

We look at several techniques for analyzing and characterizing large bipartite graphs with a focus on better explaining the performance of graph neural networks in performing link prediction. We prove several results for computing the number of a variety of small subgraphs to help characterize scale in large bipartite graphs, and we also derive an asymptotic formula to characterize the longest simple walk. We then look at using now standard convolution-filter-based graph neural network learning methods to perform link prediction in each data set. Ultimately, we find that the graph learning methods used are most affected by whether the data is well described by a power-law distribution, which indicates a scale-free structure in the data. The scale-free property is shown to degrade predictive power, and it indicates that existing convolutional-filter-based methods learn predictive tasks better when there are strong distinctions in scales in a graph despite there potentially being large numbers of disparate scales.

Correspondence: papers@team.qeios.com — Qeios will forward to the authors

1. Introduction

Prediction between dichotomous sets is at the heart of much of modern data-driven industry. The ability to accurately predict what a user wants among a set of choices determines much of the success of any number of digital platforms. In turn, there is a thriving research endeavour which has developed a range of machine learning methods, broadly described as graph neural networks (GNNs), to learn from large graphs so as to better our ability to predict user choices. Relatively exhaustive citation lists with historical context about GNNs can be found in[1][2].

 doi.org/10.32388/074O28

Given the staggering breadth of graph learning methodologies, we focus in this work on one class of GNNs called *message passing neural networks* (MPNNs)[3][4][5][6][7], which have proven to be some of the most powerful methods for learning on graphs. MPNNs rely on convolutional filters to learn the graph topology, thereby allowing for generalisability to make predictions of new links. However, as shown in[8] and explored in detail in[7], most existing MPNNs can be shown to be no more expressive than the lowest scale Weisfeiler–Lehman (WL) graph isomorphism test[9][10]. This limitation presents itself as an inability of MPNNs to learn large-scale structures such as closed walks or related subgraphs, thereby limiting the extent to which link prediction can be successful. While methods are being actively developed to address this shortcoming, see for example[7][11], both approaches and related ones suffer from computational limits induced by either the size of the adjacency matrix itself or the number of smaller scale subgraphs in the network. Moreover, the role that statistical properties of a network play, such as its degree distribution, has not received as much attention as other factors in graph learning problems, though see[12] which explores the role of preferential attachment[13] in assessing the fairness of link predictions in social networks.

Thus, in this work, assuming our graph is bipartite so that we are modelling link prediction between two well-defined classes, we present a number of analytic and statistical approaches to help better understand how MPNNs behave and are able to accurately predict linkage. In order to characterize scales in our networks, we look at counting the number of simple walks of arbitrary length in a network. We ultimately develop a number of metrics related to the largest singular value of the adjacency matrix of the network graph. Through asymptotic analysis we develop, we couple graph structure to the value of the largest singular value, thereby helping to provide interpretability and insight into otherwise large networks. Moving to the other limit of small scales, to quantify the size of small subgraphs in our network, we extend results in[14] and again develop a heuristic asymptotic formalism for determining the longest simple walk in our network. This then lets us show that the largest singular value is a reasonable estimate for distinguishing between small and large scale subgraphs in $\mathcal{G}$. These methods then give us a way of appreciating in advance where MPNNs might struggle and hopefully point the way towards better methodology. We study our methods on two bipartite data sets. The first consists of the classes “playlists″ and “tracks″ which come from the Spotify Million Song Challenge[15]. The other is the Amazon-book data set used in[16] consisting of classes “users″ and “items″. We are able to show that while the Spotify and Amazon data sets are of similar node and edge counts, the small subgraph counts

are orders of magnitude apart. Further, we anticipate markedly longer simple walks in the Spotify data which in principle should confound MPNNs.

We then present methods developed in[17] for determining whether a network has a power-law degree distribution and therefore most likely developed according to preferential attachment (PA)[13], i.e. a "rich get richer" process. Aside from being a foundational insight into the nature of a network, as we show, this also appears to be a critical issue in our characterisation of how MPNNs perform. Most interestingly, we see that the Spotify data only has a power-law distribution when looking at how playlists attach to tracks, but not in the reciprocal direction. In contrast, the Amazon network clearly is described by a power-law distribution in both directions. Thus, while we show that the Spotify data has more small and large scale structure than the Amazon data, one can also distinguish in a statistically meaningful way the difference between playlist and track nodes at every scale in comparison to the Amazon data set.

To illustrate the impact of this difference and also explore the role that varying subgraph scales play in the learning process, we look at using three typical MPNNs to perform link prediction in both data sets. These three are Light-GCN[16], GraphSage[18], and Chebnet[4]. We also look at a variant of the Bayesian-Personalized Ranking Loss (BPRL)[19] and likewise explore the role of different levels of negative sampling in our methods. As shown in[7], all three MPNNs should in principle do no better than a 1WL test, though Chebnet has certain instances where it can transcend this limitation. Likewise, looking at experimental results in[7], we would generally anticipate Chebnet performing best, all other things being equal. This result holds up over the Spotify data set where Chebnet clearly outperforms the other MPNNs. However, our numerical experiments show that while every method performs well as a class discriminator, as measured by Area-Under-the-Curve (AUC) scores, when we look at Recall-at-K (ReK) scores, the scale-free nature of the Amazon-book network lowers ReK scores but also makes GraphSage and Chebnet perform in virtually identical ways with sufficient negative sampling. So while Chebnet is best at navigating the multiscale structure of the network, we posit that learning via MPNNs is more challenging in a PA network since scale becomes less useful for discrimination, thereby making GraphSage and Chebnet perform in nearly identical ways. To address this issue fully is a question for future research.

The structure of the paper is as follows. In Section 2, we present our methods for analysing large bipartite graphs and present our results on power-law distributions. In Section 3, we present our results on using MPNNs to perform link prediction. In Section 4, we discuss conclusions and future work. This is followed by Acknowledgments and finally, an Appendix that collects proofs of technical theorems and lemmas in

the body of the paper. Note, in order to aid the reader by making concepts more concrete, we routinely refer to the classes in our bipartite networks as “playlists/users" and “tracks/items".

2. Determining Scales in Large Bipartite Graphs

The bipartite structure of the playlist/track or user/item data means the affiliated adjacency matrix A of the graph can be written as

$$A = \begin{pmatrix} 0 & B \\ B^T & 0 \end{pmatrix},$$

where we take B to be $N_+ \times N_-$ (i.e. number-of-playlists/number-of-users by number-of-tracks/number-of-items). We denote the set of nodes in $\mathcal{G}$ as $\mathcal{N}$ and the edges as $\mathcal{E}$. If we define the number of $2k$-cycle (non-simple) walks in the graph to be $n_{wlk}(2k)$, it is a classic result then that $n_{wlk}(2k) = \mathrm{tr}(A^{2k})/4k$. Thus, from studies of $\mathrm{tr}(A^{2k})$ alone, we can develop some quantitative understanding of the number of different scales in a network.

2.1. Singular-Value Analysis for Counting Walks in Large Bipartite Graphs

The most straightforward way to compute $\mathrm{tr}(A^{2k})$ for $\mathcal{N} \gg 1$ is via the following lemma.

Lemma 1. *For bipartite graph $\mathcal{G}$ with adjacency matrix A, we have*

$$n_{wlk}(2k) = \frac{1}{4k} \sum_{j=1}^{N} \sigma_j^{2k}$$

where σ_j are the singular values of B.

Proof. We readily see that

$$A^{2k} = \begin{pmatrix} (BB^T)^k & 0 \\ 0 & \left(B^T B\right)^k \end{pmatrix}$$

so

$$\mathrm{tr}\left(A^{2k}\right) = \mathrm{tr}\left((BB^T)^k\right) + \mathrm{tr}\left((B^T B)^k\right).$$

Given that $\mathrm{tr}(CD) = \mathrm{tr}(DC)$ for C $m \times n$ and D $n \times m$, we see that

$$\mathrm{tr}\left((BB^T)^k\right) = \mathrm{tr}\left((B^T B)^k\right).$$

Likewise, if we have the SVD of B so that $B = U\Sigma V^T$, then

$$n_{wlk}(2k) = \frac{1}{4k}\mathrm{tr}\left(A^{2k}\right) = \frac{1}{4k}\sum_{j=1}^{N_+}\sigma_j^{2k}.$$

□

Note, per convention, we order the singular values such that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$, where $r = \mathrm{rank}(B)$. As we see from this result, if $\sigma_1 > 1$, then we can anticipate there being long walks, and thus long scales, in $\mathcal{G}$. Therefore, as a first pass at assessing the presence of larger scales in our graph, we would like methods for estimating σ_1 which provide both quantitative estimates and also help us understand how particular values of σ_1 come about from subgraph structure.

To do this, we see that

$$B^TB = V\Sigma^2V^T = D_- + B_{2h}^-$$

where D_- is the $N_- \times N_-$ diagonal matrix of track/item degrees, say d_j^-, and B_{2h}^- is the symmetric matrix with zero-diagonal entries and off-diagonal entries

$$\left(B_{2h}^-\right)_{jk} = \sum_{l=1}^{|\mathcal{N}_+|} b_{lj}b_{lk}$$

so $(B_{2h})_{jk}$ is the number of playlists/users shared between the j^{th} and k^{th} track/item. Note, we could have just as well written

$$BB^T = U\Sigma^2U^T = D_+ + B_{2h}^+,$$

and from hereon we provide results derived from either B^TB or BB^T.

To wit, we immediately get that

$$\sum_{j=1}^{r}\sigma_j^2 = \sum_{k=1}^{N_-}d_k^- = \sum_{k=1}^{N_+}d_k^+.$$

From this, we get the simple bound

$$\sigma_1^2 \geq \frac{1}{r}\sum_{k=1}^{N_-}d_k^-,$$

which provides an easy test to determine if $\sigma_1^2 > 1$ and thus if we should expect growth in $n_{wlk}(2k)$ as k increases.

To get estimates for σ_1 in terms of basic graph properties, let

$$d_M^+ = \max_{1\leq l\leq N_+} d_l^+,\ d_M^- = \max_{1\leq j\leq N_-} d_j^-.$$

Then we see that the maximum-row-sum norm of $B^T B$ is

$$\begin{aligned}\left|\left|\left|B^T B\right|\right|\right|_\infty &= \max_{1\leq j\leq N_-}\sum_{k=1}^{N_-}\sum_{l=1}^{N_+} b_{lj}b_{lk} \\ &= \max_{1\leq j\leq N_-}\sum_{l=1}^{N_+} b_{lj}d_l^{(p)} \\ &\leq \max_{1\leq l\leq N_+} d_l^{(p)} \max_{1\leq j\leq N_-} d_j^{(tr)}.\end{aligned}$$

Therefore, we have

$$\sigma_1 \leq \sqrt{d_M^+ d_M^-},$$

which we note must be the same for BB^T. While convenient, this bound is not especially useful in practice. We can get a better variational estimate of σ_1 by noting that

$$\sigma_1^2 = \max_{\hat{\mathbf{u}}} \langle (D_- + B_{2h})\hat{\mathbf{u}}, \hat{\mathbf{u}}\rangle \,,$$

so that

$$\sigma_1^2 \leq d_M^- + ||B_{2h}||_F,\ ||B_{2h}||_F^2 = \mathrm{tr}(B_{2h}^2).$$

Proceeding in the same fashion gives us the immediate improvement

$$\sigma_1^2 \leq \min\left\{d_M^- + ||B_{2h}||_F, d_M^+ + \left|\left|\tilde{B}_{2h}\right|\right|_F\right\}$$

To get an even more refined estimate of the largest singular values, a perturbative approach can help provide yet more insight into the way the graph structure influences the magnitudes of the σ_j. We prove

Theorem 1. *If for $j \neq k$*

$$\min\left\{\frac{\max_{j,k}(B_{2h}^+)_{jk}}{d_M^+}, \frac{\max_{j,k}(B_{2h}^-)_{jk}}{d_M^-}\right\} = \epsilon \ll 1,$$

then for $d_j^{(ch)}/d_M^{(ch)} \gg \epsilon$

$$\sigma_j^2 = d_j^{(ch)} + \sum_{l\neq j}\frac{1}{d_j^{(ch)} - d_l^{(ch)}}\left(\left(B_{2h}^{(ch)}\right)_{lj}\right)^2 + \cdots$$

where $ch = \pm$ depending on which term defines ϵ.

Please see the Appendix for the proof. As can be seen from Theorem 1, if the maximum number of shared playlists/users between two tracks/items is significantly less than the maximum number of playlists/users attached to any track/item, the leading order effects which determine the magnitudes of

σ_j are the track degrees, with the next correction coming from the weighted overlap between tracks relative to their difference in degrees. Thus, we get a clear sense now that the singular values encode larger scale structure and, in turn, that larger scale structures determine the singular values.

From these results, we get several useful comparisons between our data sets. For the Spotify data, we restrict $\mathcal{G}$ so that every node has a minimum degree, say $d_{min} = 30$. Going below this threshold causes later learning approaches to become overwhelmed with what amount to anecdotal cases, leading to overfitting, thereby allowing for little meaningful generalization. In concrete terms, then, for the Spotify data, we have $|\mathcal{N}_+| = 11888$ and $|\mathcal{N}_-| = 7472$ so that $|\mathcal{N}| = 19360$ and $|\mathcal{E}| = 805071$. We plot the first 500 numerically approximated singular values of B in Figure 1.

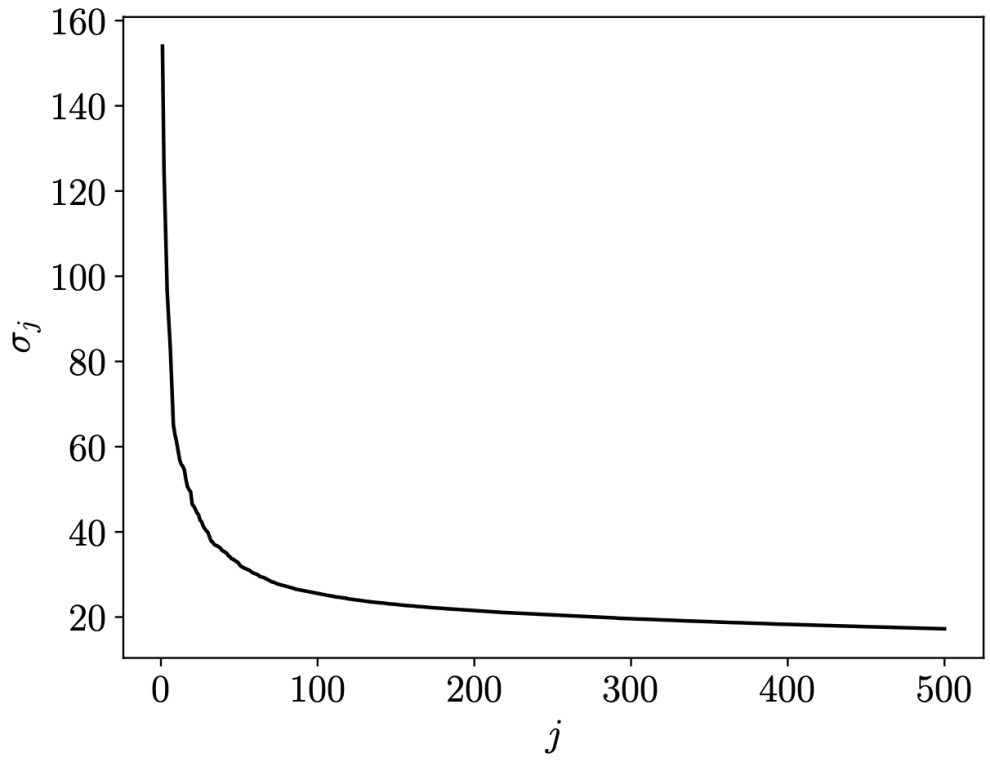

Figure 1. Plot of singular values σ_j of B from the Spotify data set for $1 \leq j \leq 500$.

As we find, $\sigma_1 \approx 154.02$, and thus $n_{wlk}(2k) = \mathcal{O}(\sigma_1^{2k})$ as $k \to \infty$. Our upper bound on σ_1 from Equation [varapprox] gives us $\sigma_1 \leq 195.4$. We see then how the influence of the interacting tracks and playlists gives rise to such a relatively large value of σ_1 and thus the potential for large closed walks within $\mathcal{G}$.

For the Spotify data set, with $\mathrm{ch} = -$, $\epsilon \approx .49$, so unfortunately, we cannot expect our result in Theorem 1 to provide a good approximation. Nevertheless, as we see in Figure 2, the expansion gives meaningful insight into the first nine singular values, with the quality of the approximation improving by $j = 9$. This gives us yet more analytic insight into singular values and how their magnitudes relate to the neighborhood structure within the graph.

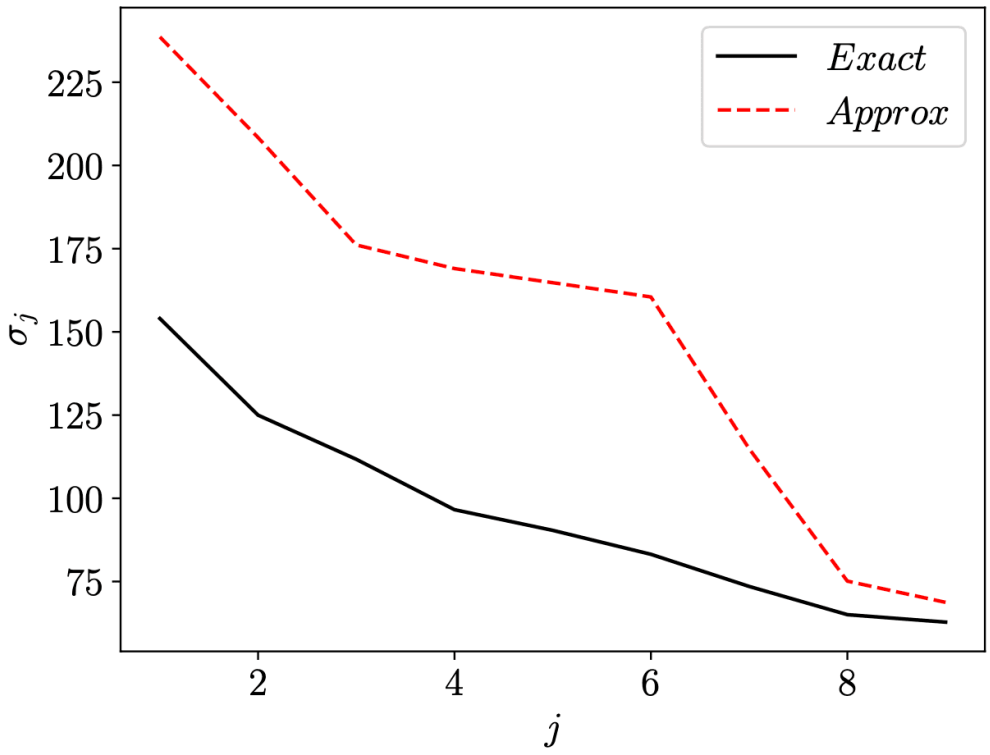

Figure 2. Plot of singular values σ_j of B and their approximations using Theorem 1 for $1 \leq j \leq 8$ for the Spotify data set.

To get a graph of comparable size, we fix the minimum degree of $\mathcal{G}$ for the Amazon data to $d_{min} = 26$. In this case, we get $|\mathcal{N}_+| = 9103$, $|\mathcal{N}_-| = 9766$, so that $|\mathcal{N}| = 18869$ and $|\mathcal{E}| = 539347$. We immediately see that the overall connectivity of the Amazon data is markedly lower than that of the Spotify data set. Computing the singular values, we get $\sigma_1 \approx 113.3$, showing that the reduced connectivity of the Amazon data set directly relates to the magnitude of the maximum singular value. We likewise find that, with $\mathrm{ch} = +$, $\epsilon \approx .37$ This again helps quantify the marked reduction in connectivity in the Amazon data

doi.org/10.32388/074O28

relative to the Spotify data set. As expected then, looking at our perturbative approach from Theorem 1 in Figure 3, we see a closer approximation of the curves.

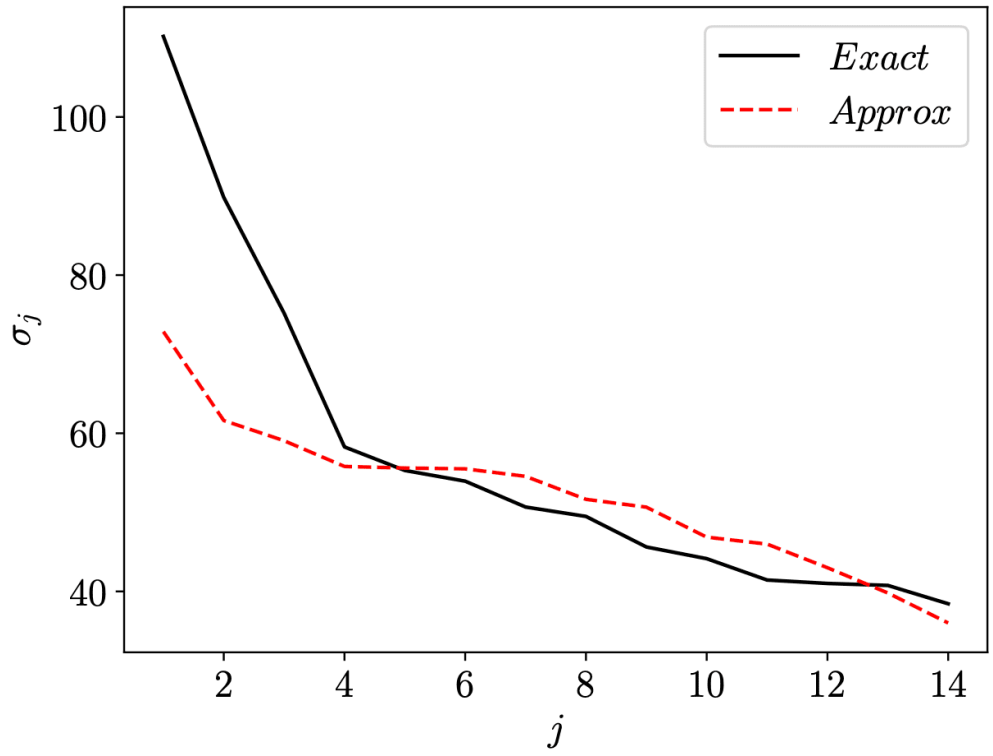

Figure 3. Plot of singular values σ_j of B and their approximations using Theorem 1 for $1 \leq j \leq 14$ for the Amazon data set.

All of these results then give a strong indication that larger-scale subgraphs exist within each $\mathcal{G}$. Moreover, we expect even more large-scale structure in the Spotify data set. We now examine why that is and provide analytic formulas to better quantify which structures are present.

2.2. Quantifying Multiscale Network Structure via Counting Small Subgraphs

Returning to the quantity $n_{wlk}(2k)$, we note that this value includes significant counting of closed paths over small subgraphs. For example, every $2k$ long closed walk necessarily includes $2|\mathcal{E}|$ paths. To account for this and thus determine a more accurate count of the large-scale structures within $\mathcal{G}$, following[14], defining the number of homomorphic images (i.e. surjective homomorphisms) of a simple k-cycle walk to a subgraph $\mathcal{H}$ to be $c_k(\mathcal{H})$ and the number of subgraphs isomorphic to $\mathcal{H}$ by $n_{\mathcal{G}}(\mathcal{H})$, then

doi.org/10.32388/074O28

$$\mathrm{tr}(A^k) = \sum_{\mathcal{H}} c_k(\mathcal{H}) n_{\mathcal{G}}(\mathcal{H}).$$

Thus, the number of simple $2k$ length walks in our bipartite graph, say $n_{\mathcal{G}}(\mathcal{C}_{2k})$ is given by

$$n_{\mathcal{G}}(\mathcal{C}_{2k}) = n_{wlk}(2k) - \frac{1}{4k} \sum_{\mathcal{H}_d,\, d<2k} c_{2k}(\mathcal{H}_d) n_{\mathcal{G}}(\mathcal{H}_d)$$

where by $\mathcal{H}_d$ we mean every subgraph (up to isomorphism) of d-nodes. For example then, the authors of[14] show that

$$n_{\mathcal{G}}(\mathcal{C}_4) = n_{wlk}(4) - \frac{1}{8}(4n_{\mathcal{G}}(\mathcal{H}_2) + 2|\mathcal{E}|)$$

where $|\mathcal{E}|$ is the number of edges and $n_{\mathcal{G}}(\mathcal{H}_2)$ is the number of connected 3 node sequences (i.e. a graph with nodes a, b, c and edges (a, b) and (b, c)). This is illustrated in Figure 4, which also shows the other subgraphs that we can compute $n_{\mathcal{G}}(\mathcal{H})$ in closed form as shown in[14]. Note, we have kept the notation of[14] for ease of cross-referencing.

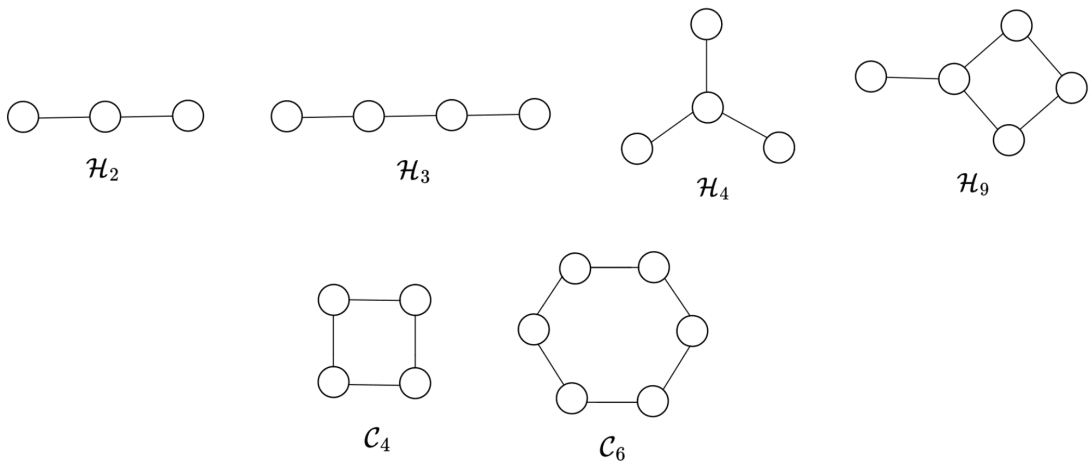

Figure 4. Subgraphs with closed form counting formulas for $n_{\mathcal{G}}(\mathcal{H})$ from[14].

We can extend the closed form formulas in[14] using the following lemmas. The proofs are found in the Appendix.

Lemma 2. *For $k \geq 2$,*

$$c_{2k}(\mathcal{H}_2) = 2^{k+1} - 4.$$

Lemma 3. *For $k \geq 2$,*

$$c_{2k}(\mathcal{C}_4) = 4(2^{2k-1} - 2^{k+1} + 2).$$

Lemma 4. *For $k \geq 2$,*

$$c_{2k}(\mathcal{H}_4) = 2 \cdot 3^k - 3 \cdot 2^{k+1} + 6.$$

Performing some relatively straightforward counting to address the cases not proved above, we then find

$$\begin{aligned} n_{\mathcal{G}}(\mathcal{C}_6) = n_{wlk}(6) - \frac{1}{12}(&72n_{\mathcal{G}}(\mathcal{C}_4) + 12n_{\mathcal{G}}(\mathcal{H}_9) + 12n_{\mathcal{G}}(\mathcal{H}_4) \\ &+6n_{\mathcal{G}}(\mathcal{H}_3) + 12n_{\mathcal{G}}(\mathcal{H}_2) + 2|\mathcal{E}|) \end{aligned}$$

We plot all of the relevant counts for the Spotify data in Figure 5 and for the Amazon data in Figure 6. For the Spotify data, $n_{\mathcal{G}}(\mathcal{C}_6) \approx 10^{13}$ and thus it dominates the counts of all other subgraphs. $n_{\mathcal{G}}(\mathcal{C}_6)$ also is the largest count for the Amazon data, but it is now $\mathcal{O}(10^{12})$, again reflecting the lower degree of connectivity in the Amazon data. Likewise, $n_{\mathcal{G}}(\mathcal{H}_9)$ is closer in magnitude to $n_{\mathcal{G}}(\mathcal{C}_6)$, so we have more small-scale clustering in the Amazon data compared to the Spotify data. These results echo what we expect from our singular-value analysis of B, and so while there must be some internal maximum for $n_{\mathcal{G}}(\mathcal{C}_{2k})$, we can expect that it is very large for relatively large values of k. Thus, we expect that large-scale structure is rampant throughout $\mathcal{G}$, indicating that long chains of playlists to tracks and back again are present in the Spotify data set.

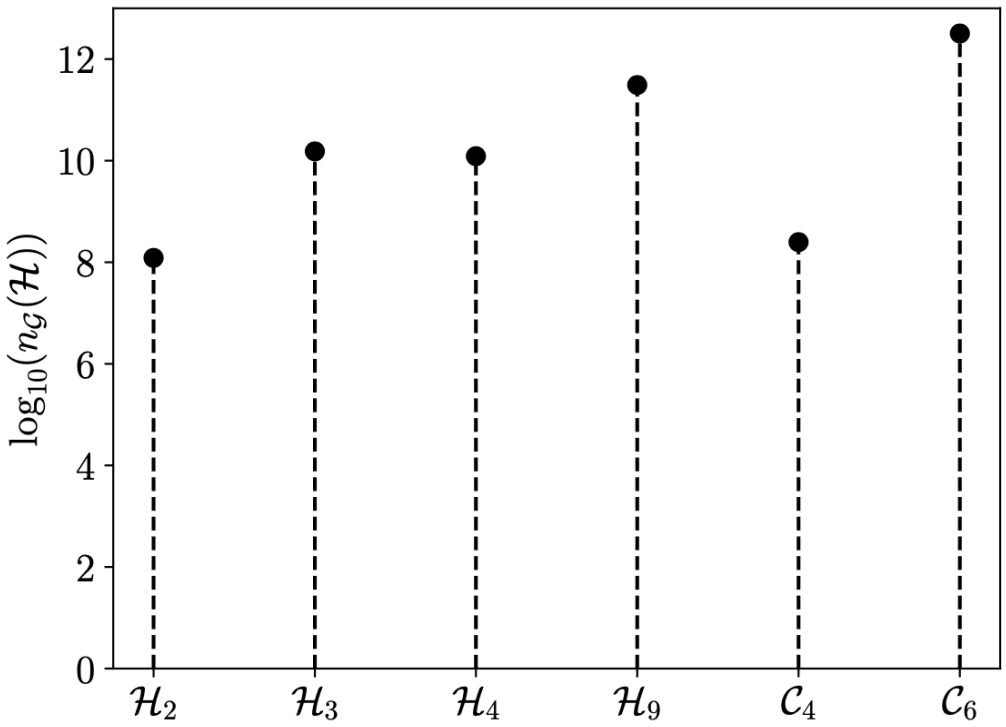

Figure 5. Semi-log plot of the counts of $n_{\mathcal{G}}(\mathcal{H})$ using formulas from[14] for the Spotify data set.

doi.org/10.32388/074O28

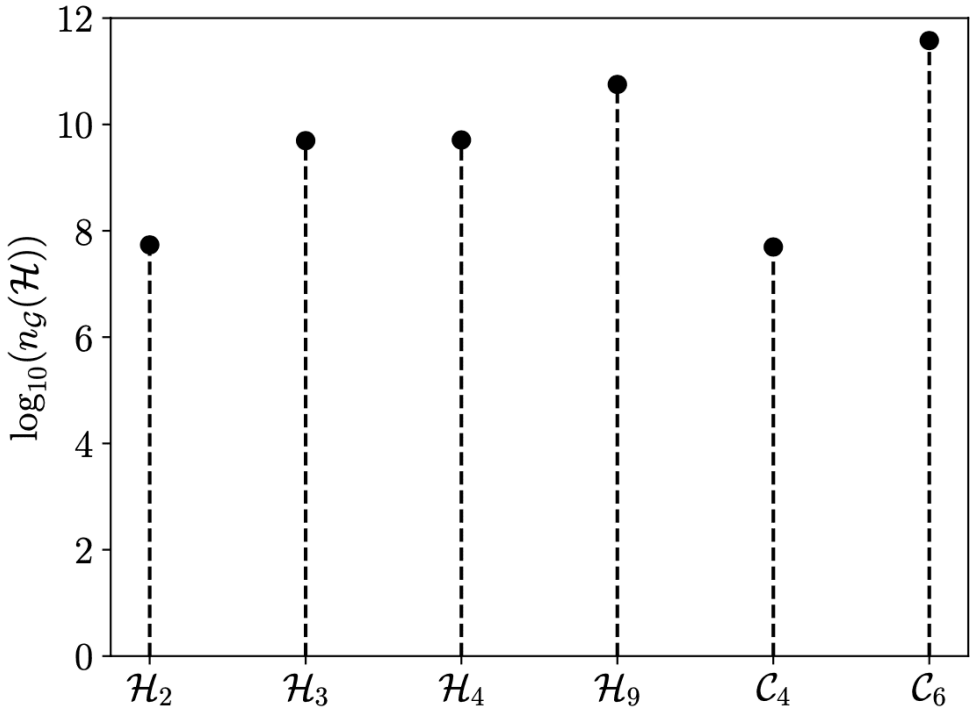

Figure 6. Semi-log plot of the counts of $n_{\mathcal{G}}(\mathcal{H})$ using formulas from[14] for the Amazon data set.

Aside from their immediate utility, we also see that longer simple closed walks over subgraphs ultimately balance the count of all walks found via the trace formula. For example, we see that $c_{2k}(\mathcal{C}_4) = \mathcal{O}(4^k)$ while $c_{2k}(\mathcal{H}_4) = \mathcal{O}(3^k)$. Thus, these terms in part balance the growth from $\mathcal{O}(\sigma_1^{2k})$, and show how we finally achieve a maximum possible simple closed walk in $\mathcal{G}$. Said yet another way, we see how surjective homomorphisms onto subgraphs define a notion of maximal scale in $\mathcal{G}$.

To see this matter more explicitly, we prove

Theorem 2. *For $k > j \geq 2$, we have*

$$c_{2k}\left(\mathcal{C}_{2j}\right) = 2j\left(T_{k,j} - p_{k,j}\right),\ T_{k,j} = j^{2(k-j)+3}((j-1)!)^2$$

where

$$\lim_{k\to\infty} \frac{p_{k,j}}{T_{k,j}} = 0.$$

Proof. We label $\mathcal{C}_{2j}$ via the sequence $\{a_1, \cdots, a_{2j}\}$. This naturally separates into j even-index labels and j odd-index labels. For a $2k$ walk, if we start at a_1 we generate, for j odd, the sequence

$$\frac{a_1}{1},\frac{\begin{pmatrix}a_2\\a_{2j}\end{pmatrix}}{2},\frac{\begin{pmatrix}a_1\\a_3\\a_{2j-1}\end{pmatrix}}{3},\cdots,\frac{\begin{pmatrix}a_1\\\vdots\\a_{2j-1}\end{pmatrix}}{j},\cdots,\frac{\begin{pmatrix}a_1\\a_3\\a_{2j-1}\end{pmatrix}}{2k-1},\frac{\begin{pmatrix}a_2\\a_{2j}\end{pmatrix}}{2k},$$

and for j even

$$\frac{a_1}{1},\frac{\begin{pmatrix}a_2\\a_{2j}\end{pmatrix}}{2},\frac{\begin{pmatrix}a_1\\a_3\\a_{2j-1}\end{pmatrix}}{3},\cdots,\frac{\begin{pmatrix}a_2\\\vdots\\a_{2j}\end{pmatrix}}{j},\cdots,\frac{\begin{pmatrix}a_1\\a_3\\a_{2j-1}\end{pmatrix}}{2k-1},\frac{\begin{pmatrix}a_2\\a_{2j}\end{pmatrix}}{2k},$$

In either case, counting shows that this generates $j^{2(k-j)+3}\left(\prod_{l=1}^{j-1} l\right)^2 = T_{k,j}$ possible sequences.

We now suppose a_3 is absent in every odd entry of the above sequence. Denoting the number of these sequences as $m_{3,k,j}$, we find that

$$m_{3,k,j} = \begin{cases} j^{(k-j)+1}(j-1)^{(k-j)+2}\left(\prod_{l=1}^{(j-1)/2}(2l)^2\right)^2 & j \text{ odd} \\ j^{(k-j)+2}(j-1)^{(k-j)+1}\left(\prod_{l=1}^{(j-2)/2}(2l)^2\right)^2 & j \text{ even} \end{cases}$$

Defining $p_{k,j}$ via the formula

$$p_{k,j} = T_{k,j} - \frac{c_{2k}\left(\mathcal{C}_{2j}\right)}{2j},$$

we see then that $0 < p_{k,j} < (2j-1)m_{3,k,j}$. In turn then, we find that

$$\frac{p_{k,j}}{T_{k,j}} < (2j-1)(1-1/j)^{(k-j)+1}\begin{cases}(1-1/j) & j \text{ odd} \\ 1 & j \text{ even}\end{cases}$$

For $j \geq 2$, we then have the result. □

Our result then shows us $c_{2k}(C_{2j}) = \mathcal{O}(2j^{2(k-j)+4}((j-1)!)^2)$ for $k \gg j \geq 2$. If we suppose that

$$\begin{aligned} n_{\mathcal{G}}(\mathcal{C}_{2k}) \approx & \frac{1}{4k}\left(\sigma_1^{2k} - c_{2k}(\mathcal{C}_{2j})n_{\mathcal{G}}(\mathcal{C}_{2j})\right), \\ \approx & \frac{1}{4k}\left(\sigma_1^{2k} - \frac{1}{2}j^{2(k-j)+3}((j-1)!)^2\sigma_1^{2j}\right). \end{aligned}$$

Using Stirling's formula so that

$$(j-1)! \approx (j-1)\ln(j-1) - (j-1),\ j \gg 1,$$

and letting $\ln(j-1) \approx \ln(j) - 1/j$, we see that

$$\ln(j^{2(k-j)+3}((j-1)!)^2\sigma_1^{2j}) \approx 2\left((k+1/2)\ln j + j(\ln\sigma_1 - 1)\right).$$

This is strictly increasing in j, and thus we should get j as close to k as possible before compromising the validity of our asymptotic approximations. We thus let $k = \alpha j$, $\alpha > 1$ with the idea that we should later let $\alpha \to 1^+$ in order to get an estimate on the maximal simple walk length.

Keeping $\alpha > 1$, we expect to get $n_{\mathcal{G}}(\mathcal{C}_{2k}) = \mathcal{O}(1)$ when

$$(\alpha - 1) j \ln \sigma_1 \approx \alpha j \ln j - j + \frac{1}{2} \ln j.$$

Using the expansion $j \approx e^{1/\alpha} \sigma_1^{(\alpha-1)/\alpha} + \tilde{c}$ where $\tilde{c} = o(e^{1/\alpha} \sigma_1^{(\alpha-1)/\alpha})$, we get

$$j \approx e^{1/\alpha} \sigma_1^{(\alpha-1)/\alpha} - \frac{1}{2\alpha} \ln \sigma_1^{(\alpha-1)/\alpha}, \ \alpha > 1.$$

and thus we get the estimate for the critical simple walk, say k_{cr}, of

$$k_{cr} \approx \alpha e^{1/\alpha} \sigma_1^{(\alpha-1)/\alpha} - \frac{1}{2} \ln \sigma_1^{(\alpha-1)/\alpha}, \ \alpha > 1.$$

For the Spotify data, letting $\sigma_1 \approx 154$, by choosing $\alpha = 4$ we get $k_{cr} \approx 223$. Choosing $\alpha = 2$ we find $k_{cr} \approx 40$. Pushing our luck, as it were, and trying $\alpha = 3/2$ gets us $k_{cr} \approx 15$. For the Amazon data, $\sigma_1 \approx 113$, for $\alpha = 4$, we get $k_{cr} \approx 176$, for $\alpha = 2$, we get $k_{cr} \approx 34$, and for $\alpha = 3/2$, we get $k_{cr} \approx 13$. Thus, while getting a definitive answer would require more sophisticated methods, we can conclude in both cases that simple cycles longer than $\mathcal{O}(\sigma_1)$ are present in relatively small numbers if they are there at all. Thus, σ_1 gives us a good estimate for the number of nodes that distinguish small and large scale subgraphs in $\mathcal{G}$. We also point out that more general-purpose algorithms for counting simple cycles exist, *cf.* [20], and exploring their use in this context is a question for future work.

2.3. Preferential Attachment and Scale-Free Networks

Beyond counting subgraphs, we can also look at $\mathcal{G}$ from a statistical point of view. Letting the degree distribution of $\mathcal{G}$ be $P(d)$, we see in Figures 7 and 8 that the log-log plots of $P(d)$ appear to be linear. A typical conclusion then would be to see if $P(d) \approx d^{-\alpha}$, or if the distribution followed a *power law*. The consequences of this, as first shown in the now seminal work of[13] are that we can then suppose a preferential-attachment (PA) model, i.e., a "rich get richer" process, generates the network and, moreover, that we expect the network to exhibit *scale-free* behavior; see also[1]. Of course, the term scale-free should be understood in an appropriate limit since the results of the prior section clearly show that both networks exhibit large numbers of small-scale subgraph structures with distinguished scales. Thus, the results of this section should be understood over subgraphs with node counts larger than σ_1. Moreover,

as explored in detail in[17], while many distributions appear to follow a power law, whether that is a statistically meaningful observation requires appropriate analysis.

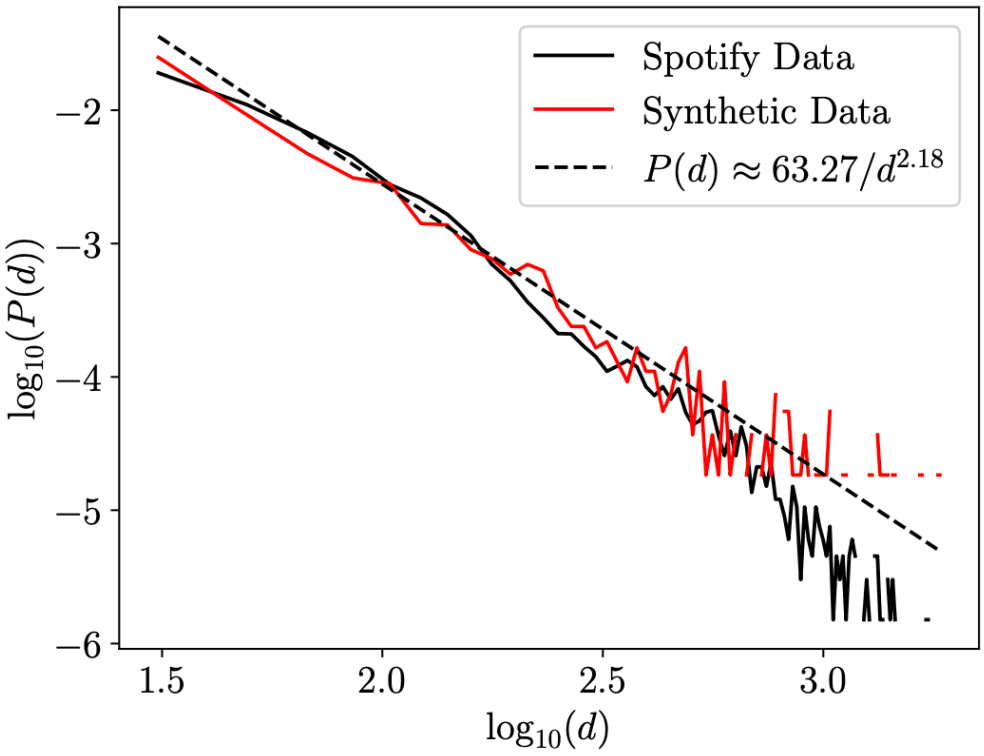

Figure 7. Log-log plot of the degree distribution for the Spotify data with the power-law MLE estimate for $\gamma = 2.18$. The p-value for the KS test is 2.19×10^{-5}, and thus we reject the null hypothesis.

doi.org/10.32388/074O28

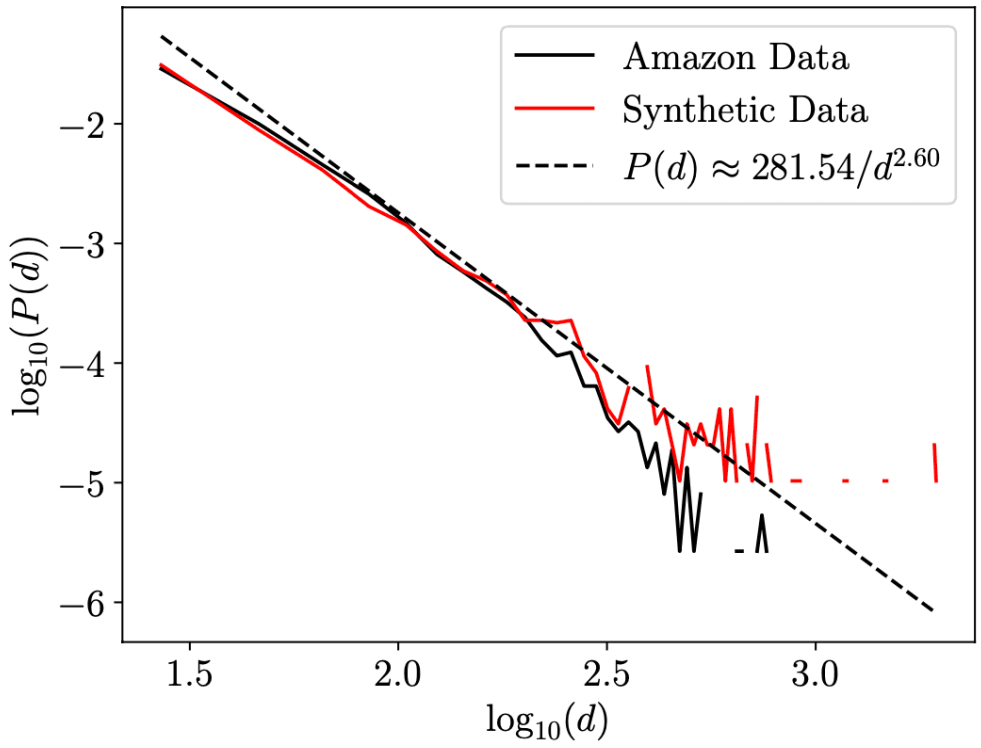

Figure 8. Log-log plot of the degree distribution for the Amazon data with the power-law MLE estimate for $\gamma = 2.6$. The p-value for the KS test is $.69$, and thus we keep the null hypothesis.

To that end, if we suppose that $P(d) = Cd^{-\gamma}$, we first note that by construction we fix $d \geq d_c$. To normalize the distribution, we find C to be

$$C = \frac{1}{\zeta(\gamma, d_c)}, \ \zeta(\gamma, d_c) = \sum_{d=0}^{\infty} \frac{1}{(d_c + d)^{\gamma}}.$$

From[17], we can find the maximum-likelihood estimate (MLE) of γ to be

$$\gamma \approx 1 + |\mathcal{N}| \left(\sum_{j=1}^{|\mathcal{N}|} \ln \frac{d_j}{d_c - 1/2} \right)^{-1}$$

Likewise, we use the Kolmogorov–Sinai (KS) test to determine whether the null hypothesis of the data being distributed according to the MLE approximation of $P(d)$ is valid or should be rejected. Following standard practice, we reject if the computed p-value is less than 5%.

Returning then to the results in Figure 7, while the Spotify data does not seem to deviate significantly from the MLE fit, the p-value for the KS test is well below the 5% rejection threshold, giving a very strong

indication that the data is not well explained by the MLE power-law distribution. To help illustrate the issue, we generate synthetic data using the MLE distribution, and as can be seen in Figure 7, the tails do deviate significantly, thereby explaining the rejection of the null hypothesis in the KS test. So while to the eye, and even somewhat intuitively, we might imagine that tracks and playlists form via PA to generate a scale-free network following a power-law distribution, that is not the case. In contrast, for the Amazon data, we readily pass the KS test with a p-value of .69. This gives us a very strong indication that the Amazon graph follows a power-law distribution and thus should be well explained via PA.

However, given that $\mathcal{G}$ is bipartite, it is natural to ask about the differences between the affiliated distributions generated by B and B^T. Looking first at B^T, which we can describe as how playlists/users attach to a given track/item, for the Spotify data, we see in Figure 9 similar results to those seen in Figure 7. In this case, though, we just pass the KS test and keep the null hypothesis. Thus, how playlists attach to tracks can be explained via PA. This intuitively makes sense since a track's popularity means more playlists should link to it. Moreover, if we look at the tails in Figure 9, the discrepancy between the measured and synthetic data is not as pronounced, helping explain why we can better explain the data through the MLE power-law distribution approximation.

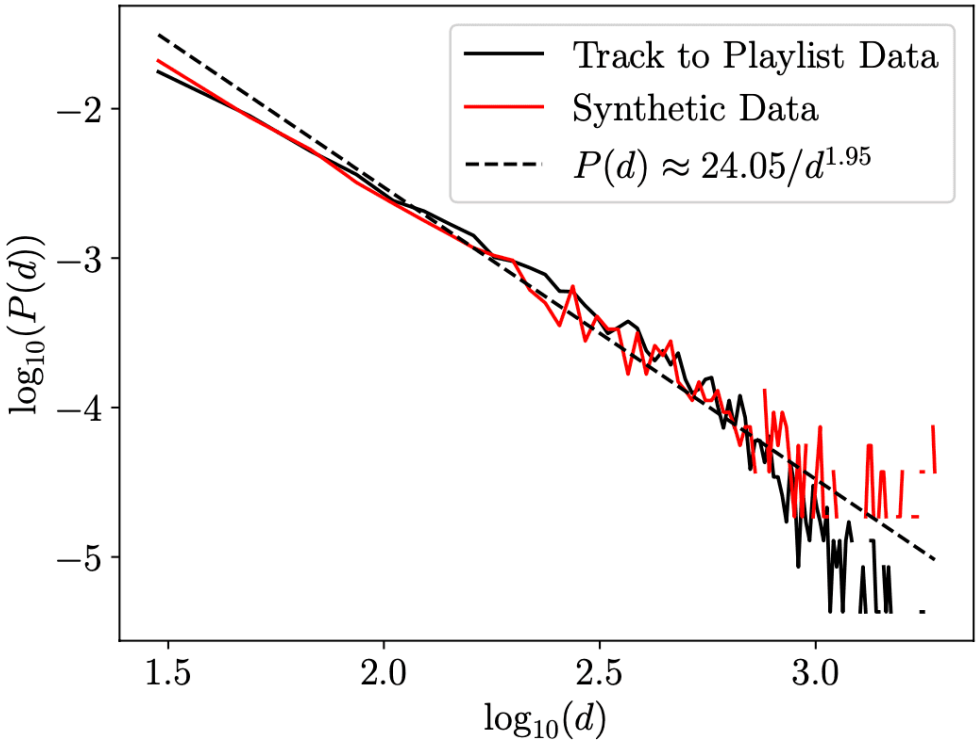

Figure 9. Log-log plot of the degree distribution for B^T for the Spotify data with power-law MLE estimate for $\gamma = 1.95$. The p-value for the KS test is .055, and thus we keep the null hypothesis.

In contrast, examining B for the Spotify data in Figure 10, we see that high-degree nodes cause even more extreme deviations away from power-law predictions. Thus, while PA can explain how tracks attach to playlists, this is not happening in a symmetric way to how playlists attach to popular tracks. How large playlists are then generated clearly follows a different generative mechanism from PA.

doi.org/10.32388/074O28

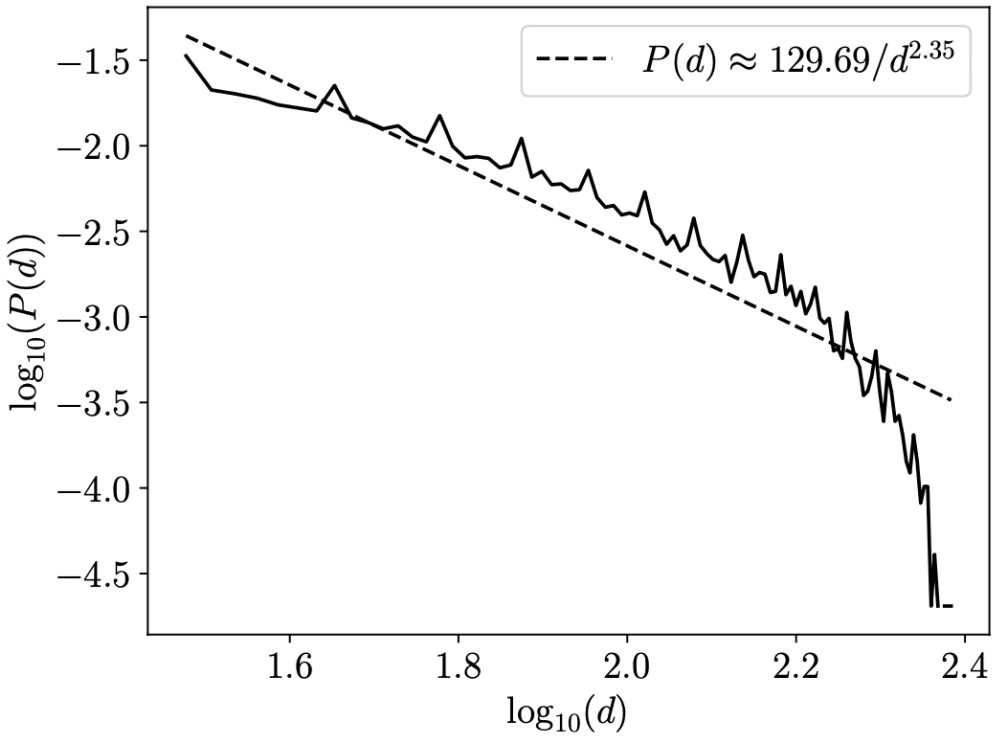

Figure 10. Log-log plot of the degree distribution for B for the Spotify data with power-law MLE estimate for $\gamma = 2.35$. The p-value for the KS test is .006, and thus we reject the null hypothesis.

While we see a strong disparity in the attachment mechanisms in the Spotify data, looking at Figures 11 and 12, we see that the Amazon network does not exhibit as pronounced a dichotomy in how users attach to items and vice versa. Moreover, the null hypothesis is kept in both KS tests, even though one can see the distribution in Figure 12 does show some of the bending seen in Figure 10.

doi.org/10.32388/074O28

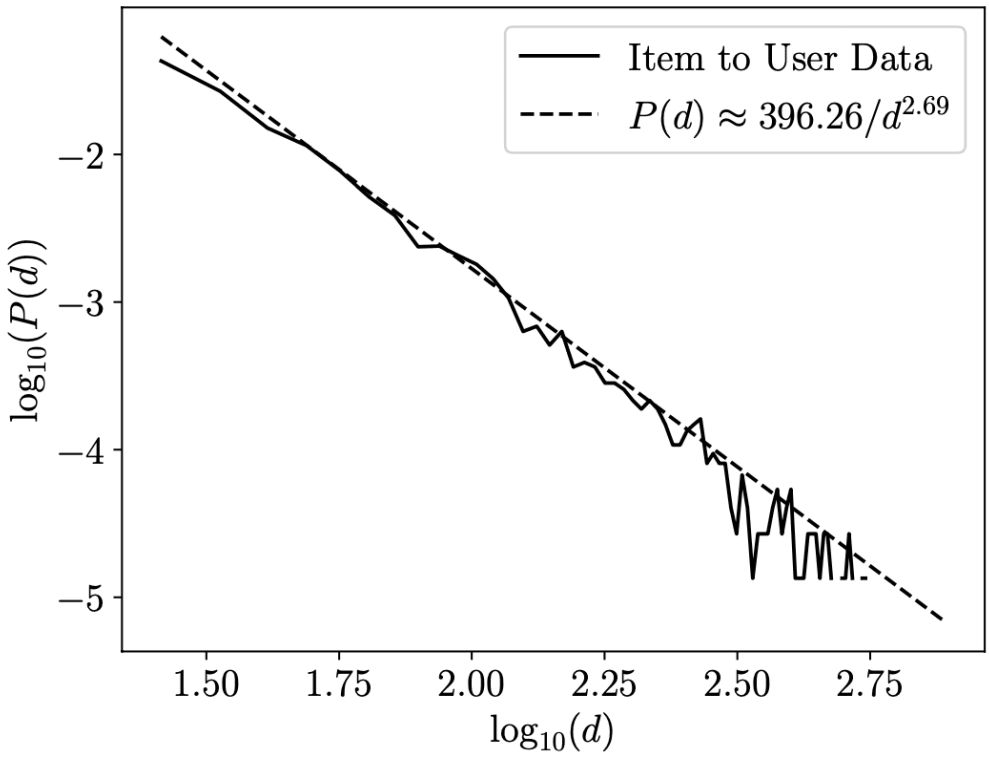

Figure 11. Log-log plot of the degree distribution for B^T for the Amazon data with power-law MLE estimate for $\gamma = 2.69$. The p-value for the KS test is .72, and thus we keep the null hypothesis.

doi.org/10.32388/074O28

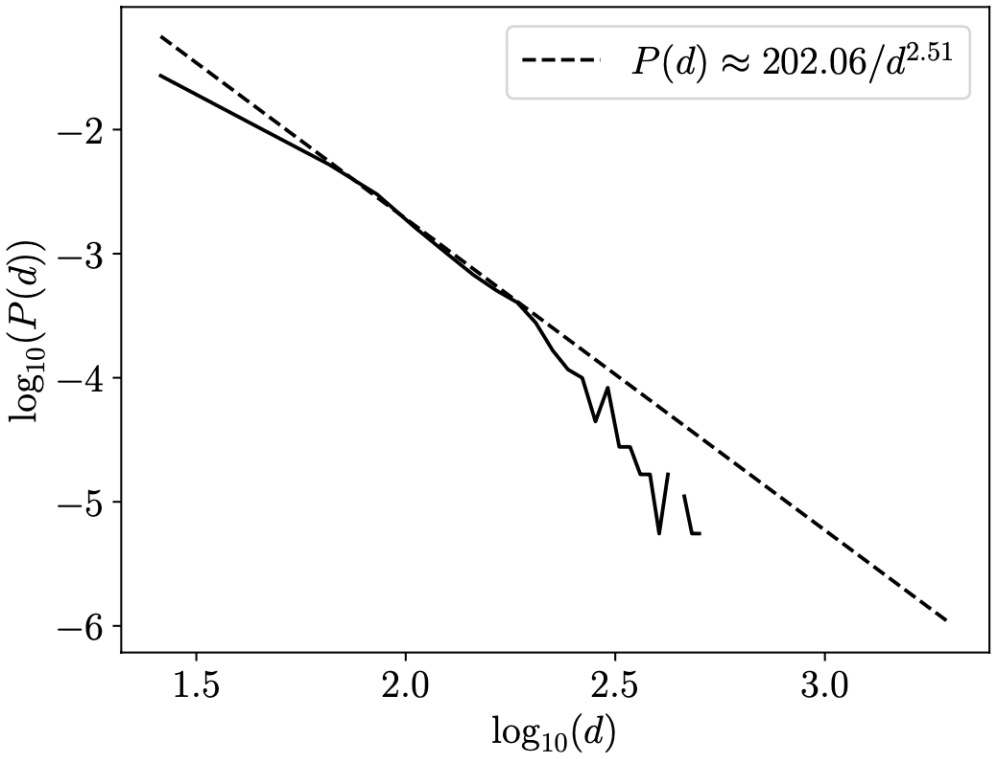

Figure 12. Log-log plot of the degree distribution for B for the Amazon data with power-law MLE estimate for $\gamma = 2.51$. The p-value for the KS test is $.47$, and thus we keep the null hypothesis.

In all, then, PA is clearly a reasonable mechanism for explaining the Amazon data set, whether we look at it from a user/item or item/user perspective. This is not the case for the Spotify data set, though, which shows a strong distinction in how tracks attach to playlists relative to the other direction. This fundamental inhomogeneity leads to struggles for the learning strategies we employ, a point we explore in the next section.

3. Link Prediction over Multiscale Networks

Using the template presented in[1], graph learning via MPNNs can be broadly explained via the following steps:

1. Separate $\mathcal{G}$ into a 'message-passing' graph $\mathcal{G}_M$ and a 'supervising graph' $\mathcal{G}_S$. The nodes are the same in both graphs, but the set of edges in $\mathcal{G}_M$ is a strict subset of those in $\mathcal{G}_S$.
2. Choose an initial, weight-dependent, embedding $\mathbf{h}_0 : \mathcal{N} \to \mathbb{R}^{N_f}$.

3. At the k^{th}-stage, for each node, say n_j, find the received *messages* $\mathbf{m}(n_j)$ where

   $\mathbf{m}\left(n_j\right) = \text{AGGREGATE}(\mathbf{h}_k\left(n_l\right), n_l \in \mathcal{N}\left(n_j\right))$

   where $\mathcal{N}(n_j)$ denotes the neighborhood, relative to $\mathcal{G}_M$, of node n_j.

4. Find $\mathbf{h}_{k+1}(n_j)$ via the formula

   $\mathbf{h}_{k+1}\left(n_j\right) = \text{UPDATE}(\mathbf{h}_k\left(n_j\right), \mathbf{m}\left(n_j\right))$

To update the weights in the embeddings, we use the recall score between playlist/user j and track/item l, say s^p_{jl} where

$$s^p_{jl} = \left\langle \mathbf{h}_{k+1}(n^+_j), \mathbf{h}_{k+1}(n^-_l) \right\rangle.$$

These 'positive' scores are computed over $\mathcal{G}_S$. For the j^{th}-playlist/user, we also sample some N_{neg} 'negative' samples, i.e., tracks/items which do not share an edge with playlist/user j in $\mathcal{G}_S$. This generates N_{neg} negative scores s^n_{jm}. We then use a variant of the Bayesian-Personalized Ranking Loss (BPRL), *cf.* [19], denoted by $\mathcal{L}$, in which we compute the average log-likelihood of the sigmoidal response of the form

$$\mathcal{L} = -\frac{1}{N_{neg}N_+}\sum_{j=1}^{N_+}\sum_{m=1}^{N_{neg}}\frac{1}{|\mathcal{N}(j)|}\sum_{l=1}^{|\mathcal{N}(j)|}\log\sigma\left(s^p_{jl} - \left|s^n_{jm}\right|\right).$$

where $|\mathcal{N}(j)|$ denotes the number of tracks/items in a neighborhood, defined relative to $\mathcal{G}_S$, of playlist/user j. As we found, using $\left|s^n_{jm}\right|$ in place of the more typical s^n_{jm} in effect helped fix a local coordinate system to the j^{th} playlist, thereby greatly enhancing training and ultimately generalized learning.

What distinguishes MPNNs is how AGGREGATE and UPDATE are implemented. As shown in[6], most (if not all) existing approaches can be written in the convolutional form

$$H_{k+1} = \sigma\left(\sum_{s=1}^{N_{stps}} C_s H_k W_{s,k}\right),\ 0 \le k \le N_{lay} - 1,$$

where H_k is the matrix formed from the separate node embeddings, C_s is a convolution matrix, $W_{s,k}$ are trainable weights, and σ is a sigmoidal response function. For example, the Light Graph Convolutional Network (LGC) method sets $N_{stps} = 1$ and C_1 is given by

$$C_1 = (D + I)^{1/2}(A + I)(D + I)^{1/2}.$$

For the Chebyshev spectral filter[4], or Chebnet, one chooses N_{stps} and then defines convolutional filters via the recursive formulas

$$C_1 = I,\ C_2 = \frac{2}{\lambda_M} L - I,\ C_s = 2C_2C_{s-1} - C_{s-2},\ 1 \leq s \leq N_{stps},$$

where the graph Laplacian $L = D - A$ with D the diagonal matrix of degrees so that

$$D = \begin{pmatrix} D_+ & \\ & D_- \end{pmatrix},$$

and λ_M is the maximum eigenvalue of L. The last example that we look at is GraphSage[18], which can be recast as a two-step filtration process in which $C_1 = I$ and $C_2 = D^{-1}A$.

As shown in[7], any MPNN can be proven to be equivalent to some version of the k-Weisfeiler-Lehman Test (kWLT); see[10] for further discussion and historical citations. While powerful, any kWLT is only a necessary condition for isomorphism, and counterexamples exist for all k[21]. This is due to each kWLT being, in effect, a graph-coloring scheme over k-tuples of the nodes, i.e., coloring on $\mathcal{N}^k$. In the 1WLT, the coloring of each node is altered by the colors of its neighbors. Each higher-order test then looks at longer-scale relationships between nodes to identify the particular color of any one node, thereby giving any kWLT, for $k \geq 2$, greater discriminatory power for larger k but also introducing a fundamental limitation due to the finite value of k. Perhaps somewhat frustratingly,[7] proves that generically LGC, GraphSage, and Chebnet are no more powerful than the 1WLT, though Chebnet can distinguish any two graphs that have different maximum eigenvalues of their respective Laplacians. This is used to explain the significant performance difference over the EXP dataset[22] as shown in[7]. However, while related, the question of whether an MPNN can accurately count subgraphs is not identical to its kWLT equivalence; again, see[7]. Likewise, the role that the preferential-attachment/scale-free property plays in link prediction problems has not, to the best of our knowledge, been explored.

3.1. Results

In response to these issues, we look at how LGC, GraphSage, and Chebnet perform on both the Spotify and Amazon data sets. The results in Figures 5 and 6 suggest that all three MPNNs will struggle due to a basic inability to capture the small-scale connected structures which appear in such large quantities throughout each network. That said, we would also anticipate that each MPNN would struggle less over the Amazon data. But this ignores the effect that PA, potentially appearing on subgraphs larger than σ_1, plays in the learning process. As we show, the absence of PA in the Spotify data helps the learning process

by essentially providing insight into important large-scale structure by way of there being a statistical distinction between the node classes.

In our numerical experiments, we fix the embedding dimension $N_f = 64$, the number of times we iterate through layers to $N_{lay} = 3$, and the learning rate of the ADAM optimizer to $.01$. Variations of these values were explored, and results were either unchanged or worse. For the Chebnet cases, on the Spotify data, we set $N_{stps} = 4$ while we found improved performance on the Amazon data when we set $N_{stps} = 3$. The data sets are separated into 70% of edges being used in training, 15% for validation, and 15% for testing. We measure the success of our predictions through both Area under Curve (AUC) and Recall-at-K (ReK) measurements. For the Spotify data, we set $K = 300$, representing 4% of the total tracks. To generate comparable results, we set $K = 392$ for the Amazon data, again representing 4% of the total items. For all cases, we compare using $N_{neg} = 1$ to $N_{neg} = 10$. In all cases, we train for 300 epochs across ten experiments for each case.

The results of our experiments are presented in Figures 13, 14, 15, and 16. For the Spotify data, the AUC scores shown in Figure 13, which in this case measure the classification problem of determining whether an edge exists between a particular playlist and track, clearly show every model is markedly better than random classification. Chebnet, in particular, stands out in this regard, with GraphSage performing the worst, though it is close to LGC's results modulo the higher standard deviation. Light-GCN is also distinguished by having the smallest standard deviation around the mean, especially when $N_{neg} = 10$. Likewise, we see Chebnet reaches a slightly higher overall AUC with $N_{neg} = 10$ and also exhibits a small reduction in the standard deviation of the AUC score.

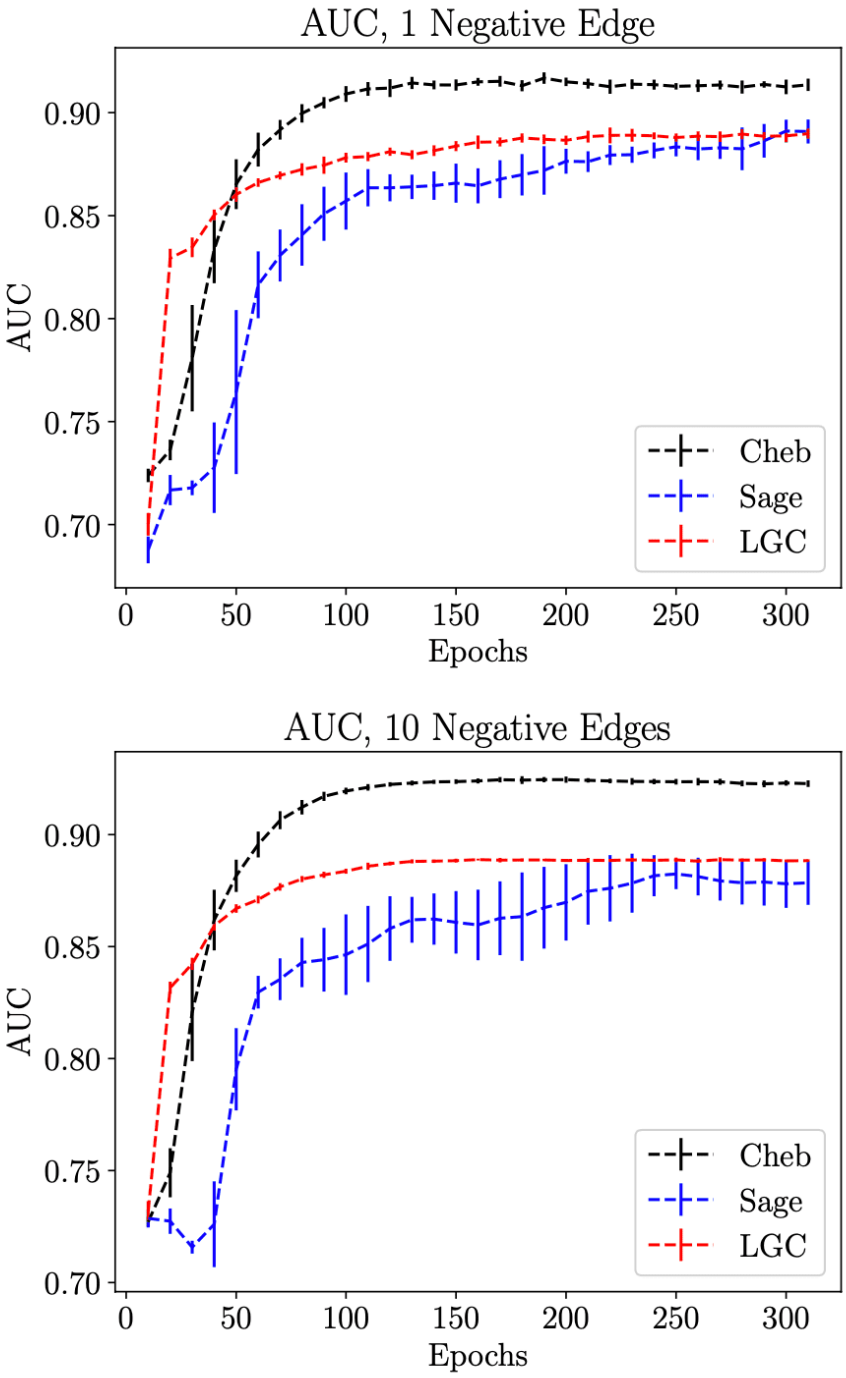

Figure 13. Comparison of AUC for Light-GCN (LGC), GraphSage, and Chebnet with negative samples set to $N_{neg} = 1$ and $N_{neg} = 10$ for the Spotify data set. Chebnet is trained with $N_{stps} = 4$. The vertical bars represent the standard deviation around the mean computed over ten trials.

doi.org/10.32388/074O28

However, the predicted ReK for LGC, GraphSage, and Chebnet shows strong distinctions, with Chebnet performing much better than its peers. Given the results in Figure 5, we posit that the differences in subgraph counting ability probably best explain the overall performance in learning. First, we note that Light-GCN, despite being no better or worse than a 1WLT, cannot do better than guessing after training. GraphSage is able to outperform LGC, but it exhibits significant spreads in its predictive power, especially when only one negative sample is used. In contrast, we see Chebnet is able to get both the best predictive performance with minimal spread in predictive outcome, especially when we use $N_{neg} = 10$. Thus, while increasing the exposure to negative samples does not necessarily create large improvements in ReK scores, we see that it plainly has an effect on the graph learning process that works in conjunction with the extent to which each MPNN is able to navigate the vast scales of subgraphs in the Spotify data.

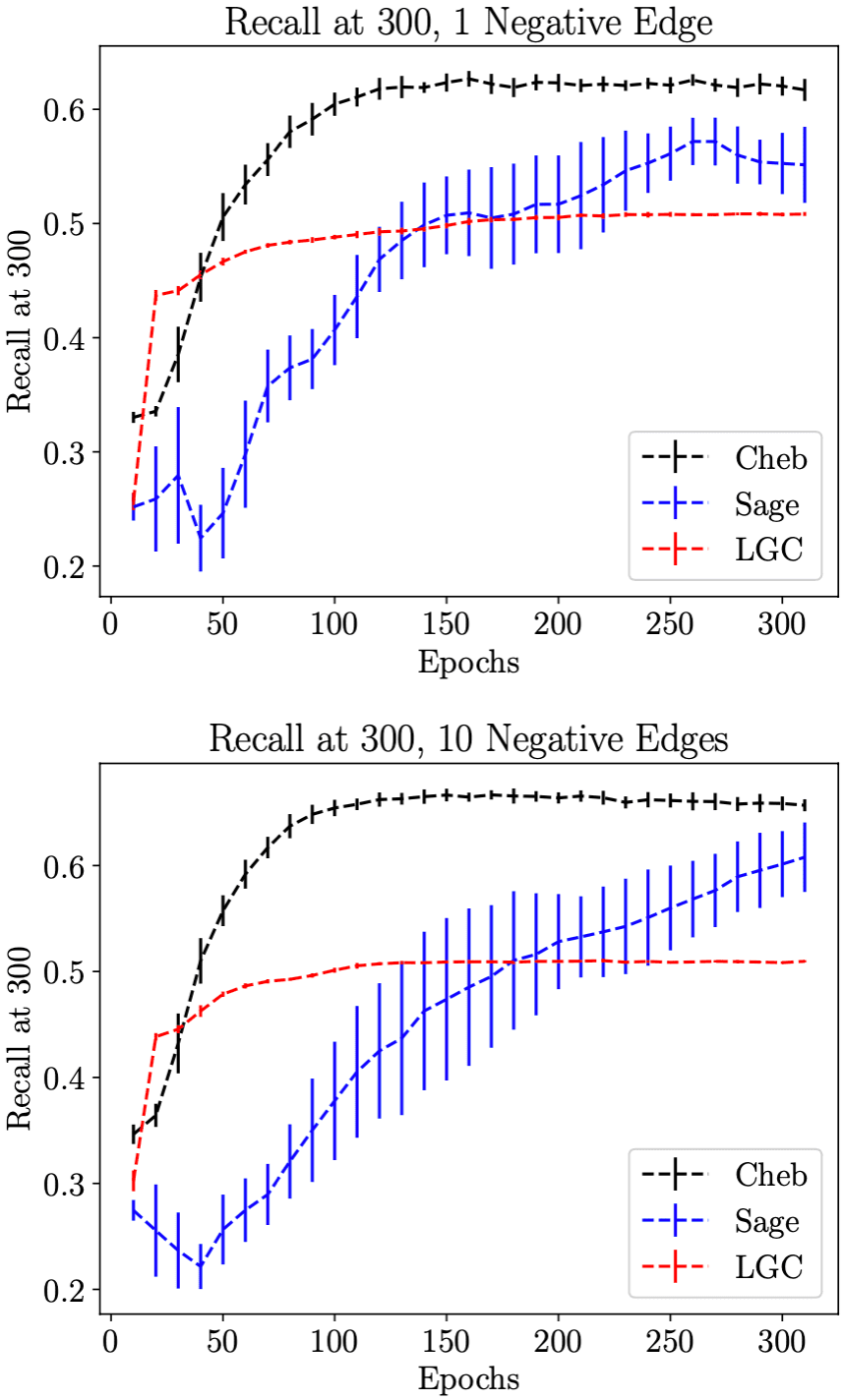

Figure 14. Comparison of Recall-at-K performance for Light-GCN (LGC), GraphSage, and Chebnet with negative samples set to $N_{neg} = 1$ and $N_{neg} = 10$ for the Spotify data set. Chebnet is trained with $N_{stps} = 4$. The vertical bars represent the standard deviation around the mean computed over ten trials.

doi.org/10.32388/074O28

The AUC scores for the Amazon data shown in Figure 15 show much the same behaviour as over the Spotify data. We note, though, that there is much less distinction in performance across the MPNNs. LGC continues to underperform, but the overall spread among the three methods is less, and GraphSage and Chebnet are almost indistinguishable. We note for $N_{neg} = 10$ that the performance of both is slightly improved, so increased exposure to negative sampling still improves training. In all, though, we see each MPNN is generally as strong a classifier over the Spotify data as the Amazon data.

doi.org/10.32388/074O28

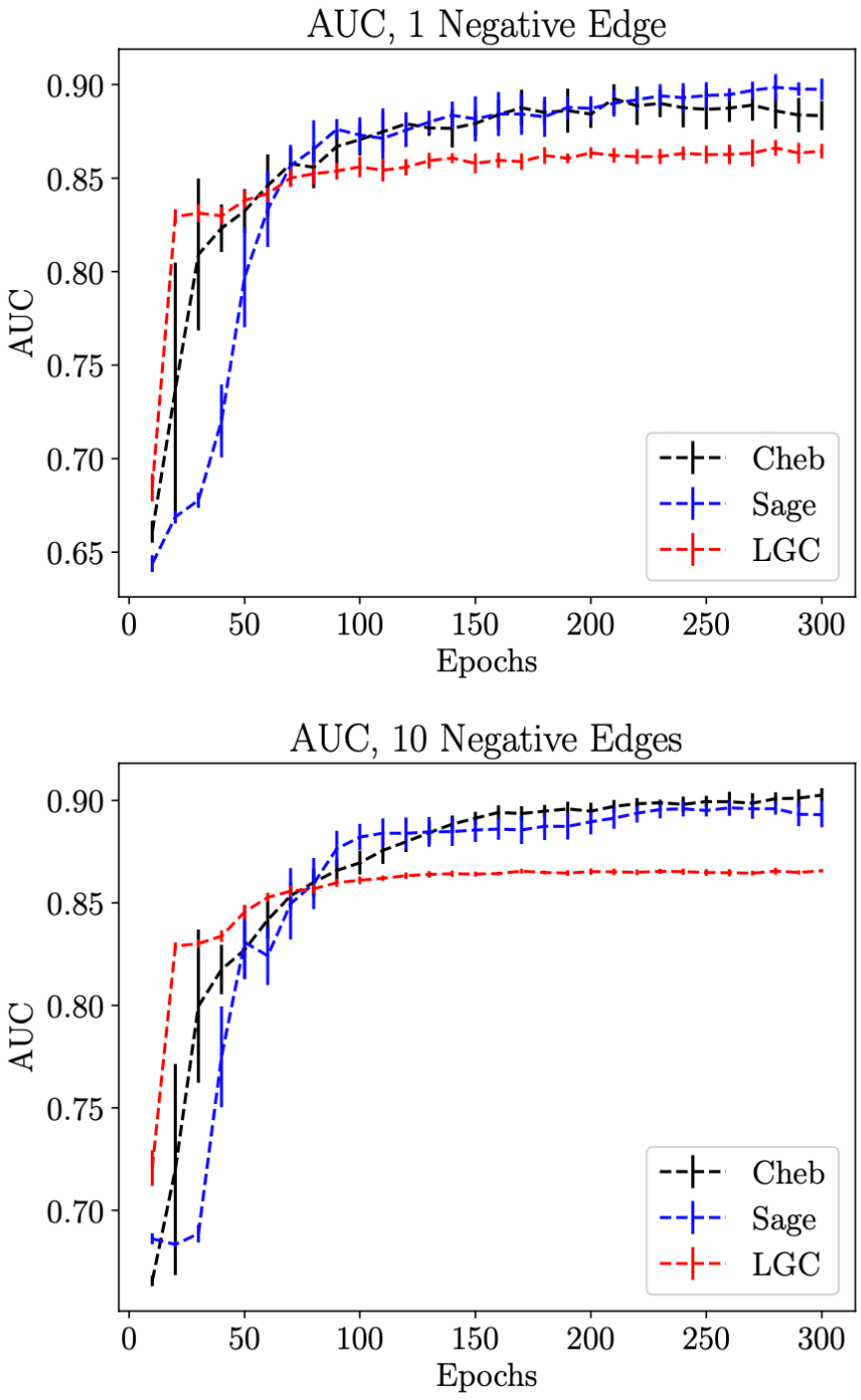

Figure 15. Comparison of AUC for Light-GCN (LGC), GraphSage, and Chebnet with negative samples set to $N_{neg} = 1$ and $N_{neg} = 10$ for the Amazon data set. Chebnet is trained with $N_{stps} = 3$. The vertical bars represent the standard deviation around the mean computed over ten trials.X

doi.org/10.32388/074O28

Looking at ReK, though, shows how different the graph learning process is on the Amazon data. First, we note that by including more negative samples, while GraphSage is now the best-performing MPNN, its distinction from Chebnet is nominal. We also see in this regard that the exposure to more negative samples has a dramatic effect on predictive power. We also observe that LGC cannot compete at link prediction, at least for the parameter choices we have made. This, then, is clearly a consequence of the large numbers of small-scale graphs as shown in Figures 5 and 6. Finally, we see the overall predictive power of all of the MPNNs is reduced on the Amazon data set, despite scaling our ReK criteria to be the same across both data sets.

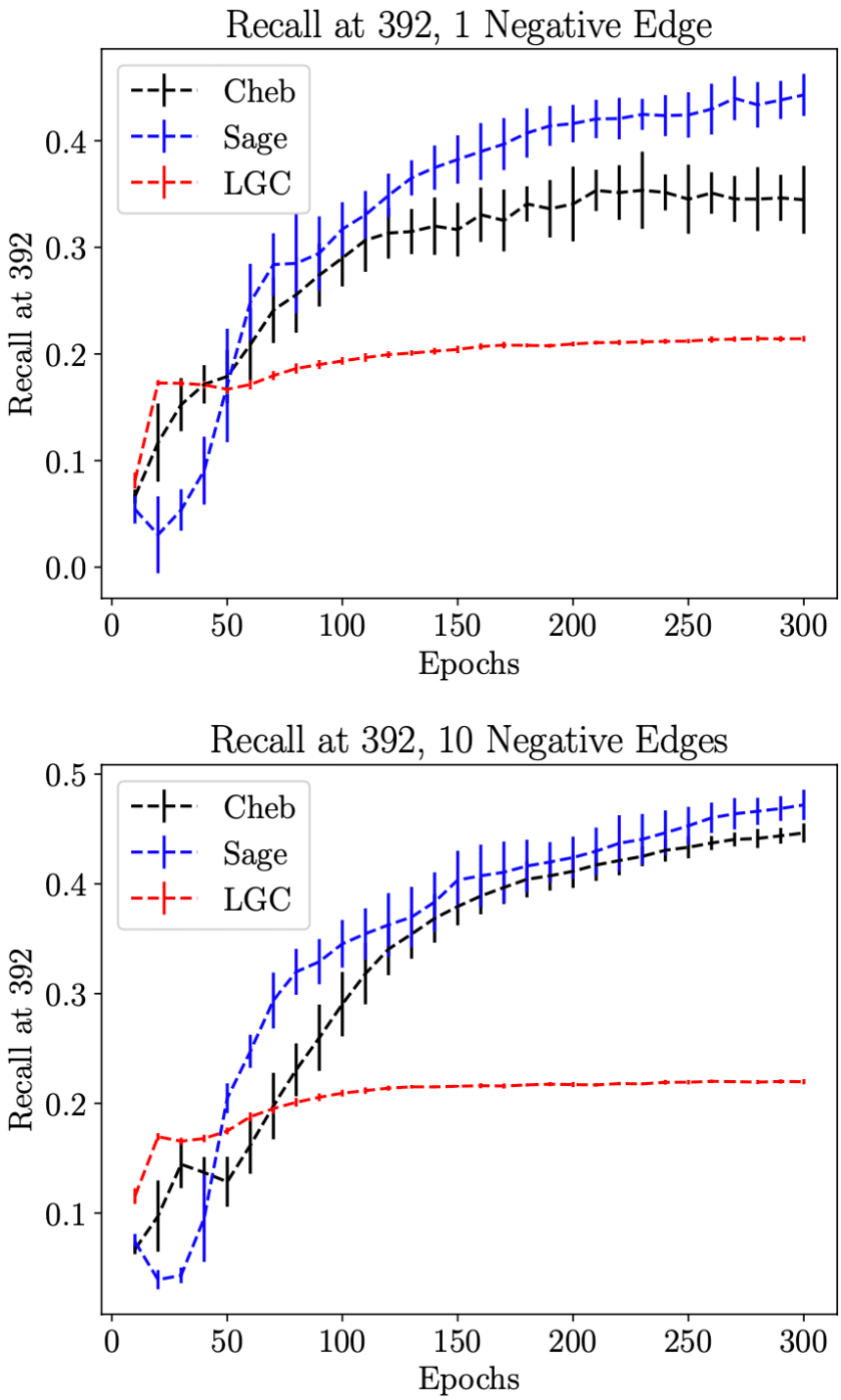

Figure 16. Comparison of Recall-at-K performance for Light-GCN (LGC), GraphSage, and Chebnet with negative samples set to $N_{neg} = 1$ and $N_{neg} = 10$ for the Amazon data set. Chebnet is trained with $N_{stps} = 3$. The vertical bars represent the standard deviation around the mean computed over ten trials.

doi.org/10.32388/074O28

4. Conclusion and Future Directions

As expected and as shown in the Spotify data set, the presence of large numbers of smaller scale subgraphs makes Chebnet the most competitive choice among the MPNNs studied in this work. Given that the subgraph counts are smaller (compare Figures 5 and 6) in the Amazon data, one might have imagined, keeping the discussion of kWL tests in mind, that the ReK scores would have presented themselves differently than they did. However, it appears that the fact that the Amazon data follows a power-law distribution with its attendant scale-free property causes all of the MPNNs to struggle. Upon further reflection, this makes sense. The question of predicting a track/item relative to a playlist/user would benefit from some strong distinction between the two classes. Clearly then, given that the Spotify data has such a strong asymmetry in the distributions of tracks-to-playlists versus playlists-to-tracks, *cf.* Figures 10 and 9, the Chebnet filter is clearly able to learn some part of this distinction and leverage it for enhanced predictive power.

Future work would first address the issue of PA degrading the performance of MPNNs. For the task of link prediction, the presence of PA most likely demands at a minimum adding graph attributes to help with distinguishing node type and improving link prediction, especially if it is done in a uni-directional way, e.g., strictly for item to user. Likewise, developing MPNNs that can cope with the large number of small scale subgraphs is already an area of active research, and the results of this work clearly motivate continued endeavour in this direction.

Appendix: Proofs of Theorems

Proof of Theorem 1

Theorem 1. *If for $j \neq k$,*

$$\min\left\{\frac{\max_{j,k}(B_{2h}^{+})_{jk}}{d_M^{+}}, \frac{\max_{j,k}(B_{2h}^{-})_{jk}}{d_M^{-}}\right\} = \epsilon \ll 1,$$

then for $d_j^{-}/d_M^{-} \gg \epsilon$

$$\sigma_j^2 = d_j^{(ch)} + \sum_{l\neq j} \frac{1}{d_j^{(ch)} - d_l^{(ch)}} \left(\left(B_{2h}^{(ch)}\right)_{lj}\right)^2 + \cdots$$

where $ch = \pm$ depending on which term defines ϵ.

Proof. Without loss of generality, we suppose that $ch = -$. Rescaling by d_M^-, we then have the perturbation problem

$$B^T B = d_M^- \left(\tilde{D} + \epsilon \tilde{B} \right),$$

where $\tilde{D} = D_-/d_M^-$ and $\tilde{B} = B_{2h}^-/\max_{j,k}(B_{2h}^-)_{jk}$. Note, through permutation, we order the values in $\tilde{D}$ in descending order to match that of the singular values.

To find $B^T B \mathbf{u}_j = \sigma_j^2 \mathbf{u}_j$, we use the expansions

$$\mathbf{u}_j = \mathbf{u}_{j,0} + \epsilon \mathbf{u}_{j,1} + \epsilon^2 \mathbf{u}_{j,2} + \cdots, \; \sigma_j^2 = \sigma_{j,0} + \epsilon \sigma_{j,1} + \epsilon^2 \sigma_{j,2} + \cdots.$$

At leading order, we get $\mathbf{u}_{j,0} = c\hat{\mathbf{e}}_j$ where c is an arbitrary constant and $\sigma_{0,j} = \tilde{d}_j$. The next order corrections are all of the form

$$\left(\tilde{D} - \sigma_{j,0} \right) \mathbf{u}_{j,k} = -\tilde{B} \mathbf{u}_{j,k-1} + \sum_{l=1}^{k} \sigma_{j,l} \mathbf{u}_{j,k-l}, \; k \geq 1.$$

Then, at every order, we get the solvability requirement

$$\sigma_{j,k} = \frac{\langle \tilde{B} \mathbf{u}_{j,k-1}, \mathbf{u}_{j,0} \rangle}{\langle \mathbf{u}_{j,0}, \mathbf{u}_{j,0} \rangle}.$$

Note, we have tacitly chosen our inhomogeneous solutions to be orthogonal to the kernel so that $\langle \mathbf{u}_{j,l}, \hat{\mathbf{e}}_j \rangle = 0$ for $l > 0$.

Defining the diagonal matrix

$$\left(\tilde{D} - \sigma_{j,0} \right)^{-P}_{kk} = \begin{cases} (\tilde{d}_k - \sigma_{j,0})^{-1} & j \neq k \\ 0 & j = k \end{cases}$$

we likewise find

$$\mathbf{u}_{j,k} = \left(\tilde{D} - \sigma_{j,0} \right)^{-P} \left(-\tilde{B} \mathbf{u}_{j,k-1} + \sum_{l=1}^{k} \sigma_{j,l} \mathbf{u}_{j,k-l} \right).$$

Stepping through these formulas up to second order, at $k = 1$, we get

$$\sigma_{j,1} = \frac{\langle \tilde{B} \mathbf{u}_{j,0}, \mathbf{u}_{j,0} \rangle}{\langle \mathbf{u}_{j,0}, \mathbf{u}_{j,0} \rangle} = 0,$$

where we get zero since $\tilde{B}_{jj} = 0$, and

$$\mathbf{u}_{j,1} = -(\tilde{D} - \sigma_{j,0})^{-P} \tilde{B} \mathbf{u}_{j,0}.$$

Proceeding to the next order, we find the correction $\sigma_{j,2}$ to be

$$\sigma_{j,2} = -\frac{\left\langle \left(\tilde{} D - \sigma_{j,0}\right)^{-P} \tilde{B}\mathbf{u}_{j,0}, \tilde{B}\mathbf{u}_{j,0} \right\rangle}{\langle \mathbf{u}_{j,0}, \mathbf{u}_{j,0} \rangle}.$$

Setting $c = 1$ and returning to our original scaling, we see that the singular values σ_j^2 have the expansion

$$\sigma_j^2 = d_j^- + \sum_{l \neq j} \frac{1}{d_j^- - d_l^-} \Big((B_{2h})_{lj} \Big)^2 + \cdots$$

□

Proofs for Counting Simple Closed Walk Homomorphisms

Lemma 2. *For $k \geq 2$,*

$$c_{2k}(\mathcal{H}_2) = 2^{k+1} - 4.$$

Proof. Labeling the vertices of $\mathcal{H}_2$ as (a,b,c), a $2k$ long sequence starting at a would look like

$$a,\ b,\ a/c,\ b,\ \cdots,\ a/c,\ b.$$

That gives 2^{k-1} total paths, and we subtract the sequence $a, b, a, b, \cdots, a, b$ for a total of $2^{k-1} - 1$ valid homomorphisms. We would get the same number if we started the sequence from c. If we started from b, we would get $2^k - 2$ valid sequences where we subtract the 2 sequences having only a or c. □

Lemma 3. *For $k \geq 2$,*

$$c_{2k}(\mathcal{C}_4) = 4(2^{2k-1} - 2^{k+1} + 2).$$

Proof. The closed walk $\mathcal{C}_4$ generates a balanced binary tree. We label the vertices with a, b, c, d in sequential order, and we first start the tree at a. Thus, for $2k$ positions, k of them can be either b or d, while $k-1$ of them can be a or c, i.e. we generate sequences of the form

$$a,\ b/d,\ a/c,\ b/d,\ \cdots,\ a/c,\ b/d.$$

There are 2^{2k-1} possible sequences. There are 2^k which do not include c or equivalently only have a. There are then $2^{k-1} - 1$ sequences with no b not counting the sequence with only a again. Likewise, there are $2^{k-1} - 1$ sequences with no d not counting the sequence with only a again. Thus, starting from a, we have $2^{2k-1} - 2^k + 2$ surjective homomorphisms. Given the symmetry of the labeling, we see we would get the same result starting from b, c, and d, thus we get the final result. □

Lemma 4. *For $k \geq 2$,*

$$c_{2k}(\mathcal{H}_4) = 2 \cdot 3^k - 3 \cdot 2^{k+1} + 6.$$

Proof. Following the method used thus far, if we start at a, we can trace out all possible sequences as

$$a,\ b,\ a/c/d,\ b,\ \cdots,\ a/c/d,\ b.$$

This gives a total of 3^{k-1} possible sequences. There are then 2^{k-1} sequences with no d and $2^{k-1} - 1$ sequences with no c not counting the sequence with only a again. This gives a total of $3^{k-1} - 2^k + 1$, and we see we get the same result starting from c or d for a total of $3(3^{k-1} - 2^k + 1)$ surjective homomorphisms. If we start from b, this gives us $3^k - 2^k - (2^k - 1) - (2^k - 2)$ surjective homomorphisms. Adding gets the final result. □

Statements and Declarations

Acknowledgments

We would like to thank Robert Simpson for participating in prior work on studying counts of subgraphs. Likewise, we would like to acknowledge that we used the code and discussion from https://medium.com/stanford-cs224w/spotify-track-neural-recommender-system-51d266e31e16 to begin our work on this project.

References

1. [a], [b], [c]*Hamilton WL (2020). Synthesis lectures on artificial intelligence and machine learning. pp. 1–159.*
2. [^]*Zhou J, Cui G, Hu S, Zhang Z, Yang C, et al. (2020). "Graph neural networks: A review of methods and applic ations." AI Open. **1**:57–81. doi:10.1016/j.aiopen.2021.01.001.*
3. [^]*Kipf TN, Welling M (2017). "Semi-supervised classification with graph convolutional networks." In: Interna tional conference on learning representations. Available from: https://openreview.net/forum?id=SJU4ayYgl.*
4. [a], [b], [c]*Defferrard M, Bresson X, Vandergheynst P (2016). "Convolutional neural networks on graphs with fast l ocalized spectral filtering." In: Proceedings of the 30th international conference on neural information proc essing systems. Red Hook, NY, USA: Curran Associates Inc. pp. 3844–3852.*
5. [^]*Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE (2017). "Neural message passing for quantum chemis try." In: Proceedings of the 34th international conference on machine learning - volume 70. Sydney, NSW, A ustralia: JMLR.org pp. 1263–1272.*

6. ^a ^b *Balcilar M, Renton G, Héroux P, Gaüzère B, Adam S, et al. (2021). "ANALYZING THE EXPRESSIVE POWER OF GRAPH NEURAL NETWORKS IN a SPECTRAL PERSPECTIVE." In: Proceedings of the 10th international conference on learning representations.*
7. ^a ^b ^c ^d ^e ^f ^g ^h ^i *Balcilar M, Heroux P, Gauzere B, Vasseur P, Adam S, et al. (2021). "Breaking the limits of mes sage passing graph neural networks." In: Meila M, Zhang T, editors. Proceedings of the 38th international c onference on machine learning. PMLR pp. 599–608.*
8. ^ *Xu K, Hu W, Leskovec J, Jegelka S (2019). "How powerful are graph neural networks?" In: International conf erence on learning representations. Available from: https://openreview.net/forum?id=ryGs6iA5Km.*
9. ^ *Weisfeiler B, Lehman A (1968). "A reduction of a graph to a canonical form and an algebra arising during t his reduction." Scientific and Technical Information. **2**:12–16.*
10. ^a ^b *Arvind V, Fuhlbrück F, Köbler J, Verbitsky O (2020). "On weisfeiler-leman invariance: Subgraph counts an d related graph properties." Journal of Computer and System Sciences. **113**:42–59. doi:10.1016/j.jcss.2020.04.0 03.*
11. ^ *Bevilacqua B, Frasca F, Lim D, Srinivasan B, Cai C, et al. (2022). "Equivariant subgraph aggregation netwo rks." In: International conference on learning representations.*
12. ^ *Subramonian A, Sagun L, Sun Y (2024). "Networked inequality: Preferential attachment bias in graph neu ral network link prediction." https://openreview.net/forum?id=4i4fgCOBDE.*
13. ^a ^b ^c *Barabási AL, Albert R (1999). "Emergence of scaling in random networks." Science. **286**(5439):509–512.*
14. ^a ^b ^c ^d ^e ^f ^g ^h ^i *Alon N, Yuster R, Zwick U (1997). "Finding and counting given length cycles." Algorithmica. **17**:209–223.*
15. ^ *Chen CW, Lamere P, Schedl M, Zamani H (2018). "Recsys challenge 2018: Automatic music playlist continu ation." In: Proceedings of the 12th ACM conference on recommender systems. New York, NY, USA: Associatio n for Computing Machinery pp. 527–528. doi:10.1145/3240323.3240342.*
16. ^a ^b *He X, Deng K, Wang X, Li Y, Zhang Y, et al. (2020). "LightGCN: Simplifying and powering graph convoluti on network for recommendation." https://arxiv.org/abs/2002.02126.*
17. ^a ^b ^c *Clauset A, Shalizi CR, Newman MEJ (2009). "Power-law distributions in empirical data." SIAM Revie w. **51**(4):661–703.*
18. ^a ^b *Hamilton W, Ying Z, Leskovec J (2017). "Inductive representation learning on large graphs." In: Guyon I, V on Luxburg U, Bengio S, Wallach H, Fergus R, et al. editors. Advances in neural information processing syste ms. Curran Associates, Inc. Available from: https://proceedings.neurips.cc/paper_files/paper/2017/file/5dd9d b5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.*

19. ^a, ^b*Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L (2009). "BPR: Bayesian personalized ranking fr om implicit feedback." In: Proceedings of the twenty-fifth conference on uncertainty in artificial intelligenc e. Arlington, Virginia, USA: AUAI Press pp. 452–461.*

20. ^^*Giscard PL, Kriege N, Wilson RC (2019). "A general purpose algorithm for counting simple cycles and simpl e paths of any length." Algorithmica. **81**:2716–2737.*

21. ^^*Cai J, Fürer M, Immerman N (1992). "An optimal lower bound on the number of variables for graph identifi cation." Combinatorica. **12**:389–410.*

22. ^^*Abboud R, Ceylan II, Grohe M, Lukasiewicz T (2021). "The surprising power of graph neural networks with random node initialization." In: Proceedings of the thirtieth international joint conference on artifical intell igence (IJCAI).*

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.