Simulation-Based Estimation of General Structural Network Models

Cameron Fen*

May 6, 2024

This paper addresses the issue of estimating structural graph/network models on data from a single graph/network. While there exist methods for estimating structural models on many independent and identically distributed (iid) graphs/networks (Banerjee et al., 2013), there are limited general-purpose algorithms to fit structural models on a given single network. With most graphs/networks, the likelihood function is both intractable, and graphs/networks can't easily be split into many iid components, so traditional methods like maximum likelihood and method of moments will not work. This study proposes an algorithm adapted from Deep Learning, dubbed Sequential Neural Posterior Estimation (SNPE), for network analysis. SNPE is a simulation-based estimator that can estimate likelihoods without a likelihood function and does not require a cross-section of iid samples. These two facts allow for general-purpose estimation of structural network/graph models. SNPE can recover calibrated parameter values that generate a ground truth graph/network from a structural model. Additionally, the algorithm is applied to estimate the homophily model (Bramoullé et al., 2012) on empirical data which was not attempted in the original paper. This study presents a promising algorithm for fitting structural models on a single network, opening avenues for future research in network estimation.

JEL Codes: C11, C68, C63, C45

Keywords: Neural Networks, Bayesian Inference, Network Estimation, Structural Models, Simulation-Based Estimators

*Cameron Fen is a PhD student at the University of Michigan, Ann Arbor, MI, 48104 (E-mail: camfen@umich.edu. Website: cameronfen.github.io.). The author thanks Florian Gunsilius, Matthew Shapiro, John Leahy, Toni Whited, David Childers, and Mark Newman for helpful feedback. All errors are my own.

I. Introduction

The estimation of structural models on networks/graphs is a less developed field compared to the dynamic estimation of non-network structural models. Current methods are specialized and cannot be applied to general structural models. A general-purpose structural graph/network estimation procedure would be helpful in the same way Method of Simulated Moments (MSM) (McFadden, 1989) opened up macroeconomic modeling and other fields. Estimating parameters of a structural model on a single network/graph is difficult because the nodes are not iid and the graph does not typically produce a tractable likelihood function. Despite these challenges, I use a machine learning technique, Sequential Neural Posterior Estimation (SNPE), which can estimate the parameters of a structural model with data from a single (potentially large) network in a full information or Bayesian manner.

To make the problem concrete, I'm defining a simple structural model based on Bramoullé et al. (2012). Take a graph/network of papers as nodes and citations as edges. Edges form in one of two ways: Either a paper randomly cites another paper with probability ω, or a paper that has cited a second paper, cites a third paper which the second paper also cited, with probability ψ. Geometrically, one can imagine this second connection as the "closing" of a triangle. More details of this model are discussed later.

The objective of this structural network estimation problem is to figure out the values of ω and ψ given a real network. The challenges of this problem are: 1) it is intractable to calculate the likelihood that a model generated the given real graph and 2) a collection of subgraphs constructed by taking sections or sampling in some manner from the whole graph generally do not share the same properties as one another or the entire graph. 1) implies that likelihood methods will not work and 2) implies that other simulation-based likelihood-free methods like Method of Moments (McFadden, 1989) are unworkable as well. However, SNPE neither requires a likelihood function nor iid subgraphs and can resolve this issue. One thing you can do with this, and most structural models, is simulate. For example, one calibrates parameters, ω and ψ, and then one forms edges based on the parameter values. Now one has simulated a network/graph from the structural model.

As long as one can do that and the structural model puts support on the true network, there is a good chance SNPE will be able to estimate. SNPE is fundamentally a Bayesian algorithm but to perform maximum likelihood, one can use flat priors and take the posterior mode. See Section II.D. for proofs and general properties of SNPE, and Section II.E. for why they generally don't hold in the network/graph regime. The section below, Section II. discusses methodology in more detail.

Many papers have implemented structural models of graphs/networks, but due to a lack of estimation tools, do not perform full estimation on real data (Calvo-Armengol and Jackson, 2004), (Carvalho and Voigtländer, 2014), (Calvó-Armengol and Jackson, 2007). Because conventional estimation techniques are generally ad-hoc or only work for data with multiple graphs/networks, many papers choose not to estimate their models on data and only report theoretical properties of their models (Jackson and Wolinsky, 1996), (Gilles, Johnson et al., 2000), (Furusawa and Konishi, 2007). As far as I know, there are no good alternatives for estimating general structural models of graphs/networks. One can design particular "reduced-form" or statistical models constructed to yield tractable likelihoods. The Exponential Random Graph Model (ERGM) (Robins et al., 2007) is one such statistical approach. The downside of this approach is that the model parameters have no meaning and just determine the shape of the exponential family distribution. In contrast, for example, ω corresponds to the probability a citation is formed randomly in the Bramoullé et al. (2012) model. My approach resolves this issue, allowing general structural models in the vein of Bramoullé et al. (2012) to be estimated, even if the structural model yields no closed-form likelihood function. Additionally, SNPE is a full information estimation algorithm unlike, for example, Banerjee et al. (2013).

The next section, Section II., discusses the SNPE simulation-based likelihood method. It also discusses the mixture of Gaussian density estimator used in SNPE and the Graph Convolutional Neural Networks (GCNN), where a neural network converts a graph into a set of statistics so that the density estimator can be conditioned on. The results section, Section III., applies the estimation routine on three different networks, demonstrating the ability of the algorithm to recover the calibrated parameters of these models. The empirical section, Section IV., applies the algorithm to the Bramoullé et al. (2012) structural model on the Karate Club dataset (Zachary, 1977), extending

the original paper by applying both the novel estimation technique and using a real-world empirical dataset.

II. Methods

I will present an overview of a simulation-based estimation approach, Sequential Neural Posterior Estimation (SNPE). This section will contain an exposition of the general technique, the Conditional Mixture of Gaussians (CMoG) density estimator used to make SNPE work, and the Graph Convolutional Neural Network (GCNN) that converts graph data into numerical data, which a CMoG can be conditioned on.

II.A. Background on Sequential Neural Posterior Estimation (SNPE)

This section will focus on the novel extensions of SNPE used in this paper and a rough background on SNPE methods described in more detail in Fen (2022). First I will start with an overview of Bayesian statistics.

Bayesian estimation, like in SNPE, is concerned with recovering a posterior distribution. Given a prior belief: $P(\theta)$ and new information Y created via a joint relationship with θ, Bayes' rule describes the optimal updating rule for the posterior belief of θ, $P(\theta|Y)$:

$$P(\theta|Y) = \frac{P(Y|\theta)P(\theta)}{P(Y)}$$

θ and Y could and generally are multi-dimensional. The baseline algorithm used for Bayesian inference is Markov Chain Monte Carlo (MCMC), but newer techniques like variational inference (Blei, Kucukelbir and McAuliffe, 2017) and SNPE, the technique applied in this paper, have advantages.

SNPE is a Bayesian simulation-based algorithm that can recover the posterior distribution of the parameters without likelihood function evaluations. As mentioned in the Introduction, Section

I., SNPE works by sampling from the prior parameters space, $\theta_i \sim P(\theta)$[1], and then a graph, Y_i, conditional on the first round prior samples, $Y_i \sim P(Y|\theta_i)$. Then using samples $Y, \theta \sim P(Y, \theta) = P(Y|\theta)P(\theta)$, one can estimate a machine learning conditional density estimator $P_\phi(\theta|Y)$. This will be the CMoG discussed in Section II.B.. Conditioning Y on the true graph data Y' gives the posterior, $P_\phi(\theta|Y = Y')$. Note how the likelihood function, $P(Y|\theta)$, is sampled from, but never evaluated, which is essential for estimation on graph/network data.

The most important step is the estimation of the density $P(\theta|Y)$ from joint samples Y, θ. Baseline density estimation methods like kernel density estimation (KDE) in the related simulation-based MLE algorithm proposed by Kristensen and Shin (2012) are not up to the task. KDEs can't handle conditional densities, end-to-end estimation with a GCNN (discussed in Section II.C.), and more than a couple of posterior dimensions in θ. SNPE uses a series of different deep-learning-based density estimators, that can overcome these issues. The most common SNPE density estimators with other datasets are the normalizing flow or the GAN. These are discussed more comprehensively in Fen (2022) –Section III.C–and sections following it. However, the CMoG density estimator is more suited to models with fewer parameters like structural models of networks/graphs and so that will be the focus of this paper.

II.B. The Density Estimators: Conditional Mixture of Gausians (CMoG)

This section is a exposition of the CMoG. Given data simulated from the joint distribution of data and parameters: $Y_i, \theta_i \sim P(Y, \theta) = P(Y|\theta)P(\theta)$, SNPE uses a density estimator like the CMoG to learn the posterior, $P(\theta|Y)$. When Y is conditioned on Y', one gets the posterior, $P(\theta|Y = Y')$. Any density estimator, not just the CMoG, takes samples and attempts to recover the probability density function that generates those samples.

The CMoG approach is a powerful tool for density estimation in low to medium-dimensional spaces, and in the context of simulation-based Bayesian inference, can enable the estimation of conditional posteriors. As outlined in Bishop (1994), a mixture of Gaussians (first unconditional) is a linear combination of n Gaussian distributions, each with its own mean, μ_i; covariance, Σ_i; and

1. In the Bramoullé et al. (2012) model, θ would stand in for both ω and ψ

probability weight, π_i. The probability density function (pdf) of a (multivariate) point, θ, under this mixture can be calculated using the equation:

$$P(\theta) = \sum_{i}^{n} \pi_i * N(\theta; \mu_i, \Sigma_i)$$

Here $N(\theta; \mu_i, \Sigma_i)$ indicates the probability of θ under the ith multivariate normal with mean μ_i and covariance Σ_i. Here is an image illustrating the structure of an unconditional mixture of Guassians in one dimension:

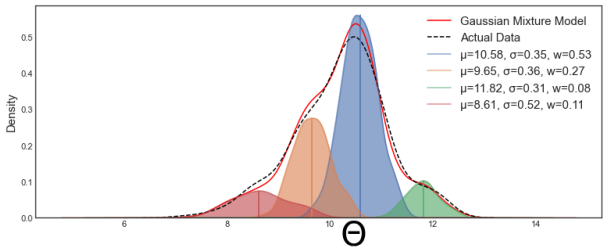

FIGURE I: A PICTURE DESCRIBING HOW A MIXTURE OF GAUSSIANS PERFORMS DENSITY ESTIMATION

Note that this picture shows how to stack arbitrarily small Gaussians to fit a pdf as well as necassary. To perform conditional density estimation (as opposed to unconditional density estimation), a neural network is used to model the conditional relationship between the parameters θ and the data Y. The neural network takes Y as input and returns the parameters for the CMoG, including the weights, means, and covariances for each Gaussian component. In this case the posterior conditional density $P(\theta|Y)$ can be represented by this equation:

$$P(\theta|Y) = \sum_{i}^{n} \pi_i(Y) * N_i(\theta; \mu_i(Y), \Sigma_i(Y)) \tag{1}$$

Using a neural network as a conditional functional approximator provides flexible and adaptable density estimates. Changes in the input data Y can result in changes to the CMoG, allowing the model to change distributions depending on the values of the conditioning data. Ultimately the

CMoG can be optimized via MLE to fit the conditional density given by a set of samples, $\{\theta_i, Y_i\}_i^I$. This approach estimates accurate conditional posteriors, even in relatively high-dimensional spaces, and is a useful tool in simulation-based inference.

Given any set of data, Y, a neural network can learn the CMoG parameters by mapping Y to the CMoG parameters - $\pi_i(Y)$, $\mu_i(Y)$, and $\Sigma_i(Y)$. However, if Y is a graph/network, more specialized models like GCNN should be used to convert a graph/network to the parameters of the CMoG. I will now discuss how a GCNN works.

II.C. Graph Convolution Neural Network (GCNN)

A GCNNs (Kipf and Welling, 2016) is necessary to transform graph/network data into numerical statistics or parameter values ($\pi_i(.)$, $\mu_i(.)$, and $\Sigma_i(.)$) for the mixture of Gaussians. This section will discuss the foundation behind GCNNs. Other embeddings like the FEATHER embedding can also work and are discussed in Appendix A. along with its advantages and disadvantages.

Starting on a basic level, the most basic type of neural network is the Feed-Forward neural network which is composed of multiple layers of interconnected nodes. Each layer takes the output of the previous layer and maps it to a new set of outputs. The basic unit of a Feed-Forward Neural Network is the layer, which is defined by the equation:

$$y_j = \sigma(A_i y_i + B_i) \tag{2}$$

Here, y_i and y_j represent the input and output of the layer, respectively, both of which are vectors. The matrix A_i maps the input, y_i, to the output, y_j, and B_i is a vector intercept term. Thus, the equation is a vector regression. The function σ is a nonlinearity that adds flexibility to the network, allowing it to learn complex relationships between inputs and outputs. A popular choice of nonlinearity is the Rectified Linear Unit (ReLU) shown in Figure II, which works well in many applications (Agarap, 2018).

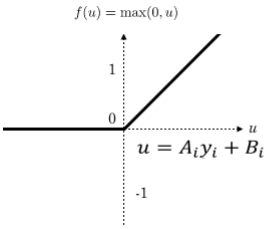

FIGURE II: RELU ACTIVATION FUNCTION: OFTEN USED IN NEURAL NETWORKS TO PROVIDE NON-LINEARITY

The idea behind a Feed-Forward neural network is to compose many copies of Equation (2), one on top of another. Thus given an input y_i, the first equation produces y_j, then y_j is an input to the next layer which is also a vectored-valued equation: $y_k = \sigma(A_j y_j + B_j)$, with different parameter values for A_j and B_j compared to A_i and B_i. One can continue composing additional layers on top of output y_j. If a neural network does not have nonlinearities, regardless of how many layers you have, it can only learn linear relationships. If neural networks have fairly generic nonlinearities, then, as parameter count increases, they can learn any continuous function (Hornik, Stinchcombe and White, 1989). Additionally, one can increase the dimensionality of the intermediate vector y_j, as its elements aren't constrained by the dimension of the input or output. Adding additional layers or increasing the dimensionality of intermediate layers increases the parameter count and neural network flexiblibility.

While a feed-forward neural network can be used to process many types of inputs, including graphs, it does not take into account the graph structure. The edge relationship between some nodes and not others can be exploited to improve the accuracy and information conveyed. This is where GCNNs come in. Nodes more closely connected via edges should effect each other more directly than nodes that are more distantly or not connected via edges. GCNN uses a convolution operation, similar to that used in image processing, to extract features from the graph[2]. A convolutional neural network modified to work with graph/network data accounts for graph/network topology and the features of individual nodes and edges when making predictions (Wu et al., 2020).

GCNNs have shown great potential in working with graph data, and their architecture is specifi-

2. See for instance: O'Shea and Nash (2015) for a tutorial on convolutional neural networks for image processing.

cally designed for this purpose. Typically, GCNNs define an adjacency matrix, denoted by A, which characterizes the edges between nodes in a graph. This matrix is often normalized by dividing by the degree matrix, a n by n matrix whose diagonal contains the degree of each of the n nodes[3]. The resulting normalized Laplacian matrix, as it's called, is denoted by A^*. In GCNNs, the input of a given layer, y_i, which can contain a vector of node characteristics, is fed into a GCNN Layer. This layer is defined as $y_j = \sigma(W_i y_i A^* + B_i)$, where W_i and B_i are learnable parameter matrices (vectors), and σ is a nonlinearity, all of which are the same as the feedforward case. The difference is that A^*, which denotes edge and degree relationships, allows only nodes connected via edges to affect one another. If one adds additional layers, the connections start by aggregating node characteristics via neighbors, then with a second layer, neighbors of neighbors, and with a third, neighbors of neighbors of neighbors, and so on. Thus output y_j is something like a sum of the effect on nearby nodes with the node in question times parameter values W_i and B_i. By using the normalized (or unnormalized) Laplacian, GCNNs extend the feed-forward neural network architecture to a structure that fits the structure of graph/network, even before estimating the parameter values. This inductive bias allows GCNNs to efficiently match data generating processes of and on networks.

Combining the SNPE algorithm with a GCNN, $X = G(Y)$, leads to an algorithm that can estimate structural models on graphs like the below Algorithm 1

3. More specifically, with A being the adjacency matrix, D being the degree matrix, and I being the identity matrix, the normalized Laplacian, $\tilde{L}$ is: $\tilde{L} = I - D^{-1/2}AD^{-1/2}$

Algorithm 1: SNPE Algorithm

Input: Simulator $P(Y|\theta)$, prior $P(\theta)$, data Y', graph statistics X, Graph embedding method, $G(.)$, flow $f_\phi(\theta|G(Y))$, Rounds R, Samples S;

Initialize: Posterior $P^{(0)} = P(\theta)$, data set D = {};

for $i \leftarrow 1$ **to** R **do**

Sample $\theta^{(n)} \sim P^{(i-1)}$ for $n = 1...S$ with Monte Carlo;

Simulate $Y^{(n)} \sim P(Y|\theta^{(n)})$ for $n = 1...S$;

Concatenate data $D = D \cup \{Y^{(n)}, \theta^{(n)}\}_{n=1}^{S}$;

while $f_\phi(\theta|X = G(Y'))$ *not converged* **do**

Sample $\{Y^{(i)}, \theta^{(i)}\}_i^B \sim D$ from D;

Train $f_\phi(\theta|X = G(Y))\dfrac{p^{(i-1)}}{p(\theta)}$ on $\{Y^{(i)}, \theta^{(i)}\}_i^B$;

end

Update posterior $p^{(i)} = f_\phi(\theta|X = G(Y'))$;

end

II.D. General Properties of SNPE and Embeddings

This section will prove that SNPE will converge to the Bayesian posterior given enough data and a density estimator that has a parametrization that can reproduce the posterior. These results won't quite hold for graph data (discussed below in Section II.E.), but give an intuition why this algorithm can be effective in the graph/network regime.

Statement 1: Given samples $Y_n, \theta_n \sim P(Y, \theta) = P(Y|\theta)P(\theta)$, as n goes to infinity, a density estimator, $f_\phi(\theta|G(Y))$, constructed in the manner described in algorithm 1, will converge to the true posterior as long as the posterior is in the support of the density estimator.

Proof: See proof of Proposition 1 in Papamakarios and Murray (2016) and note that, in this case, the density estimator is a probability density that is reweighted via importance sampling by $\dfrac{p^{(i-1)}}{p(\theta)}$. So $f_\phi(\theta|G(Y))$ will not be proportional but converge exactly to the posterior.

Statement 2: A (conditional) mixture of Gaussians, with enough Gaussian components will converge to any continuous distribution in the L^2 norm.

Proof: See, for example, Calcaterra and Boldt (2008) among many sources.

II.E. Limitations of Network-based Embedding Methods

This section discusses the main weakness of GCNNs, related techniques like graph transformers (Yun et al., 2019), and fixed graph embedding methods like FEATHER in Appendix A.

The results in Fen (2022) and partially stated above in II.D., borrowed from Papamakarios and Murray (2016), Huang et al. (2018), Goodfellow et al. (2014), and Bishop (1994) demonstrate that many machine learning density algorithms in conjunction with the SNPE algorithm should be universal approximator of probability distributions. That is, using a large enough model as a density estimator can arbitrarily closely approximate any continuous probability distribution (Huang et al., 2018), Calcaterra and Boldt (2008). However, this result no longer holds with graph data. GCNNs, graph attention transformers (Veličković et al., 2017), and related fixed embedding methods like FEATHER cannot distinguish between certain graphs that are non-isomorphic (Xu et al., 2018). Thus, they are not universal approximators.

Although the result of universal approximation no longer holds, GCNNs are a powerful model with useful inductive biases for embedding graphs. While the results in Papamakarios and Murray (2016) suggest that SNPE converges to the Bayesian posterior, the assumption that the conditioning variable captures all variability in the data is not met with GCNNs (or the FEATHER algorithm).

Another issue is that since networks are discrete objects, $G(Y)$, will be discrete too. However, to ensure G puts support on the true data, I add noise to the graph embedding, smoothing out the values. Another minor issue is saving graph data is generally quadratic in memory as a function of nodes, which means that if memory becomes a constraint one may be forced to use a fixed embedding like FEATHER. For more information see Appendix A.

Despite the lack of guarantees, these approaches are specifically tailored toward graphs and improve performance over other methods. This situation is analogous to proof regarding the efficacy of value function iteration which holds in the abstract but often no longer holds when using a tabular solution algorithm.

III. Results

This section demonstrates the efficacy of SNPE in recovering the calibrated parameters that generated a network/graph from a structural model. To the best of my knowledge, there are no algorithms that can estimate the parameters of general-purpose structural models on a single graph/network. If you build your model exactly right, for example, the Exponential Random Graph Model (ERGM) (Robins et al., 2007), perhaps you can derive a likelihood function. If you have iid graphs, you can use the Method of Moments. However, if you have neither of these things – the vast majority of graphs/networks – you likely cannot perform the rigorous estimation. Because of the lack of alternatives, I don't have a baseline model with which to compare my algorithm.

Additionally, I have found it difficult to reproduce the results of models like stochastic block models (Holland, Laskey and Leinhardt, 1983) or ERGMs using my algorithm. This has to be investigated further.

However concerning other models, to verify estimation works, I calibrate a given structural model with some parameter values. I then generate one graph/network simulated from this model at those parameters. Using this as my ground truth data, I hope my estimation technique can identify the parameters that generated the ground truth network/graph via SNPE.

I show how one can estimate parameters from the Newman-Watts-Strogatz graph (Newman and Watts, 1999), a widely-used network model. The algorithm is then tested on the Power-Law Cluster (Holme and Kim, 2002) and Relaxed Caveman graphs (Fortunato, 2010). These networks are used to model things like clustering behavior, power law dynamics, and degrees of separation, which have economic uses like modeling production networks, urban/suburban connectivity, and social networks. Finally, I perform an empirical exercise estimating the Bramoullé et al. (2012)

structural model on the Karate Club (Zachary, 1977) network.

These models are well-known and can easily generate simulated networks via the NetworkX[4] python package, making verification easier to execute. While these models are somewhat toy, as the literature generally lacks methods for estimation of structural models of graphs/networks, most models often have 2 or 3 parameters to help with intuition rather than the ability to fit the data. For example, Newman and Watts (1999) models the famous idea that any two people are separated by at most 6 degrees and shows in their structural network that the degree of separation is a function of three independent variables. From what I can tell, this is not quite an estimation, but using theoretical tools to learn the properties of these graphs. It seems to me that because estimation is so difficult, most papers focus on understanding the properties of a given graph which is easier with fewer parameter.

Estimating structural network/graph models on real-world data requires the structural model to put non-zero support on the empirical graph data set. Because in reality, the graph/network data will generally be misspecified, the structural model should put non-zero support on all graphs/networks, perhaps winnowing down to all graphs/networks with the same number of nodes. Many structural network models do not put support on every network/grahs with a given number of nodes. Because of this, they can only be tested via simulation and not on networks/graphs from the real world. Aside from my empirical exercise which does construct a structural model that puts support on all networks/graphs of n nodes, I use simulated data to test my approach.

In the simulated data approach, I first calibrate the parameters of the structural model to be some values. I then generate a single network/graph from this structural model and use this as the ground truth data. Using the SNPE algorithm, I try to recover the calibrated parameters. All priors are uniform so as not to bias the algorithm towards the correct solution.

In the diagrams in the next few sections (like Figures III, IV, V) the blue lines along the diagonal plots indicate the posterior derived from SNPE. The red line indicates the true parameter value, the y-axis is probability mass, and the x-axis contains the prior range for the parameter. The heat maps indicate two-way probability density based on the parameter on the row and the column

4. https://networkx.org/

further confirming the accuracy of the parameter estimation. In these charts, high probability mass is indicated by colors moving from dark blue to green to yellow, while the red dot is the true parameter value.

The following charts demonstrate that the estimated posterior often concentrates around the true parameter values, or at least has a posterior mode in a reasonably close location. These findings show the proposed algorithm can be applied to a wide range of network models and effectively recover the underlying structural parameters.

III.A. Newman-Watts-Strogatz Small World Graph

The focus of this section is to discuss the results obtained for the small-world graph, as proposed by Newman and Watts (1999). That paper builds off the Watts and Strogatz (1998) model. Watts and Strogatz (1998) extends the Erdos and Renyi (1959) random graph/network, where all nodes form edges with a constant random probability. Unlike real-world graphs/networks or social networks, in a random graph/network, nodes are relatively indistinguishable from one another. However, in the real world, being friendships, or more relevant, production networks or urban economies, some nodes are hubs with high edge degrees, some are spokes, and some are sparely connected. The Watts and Strogatz (1998) paper attempts to study the maximum number of edges to connect two nodes, using a structure resembling the real world hub and spoke system over the less realistic Erdos and Renyi (1959) random graph. In this setting, they test the idea of at most six degrees of separation between any two nodes. The Newman and Watts (1999) model modified the original model to allow for certain theoretical results on the degree of separation.

The Newman and Watts (1999) model has two parameters, namely, k and p. In this model, the number of nodes is arranged in a ring, and each node is connected to the k nearest neighbors in the ring. For each edge, there is a probability, p, that the first node, u, of an edge connecting u with v, will form a new edge with a random node, w. The Watts and Strogatz (1998) model then removes the edge u, v. In the original Watts and Strogatz (1998) sense, this helps to interpolate between the ring structure and a random graph, with p as the interpolation knob. If p is zero, you get a ring where each node is a hub to k nearest neighbors. If p is 100 %, then each of the k edges

are randomly rewired with another node, like in a random graph. Intermediate p allows the graph to form a hub and spoke system, where some nodes are hubs with more than k edges, and some are periphery nodes. In contrast, Newman and Watts (1999) does not remove any edges in the ring but only forms new edges. The original Watts and Strogatz (1998) will not work for real data because the graph always has a constant number of edges. If the real-world data differs in edge number, no probability is put on the true graph/network data. While I am using simulated data, having the true network be essentially a knife-edge solution, leads to estimation difficulties. Not to mention this structural model will not work with real-world data.

Moving on to estimation strategy, in cases where the parameters are integer-valued, i.e. in this case, k, I sample from the posterior continuous distribution of interest, but floor the numerical value to return an integer. Additionally, as the number of nodes is known in the dataset, the model generates a network with the same number of nodes as the data.

In my analysis, the calibrated value the algorithm should attempt to match is k fixed at 12, and p set to 0.33. The results in Figure III indicate that the proposed algorithm successfully recovers a posterior distribution whose mode is close to the true parameter values of k and p, demonstrating the algorithm's efficacy in recovering the underlying structural parameters for this specific type of graph. The posterior graph looks discrete because a histogram with many small buckets was used to illustrate the posterior.

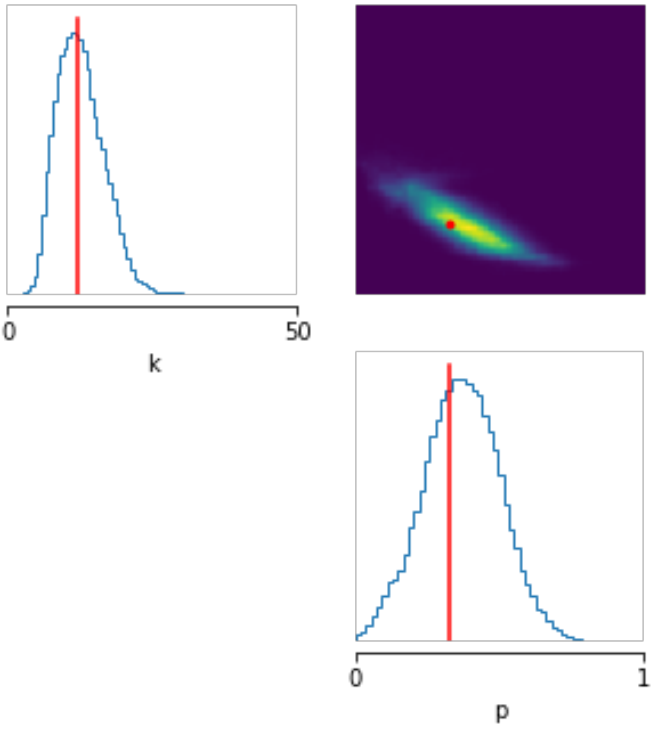

Figure III: SNPE Parameter Results of Estimating the Newman-Watts-Strogatz Small World Network

III.B. Power-Law Cluster Graph

One nice property of the Newman and Watts (1999) graph is that you have clustering behavior with some nodes being hubs and others periphery nodes. However, in the real world, many networks have vertices degree which are distributed via a power law. I would point to graphs/networks that may have both clustering and power law behavior in economics like production networks across small, medium, and large companies, urban road infrastructure, and the number of friendships people have. The Albert and Barabási (2000) graph exhibits power-law degree distribution and the Holme and Kim (2002) power-law cluster graph I use in this section extends the Albert and Barabási (2000) network/graph creation algorithm that also gives the power-law network/graph a clustering dynamic. While the Albert and Barabási (2000) paper is extremely well known, a

one-parameter estimation is not that interesting. Thus, this network/graph extension, while still well-known, adds a second parameter.

The Albert and Barabási (2000) has a single parameter, m, or the number of edges for each vertex. Over m rounds, one takes one edge starting from a given vertex, v, and that edge terminates to any other given vertex, w, with probability equal to the degree of w, divided by the total degree of all vertices in the graph/network: $P_w = \frac{k_w}{\sum_{l \in G} P_l}$. In each round, one goes through each vertex and then terminates after m rounds. The Holme and Kim (2002) extension is that, with probability p, for each edge created at a vertex v and terminating at vertex w in a given round, randomly connect v with a neighbor of w thereby closing a triangle. This adds clustering/hub and spoke behavior to the power-law distribution of the Albert and Barabási (2000) graph.

In this section, I illustrate the ability of SNPE to recover parameters of the Power-Law cluster graph (Holme and Kim, 2002) on simulated data. For our analysis, the random edge parameter is set to 3, and the triangle probability is set to 0.35. These parameter values result in fewer triangles being created, as the parameter values are on the low side. By inspection of Figure IV, the results indicate good identification of the edge parameter, with the mode at 3, but the triangle parameter is less well identified. This is consistent with the literature on Exponential Random Graph Models (ERGM), where it is widely known that triangles are difficult to identify (Handcock et al., 2020).

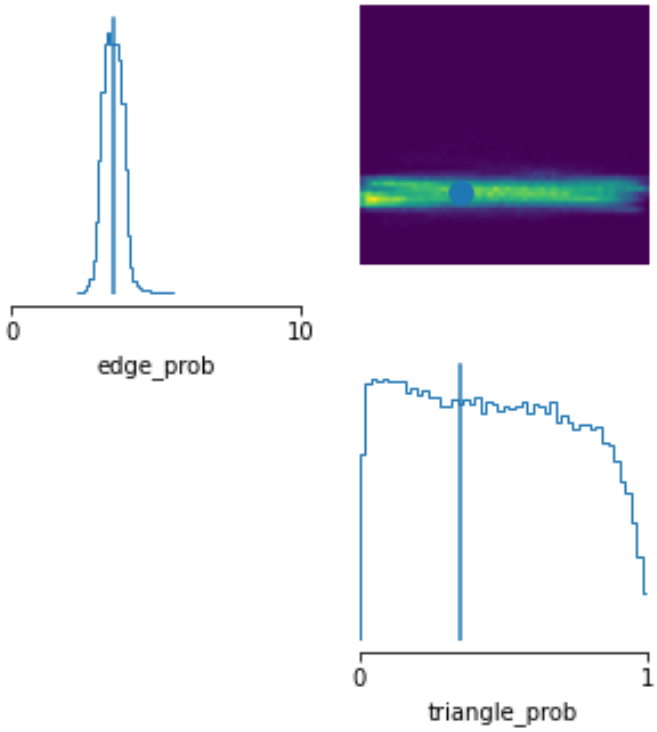

FIGURE IV: SNPE PARAMETER RESULTS OF ESTIMATING THE POWER-LAW CLUSTER GRAPH

III.C. Relaxed Caveman Graph

In this section, I estimate the Relaxed Caveman Graph proposed by Fortunato (2010). First, one defines n cliques of size m. A clique is a set of nodes that is fully connected. Then each edge of the network/graph is rewired to a random node with probability p. For the analysis, I use 10 cliques of size 10 with a rewiring probability of 60 percent. The high rewiring probability is intended to make the graph look less distinctive and resemble graphs seen in nature. Again this graph aims at exploring clustering behavior among nodes, which is a significant literature that matches data from economic, political, and sociological fields.

In contrast to the GCNN used in previous sections, I use the FEATHER graph embedding proposed by Rozemberczki and Sarkar (2020) to convert a graph into numerical statistics. This is discussed in Appendix A.

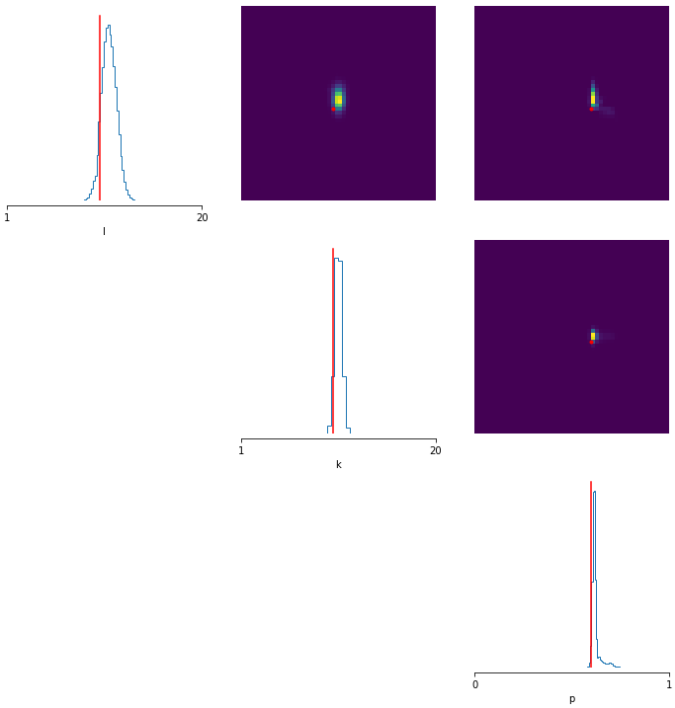

FIGURE V: SNPE PARAMETER RESULTS OF ESTIMATING THE RELAXED CAVEMAN GRAPH

The posterior distribution for the Relaxed Caveman network/graph shows good convergence of the algorithm, with the true parameters being very close to the mode of the posterior distribution. The posterior mode is 11 cliques of size 10 with a rewiring probability of 61%. This aligns with the true values of 10 cliques of size 10 with 60% rewiring. Here the number of nodes is not specified at the beginning. The 95% Bayesian credible intervals are between 9 and 12 cliques, 9 and 11 members per clique, and .59-.69 rewiring probability. The model seems too confident around the rewiring probability, but otherwise, the algorithm seems to be effective. Overall, the results suggest that the proposed algorithm can successfully estimate the parameters of the Relaxed Caveman graph, with some minor deviations in the estimated posterior distribution.

IV. Empirical Application: Homophily Connection Networks

This section outlines an empirical application of an algorithm to real data based on the citation graph homophily paper by Bramoullé et al. (2012). The algorithm extends the results of the original paper by structurally estimating the model, which was not done in the original study.

Instead of performing a full-scale estimation, Bramoullé et al. (2012) uses citations and citations of cited papers to come up with statistics that approximate the true underlying parameter value for ω ad ψ. The paper assumes that n nodes are born one after another. Each node that is born sends $m > 1$ connections to previously born nodes. A fraction of these links, m_r, are connections at random, and the remaining fraction of these links, $m_s = m - m_r$, are search links – connections formed by citing a paper cited by a paper you cited.

However, this approach is less realistic for a graph/network of friends in the Karate Club data set Zachary (1977) at a (two year) snapshot in time, which is the empiral data I use. The idea that the first couple node have a huge advantage in making friends doesn't match reality. Additionally, the original Bramoullé et al. (2012) model, seems to assign almost no support to a wide variety of relatively reasonable graphs, in favor of a scaling power law where the first couple of initialized nodes get the lion share of connections. Considering the necessity of a network/graph estimation model to put support on the entire space because the model is likely misspecified, I modify the Bramoullé et al. (2012) model to be more applicable to this scenario and to put support essentially on any network/graph with a certain number of nodes.

In the modified model, we are modeling the connection of Karate Club friendships data at a single snapshot of time. Each node is a person, and edgers suggest some sort of interaction or friendship. All nodes are initialized at the beginning and we go through every node once per round. When at a given node, u, there is a probability, ω, that that node has a direct connection with another randomly chosen node. Additionally, at each node in each round, there is a probability, ψ, that the given node, u, with a randomly chosen neighbor, w, will connect with a randomly chosen neighbor of w, but not neighbor of node u. This is the closing of a triangle. No edges are formed,

if no triangles can be closed, regardless of the value of ψ. Since temporal relationships have been destroyed, this is an undirected network with the edge representing a mutual interaction. There are 15 rounds in total which, given both random and search connection types, allows more than enough edges to form a complete graph using the 34 node Karate Club dataset[5]

Unlike this paper, the original Bramoullé et al. (2012) paper did not provide an estimation procedure for their approach. They only used statistics from a network of physics papers/citations to approximate the random connection and search connection rate. Any connection that closed a triangle, was considered a ψ-search/homophily connection. Any other connection was considered a ω-random connection. To the best of my knowledge, there were no tools for the estimation of their model and so they did not attempt it. This approach could overcount search/homophily citations, as some connections that close a triangle could be randomly created. On the other hand, if the model is misspecified, it could also undercount search connections. For example, one node, u, cites (or befriends) another node, x, because they found a third node, v, that cited (or befriends) the second node, x. However, u does not report being connected to x, even though that's how u learned about v. This is like the closing of a triangle, but the referring person, v, has a missing connection with the original node u.

IV.A. Karate Club Friendships Graph

This paper estimates a modified Bramoullé et al. (2012) model applied to the empirical Zachary Karate Club dataset. The first parameter in the diagonal of Figure VI is the random connection rate ω. The second is the homophily connection probability ψ

5. All nodes in a complete graph will have a degree of 33. There are 15 total rounds with 2 edges formed each round ($2 * 15 = 30$). Additionally, each edge increases the degree of two nodes by 1

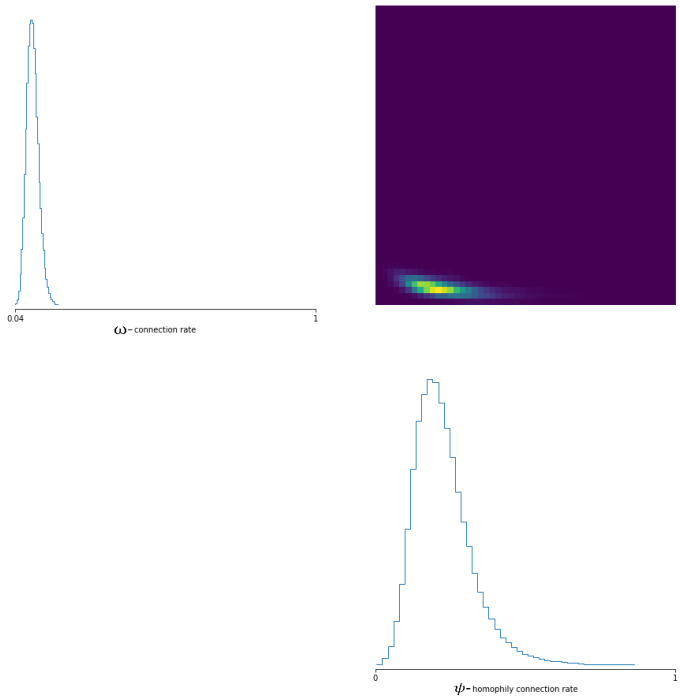

FIGURE VI: SNPE PARAMETER RESULTS OF ESTIMATING BRAMOULLÉ ET AL. (2012) MODEL ON REAL-WORLD KARATE CLUB DATA

I estimate the model on the Karate Club dataset and obtain the posterior distribution shown in the above Figure VI. This figure displays the Bramoullé et al. (2012) model estimated on the Karate Club data. The Poserior mode is .08 for the ω parameter and .23 for the ψ parameter. Since this is a test on empirical data, there are no ground truth parameter values I can cross reference this with. However, one can take comfort that the posterior distribution is unimodal and has a mode concentrated around realistic values. As a test to make sure the model can identify parameters in a simulated setting, I then take the derived empircal parameters' mode, calibrate the same Bramoullé et al. (2012) model, and estimate on the simulated data generated by the calibrated model. The results of this estimation are shown in the figure below:

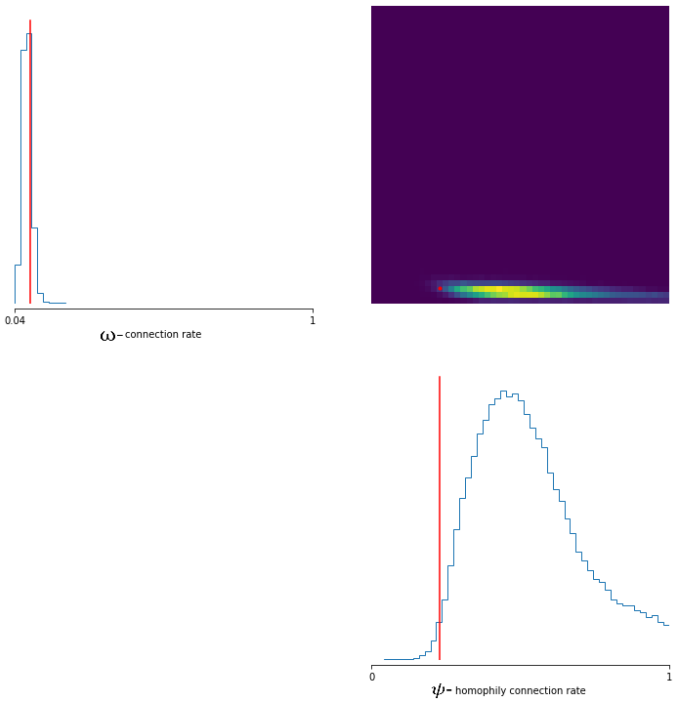

FIGURE VII: SNPE PARAMETER RESULTS OF ESTIMATING BRAMOULLÉ ET AL. (2012) MODEL ON SIMULATED DATA

Testing the model on simulated data, it is evident from Figure VI that the model does a good job of identifying the ω-random connection parameters. However, inspection suggests the model performs poorly in identifying the ψ-homophily search parameter. However the mode of the posterior is .045 and .185, which is close to the calibrated values of .08 and .23. Unsurprisingly, the model is not confident in the mode as the 95% credible interval is between 4% and 68% for the random connection rate and 8% to 99% for the homophily search parameter. These findings suggest that the Bramoullé et al. (2012) model provides a reasonable but not amazing explanation of the observed friendship connections in the Karate Club dataset, as well as providing insights into the formation of social networks more generally.

V. Conclusion

In conclusion, this paper introduces a novel method to estimate structural network models on a single network. The approach is general-purpose and can estimate many models, regardless of the structure, as long as one can simulate graphs from the structural model and the structural model puts nonzero support on the empirical data used for estimation. This method is also full information or Bayesian depending on econometrician preference. Although this approach is relatively unexplored, the results of this study demonstrate that it is a promising approach that can lead to a better structural understanding of networks in the real world. However, this approach is not without its limitations. One of the central issues is to clarify both theoretically and empirically what sort of network models this approach can effectively estimate. Additionally, many structural models have to be restructured so that the model puts positive support on the data used for estimation. I also have trouble estimating models like ERGMs, but not other models, like the ones displayed. Nevertheless, there are many models that have not been estimated in the literature that can be estimated in this way with some modifications to, for example, adjust for the support issue. In light of these potential avenues for further research, this approach seems like a promising tool for future analysis and has the potential to advance the study of network models.

References

Agarap, Abien Fred. 2018. "Deep learning using rectified linear units (relu)." *arXiv preprint arXiv:1803.08375*.

Albert, Réka, and Albert-László Barabási. 2000. "Topology of evolving networks: local events and universality." *Physical review letters*, 85(24): 5234.

Banerjee, Abhijit, Arun G Chandrasekhar, Esther Duflo, and Matthew O Jackson. 2013. "The diffusion of microfinance." *Science*, 341(6144): 1236498.

Bishop, Christopher M. 1994. "Mixture density networks."

Blei, David M, Alp Kucukelbir, and Jon D McAuliffe. 2017. "Variational inference: A review for statisticians." *Journal of the American statistical Association*, 112(518): 859–877.

Bramoullé, Yann, Sergio Currarini, Matthew O Jackson, Paolo Pin, and Brian W Rogers. 2012. "Homophily and long-run integration in social networks." *Journal of Economic Theory*, 147(5): 1754–1786.

Calcaterra, Craig, and Axel Boldt. 2008. "Approximating with gaussians." *arXiv preprint arXiv:0805.3795*.

Calvo-Armengol, Antoni, and Matthew O Jackson. 2004. "The effects of social networks on employment and inequality." *American economic review*, 94(3): 426–454.

Calvó-Armengol, Antoni, and Matthew O Jackson. 2007. "Networks in labor markets: Wage and employment dynamics and inequality." *Journal of economic theory*, 132(1): 27–46.

Carvalho, Vasco M, and Nico Voigtländer. 2014. "Input diffusion and the evolution of production networks." National Bureau of Economic Research.

Erdos, P, and A Renyi. 1959. "On random graphs I." *Publ. math. debrecen*, 6(290-297): 18.

Fen, Cameron. 2022. "Fast Simulation-Based Bayesian Estimation of Dynamic Models using Normalizing Flow Neural Networks."

Fortunato, Santo. 2010. "Community detection in graphs." *Physics reports*, 486(3-5): 75–174.

Furusawa, Taiji, and Hideo Konishi. 2007. "Free trade networks." *Journal of International Economics*, 72(2): 310–335.

Gilles, Robert P, Cathleen Johnson, et al. 2000. "original papers: Spatial social networks." *Review of Economic Design*, 5(3): 273–299.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. "Generative adversarial nets." *Advances in neural information processing systems*, 27.

Handcock, Mark S, David R Hunter, Carter T Butts, Steven M Goodreau, Pavel N Krivitsky, and Martina Morris. 2020. "ergm: Fit, Simulate and Diagnose Exponential-Family Models for Networks."

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

Holme, Petter, and Beom Jun Kim. 2002. "Growing scale-free networks with tunable clustering." *Physical review E*, 65(2): 026107.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. 1989. "Multilayer feedforward networks are universal approximators." *Neural networks*, 2(5): 359–366.

Huang, Chin-Wei, David Krueger, Alexandre Lacoste, and Aaron Courville. 2018. "Neural autoregressive flows." 2078–2087, PMLR.

Jackson, Matthew O, and Asher Wolinsky. 1996. "A strategic model of social and economic networks." *Journal of economic theory*, 71(1): 44–74.

Kipf, Thomas N, and Max Welling. 2016. "Semi-supervised classification with graph convolutional networks." *arXiv preprint arXiv:1609.02907*.

Kristensen, Dennis, and Yongseok Shin. 2012. "Estimation of dynamic models with nonparametric simulated maximum likelihood." *Journal of Econometrics*, 167(1): 76–94.

McFadden, Daniel. 1989. "A method of simulated moments for estimation of discrete response models without numerical integration." *Econometrica: Journal of the Econometric Society*, 995–1026.

Narayanan, Annamalai, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. 2017. "graph2vec: Learning distributed representations of graphs." *arXiv preprint arXiv:1707.05005*.

Newman, Mark EJ, and Duncan J Watts. 1999. "Renormalization group analysis of the small-world network model." *Physics Letters A*, 263(4-6): 341–346.

O'Shea, Keiron, and Ryan Nash. 2015. "An introduction to convolutional neural networks." *arXiv preprint arXiv:1511.08458*.

Papamakarios, George, and Iain Murray. 2016. "Fast ε-free inference of simulation models with bayesian conditional density estimation." *Advances in neural information processing systems*, 29.

Robins, Garry, Pip Pattison, Yuval Kalish, and Dean Lusher. 2007. "An introduction to exponential random graph (p*) models for social networks." *Social networks*, 29(2): 173–191.

Rozemberczki, Benedek, and Rik Sarkar. 2020. "Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models." 1325–1334.

Rozemberczki, Benedek, Oliver Kiss, and Rik Sarkar. 2020. "Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs." 3125–3132, ACM.

Veličković, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. "Graph attention networks." *arXiv preprint arXiv:1710.10903*.

Wang, Lili, Chenghan Huang, Weicheng Ma, Xinyuan Cao, and Soroush Vosoughi. 2021. "Graph Embedding via Diffusion-Wavelets-Based Node Feature Distribution Characterization." 3478–3482.

Watts, Duncan J, and Steven H Strogatz. 1998. "Collective dynamics of 'small-world'networks." *nature*, 393(6684): 440–442.

Wu, Z, S Pan, F Chen, G Long, C Zhang, and PS Yu. 2020. "A Comprehensive Survey on Graph Neural Networks." *IEEE Transactions on Neural Networks and Learning Systems*.

Xu, Keyulu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*.

Yun, Seongjun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. "Graph transformer networks." *Advances in neural information processing systems*, 32.

Zachary, Wayne W. 1977. "An information flow model for conflict and fission in small groups." *Journal of anthropological research*, 33(4): 452–473.

Department of Economics, University of Michigan, Ann Arbor

Appendix

A. FEATHER Network Embedding

An alternative to GCNNs is to use an out-of-the-box graph embedding technique without learning parameters. The FEATHER algorithm (Rozemberczki and Sarkar, 2020) is an alternative that doesn't require parameter tuning. As the SNPE often requires 50 thousand, 100 thousand, or 500 thousand samples, storing graphs which are, unless sparse, quadratic in memory as a function of nodes. This means that storing 500 thousand large graphs is probably intractable on a single computer memory-wise. To address this, one can use fixed embeddings like FEATHER. Because the parameters of FEATHER never change, one can throw away the graph and only store the embedding, which is constant/or linear in memory with respect to the number of nodes.

FEATHER learns node embeddings from the characteristic function of the probability density function (pdf) implied by a random walk from a node. This is achieved by learning the function that maps the node-level features to the complex plane. The algorithm then evaluates the probability associated with each node of the random walk on the network for a certain number of iterations and then converts the probabilities to their characteristic function. The network-level embedding is obtained by averaging the node embeddings.

For each node, the FEATHER algorithm attempts to learn the characteristic function for the probability distribution of a random walk starting from the source node. Given a network Y with nodes and edges, V and E, the FEATHER algorithm learns the function:

$$E[e^{i\theta x}|Y,u] = \sum_{w \in V} P(w|u)(\cos(\Theta x_w) + i\sin(\Theta x_w))$$

Here $P(w|u)$ is the probability that node w is reached from node u after r random walk steps. x_w are the node-level features. θ and r are hyperparameters and one should produce a number for a variety of values. As my graphs has no node features, the features used are the default features of eccentricity, transitivity, and degree. One can evaluate the random walk on a graph for r iterations from 1 to n steps from the original node. Then one can also evaluate the characteristic function at

designated points of Θ. To get a graph-level embedding from these node embeddings, the algorithm averages across node embeddings.

While FEATHER is the embedding technique used in this paper, many other approaches can be used. Other graph embedding techniques include graph2vec (Narayanan et al., 2017), wavelets-based embeddings (Wang et al., 2021), and many others from libraries such as the Karate Club graph library (Rozemberczki, Kiss and Sarkar, 2020). This library has no relation to the Zachary Karate Club dataset (Zachary, 1977), other than as an acknowledgement of the impact the Zachary data set has on the field.