Simulation-Based Estimation of General Structural Network Models

Cameron Fen*

November 9, 2023

This paper addresses the issue of estimating structural models on data from a single graph/network. For example, in macroeconomics, one could use the production network of the US, where nodes are firms and edges are supplier/customer relationships, and one learns the value of the parameter that dictates the probability of an edge forming. This is a longstanding problem in Economics and related fields. While there exist methods for estimating structural models on many iid graphs/networks, there is limited research on a general-purpose algorithm to fit structural models on a single network. This study proposes an algorithm adapted from deep learning, dubbed Sequential Neural Posterior Estimation (SNPE), for network analysis, which enables Bayesian and likelihood estimation of arbitrary structural models, given relatively standard conditions. SNPE is a simulation-based estimator, which can produce likelihoods via samples from a distribution. Networks sampled from a model are converted to numerical statistics via graph neural networks and other methods. Then one derives the posterior by conditioning the distribution of parameters on the statistics derived from the data-generating graph. Simulated tests demonstrate the effectiveness and accuracy of this approach. To demonstrate the capability of this model, the algorithm is applied to estimate a homophily citation model (Bramoullé et al., 2012) on empirical data, which was not attempted by 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 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 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 arbitrary structural models. The need for a general-purpose structural network estimation procedure is necessary, as the Method of Simulated Moments (MSM) (McFadden, 1989) did for macroeconomics and other fields. Estimating parameters of a structural model on a single 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, SNPE, which allows for the estimation of structural models for data from a single (and potentially large) network, overcoming these issues.

To make the problem concrete, I'm going to define a simple structural model based on Bramoullé et al. (2012). Take a network of people as nodes and friendships as edges. Edges form in one of two ways: Either you match with a probability ω with a random node/person, or you match with a friend of a friend with probability ψ. The objective of the 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 the model generated the real graph so one cannot use likelihood function methods, and 2) nodes are not iid as nodes with more edges will get relatively richer, and so method of moment methods will not work. However, SNPE neither requires a likelihood function nor iid nodes and can resolve this issue. Of course, this model is not the only model this approach can solve. As long as one can simulate graphs from the structural graph model and the structural model puts support on the true data, there is a good chance SNPE will be able to estimate.

SNPE is a simulation-based Bayesian or MLE method. SNPE works by first sampling from the prior and then sampling from the model. In this case, the model produces a network, Y, conditional on the first stage prior samples: ω and ψ. Now one would like to estimate the conditional density $P(\omega, \psi|Y)$, as conditioning Y on the real graph Y' would give the posterior. The methods section below as well as the SNPE section in Fen (2022) describe how to do this. The one problem compared to SNPE structural estimation is the conditional density estimator requires the conditioning variable, Y, to be a numerical vector. In order to convert the graph into a set of (hopefully sufficient) statistics, I use either a graph convolutional neural network or some sort of algorithmic network embedding. I define the conversion of Y to a set of statistics by the following equation: $X = G(Y)$. Now I have X, ω, ψ joint samples where X is a numerical vector and can now fit a conditional estimator, $P(\omega, \psi|X)$. Ultimately, one can derive the posterior by conditioning X on $G(Y')$, where Y' is the ground truth graph. The ultimate posterior is $P(\theta, \omega|X = G(Y'))$. This derives the posterior in the network estimation problem. If one wants to perform maximum likelihood, one can use flat priors and take the mode of the posterior.

There are many papers that have implemented structural models, but due to 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 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. If one wants to structure the structural model that generates networks in particular ways, one can evaluate the likelihood. One such example is the Exponential Random Graph Model (Robins et al., 2007). The problem with this type of model is the likelihood is defined in an ad-hoc manner, so it's both difficult to simulate from and give a structural interpretation to the parameters. This is unlike, for example, the Bramoullé et al. (2012) model, which has a clear interpretation of parameters and is easy to simulate but is difficult to estimate. My approach resolves this issue allowing structural models in the vein of Bramoullé et al. (2012) to be estimated.

The next section, Section II., discusses the SNPE simulation-based likelihood method. It also discusses network embedding methods, where the model converts a graph into statistics so that a density estimator can be conditioned on a statistic representing the graph. 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, which was not estimated in the original paper.

II. Methods

In this section, I will present an overview of a simulation-based estimation approach, Sequential Neural Posterior Estimation (SNPE). The SNPE algorithm is a Bayesian inference technique that uses a machine learning conditional density estimator to learn the posterior distribution of the parameters of interest. In addition to the SNPE algorithm, I will also discuss embedding networks. Embedding networks are used to convert graph-structured data into a set of sufficient statistics as the SNPE density estimators only work with numerical data. Two common types of embedding networks are graph convolutional neural networks (GCNN) (Kipf and Welling, 2016) and FEATHER (Rozemberczki and Sarkar, 2020), which will both be discussed in Section II.B.. For more information on normalizing flow models and the SNPE algorithm, the reader is referred to the section "Background on SNPE and MCMC" in Fen (2022).

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

The application of SNPE for simulation-based Bayesian inference has gained increasing attention in the sciences (Cranmer, Brehmer and Louppe, 2020). The SNPE algorithm has been described in detail in Fen (2022), which provides a comprehensive account of the method. This paper will only provide a brief overview of the algorithm but will provide a more detailed discussion of the novel aspects of the paper, discussing graph neural networks and other embedding methods that are used to convert graph data into numerical values.

SNPE is a simulation-based algorithm that can recover the posterior distribution of the parameters by sampling from the parameters space, $P(\theta)$, and the model, $P(Y|\theta)$, and estimating the density $P(\theta|Y)$. Here θ is a catchall parameter that, for example, in the Bramoullé et al. (2012) example, encompasses ω and ψ. The estimation of the density is done by fitting a density estimator of the conditional posterior on the joint samples (Y, θ). The density estimators will be briefly discussed below in this section but more comprehensively in Fen (2022). The resulting posterior estimate is conditioned on the real data, Y', and can be used to make predictions or inferences about the parameters of the model. The SNPE algorithm shares similarities with the approach to simulation-based MLE proposed by Kristensen and Shin (2012). Notably, because the algorithm is simulation-based, it does not require a likelihood function which is almost always intractable for structural network models. Additionally, it doesn't require nodes to be iid, which is essential for current methods of structural network estimation.

To apply SNPE in the case of graph data, it is necessary to use a GCNN or a fixed embedding algorithm to convert the network data into a set of numerical values. This is because in this case, Y and Y' are model-simulated network data and true data, respectively, and are in graph form. However, the conditional inputs in the machine learning conditional density estimator require numerical inputs. Thus, the approach needs a graph neural network to convert the network data into numerical values, X and X' respectively, for the machine learning conditional density estimator to train on. An alternative approach, the FEATHER algorithm (Rozemberczki and Sarkar, 2020) uses the characteristic function of a random walk to create numerical values. I will discuss both these graph embedding techniques and how they work in Section II.B.. Below under the label Algorithm II.A. is the SNPE algorithm, using a graph embedding technique $G(.)$, either a GCNN or fixed embedding:

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 $f_\phi(\theta|X = G(Y))\frac{p^{(i-1)}}{p(\theta)}$ on $\{Y^{(i)}, \theta^{(i)}\}_i^B$;
    end
    Update posterior $p^{(i)} = f_\phi(\theta|X = G(Y'))$;
end
```

The methodology employed in this study builds on previous research that utilized the kernel density estimator (KDE) for simulation-based inference (Kristensen and Shin, 2012). While the KDE has been shown to be a flexible and versatile method, it is not without significant limitations. In particular, the KDE struggles with conditional estimation and the ability to jointly optimize a graph neural network whose job is to convert graphs to sufficient statistics, with its own density estimation.

To address these limitations, this study utilizes advanced machine-learning techniques to estimate conditional densities in a gradient-based manner allowing us to evaluate the graph neural network in an end-to-end manner. The methods discussed in Fen (2022) offer several promising alternatives to the traditional KDE approach. One such method is the normalizing flow, which is able to handle high-dimensional data and estimate conditional densities. Another approach discussed in the paper is the use of a Generative Adversarial Network (Goodfellow et al., 2014) (GAN), which can estimate the likelihood function, but requires either rejection sampling or variational inference projection onto a flow for sampling. A third approach involves using a Conditional Mixture of Gaussians (CMoG), which is more straightforward but cannot handle higher dimensional problems. Since CMoG is the easiest approach to follow and typical structural models on graphs have fewer parameters, I will discuss this approach. The other density estimators are discussed in Fen (2022), Section III.C, and sections following it.

The CMoG approach is a powerful tool for density estimation in high-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 is a linear combination of N Gaussian distributions, each with its own mean μ_i, covariance Σ_i, and weight pi_i. The probability density function (pdf) of a point 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 theta 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:

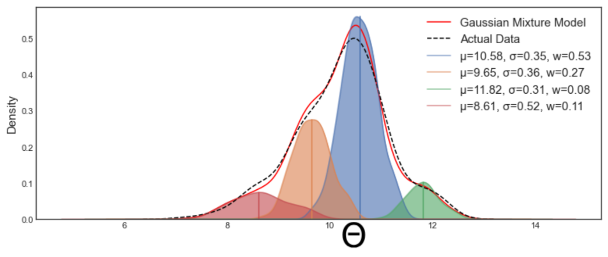


Figure I: Mixture of Gaussians

To obtain the conditional density estimator, a neural network is used to model the conditional relationship between the parameters θ and the data X. The neural network takes X as input and returns the parameters for the CMoG, including the weights, means, and covariances for each Gaussian component. These parameters can then be used to estimate the conditional posterior density $P(\theta|X)$ using the equation provided in the previous paragraph:

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

The use of a neural network to estimate conditional density also provides flexibility and adaptability to the density estimator. Changes in the input data X can result in changes to the CMoG, allowing for the model to adapt to changes in the data. Additionally, the parameters of the neural network can be optimized via MLE to improve the accuracy of the density estimator. This approach allows for the estimation of accurate conditional posteriors, even in relatively high-dimensional spaces, and can be a useful tool for simulation-based inference.

One can also optimize the likelihood of the CMoG, $P(\theta|X)$, in conjunction with a GCNN. In this case $\pi_i(.)$, $\mu_i(.)$, and $\Sigma_i(.)$, would all be graph neural networks and they convert graphs Y, into parameters π_i, μ_i, and Σ_i:

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

While I've discussed how SNPE works from a high level, I've abstracted from how to build a machine-learning model that can convert graphs into statistics. Discussing GCNNs and fixed graph embeddings will be discussed in the next session.

II.B. Graph Convolution Neural Networks

In order to use graphs as inputs for density estimation, they must first be converted into numerical values which hopefully are sufficient statistics. This is typically achieved using graph neural networks or fixed graph embeddings. One 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:

$$x_j = \sigma(A_i x_i + B_i) \tag{1}$$

Here, x_i and x_j represent the input and output of the layer, both of which are vectors, respectively. The matrix A_i maps the input, x_i, to the output, x_j, and B_i is an intercept term. 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 and has been shown to work well in many applications (Agarap, 2018).

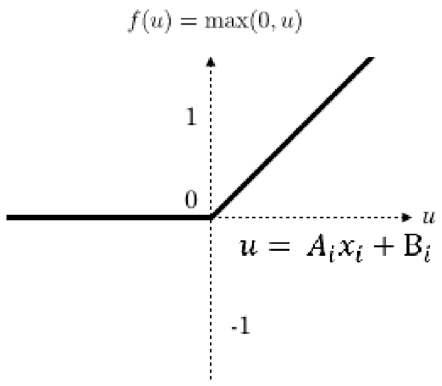


FIGURE II: RELU ACTIVATION FUNCTION

The idea behind a Feed-Forward Neural Network is to compose many copies of equation (1), one on top of another. Thus given an input x_i, the first equation produces x_j, then x_j is an input to the next layer which is also a vectored-valued equation: $x_k = \sigma(A_j x_j + B_j)$, with different parameter values for A and B. One can continue composing these functions one on top of another. Additionally, one can increase the dimensionality of the intermediate vector x_j, as its elements aren't constrained by the dimension of the input or output. Both methods increase the parameter count and allow the neural network to be more flexible.

While a Feed-Forward Neural Network can be used to process a variety of inputs, including graphs, it does not take into account the inherent structure of the input. Graphs have a natural structure that can be exploited to improve the accuracy and information conveyed from a statistic generated from a function that takes in a graph and outputs a numerical set of statistics corresponding to the graph. This is where Graph/network Convolutional Neural Networks (GCNNs) come in. GCNNs are a type of neural network designed specifically to work with graph-structured data. They use a form of a convolution operation, similar to that used in image processing, to extract features from the graph. This allows the network to take into account the topology of the graph, as well as the features of individual nodes and edges when making predictions (Wu et al., 2020).

GCNNs (Kipf and Welling, 2016) have shown great potential in working with graph data, and their architecture is specifically 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 usually normalized by dividing by the degree matrix, and the resulting normalized matrix is denoted by A^*. In GCNNs, the input of a given layer, X_i, which can contain a network of node characteristics, is fed into a Graph/network Convolutional Layer. This layer is defined as $X_j = \sigma(W_i X_i A^* + B_i)$, where W_i and B_i are learnable parameter matrices (vectors), and σ is a nonlinearity. Here X_j is the output node characteristics, which results from applying the GCNN layer to X_i. By using the adjacency matrix, GCNNs extend the feed-forward neural network architecture to enable the characteristics of nearby nodes to affect each other more strongly than those of far away nodes. This inductive bias allows the model to efficiently handle data that is defined on networks. For instance, a randomly initialized graph neural network can nearly learn how to cluster network data properly, as demonstrated in the Karate Club dataset, which is classified by color using a modularity-based algorithm (Brandes et al., 2007). These colors represent clusters of nodes that share a community together, due to sharing an abnormally high number of edges within the cluster. Figure III shows the Karate Club data set with nodes colored according to their cluster:

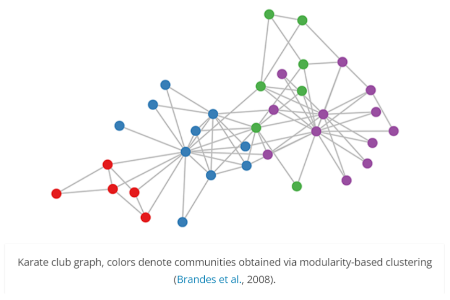
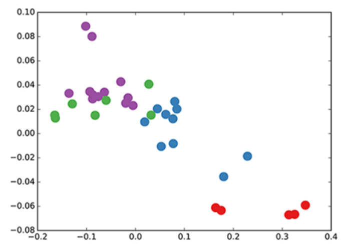

Karate club graph, colors denote communities obtained via modularity-based clustering (Brandes et al., 2008).

Figure III: Karate Club Network Picture

The randomly initialized GCNN then produces a 2D embedding and without training, nodes belonging to the same cluster are given embedding values close to one another. Below, Figure IV shows the same cluster colors produced by a randomly initialized graph neural network:



GCN embedding (with random weights) for nodes in the karate club network.

FIGURE IV: THE KARATE CLUB NODES IN EMBEDDING SPACE

With training, one can learn even more effective embeddings that can take a graph and produce numerical values for the simulation-based inference approach. Given a Conditonal Mixture of Gaussians trained to estimate $P(\theta|X)$ and a graph Y, the GCNN,, $G()$, takes Y and returns a numerical embedding X that can be fed into the density estimator to properly condition on the graph. The GCNN is trained end-to-end with the density estimator and so should be able to find the best possible statistics that maximize the fit of the density estimator.

II.C. FEATHER Network Embedding

An alternative is to use an out-of-the-box graph embedding technique without learning parameters. In this paper, I use the FEATHER algorithm (Rozemberczki and Sarkar, 2020) as the graph embedding technique. FEATHER learns node embeddings from the characteristic function of the probability density function (pdf) implied by a random walk from a node, and then averages the node embeddings to obtain the graph embedding. This is achieved by learning the function that maps the node-level features – which in our case are the default features of eccentricity, transitivity, and degree – 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. As my graphs have no node features, the features 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. In order 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, there are many other approaches that 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 (Rozemberczki, Kiss and Sarkar, 2020). The reason for using a fixed embedding is that, unless one is dealing with sparse graphs, the amount of memory required to store a graph is quadratic in the number of nodes. As our method, Sequential Neural Posterior Estimation (SNPE), often requires a large number of graphs for estimation, memory constraints can quickly become an issue. However, with an embedding method that does not require any modifiable weights, one can derive the embedding and then throw away the graph, saving only the embedding for future iterations. This allows one to work with much larger graphs without running into memory constraints.

II.D. Limitations of Network-based Embedding Methods

This section discusses the main weakness of GCNNs, and related techniques like graph transformers, and graph embedding methods. One main result discussed in Fen (2022) concerns universal approximator properties of Normalizing Flows and Mixture of Gaussians. The graph SNPE algorithm does not have this quality. Likewise, large social networks are also beyond the computational capacity without a large amount of memory, although with fixed embeddings, work on them is more possible.

The results in Fen (2022) and borrowed from Huang et al. (2018), Goodfellow et al. (2014), and Bishop (1994) demonstrate that the normalizing flow, GAN, and a Mixture of Gaussians in the SNPE algorithm are universal approximator of probability distributions and that any large enough density estimator of these forms can model any continuous probability distribution. However, this result no longer holds with graph data. The reason is that GCNNs, their cousins graph attention transformers (Veličković et al., 2017), and related embedding methods, can not 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 that has 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 when using most network-based embedding models–be they GCNNs, or fixed embedding algorithms. These approaches are specifically tailored toward graphs and give much better performance than other methods.

III. Results

This section presents empirical results that demonstrate the efficacy of the proposed algorithm in recovering the parameters of a structural model from the graph generated by those parameters. To this end, I begin by discussing the Newman-Watts-Strogatz graph (Newman and Watts, 1999), a widely-used network model. The algorithm is then tested on other network models, namely the Power-Law Cluster (Holme and Kim, 2002) and Relaxed Caveman graphs (Fortunato, 2010). Finally, I perform an empirical exercise estimating the Bramoullé et al. (2012) structural model on the Karate Club network.

To evaluate the algorithm's performance, a simulated network is drawn from the structural model using a specific set of parameters, and the structural model parameters are then estimated based on the given network.

The results demonstrate the algorithm's effectiveness in recovering the structural model parameters from the generated graph for all three types of network models considered. The posterior estimates closely match the true parameter values, as indicated by the convergence of the blue lines to the red lines in the charts. The heat maps further confirm the accuracy of the parameter estimation, with high probability density concentrated around the true parameter values. These findings provide evidence that the proposed algorithm can be applied to a wide range of network models and can effectively recover the underlying structural parameters, which is crucial for accurately characterizing and understanding real-world networks.

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). This 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 terminating node, u, of that edge, u, v, will form a new edge with a random node, w. To embed the generated graph, a graph convolutional neural network is employed.

The results are presented in the charts below, where the blue line corresponds to the posterior of the estimation routine, and the red line represents the true calibrated parameter value. The off-diagonal charts show a heat map of the two-way distribution implied by the parameter that defines the row and the column that intersects on a given chart. In cases where the parameters are integer-valued, a continuous uniform distribution was still used for sampling, but the floor function was applied to generate an integer. As the number of nodes is known in the dataset, the model generates a network with the same number of nodes as the dataset.

In my analysis, the value of k is fixed at 12, and p is set to 0.33. The results indicate that the proposed algorithm successfully recovers a posterior distribution whose mode is quite 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.

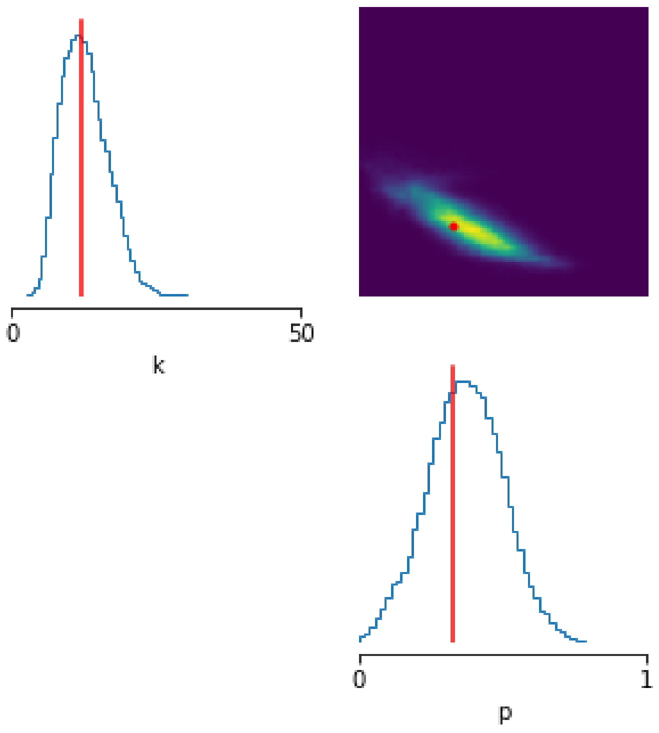


FIGURE V: SNPE RESULTS OF ESTIMATING THE NEWMAN-WATTS-STROGATZ SMALL WORLD NETWORK

III.B. Power-Law Cluster Graph

In this section, I will focus on the estimation of the Power-Law cluster graph on simulated data, as proposed by Holme and Kim (2002). The model has two parameters: the number of random edges to add to each node where edges are added preferentially to vertices with higher degrees. The second parameter is the probability of closing a triangle with a randomly chosen neighbor after an edge is added. The random edge parameter must be an integer; therefore, uniform random variables are generated and then rounded to the nearest integer. 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. The results indicate good identification of the edge parameter, 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). To embed the generated graph, a graph convolutional neural network is employed. Overall, the results suggest that the proposed algorithm can successfully estimate the edge parameter in the power law cluster graph, despite challenges in identifying the triangle parameter.

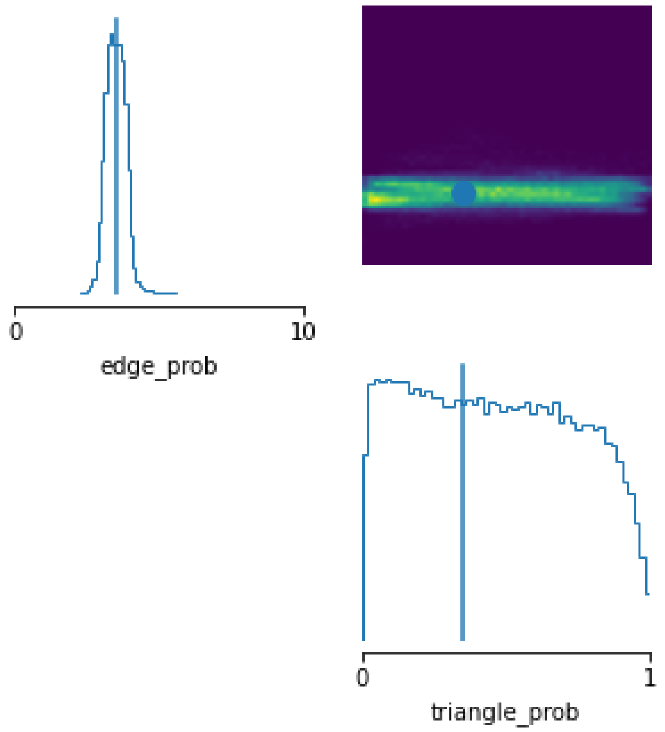


Figure VI: Power Law Cluster Graph

This chart shows the model accurately recovering the random edge probability of 3, However, there is difficulty recovering the triangle probability as mentioned above.

III.C. Relaxed Caveman Graph

In this section, I turn my attention to the Relaxed Caveman Graph proposed by Fortunato (2010), which is formed by randomly rewiring an edge formed by a clique in a caveman graph with probability p. To generate the Relaxed Caveman graph, n cliques of size m are formed, and then one edge in each complete graph clique is deleted to form another edge that connects all the cliques in a cycle. 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 more often. In contrast to the graph convolutional neural network used in previous sections, I use the FEATHER graph embedding proposed by Rozemberczki and Sarkar (2020) to embed the Relaxed Caveman graph.

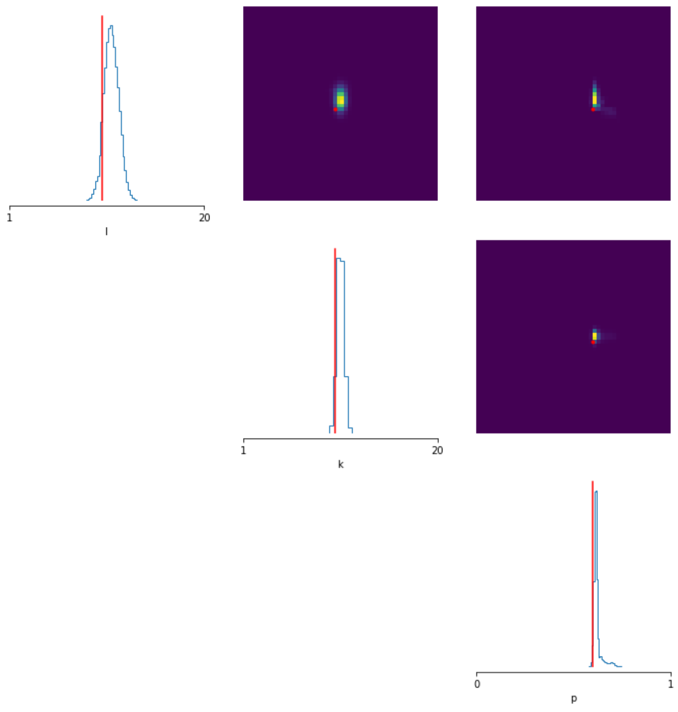


FIGURE VII: RELAXED CAVEMAN GRAPH

The posterior distribution for the Relaxed Caveman graph shows good convergence of the algorithm, with the true parameters being very close to the mode of the posterior distribution. However, there is a slight shift in the mode of the posterior distribution from the true value. This is a mistake that can often be made using SNPE. Although Metropolis-Hastings Markov Chain Monte Carlo (MCMC) often has poor mixing problems, SNPE algorithms rely on sampling and density estimation and errors sometimes involve the entire distribution being shifted slightly from the correct location. 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 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 is used to extend the results of the original paper by structurally estimating the model, which was not done in the original study.

The paper by Bramoullé et al. (2012) did not estimate their model on real data. Instead, they use citations and citations of cited papers to come up with statistics that approximate what the true underlying parameter value is. 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 of friends in the Karate Club data set since the number of friends one has does not have a temporal dimension. Additionally, the original Bramoullé et al. (2012) model, seems to assign almost no support to a wide variety of relatively reasonable graphs – for example common friendship, supplier/customer, or citation graphs. Therefore, I modify the Bramoullé et al. (2012) model to be more applicable to this scenario. In the modified model, all nodes are initialized at the beginning and there is a probability that each node has a direct connection with another node at the beginning of each round. Additionally, there is a separate probability that a node is connected to the connection of a connection. Since temporal relationships have been destroyed, this is an undirected network with the node representing a citation from the earlier paper to the later paper. There are 15 rounds in total which is a reasonable amount for the 34-node Karate Club dataset.

The original Bramoullé et al. (2012) paper did not provide an estimation procedure for their approach, so they used statistics from the network discussed above to approximate the random connection and homophily connection rate. However, if the data was generated by this model, this approach could overcount search citations, if there are some citations that are randomly created even though there is an intermediary that is cited by the citing paper and cites the cited paper. On the other hand, if the model is misspecified, it could also undercount search citations if someone cites the paper knowing an intermediary paper but does not cite the intermediary paper.

IV.A. Karate Club Friendships Graph

This paper presents an analysis of a homophily model applied to the Karate Club dataset, a dataset of friendship connections among Karate Club participants. The Bramoullé et al. (2012) model assumes that individuals make direct friends and then make friends from their friends' friends. To test the validity of the model, I estimate it on the empirical data and then generate simulated data using the model's estimated parameters. I then estimate the model on the simulated data to check for convergence to the true solution.

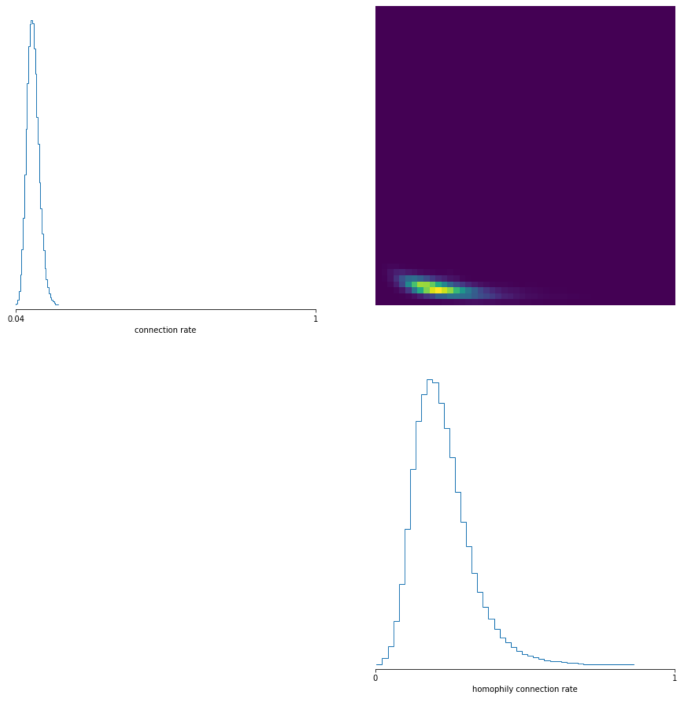


FIGURE VIII: SNPE ESTIMATES HOMOPHILY MODEL ON KARATE CLUB DATA

I estimate the model on the Karate Club dataset and obtain the posterior distribution shown in the above Figure VIII. This figure displays the homophily model estimated on the Karate Club data. When estimated on real data, Figure VIII shows a posterior that concentrates around the true parameter value for both parameters. The mode for the marginal of the first parameter, random connection rate, is around .08. The mode for the marginal of the second parameter, homophily connection rate, is around .23. I then estimate the model on the simulated data. To recover the true parameters on the simulated data, I set the random connection rate to .08 and the homophily search connection rate to .23, which were the approximate means of the empirical posterior. The results of this estimation are shown in the figure below:

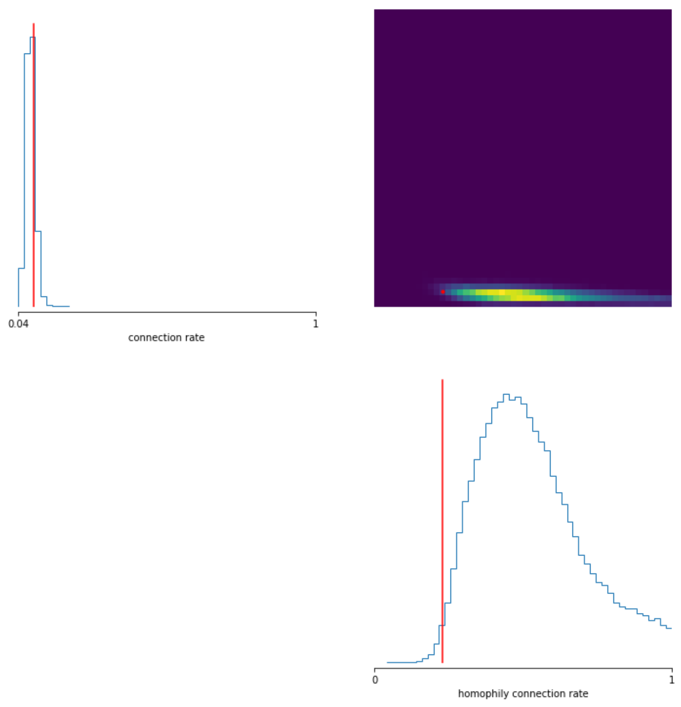


Figure IX: SNPE Estimates Homophily Model on simulated data

Testing the model on simulated data, it is evident from Figure VIII, that the model does a good job of identifying the random connection parameters. However, the model performs poorly in identifying the homophily search parameter. I attribute this to the fact that the simulated data, in line with the Karate Club data, has only 34 nodes, and there will be some variance in the simulated data, and a connection of a connection will be more difficult to extract from data than direct connections. Nevertheless, the true parameter is still within the support of the posterior for the homophily search parameter. These findings suggest that the homophily model provides a reasonable explanation of the observed friendship connections in the Karate Club dataset, as well as provides insights into the formation of social networks more generally.

V. Conclusion

In conclusion, this paper has introduced 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. 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. Nevertheless, there are many models that have not been estimated in the literature that can be estimated in this way with some modifications to 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.*

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

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.

Brandes, Ulrik, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. 2007. "On modularity clustering." *IEEE transactions on knowledge and data engineering*, 20(2): 172–188.

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.

Cranmer, Kyle, Johann Brehmer, and Gilles Louppe. 2020. "The frontier of simulation-based inference." *Proceedings of the National Academy of Sciences*, 117(48): 30055–30062.

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."

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

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.

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.

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.*

Department of Economics, University of Michigan, Ann Arbor