uc3m | Universidad Carlos III de Madrid

Master Degree in Big Data Analytics

2024-2025

Master Thesis

"Community Detection from ARD Data via Graph Estimation and Graph Neural Networks"

Ángela Caniego González

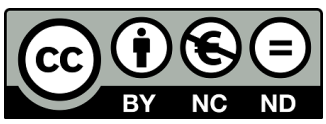
Rosa Elvira Lillo Rodríguez

Jose Aguilar Castro

Madrid, June 2025

AVOID PLAGIARISM

The University uses the **Turnitin Feedback Studio** program within the Aula Global for the delivery of student work. This program compares the originality of the work delivered by each student with millions of electronic resources and detects those parts of the text that are copied and pasted. Plagiarizing in a TFM is considered a **Serious Misconduct**, and may result in permanent expulsion from the University.



SUMMARY

Keywords:

DEDICATION

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

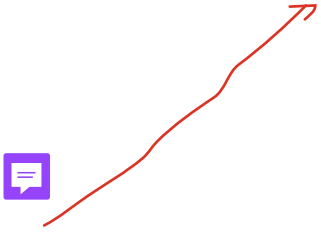
Complex network analysis has become a fundamental tool for understanding social, biological and technological phenomena. However, in many real-world situations, obtaining complete data on the connections between individuals or entities is not possible. This may be due to privacy constraints, logistical limitations, or even the lack of mechanisms to record interactions directly. In response to these difficulties, alternative methods have emerged that allow the structure of a network to be inferred from aggregate or indirect data. One of the most promising approaches is the use of Aggregated Relational Data (ARD), which allows the estimation of network and community properties from partial and summarised information. This document proposes a two-step methodology: first, the estimation of graphs from ARD; and second, the detection of hidden communities using Graph Neural Networks (GNNs), which learn representations of nodes useful for further structural analysis.

1.1. Motivation

The difficulty of collecting complete social network or contact data has motivated the development of alternative methods that can infer the underlying structure from indirect observations. Indirect surveys, where participants are asked how many people they know who belong to a certain group, allow ~~to infer~~ properties of the social graph without requiring direct observation, using statistical and probabilistic methods. These techniques are especially valuable in contexts where privacy, ~~speed~~ or limited resources prevent exhaustive surveys or network censuses from being carried out.

A prominent example of the usefulness of this approach is the CoronaSurveys project, developed during the COVID-19 pandemic. This ~~system~~ used indirect online surveys to estimate the number of active cases, new infections and deaths in different countries. Instead of asking individuals about their own health status, they were asked how many people in their environment they knew who were affected by the virus. This methodology allowed for robust and rapid estimates with minimal logistical cost and was particularly useful in ~~times~~ of scarce official data. The success of CoronaSurveys demonstrates the power of indirect ARD-based methods to approximate real network phenomena.

This project continues with that philosophy: taking advantage of the potential of aggregated data to reconstruct network structures and analyse emerging phenomena, such as the existence of hidden communities. ~~Instead of focusing solely on counting cases~~, ~~here~~ we study a methodology to estimate the entire network of relationships between individuals from ARD data and, on that estimated network, apply advanced deep learning techniques to discover latent structural clusters.



Moreover, with the increasing availability of machine learning techniques and neural networks specialised in graphs, such as Graph Neural Networks (GNNs), it is possible to extract high-level information from complex structures. GNNs allow learning vector embeddings of nodes that capture local and global relationships, and facilitate tasks such as link prediction, node classification or, as in this case, community detection.

1.2. Objective

The main objective of this work is to design, implement and evaluate a two-phase pipeline to detect communities in social networks estimated from ARD data. For this purpose:

1. In a first stage, the Bayesian model proposed by McCormick et al. (2010) is used to infer the existence of links between nodes from ARD-type responses. This model generates multiple sample graphs representing the posterior distribution over possible estimation networks.

2. In a second stage, the estimated graphs and a GNN, particularly the Deep Graph Infomax (DGI), is applied to obtain embeddings of nodes. From these vectors, clustering algorithms are applied to identify latent communities.

This pipeline offers a hybrid solution that combines classical statistical models with modern deep learning techniques, making use of the strength of both approaches.

Complementarily, quality metrics and visual techniques are also explored to assess both the accuracy of the estimated graphs compared to the true network and the consistency of the detected communities, also comparing these communities to real communities identified in data. In addition, this document analyses the robustness of the proposed methodology to variations in the simulation parameters, and compares the results with the simulated network structure.

Specifically, This approach integrates knowledge from Bayesian statistics, data simulation, graph theory and deep learning, and aims to demonstrate that, even with limited or indirect data, it is possible to reconstruct and analyse complex social structures.

1.3. Overview of the Project

This document is structured in four main chapters, in addition to this introduction.

- In **Chapter 2**, the theoretical and methodological framework of network estimation from ARD is presented. It discusses the assumptions behind the McCormick et al. model, details the synthetic data simulation process, and analyses the performance of the model under different parameters. Node and network level metrics are included, as well as visualisations.

- In **Chapter 3**, Graph Neural Networks are introduced, their fundamentals, applications and, in particular, the Deep Graph Infomax (DGI) approach used for obtaining node embeddings. It explains how GNNs are trained on each estimated graph, how embeddings are obtained, and what evaluation strategies are used.
- In **Chapter 4**, the entire pipeline is applied to a specific case study. Since real ARD data are scarce, a real network with metadata is used from which ARD responses are simulated. This allows comparing the estimated network with the original network and validating the quality of the detected communities. Quantitative and visual analyses are included.
- Finally, the limitations of the approach, possible improvements and future lines of research are discussed, both from a methodological and applied point of view.

Through this research, we seek to contribute to the study of networks in contexts where direct data collection is not possible, proposing a solid analytical tool that could be extended to different disciplines, such as sociology, epidemiology or political science.

2. GRAPH ESTIMATION FROM ARD

2.1. ARD and Graphs

In many disciplines the understanding of the structure of relationships between individuals is essential. A network or graph is a mathematical representation of these relationships, in which elements (vertices or nodes) are connected by links (edges) [1]. These structures allow the modeling of social interactions, information flows, disease transmission, and many other complex dynamics. Formally, a graph $G = (V, E)$ consists of a set of nodes V and a set of edges $E \subseteq V \times V$. If the links have a direction, for example in communication or financial transaction networks, the graph is said to be directed; otherwise, it is said to be undirected. ~~Throughout~~ this paper, we will focus exclusively on undirected graphs. A common way to represent a graph is by its adjacency matrix A, where each element A_{ij} indicates the presence ($A_{ij} = 1$) or absence ($A_{ij} = 0$) of a connection between nodes i and j.

In practice, collecting complete data about real social networks is usually hard, costly or invasive. ~~Faced with~~ this limitation, an alternative is the use of ~~Aggregated Relational Data (~~ARD~~)~~ [2]. This type of data is based on indirect surveys where participants are asked how many people or entities they know that belong to a specific group (for example, "how many female doctors do you know?"), without asking them to identify each person. These subgroups are normally defined by observable characteristics like profession, gender, religion, location, etc. Survey answers can be cross-referenced with official data about the size of these subgroups (census or national surveys) to infer connexion patterns and underlying networks.

Data in ARD form has shown to be useful in many social and health contexts. For instance, it has been used to estimate the size of hidden communities in the HIV context in countries like Singapore and Rwuanda [3], where the number of people belonging to risk groups was estimated (drug users, sex workers, etc.), and also social isolation was studied, finding that individuals with HIV had smaller networks than average [2]. In development economics [4], it was studied to what extent ARD answers could replace the costly collection of complete networks. They used as a case study the well-known work of Banerjee et al. (2016) [5], which collects detailed social network information from nearly 16,500 households in 75 rural villages in Karnataka, India. Authors showed that, even collecting ARD only from a 30% sample of households, it was possible to estimate with remarkable precision key network characteristics. Furthermore, highlighting that the ARD approach ~~could have~~ reduced collecting costs up to a 70-80%, without losing relevant information for the analysis. More recently, during the COVID-19 pandemic, the project CoronaSurveys used ARD indirect surveys to estimate the number of infections in several countries [6], demonstrating that this methodology could provide reliable estimations even in the

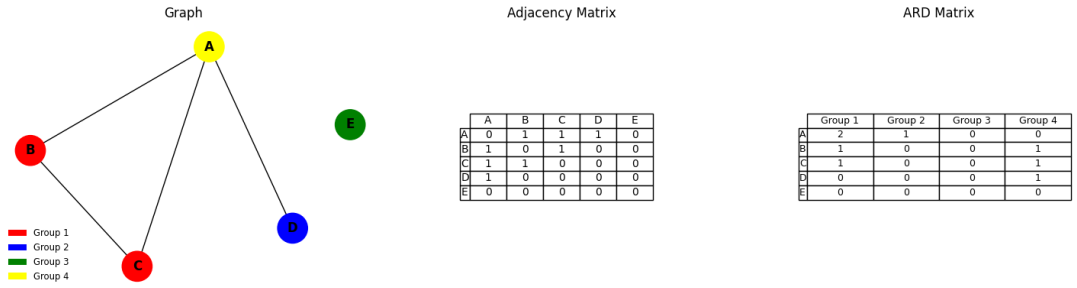
absence of mass diagnostic tests or robust surveillance systems.

From a technical point of view, ARD answers can be modeled ~~as functions~~ from a graph represented by its adjacency matrix A. Let each node i, out of N nodes, define a set of its neighbors where each neighbor belongs to a subgroup $k \in \{1, \ldots, K\}$. Then, for each node i, the counting vector $\mathbf{y}_i = (y_{i1}, \ldots, y_{iK})$ can be built, where y_{ik} represents the number of neighbors of i that belong to group k. The set of vectors $\{\mathbf{y}_i\}_{i=1}^N$ constitutes the set of ARD observations. Mathematically, these ARD answers can be interpreted as the aggregation of the adjacency matrix weighted with information about nodes:

$$y_{ik} = \sum_{j=1}^{N} A_{ij} \cdot \mathbb{1}\{w_j = k\}, \tag{2.1}$$

where w_j is the group of node j, N the number of nodes and $\mathbb{1}$ is the indicator function. The goal is to reverse the process, given the ARD matrix with rows $\{\mathbf{y}_i\}_{i=1}^N$ and some information about the size of groups, estimate the complete network given by the adjacency matrix A.

As an illustrative example, let's consider a small graph defined by five nodes where each node belongs to a subgroup represented by a color. We can build the adjacency matrix A where A_{ij} indicates whether there is a conexion or not between nodes i and j. Then, we can build the ARD matrix Y where each row represents a node and each column represents a subgroup, and Y_{ik} represents the number of neighbors of node i that belong to subgroup k. In Figure 2.1 the visualization of this example of a five-node graph is shown. To the left, the graph where colors represent the different subgroups; in the center, the adjacency matrix A; and to the right, the ARD matrix.



	A	B	C	D	E
A	0	1	1	1	0
B	1	0	1	0	0
C	1	1	0	0	0
D	1	0	0	0	0
E	0	0	0	0	0

	Group 1	Group 2	Group 3	Group 4
A	2	1	0	0
B	1	0	0	1
C	1	0	0	1
D	0	0	0	1
E	0	0	0	0

Fig. 2.1. Example of five-node graph, adjacency matrix and ARD matrix.

2.2. Network Estimation Model

This section describes the latent surface model proposed by McCormick and Zheng (2015) [2], which allows estimating social networks from ~~aggregated relational data~~ (ARD). This approach is particularly useful in contexts where complete observations on the connections between individuals are not available, but indirect information is available on how many people ~~each~~ individual knows within different population subgroups. From these summary observations, the model allows inferences to be made about both individual characteristics and the overall structure of the underlying network.

The central idea is to represent the latent network as a model of network formation where the nodes are positioned in a latent social geometry, particularly on the surface of a hypersphere of dimension p. In this space, the probability of a connection between two nodes depends on their individual connection propensity and their positions in the latent social space, the probability of connection is inversely proportional to their geodesic distance: the smaller the distance, the greater the probability of a link [4]. This geometric structure allows ~~us to capture~~ patterns of homophily (similar nodes tend to link) and transitivity (tendency of nodes to cluster) typical of real networks, while being a computationally efficient method. Starting from this latent space representation, we then define a generative model of network formation and explicitly link it to the ARD responses, allowing ~~us to estimate~~ the model parameters directly from ARD. Once these parameters are inferred, it is possible to sample entire networks that reproduce structural properties consistent with the original (but unobserved) network from which the ARD was derived.

One of the main advantages of this approach is that inference does not require observing any edge of the network directly. Instead, it starts from a random sample of nodes for which ARD responses are collected, respondents count of how many people they know who belong to different subgroups. These observations, combined with a parametric model of link formation (known as the latent surface model) allow ~~us to derive~~ a likelihood function based solely on the ARD responses. From this likelihood, the latent parameters of the model can be estimated: individual gregariousness effects and locations in the latent space. These parameters, together with the estimated degrees per node and an estimation for the intensity of the latent component, allow ~~us to compute~~ pairwise connection probabilities.

The Bayesian framework facilitates this inference, as it allows full posterior distributions to be obtained on the model parameters. Simulations conducted by ~~the authors~~ [4] show that the choice of priors has limited impact on the quality of the estimates, provided that reasonable structures are used. In addition, this approach offers the possibility of incorporating auxiliary information (like demographic covariates) to extend the estimation to nodes for which no ARD information is available.

Although the model is designed to work in scenarios where only a sample of nodes with ARD data is available, in this document, we focus on the case where ARD data is

available for the entire population. This allows us to evaluate the performance of the model under controlled conditions and to maximize the information available in the network inference. However, in more realistic scenarios with partial ARD survey samples, it is possible to extend the estimates to the rest of the population using observable covariate information (as long as this information covers the whole population) and machine learning methods. In particular, [4] proposes the use of k-nearest neighbors (k-NN) algorithms to impute both individual effects and latent locations of nodes without ARD, based on their similarity to observed nodes in the covariate space. This step allows reconstructing the entire network even when survey information is limited to a sample of the individuals.

Overall, this latent surface model represents a solution to the problem of inferring networks from aggregated data. By explicitly modeling the structure of homophily and individual variability in the propensity to form links, and by integrating available sociodemographic information, it allows ~~to generate~~ not only a single network, but a posterior distribution over possible networks compatible with the observed data. This property is particularly valuable for assessing uncertainty and for further analysis, such as link prediction or community detection.

From now on, we describe the proposed model in more detail, including the main assumptions, the mathematical formulation and the estimation algorithm following the approach developed in [4].

General Setup and Assumptions

Consider an undirected, unweighted graph $G = (V, E)$, where V is the set of nodes, with $|V| = n$, and $E \subset V \times V$ the set of edges. The associated adjacency matrix is $A = (a_{ij})$, where $a_{ij} = 1$ if there is an edge between nodes i and j, and $a_{ij} = 0$ otherwise. We also denote $g_{ij} = \mathbb{1}$ for every $(i, j) \in E$ to express the existence of a link.

Unlike traditional approaches, in this model we do not directly observe a_{ij} connections, but only ~~aggregate relational data~~ (ARD) for all nodes in the network. That is, for each node $i \in V$, a vector of counts $\mathbf{y}_i = (y_{i1}, \ldots, y_{iK})$ is observed, where each component y_{ik} represents the number of people known to i who belong to a subgroup $G_k \subset V$. Each group G_k is defined as the set of nodes that share a common observable characteristic.

Formally, each ARD response can be expressed as:

$$y_{ik} = \sum_{j \in G_k} g_{ij}, \tag{2.2}$$

which is equivalent to counting the number of neighbors of node i belonging to group G_k. Notice that the specific nodes connected to i are not observed, only the total number of connections to that group.

Moreover, it is assumed that for each node $i \in V$ a vector of F observable characteris-

tics $X_i \in \mathbb{R}^F$ is available, with information about age, gender, or other sociodemographic covariates. These covariates can be used as auxiliary predictors in the case where ARD data is not available for the entire population, although in this document they are not required because ARD data is available for the entire population, so X will not be used until the next chapter: community detection.

Latent Surface Model

The central idea of the model is to assume that each node $i \in V$ occupies a latent position, $z_i \in \mathbb{S}^p$, on the surface of a hypersphere $\mathbb{S}^p \subset \mathbb{R}^{p+1}$. These positions encode latent social relationships: nodes closer to each other in this space are more likely to be connected, which captures structural homophily in social networks.

The probability of the existence of an edge between nodes i and j is given by a log-linear exponential model of the form:

$$\mathbb{P}(g_{ij} = 1 \mid \nu_i, \nu_j, \zeta, z_i, z_j) \propto \exp\left(\nu_i + \nu_j + \zeta \cdot z_i^\top z_j\right), \tag{2.3}$$

where:

- $\nu_i \in \mathbb{R}$ is a random individual efect that represents gregariouness or tendency of node i to form links,
- $\zeta > 0$ is a global parameter that measures the intensity of the spatial latent component (the importance given to the latent positions),
- $z_i^\top z_j$ corresponds to the cosine of the angle between the latent positions i and j, acting as a similarity measure.

This model belongs to the family of latent geometry models and, by employing a metric based on the angular distance on the sphere, it respects the triangular inequality, allowing to represent transitivity phenomena: if i is close to j and j to k, then i and k will also tend to be close. This property adequately reflects typical structures of real social networks, such as the formation of communities.

A Bayesian framework is adopted where prior distributions are assigned to the different parameters and components of the model. Starting with the latent space, the latent positions $z_i \in \mathbb{S}^p$ are modeled on the surface of the hypersphere using a von Mises–Fisher (vMF) distribution. This distribution is analogous to a Gaussian multivariate but defined in spherical spaces [7], it has two main parameters: mean direction $\mu \in \mathbb{S}^p$, which defines the center of concentration on the sphere; and a concentration parameter $\eta \geq 0$, which controls how grouped together the samples are around the mean direction. When $\eta = 0$, the distribution is uniformly random on the surface of the sphere, and as the concentration η increases, the distribution becomes more condensed around μ. In the model, it is assumed that:

$$
\begin{aligned}
z_i &\sim \mathcal{M}(\mu_z, 0) \\
z_{j \in G_k} &\sim \mathcal{M}(\mu_k, \eta_k),
\end{aligned}
\tag{2.4}
$$

meaning that individual latent positions z_i are randomly distributed on the sphere, which is consistent with the assumption that sampled ARD individuals are randomly drawn from the population. On the other hand, nodes belonging to a specific group G_k defined as $z_{j \in G_k}$ are modeled with more structured vMF distribution, where μ_k represents the latent center of the group k and η_k is the concentration around which members of k group around μ_k.

Theses parameters are estimated through ARD observations, allowing for inference on the latent structure among groups. The ARD responses allow us to infer the relative positions of the subgroups in the latent space, determining implicit distances between them. This structure reflects that some groups are socially closer to each other, while others appear more dispersed. These distances directly influence the probability of connection between individuals from different groups. Figure 2.2 graphically illustrates how three different vMF distributions distinguished by color concentrate the probability mass around a mean direction μ_k on the sphere (represented as larger points). It shows how the density varies for different values of η_k through random samples drawn, the lower the parameter η_k, the more disperse distribution. This is how the model assumes that the K groups of features or characteristics are distributed on the sphere.

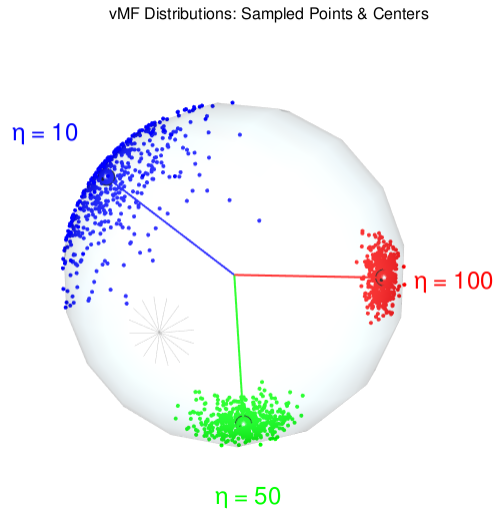


Fig. 2.2. Example of three vMF distributions.

Following the model, the expected ARD response by individual i for group k is expressed as:

$$
\lambda_{ik} = \mathbb{E}[y_{ik}] = d_i \cdot b_k \cdot \left(\frac{C_{p+1}(\zeta) C_{p+1}(\eta_k)}{C_{p+1}(0) C_{p+1}\left(\sqrt{\zeta^2 + \eta_k^2 + 2\zeta\eta_k \cos(\theta_{ik})} \right)} \right),
\tag{2.5}
$$

where:

- d_i is the expected degree of node i,
- b_k is the share of links made with group k,
- θ_{ik} is the angle between z_i and the latent center μ_k of group k,
- $C_p(\cdot)$ is a normalization constant of the vMF distribution in dimension $p+1$ [8].

The individual gregariousness parameter ν_i can be estimated from the expected degrees d_i. Following [2], the expected degree of a node i as a function of ν_i and the rest of the parameters can be expressed as:

$$d_i = n \cdot \exp(\nu_i) \cdot \mathbb{E}[\exp(\nu_j)] \cdot \frac{C_{p+1}(0)}{C_{p+1}(\zeta)}. \tag{2.6}$$

This expression allows us to clear ν_i, thus being able to estimate the latent gregariousness of each individual without directly observing its number of connections and taking advantage of the structure of the Bayesian model.

The rest of components of the model are completed specifying a priori distributions for the remaining parameters, including:

- Log-Normal priors for degrees d_i, shares b_k y concentrations η_k:

$$\log d_i \sim \mathcal{N}(\mu_d, \sigma_d^2), \quad \log b_k \sim \mathcal{N}(\mu_b, \sigma_b^2), \quad \log \eta_k \sim \mathcal{N}(\mu_\eta, \sigma_\eta^2) \tag{2.7}$$

- Gamma prior for the global parameter ζ:

$$\zeta \sim \text{Gamma}(\gamma_\zeta, \psi_\zeta) \tag{2.8}$$

To develop the likelihood function of the model conditional to the latent variables, it is crucial to take into account the assumption that probabilities of creating links between pairs of nodes are conditionally independent given the latent variables. Under this, each ARD response component y_{ik} is modeled as the sum of independent Bernoulli trials, one for each node $j \in G_k$. To make the model computationally tractable, this Bernoullis sum is approximated by a Poisson distribution:

$$y_{ik} \mid d_i, b_k, \zeta, \eta_k, \theta_{ik} \sim \text{Poisson}(\lambda_{ik}), \tag{2.9}$$

where λ_{ik} (2.5) is the expected number of ties of i to group k.

Now, the complete posterior distribution of the model is proportional to the product between the likelihood function and the priors, let $\boldsymbol{\theta}$ denote the set of latent parameters:

$$\boldsymbol{\theta} \mid y_{ik} \propto \prod_{i=1}^{n}\prod_{k=1}^{K} \exp(-\lambda_{ik})\lambda_{ik}^{y_{ik}} \cdot \prod_{i=1}^{n} \cdot \text{Normal}(\log(d_i) \mid \mu_d, \sigma_d^2) \\ \times \prod_{k=1}^{K} \text{Normal}(\log(b_k) \mid \mu_b, \sigma_b^2) \cdot \prod_{k=1}^{K} \text{Normal}(\log(\eta_k) \mid \mu_\eta, \sigma_\eta^2) \cdot \text{Gamma}(\zeta \mid \gamma_\zeta, \psi_\zeta). \tag{2.10}$$

This ~~posterior~~ allows us to infer the expected degrees of each node, the proportion of connections to each category, the intensity of the latent component, the relative positions of the groups in the sphere and their degree of concentration.

Algorithm

To estimate the latent parameters of the model, a Bayesian inference approach using Markov Chain Monte Carlo (MCMC) methods is used, the detailed algorithm can be found in [4]. Since the posterior distribution is complex and cannot be obtained analytically, an iterative scheme is used where, in each iteration, the different latent parameters (and hyperparameters) are updated: the latent positions z_i, the centers μ_k and concentrations η_k of the groups, the expected degrees d_i, the proportion of links to each group b_k, and the global parameter ζ. Some of these updates are performed by Metropolis-Hastings-type steps (e.g., z_i, d_i, η_k), while others, when possible, are updated with Gibbs steps by having conjugate conditional distributions (e.g., μ_k). The Metropolis-Hastings (MH) step is a common MCMC method that consists of proposing a new value and accepting it with a certain probability that depends on the improvement in the likelihood, while the Gibbs step is a special case of MH that samples directly (the acceptance probability is always 1) from the full conditional distribution of a parameter given the remaining parameters [9].

Each new value proposal is evaluated based on its impact on the likelihood, which is calculated from the Poisson distribution of ARD responses conditional on the current parameters. If the new value improves (or sufficiently maintains) the likelihood, it is accepted; otherwise, it may be rejected according to the acceptance rule of the MH algorithm. This process is repeated for a high number of iterations T, for example, $10,000$ iterations, with an initial "burn-in" phase (the first $T/2$ iterations), which is discarded to ensure that the chain has converged to the stationary distribution. Furthermore, to reduce the effect of autocorrelation between samples, several independent MCMC chains are usually run and spaced samples are taken from the second half. These samples from the posterior distribution are used to estimate the gregariousness parameters ν_i and later generate simulated networks $\{G^{(s)}\}_{s=1}^{S}$, where $s \in \{T/2+1, ..., T\}$, for each independent MCMC chain.

Graph Estimation

As seen above, starting from the observed values y_{ik} and using Bayesian inference with MCMC-type sampling steps on the latent variables and hyperparameters, this model allows inference ~~for~~ the individual-effects parameters ν_i and the latent positions z_i of nodes. Once estimated, a sample of graphs $\{G^{(s)}\}$ can be built where the probability of connection between each pair of nodes i and j is estimated as:

$$\mathbb{P}(g_{ij} = 1 \mid \nu_i, \nu_j, \zeta, z_i, z_j) = \frac{\exp(\nu_i + \nu_j + \zeta \cdot z_i^\top z_j)}{\sum_{i,j} \exp(\nu_i + \nu_j + \zeta \cdot z_i^\top z_j)} \cdot \sum_i d_i, \tag{2.11}$$

which is derived from the link formation model ~~2.3~~. In order to obtain valid probabilities between 0 and 1, the expression (2.3) was normalized across all possible node pairs. With this normalization ~~it~~ is also ensured that the expected total number of links matches the sum of the estimated degrees $\sum_i d_i$, which maintains consistency between the probabilistic model and the structural constraints of the graph. Specifically, we have:

$$\sum_j \mathbb{P}(g_{ij} = 1) \approx d_i, \quad \forall i. \tag{2.12}$$

Since the formation model assumes conditional independence of link formation given the latent variables, we can then generate full graphs by drawing independent Bernoulli trials for each node pair (i, j) using the probabilities defined in expression (2.11).

Each considered iteration of the algorithm (the second half of the iterations of each chain) generates a synthetic graph with latent properties compatible with the observed ARD. This allows us to obtain not only a single network but also a complete distribution of networks, from which metrics with credibility intervals can be computed.

Advantages and Limitations

The latent surface model has several advantages:

- The estimation problem is reduced to a moderate dimension space.
- It captures local structures such as homophily and transitivity.
- It allows imputing unobserved relationships in a way that is consistent with ARD.

However, it might also have some limitations:

- Precision depends on the number and quality of ARD subgroups used.
- It assumes conditional independence in link formation given the latent variables, which could be restrictive.

2.3. Simulation Framework

2.3.1. Synthetic Network Generation

In order to study the performance of the proposed model, in this section we implement a simulation model similar to the one developed in [4]. There are two purposes: verifying the performance of the model in a controlled environment without specification errors, and analyzing the influence of certain key parameters, specifically the concentration of latent groups, on the structure of the ARD data and the posterior inference.

For that matter, a synthetic network is generated from the network formation model described earlier. This model is based on a latent representation of the nodes on the surface of a hypersphere of dimension $p + 1$, usually with $p = 2$. The simulation process follows the next steps:

1. First, n latent positions $\{z_i\}_{i=1}^n$ are generated, uniformly distributed over the sphere $\mathbb{S}^{p+1}$.

2. Each node is assigned a gregarious parameter $\nu_i \sim \mathcal{N}(\mu, \sigma^2)$.

3. Together with these latent variables and the global parameter ζ, the graph $G = (V, E)$ is generated using the link probability defined in (2.3).

4. Once the structure of the network is defined, the next step is to define K binary characteristics (latent groups), each one associated to a center μ_k also uniformly distributed over the sphere.

5. For each group, the concentration parameter η_k is fixed, determining how disperse or gathered are its members around the center μ_k. Each node i belongs to group k with a probability proportional to the density of the vMF distribution $\mathcal{M}(\mu_k, \eta_k)$ evaluated in its latent position z_i.

6. Finally, the ARD matrix $Y = (y_{ik})$ is constructed, where each entry y_{ik} represents the number of neighbors the node i has that belong to group k, obtained through the connections in the generated graph and the group assignations.

In this simulation, the fixed parameters employed are the same as in [4]: $n = 250$, $\zeta = 0.3$, $\mu = -1.27$, $\sigma = 0.5$, y $K = 12$.

It is important to emphasize that, in this simulation model, the network structure is generated independently of the characteristics or groups to which the nodes belong. First, the network topology is defined solely based on the latent positions (z_i) and link parameters (ν_i), and only afterwards are the K characteristics assigned to the nodes based on their proximity to the μ_k centers. This implies that if the parameters and the random seed are kept fixed during the simulation model, the graph structure remains constant despite

changes in the concentration parameters η_k. The changes affect only the group assignments and, therefore, the resulting ARD matrix, not on the graph structure or adjacency matrix.

We now analyze the results obtained from the simulation model, focusing on how these outcomes vary when only the concentration parameters η_k are modified. In this setting, all other parameters are held constant. The values of η_k are drawn from a uniform distribution $\mathcal{U}(a, b)$, and we vary the range by adjusting the minimum and maximum values. Specifically, we consider four different configurations for the distribution of η_k: one with values uniformly distributed between 0.5 and 1, another between 1 and 2, a third one between 5 and 10, and finally, a case with values between 30 and 50. As a result, the same network structure is obtained in all scenarios, but the node features and, consequently, the ARD matrices, differ based on the sampled η_k values.

In Figure 2.3, we show the graphical representation of the network along with the adjacency matrix corresponding to the first 10 nodes and a histogram to see the node degree distribution. This visualization remains identical across the different settings of η_k, as the network structure itself does not change. Additionally, we present a summary table (Table 2.1) that reports key global metrics of the network.

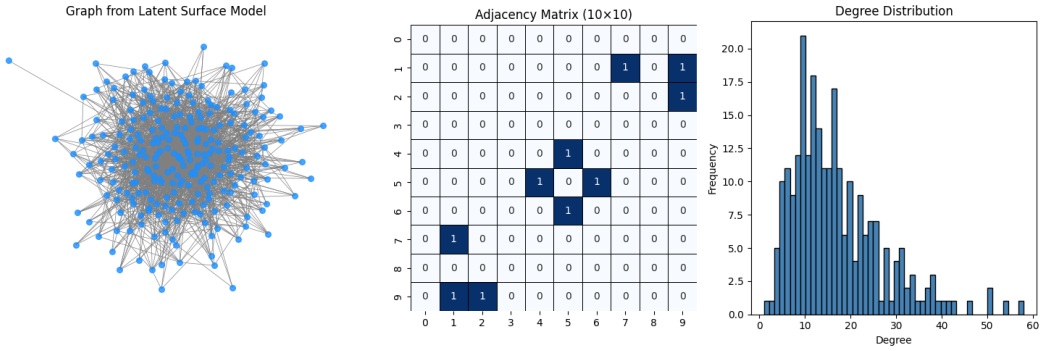


Fig. 2.3. Simulated graph obtained according to formation model.

Metric	**Value**
Number of Nodes	250
Number of Edges	2104
Average Degree	16.83
Global Clustering Coefficient	0.11
Average Shortest Path Length	2.251
Max Eigenvalue	22.79

Table 2.1. Summary of simulated graph metrics.

The degree histogram shows a unimodal and asymmetric distribution resembling a Poisson, indicating some homogeneity in node connectivity. The graph metrics reflect a moderately dense network with low local clustering and short paths between nodes, typ-

ical of "small-world" networks generated under random models. Additionally, the presence of high eigenvalues suggests some concentration of connectivity in central nodes.

To illustrate the impact of varying concentration, we compare the ARD matrices obtained under the different concentration levels by displaying the first ten rows (i.e., the ARD information about the first ten nodes), along with their corresponding node degrees in the last column. These matrices are shown in Table 2.2, arranged in a 2×2 grid and ordered from left to right, top to bottom corresponding respectively to the four configurations of η_k drawn from $\mathcal{U}(0.5, 1)$, $\mathcal{U}(1, 2)$, $\mathcal{U}(5, 10)$, and $\mathcal{U}(30, 50)$. Each matrix represents the ARD collected from the simulated network: each row corresponds to a node and each column represents a group or characteristic. Thus, the ARD matrix has dimensions $n \times K$, where n is the number of nodes and K the number of groups. The entry (i, k) indicates the number of people known by individual i who belong to group k. An additional column is included to show the degree of each node (i.e., the total number of ties), providing insight into the overall connectivity of each individual in the network compared to the number of connections detected by ARD.

While the node degrees remain unchanged (constant resulting graph topology) ~~we~~ we observe that higher group concentration (larger values of η_k) leads to significantly lower ARD counts. This occurs because when groups are highly concentrated, many nodes fall outside all defined groups, resulting in zero neighbors in the observed subgroups. As a consequence, the row sums of the ARD matrix no longer match the true degrees of the nodes.

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	**Degree**
24	26	32	27	24	29	24	28	30	22	23	25	**33**
13	8	13	12	8	10	10	10	12	13	11	14	**14**
12	11	12	7	10	8	11	9	10	9	11	12	**12**
14	12	13	12	11	12	12	12	14	11	14	11	**14**
8	8	10	8	9	10	10	12	12	10	9	10	**12**
12	10	11	10	6	12	9	11	11	8	9	11	**12**
21	21	24	23	18	22	20	23	21	21	17	20	**25**
12	12	14	11	10	15	14	12	11	7	12	12	**15**
4	4	4	5	4	5	3	5	5	5	4	4	**5**
13	15	12	15	11	16	13	15	16	13	13	9	**16**

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	**Degree**
19	21	31	20	14	22	18	25	23	19	17	21	**33**
9	8	12	11	8	8	8	10	11	8	9	10	**14**
9	10	11	6	9	5	10	7	8	5	10	11	**12**
12	11	11	10	11	9	9	11	11	7	12	9	**14**
4	5	10	7	8	9	7	10	10	7	9	6	**12**
6	9	10	7	6	9	7	10	8	5	7	10	**12**
16	14	22	20	18	19	13	21	18	15	17	15	**25**
10	12	12	6	9	9	13	11	10	6	11	12	**15**
3	3	4	5	3	4	2	5	5	3	4	3	**5**
10	12	11	12	8	10	10	14	13	10	11	8	**16**

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	**Degree**
2	7	20	8	9	14	9	13	7	6	6	5	**33**
1	3	6	2	4	7	3	6	3	1	1	2	**14**
4	3	3	1	5	2	6	1	1	0	4	1	**12**
8	6	4	4	3	4	2	5	4	2	5	3	**14**
1	2	6	2	4	3	3	3	2	2	3	3	**12**
2	4	6	3	5	6	4	5	3	1	3	3	**12**
8	7	6	8	4	6	3	7	9	7	6	4	**25**
3	6	7	4	3	2	5	2	4	3	3	2	**15**
2	0	0	2	0	2	0	2	2	1	1	0	**5**
4	4	5	3	4	3	1	3	3	3	4	3	**16**

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	**Degree**
1	2	4	5	0	4	5	4	3	1	1	1	**33**
0	0	1	2	0	2	2	2	1	0	0	0	**14**
3	0	2	0	1	0	3	0	0	0	1	0	**12**
3	1	1	0	0	1	0	1	1	2	1	1	**14**
1	2	1	0	1	2	0	1	0	1	0	1	**12**
1	0	2	0	1	2	1	2	0	1	0	0	**12**
2	3	2	2	1	4	1	4	0	1	0	3	**25**
2	1	2	1	0	0	2	0	1	2	1	1	**15**
1	0	0	1	0	1	0	1	0	0	0	0	**5**
2	1	2	2	2	0	0	1	3	2	1	1	**16**

Table 2.2. ARD Matrices across different parameters (first 10 nodes with degree)

This phenomenon implies a loss of information about the underlying network in the ARD data: the more concentrated the groups are, the fewer nodes contribute to the ag-

gregated counts. Since the ARD matrix is the core input to the latent surface estimation model, this reduction in informative content translates, as we will show in the following section, into a decline in the model's performance in both network reconstruction.

2.3.2. Graph Estimation under Different Parameters and Evaluation

In summary, so far we have followed the formation model proposed for the simulation, which has granted us a way to generate an underlying network or graph that represents the connections between nodes, as well as a way to assign individual characteristics or group membership to each node. From this complete configuration, we construct the ARD matrices, which collect the information necessary to indirectly estimate networks. With these ARD matrices in hand, we can finally apply and evaluate the model proposed by McCormick et al. [2] and described in the previous section for network estimation, given that their algorithm needs this type of data as input.

The first step will be to compare the results obtained from each of the four ARD matrices presented in Table 2.2. Specifically, we analyze the estimated graphs in each case and compare them with the original underlying graph that gave rise to the respective ARD matrices, generated previously and shown in Figure 2.3.

From each ARD matrix, we run network estimation model using MCMC with the same parameters we will use throughout the remainder of the document. The parameters are: running 3 independent MCMC chains, each with 3000 iterations, and we keep one sample every 10 iterations (as specified by the thinning parameter). From each chain, we discard the first half as burn-in, keeping the second 150 samples. This results in a total of 450 posterior draws for each ARD matrix. Each posterior draw corresponds to one full set of estimated parameters, including an estimated network. Therefore, we obtain 450 estimated graphs for each ARD matrix, which we will use in the following analysis.

Figure 2.4 shows boxplots for four global graph summary metrics: average degree, clustering coefficient, mean shortest path length, and the maximum eigenvalue of the adjacency matrix. In each figure, the results obtained from the graph estimations built with each of the four simulated ARD matrices (res1, res2, res3, and res4) are plotted, corresponding respectively to concentration parameters η_k drawn from the distributions $\mathcal{U}(0.5, 1)$, $\mathcal{U}(1, 2)$, $\mathcal{U}(5, 10)$, and $\mathcal{U}(30, 50)$. The dashed red line represents the value of the corresponding metric on the original graph (i.e., the simulated synthetic graph from which ARD was generated).

A clear trend is observed: as the groups become more concentrated (η_k increases), the estimated graph metrics progressively move away from the metrics from the original graph. This result highlights an important limitation of the network formation model used: although not usually pointed out in the literature, the model's performance improves when the groups are more dispersed; in fact, it directly depends on this assumption, since the ARD matrix then contains more relevant information about the underlying structure of

the network.

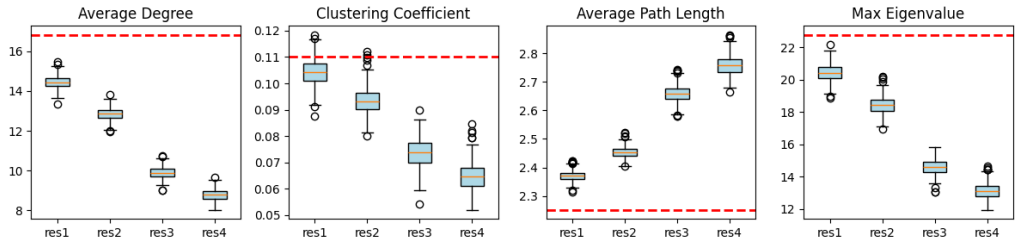


Fig. 2.4. Comparison of global metrics of estimated graphs across different group concentration configurations with the original graph.

To evaluate the performance of the network estimation algorithm, we selected two contrasting parameter settings: the best-performing case from the previous analysis, with group concentration parameters drawn from the $\mathcal{U}(0.5, 1)$ distribution, and the worst-performing case, using parameters from the $\mathcal{U}(30, 50)$ distribution. For each setting, we generated 10 distinct graphs, keeping all parameters constant but varying the simulation seed, which guarantees distinct network structures (with the same number of nodes) and ARD matrices in each case. We applied the estimation algorithm to each of these 10 ARD matrices. Unlike the initial estimation, where 450 graphs per ARD matrix were generated from multiple MCMC samples, in this case we chose to simplify the comparison: taking the average of the 450 estimated parameter sets, we generated a single estimated graph for each of the 10 ARD matrices. The result is 10 estimated graphs, each corresponding to an original simulated network. With these pairs of graphs (original and estimated) we perform comparative analyses both node-level and graph-level using scatter plots, including a dashed red line representing the ideal diagonal $y = x$, which allow us to observe the fitness of the model and how close the estimates are to the original value.

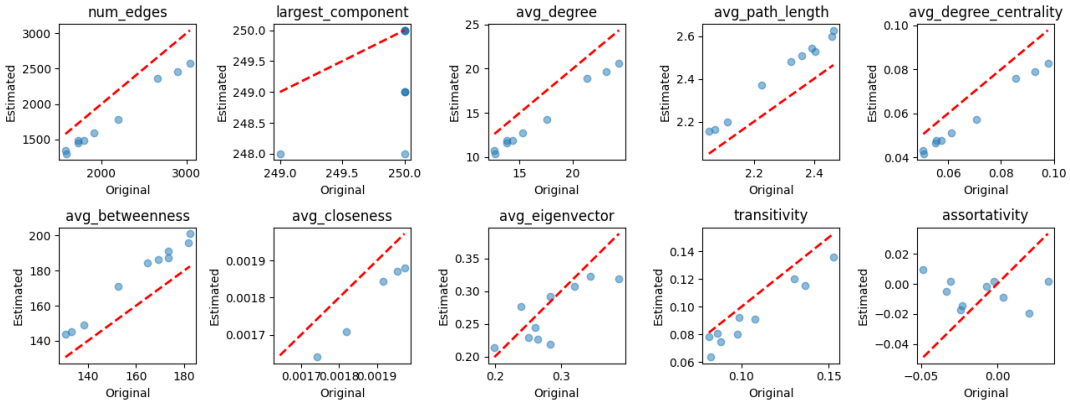


Fig. 2.5. Graph-level metrics comparing the ten estimated graphs sampling η_k from $\mathcal{U}(0.5, 1)$ with the synthetic generated original graphs.

In Figure 2.5 we see the comparisons of various graph-level structural metrics between the original and estimated values. Overall, all metrics exhibit a clear linear trend,

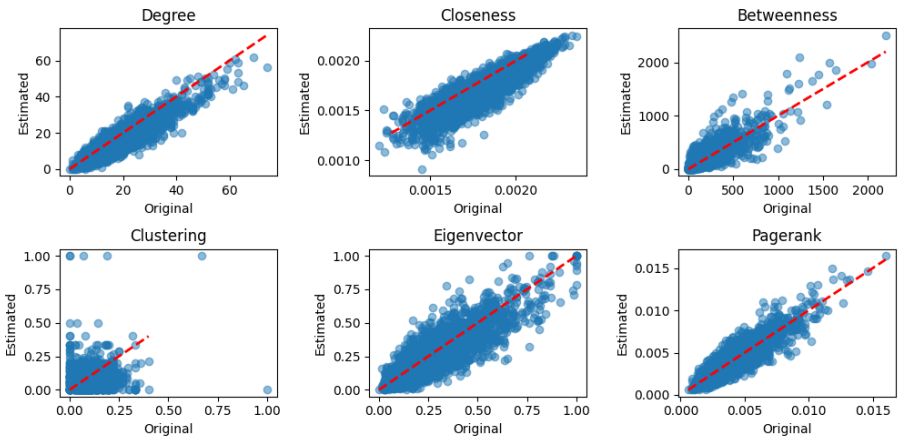


Fig. 2.6. Node-level metrics comparing the ten estimated graphs sampling η_k from $\mathcal{U}(0.5, 1)$ with the synthetic generated original graphs.

indicating that the model is capable of preserving proportionality between the original and estimated graphs. However, in some metrics, the estimated values systematically fall above or below the reference line (dashed red line), suggesting certain biases. For instance, the estimated graphs generally have a lower number of edges (num_edges), lower average degree (avg_degree), and higher average path length (avg_path_length), which indicates that the generated structures tend to be somewhat less connected than the original ones. Regarding the size of the largest component (largest_component), although it may appear poorly estimated visually, the estimation is actually quite accurate: the original graph values range between 249 and 250, and the estimated ones stay within 248, ~~249, or~~ 250, suggesting that the main connectivity is largely preserved. The metric that deviates the most from a linear trend is assortativity, which shows high variability in the original graphs, while the estimated values are more tightly clustered around zero. This may indicate that the model produces graphs with less variation in the connection patterns between similar nodes—that is, with a more neutral or randomized structure regarding degree assortativity.

Figure 2.6 presents the comparisons for various node-level metrics. There is a strong agreement between the estimated and original values for most measures, particularly for degree, pagerank, closeness centrality, and eigenvector centrality, indicating that the model effectively preserves the nodal distributions. In the case of betweenness centrality, although a clear linear trend is still observable, the points are more widely dispersed around the reference line, suggesting greater variability in the estimation. Clustering coefficient, on the other hand, shows no apparent linear relationship; the points form a scattered cloud, indicating that the model struggles to accurately reproduce local clustering structures. This discrepancy may stem from the fact that these metrics, especially clustering, are highly sensitive to local network topology. As observed at the graph level, the estimated graphs tend to be less connected overall, which can lead to fewer alternative paths and lower local cohesion, directly affecting these measures. Nevertheless, the general trend remains consistent, and the model demonstrates a good capacity to approximate

the original structure.

We now repeat the analysis for the previous worst-case scenario, using group concentration parameters sampled from the $\mathcal{U}(30, 50)$ distribution. The corresponding results are shown in Figures 2.7 and 2.8. In general, performance is clearly worse compared to the $\mathcal{U}(0.5, 1)$ case. At the graph level, although a linear relationship can still be visible, the estimated values deviate substantially more from the ideal reference line, indicating that while some proportionality is preserved, the estimates are much farther from the true values. A similar pattern is observed in the node-level comparisons: the alignment remains approximately linear, but with greater dispersion and consistently larger deviations from the diagonal. These visual patterns suggest that estimation quality deteriorates as the concentration parameter increases.

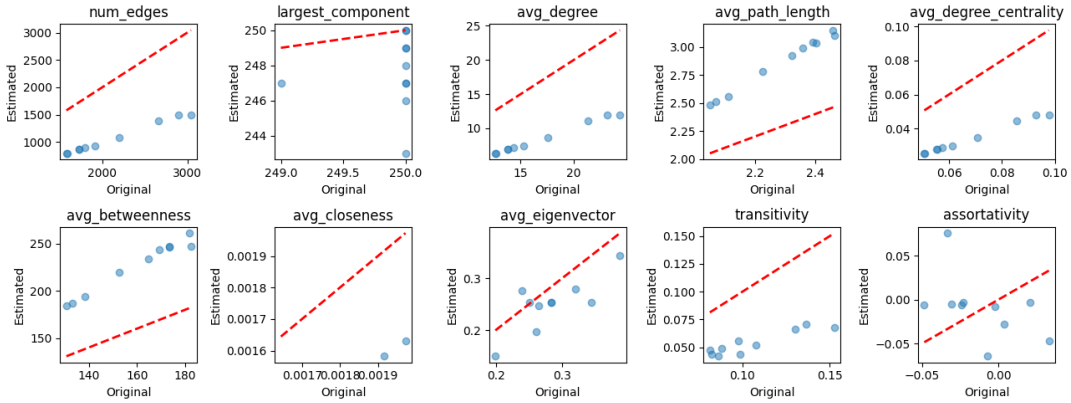


Fig. 2.7. Graph-level metrics comparing the ten estimated graphs sampling η_k from $\mathcal{U}(30, 50)$ with the synthetic generated original graphs.

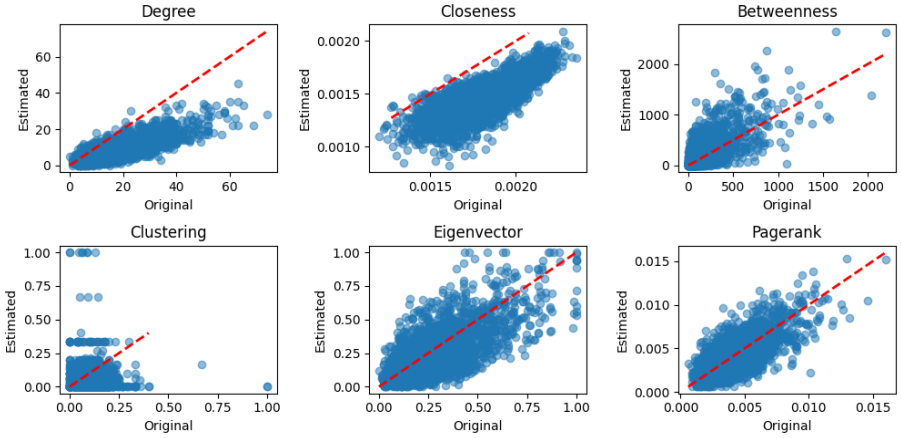


Fig. 2.8. Node-level metrics comparing the ten estimated graphs sampling η_k from $\mathcal{U}(30, 50)$ with the synthetic generated original graphs.

To quantify these differences, we now compute evaluation metrics, including the coefficient of determination (R^2), mean squared error (MSE), and mean absolute error (MAE),

which provide a numerical assessment of the similarity between the original and estimated values. As shown in Table 2.3, estimation performance degrades substantially under the higher concentration setting. Both graph-level and node-level metrics exhibit a dramatic drop in R^2, along with significant increases in MSE and MAE, reinforcing the conclusion that higher concentration parameters lead to poorer graph estimation.

Graph-Level

Distribution	R^2	MSE	MAE
$\mathcal{U}(0.5, 1)$	0.326	12,476.35	38.22
$\mathcal{U}(30, 50)$	-20.275	130,574.24	125.19

Node-Level

Distribution	R^2	MSE	MAE
$\mathcal{U}(0.5, 1)$	0.381	2,306.32	12.80
$\mathcal{U}(30, 50)$	-0.705	7,251.73	23.14

Table 2.3. Evaluation metrics comparing concentration parameter sampling η_k from $\mathcal{U}(0.5, 1)$ or $\mathcal{U}(30, 50)$.

3. COMMUNITY DETECTION WITH GNNS

3.1. Introduction to Graph Neural Networks

Graph Neural Networks (GNNs) are an extension of deep learning to the analysis of structured data such as graphs, a form of non-Euclidean representation where data does not reside on a regular grid, but rather in nodes interconnected by edges. Unlike tabular or sequential data, graphs model complex relationships between entities, such as friendships in social networks, molecular interactions, or links between documents, through nodes and edges. This makes many real-world applications, from social networks to bioinformatics, require models capable of reasoning over such complex structures. GNNs thus emerge as models whose most natural and powerful application lies in network-structured scenarios [10], [11].

In traditional machine learning, working with graph-structured data typically involved a preprocessing step that transformed the graph into fixed-size feature vectors, often sacrificing essential topological information such as local or global connectivity [12]. Statistical models such as random walks, Markov chains, Support Vector Machines with graph-based kernels, or recursive neural networks attempted to preserve this structure during processing. Recursive neural networks, for instance, performed well on trees or directed acyclic graphs, but required complex preprocessing in the case of general graphs, and did not always guarantee convergence to a stable solution. These approaches proved inefficient or inadequate for large, cyclic, or dynamically evolving graphs.

On the other hand, the rise of Convolutional Neural Networks (CNNs) revolutionized image analysis by exploiting the spatial regularity of data (a grid of pixels). However, when CNNs were directly applied to graphs, their limitations quickly became apparent: graphs lack a regular structure, the number of neighbors per node varies, there is no fixed notion of locality as in images, and nodes are not canonically ordered. These differences prevented CNNs from achieving strong performance on structured data such as social networks, molecules, or knowledge graphs.

It was precisely the inability of traditional models (both statistical and neural) to handle the arbitrary, dynamic, and non-Euclidean structure of graphs that motivated the development of GNNs. These networks explicitly incorporate the graph topology into the learning process, allowing each node (as a unit of the network) to update its internal representation using not only its own features, but also those of its neighbors, thus preserving the structure of the graph throughout training.

Historically, the first ideas for neural networks on graphs emerged in the early 2000s with models such as the original Graph Neural Networks [12]. However, it was from 2016 onward, with the rise of deep learning and works such as Graph Convolutional Networks

(GCN) [13], that GNNs gained popularity and became essential tools for representation learning on graphs.

GNNs represent a unified framework that generalizes and extends both recursive neural network models and diffusion-based approaches, allowing them to work directly with directed, undirected, cyclic, or acyclic graphs. The fundamental principle behind GNNs is the message passing or information diffusion mechanism: each graph node is represented by a unit whose vector representation is iteratively updated by combining its current state with information from its neighbors, its attributes, and the local structure of the graph. This process, repeated layer by layer, enables each node to incorporate both local and global knowledge, capturing complex patterns of connectivity. In practice, most GNNs aim to learn a vector embedding per node that encodes both features and structure [14].

A GNN is composed of layers that perform aggregation and update operations. Formally, at each layer l of a GNN, the representation of node i is updated as:

$$h_i^{(l)} = \text{UPDATE}^{(l)}\left(h_i^{(l-1)}, \text{AGGREGATE}^{(l)}\left(h_j^{(l-1)} : j \in \mathcal{N}(i)\right)\right) \tag{3.1}$$

where $\mathcal{N}(i)$ represents the set of neighbors of node i. The vector $h_i^{(l)}$ contains an embedded representation of node i in the layer l, which combines both its own information as well as the information from its surroundings or neighborhood. After several layers, a final representation $h_i^{(L)}$ is obtained, which can be used for multiple tasks, such as: node classification, link prediction or community detection.

Depending on the task, GNNs can work on two different types of applications:

- Node-focused approach: predicting labels or characteristics about individual nodes. Typical examples include: the classification of web pages or the identification of entities in knowledge graphs.
- Graph-focused approach: predicting global-level information about the graph. These applications include: determining if a molecule represented as a graph has toxic properties or classifying documents represented as semantic graphs.

GNN training can be transductive, when the model is trained on the full graph (as in GCN), or inductive, when the goal is to generalize to unseen nodes or graphs (as in GraphSAGE). Loss functions vary depending on the task: cross-entropy for classification, reconstruction loss for link prediction, etc.

GNNs have rapidly evolved, leading to various architectures that optimize different stages of the aggregation process. Among the most well-known architectures are:

- **GCN** (Graph Convolutional Network) [13]: extends classical convolution to graphs, using the Laplacian operator to define a normalized mean aggregation of neighbor representations. It is efficient and suitable for semi-supervised classification on known graphs.

- **GAT** (Graph Attention Network) [10]: introduces an attention mechanism that learns to weight the contribution of each neighbor differently. This allows for capturing more complex and adaptive relationships between nodes, being useful when neighbor information is heterogeneous.

- **GraphSAGE** [11]: proposes an inductive architecture that uses neighbor sampling and aggregation functions (mean, LSTM, pooling) to build node embeddings. It is particularly effective on large graphs or in scenarios where generalization to new nodes is required.

GNNs thus represent a natural evolution of CNNs and RNNs, adapted to the graph domain. While a CNN operates on image pixels (regular structure), and an RNN on word sequences, a GNN can operate on arbitrary structures where each node represents an entity with non-uniform connections. For this reason, GNNs have become essential tools for representation learning in domains with network-structured data, demonstrating strong performance across multiple domains. In social networks, they are used to detect communities, predict links, or recommend personalized content. In bioinformatics, they enable the prediction of protein functions, molecular interactions, or mutation effects. In recommendation systems and fraud detection, GNNs extract complex behavioral patterns between users and transactions. Furthermore, they have begun to be used in NLP, computer vision (e.g., for scene modeling), and computational physics [15].

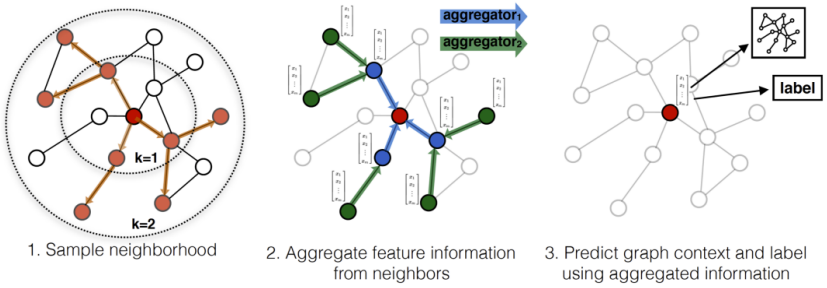


Fig. 3.1. Visual illustration of the GraphSAGE sample and aggregate approach [11].

3.1.1. Graph Convolutional Networks (GCN)

~~Graph Convolutional Networks~~ ~~(~~GCNs~~)~~ are a fundamental architecture within the family of GNNs. They extend the notion of convolution from regular domains (such as images) to irregular graph structures by aggregating feature information from the neighborhood of a node [13].

In a GCN layer, each node updates its representation by combining its own features with those of its neighbors, using the connectivity encoded in the graph's adjacency matrix. Formally, a single GCN layer performs the transformation described in 3.2.

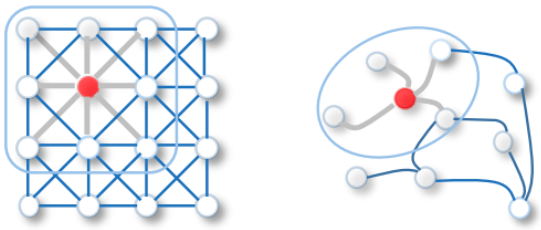


Fig. 3.2. 2D Convolution (left) vs. Graph Convolution (right) [16].

$$H = \sigma\left(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}XW\right), \tag{3.2}$$

where $X \in \mathbb{R}^{N\times F}$ is the matrix of input node features, W is a learnable weight matrix, $\hat{A} = A + I$ is the adjacency matrix with added self-loops, $\hat{D}$ is the corresponding degree matrix, and σ is a non-linear activation function, typically ReLU. This formulation ensures that each node's new representation is a normalized aggregation of its own features and those of its neighbors [13].

The output of a GCN layer is a new feature matrix $H \in \mathbb{R}^{N\times F'}$, where each row corresponds to the updated embedding of a node. Here, F' represents the number of output features or embedding dimensions, which is analogous to the number of filters in a ~~convolutional neural network~~ for images. This dimensionality is determined by the architecture and reflects the richness of the learned representation. Stacking multiple GCN layers allows information to propagate through the graph beyond immediate neighbors, enabling each node to incorporate knowledge from increasingly distant parts of the network [16]. A general overview of the architecture of the GCN model is shown in Figure 3.3.

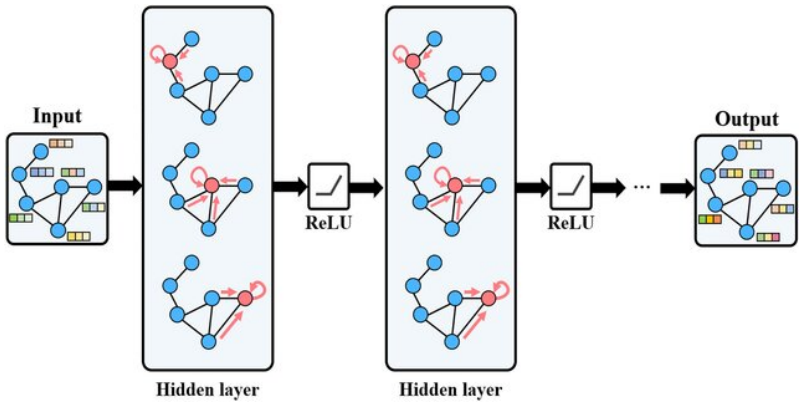


Fig. 3.3. Overview of the GCN architecture[17].

~~Graph Convolutional Networks~~ belong to the family of spectral methods, which define convolution in the frequency domain by leveraging the eigendecomposition of the graph Laplacian [10]. In contrast, spatial methods such as GraphSAGE or GAT perform message passing directly in the graph domain, aggregating information from neighbors using flexible aggregation functions. GCNs implement a simplified spectral convolution that avoids full eigendecomposition by using an efficient first-order approximation, resulting

in fast and scalable operations. Compared to GAT, which uses attention mechanisms to assign different weights to each neighbor, GCN treats all neighbors equally (after normalization), making it less expressive but computationally lighter. GraphSAGE, on the other hand, enables inductive learning by sampling and aggregating neighbors using functions such as mean or LSTM pooling, which makes it more suitable for large-scale or dynamic graphs. In summary, GCN provides an efficient and well-founded approach for semi-supervised learning on static graphs, while other models might introduce more flexibility or scalability at the cost of added complexity.

3.2. Community Detection via GNNs

Community detection is a fundamental task in network analysis: it consists of identifying subsets of nodes, either because they are densely connected to each other and sparsely connected to the rest of the graph, or because they share common attributes. Traditionally, this task has been addressed using methods such as modularity optimization, spectral clustering, or density-based techniques. However, GNNs offer a novel approach by enabling the extraction of vector representations of nodes called embeddings, upon which clustering techniques such as k-means can be applied [15].

In this work, we adopt an unsupervised approach called Deep Graph Infomax (DGI), proposed by Veličković et al. [18], which has proven effective in generating informative embeddings from graphs. We choose DGI because it is not supervised (there is no prior information about communities among nodes), can be applied in transductive settings (which aligns with our goal of extracting information about an underlying graph structure), and can incorporate node metadata. The output of DGI is a set of node embeddings, which we then use for community detection.

3.2.1. Deep Graph Infomax (DGI)

~~Deep Graph Infomax~~ (DGI~~)~~ is a self-supervised learning method for graphs proposed by Veličković et al. (2019), whose goal is to learn useful node representations by maximizing local-global mutual information, that is, between the individual representations of nodes and a global representation of the entire graph. Inspired by Deep InfoMax (DIM) [19], DGI leverages principles of contrastive learning, a paradigm that trains models to distinguish between positive (true) and negative (corrupted or unrelated) pairs, in order to learn more meaningful representations.

This approach avoids the need for manual labels (characteristic of supervised learning) and instead generates so-called negatives automatically through a corruption function applied to the graph. As a result, DGI is suitable for both transductive and inductive settings, as it does not rely on labeled nodes during training and can generalize to unseen nodes.

The DGI model consists of several key components, namely: an encoder, a readout function, and a discriminator. Let (X, A) be the pair containing, respectively, the nodal features matrix (X) and the adjacency matrix(A) of the graph. These elements can be summarized as follows:

- **Encoder** $(\mathcal{E})$: A graph neural network, such as GCN, is used to transform the input features X and the adjacency matrix A into a representation h_i for each node. This step produces a set of node embeddings $H = \mathcal{E}(X, A) = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\}$.

- **Readout function** $(\mathcal{R})$: From the set of node embeddings H, a global representation of the graph is computed, denoted $\vec{s}$, using an aggregation function. In the original work, this function $\mathcal{R}(H) = \vec{s}$ is a simple average followed by a non-linear activation (e.g., sigmoid or tanh). This representation serves as a kind of summary of the entire graph.

- **Corruption** (C): To simulate a fake or negative graph, a corruption function is applied to the features of the original graph. For example, node features can be randomly permuted, thus generating a new input set $(\tilde{X}, \tilde{A}) \sim C(X, A)$ and corresponding representations $\tilde{H} = \mathcal{E}(\tilde{X}, \tilde{A}) = \{\tilde{\vec{h}}_1, \tilde{\vec{h}}_2, \ldots, \tilde{\vec{h}}_M\}$.

- **Discriminator** $(\mathcal{D})$: A discriminator $\mathcal{D}(\vec{h}, \vec{s})$ is trained to take as input a pair formed by a node embedding $\vec{h}_i$ and the global graph summary $\vec{s}$, producing a score that reflects whether the node comes from the true graph or from a corrupted version. Its goal is to assign high scores to real node-summary pairs $(\vec{h}_i, \vec{s})$, and low scores to negative (corrupted) ones $(\tilde{\vec{h}}_i, \vec{s})$. A common implementation uses a bilinear scoring function followed by a sigmoid activation:

$$\mathcal{D}(\vec{h}_i, \vec{s}) = \sigma\left(\vec{h}_i^{\top} W \vec{s}\right), \tag{3.3}$$

  where W is a learnable weight matrix and σ denotes the sigmoid function. This score measures the alignment between local (node-level) and global (graph-level) representations, and serves as the basis for the contrastive objective used in DGI.

- **Objective** $(\mathcal{L})$: The training process aims to maximize the mutual information between each node and the graph summary. This is achieved through a contrastive loss function, typically a binary cross-entropy, which differentiates between positive (real) and negative (corrupted) pairs. In practical terms, the goal is to maximize the alignment between real nodes and $\vec{s}$, and minimize it for corrupted nodes.

$$\mathcal{L} = \frac{1}{N+M}\left(\sum_{i=1}^{N} \mathbb{E}_{(\mathbf{X},\mathbf{A})}\left[\log \mathcal{D}(\vec{h}_i, \vec{s})\right] + \sum_{j=1}^{M} \mathbb{E}_{(\tilde{\mathbf{X}},\tilde{\mathbf{A}})}\left[\log\left(1 - \mathcal{D}(\tilde{\vec{h}}_j, \vec{s})\right)\right]\right), \tag{3.4}$$

The loss function $\mathcal{L}$ defined in equation 3.4 corresponds to a contrastive formulation based on binary cross-entropy. Its goal is to train the discriminator $\mathcal{D}$ to assign high

scores (close to 1) to positive pairs formed by node embeddings $\vec{h}_i$ and the global summary of the graph $\vec{s}$, and low scores (close to 0) to negative pairs that combine $\vec{s}$ with corrupted embeddings $\tilde{\vec{h}}_j$. Equation 3.4 is meant to be maximized, its design aims to increase the score of the discriminator for positive pairs and decrease it for negative ones. In this way, the model learns informative node representations that capture the structural information of the graph. In some implementations, the sign of this function is flipped to follow the standard minimization setup, but conceptually, the original formulation seeks to maximize mutual information. Each term in the loss represents an expectation over real or corrupted samples, allowing stochastic generation of negative examples, and the total sum is averaged over $N + M$ samples to ensure balanced training.

Following [18], the complete process, illustrated in Figure 3.4, is defined in the following steps:

1. A negative sample is generated by applying the corruption function: $(\tilde{X}, \tilde{A}) = C(X, A)$.
2. A set of node embeddings is obtained for the original graph: $H = \mathcal{E}(X, A)$.
3. A set of embeddings is obtained for the corrupted graph: $\tilde{H} = \mathcal{E}(\tilde{X}, \tilde{A})$.
4. The original graph is summarized using the readout function: $\vec{s} = \mathcal{R}(H)$.
5. The parameters of the encoder $\mathcal{E}$, readout function $\mathcal{R}$, and discriminator $\mathcal{D}$ are updated using gradient descent, maximizing the contrastive objective function 3.4.

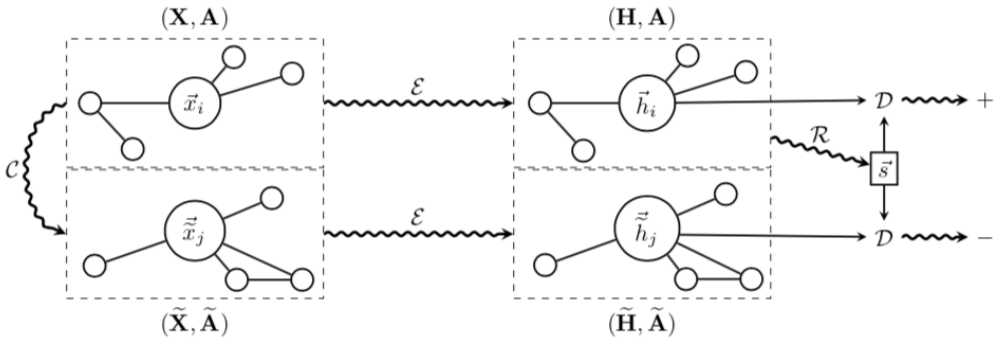


Fig. 3.4. A high-level overview of the architecture of Deep Graph Infomax [18].

DGI stands out as a self-supervised method that does not require manual labels, since it automatically generates positive and negative pairs from the graph itself, making it scalable and flexible. Its representations combine both the structural information of the graph and the node features, leveraging the available metadata. Moreover, it can be applied in both transductive and inductive scenarios, and its architecture is independent of the specific encoder used, making it compatible with various GNN variants such as GCN, GAT, or GraphSAGE.

Once the model has been trained, the resulting node embeddings $\vec{h}_i$ can be used in a variety of downstream tasks, such as clustering nodes based on structure and similarity, or visualizing them in latent space using techniques like t-SNE.

3.3. Pipeline

In this document, we integrate the Deep Graph Infomax (DGI) model into a pipeline designed for community detection in social networks based on ARD data. The main goal is to identify node groupings or communities using only two sources of information: on the one hand, aggregated relational data (ARD) responses; and on the other, metadata about the nodes such as age, gender, or occupation. We assume a setting where the true structure of the network (connections and topology) is unknown, and the objective is to reconstruct a sufficiently informative structural representation to extract latent communities.

The proposed pipeline consists of the following main stages:

1. **Graph Estimation from ARD**: as explained in Chapter 2, from the ARD responses and using the Bayesian model proposed by McCormick et al., we generate samples of estimated network structures.

2. **DGI Training**: for each estimated graph, a GCN is trained as the encoder and DGI is applied to obtain node embeddings. An early stopping strategy based on validation loss is used, and the hyperparameters (number of layers, embedding dimension, learning rate) are tuned empirically. This process yields a set of node embeddings for each graph, which are then combined into an ensemble to produce a single embedding per node, leveraging all estimated graphs during training.

3. **Embedding Clustering**: once the node vectors are obtained, k-means (the clustering algorithm chosen for this work) is applied to identify communities. The number of clusters can be set based on known group counts, selection methods such as WCSS, or empirical testing.

4. **Evaluation**: evaluating this pipeline presents particular challenges, as it starts from tabular data (ARD and node attributes), while the final output consists of structural node groupings derived from an unobserved network. For this reason, evaluation is carried out in several stages. First, during graph estimation, the simulated networks are compared to the real network (which is known in our case study) using structural metrics such as average degree and path length. Second, the DGI model, being self-supervised, is not evaluated using traditional accuracy metrics, but its objective function can be tracked during training, and the quality of the embeddings can be assessed via 2D projections using t-SNE to examine whether communities are spatially separable. Finally, after clustering, clustering metrics such as the silhouette score are used, along with qualitative (interpretation of the groups) and quantitative

evaluations, when external information is available to validate the detected communities.

In the following chapter, this pipeline is applied to a real case study using a known network with metadata about its nodes, from which ARD responses are simulated. This controlled setup allows us to directly validate the model's ability to detect latent communities that are coherent with the true underlying structure.

4. CASE STUDY

4.1. Setting and Data

In this case study, we apply the proposed pipeline to a real-world scenario involving social network data. Specifically, we use a subset of the Facebook dataset presented by Leskovec and McAuley [20], which contains ego-centric networks with node features and user-defined social circles.

The dataset consists of anonymized data collected from Facebook users via a survey application. Each ego network represents the connections among the friends of a single user (excluding the ego itself), along with their features and the social circles (or "friend lists") defined by the user. The features are anonymized binary indicators (e.g., education, work, location), with a total of 224 per node. For this study, we use the ego network of user 0, which includes:

- 333 nodes and 2519 edges, with an average degree of 15.13 and a maximum degree of 77.
- 5 connected components, the largest containing 324 nodes.
- A diameter of 11 and an average shortest path length of 3.75 in the largest component.
- A clustering coefficient of 0.508 and 10740 triangles.
- Positive assortativity (0.236) and transitivity (0.426).

Figure 4.1 displays the structure of the ego network of Facebook user 0. In addition, Figure 4.2 shows the degree distribution of the same graph, illustrating a highly right-skewed pattern where most nodes have moderate degree, but a few central nodes exhibit significantly higher connectivity.

The graph contains 333 nodes and 2519 undirected edges, with an average degree of 15.13 and a maximum degree of 77. It is moderately sparse and contains 5 connected components, the largest comprising 324 nodes. Within this main component, the diameter is 11, and the average shortest path length is 3.75, indicating relatively efficient communication paths. The network shows a high clustering coefficient (0.508), with 10740 triangles, reflecting dense local connectivity. Moreover, it exhibits positive degree assortativity (0.236), meaning that nodes tend to connect to others with similar degree, and a global transitivity of 0.426, indicating a strong tendency to form triadic closures, both typical features of social graphs.

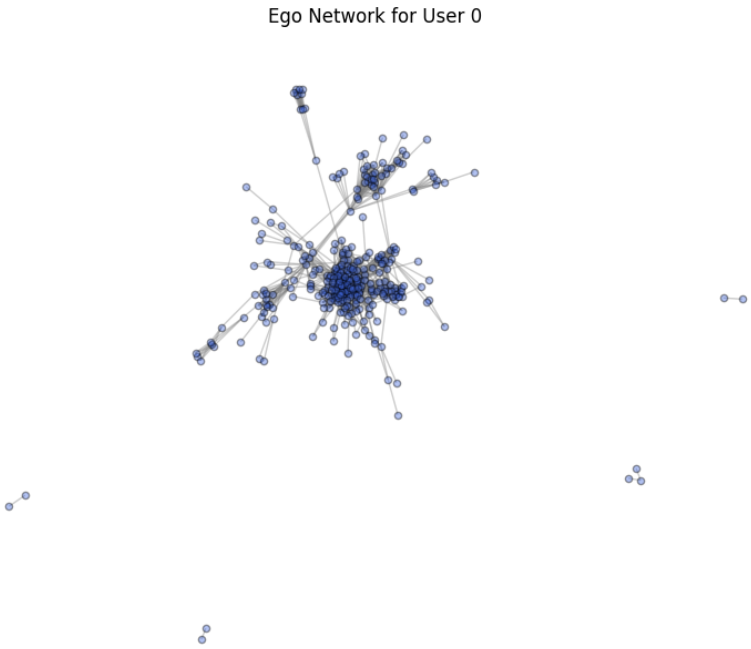


Fig. 4.1. Facebook ego network of user 0.

To further understand node importance, Figure 4.3 presents visualizations of the network with nodes highlighted according to four centrality measures: degree, betweenness, closeness, and eigenvector centrality.

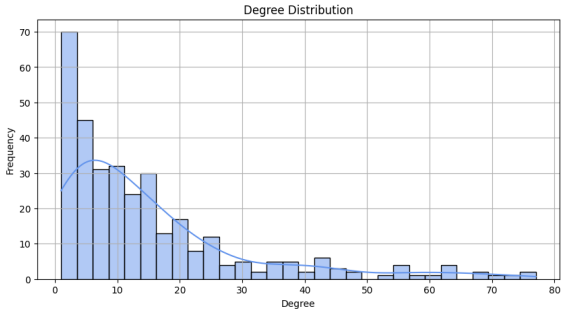


Fig. 4.2. Degree distribution of the Facebook ego network of user 0.

The visualizations reveal the structural roles of different nodes in the network. Nodes with high degree centrality are spread throughout the graph, often located at the core of subgraphs or communities, acting as hubs with many direct connections. Betweenness centrality highlights bridge nodes that lie on the shortest paths between other nodes, indicating key points for potential control over information flow. Closeness centrality emphasizes nodes that can reach others quickly; in the corresponding plot, most nodes exhibit similarly high values, suggesting that the overall network structure allows for relatively short paths between any pair of nodes. Lastly, eigenvector centrality assigns higher scores to nodes connected to other well-connected nodes. Here, only a few nodes achieve high eigenvector centrality, and they are concentrated near the network's core, indicating that

influence and connectivity are localized within specific cohesive clusters. These patterns suggest that the network is not uniformly connected, as also seen in the degree distribution histogram, but rather displays a slight core-periphery structure, a characteristic commonly observed in real-world social networks.

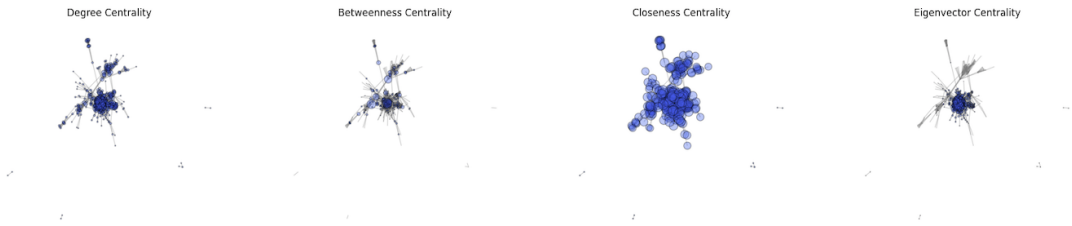


Fig. 4.3. Node importance in the Facebook ego network of user 0, highlighted by different centrality metrics.

A total of 24 user-defined social circles are present, which reflect overlapping communities. Nodes can belong to multiple circles simultaneously and some nodes don't even belong to any circle. Out of the 333 nodes, 277 are assigned to at least one circle, while 39 belong to more than one, and 56 nodes do not appear in any circle. To focus the analysis on meaningful groupings, we retain only those circles that include at least 5% of the total nodes in the graph. Nodes that are not assigned to any of these significant circles are grouped under the label `circleNA`. Table 4.1 summarizes the distribution of nodes across the selected circles. In Figure 4.4, the network is displayed with nodes colored by circle assignment and nodes belonging to multiple circles are colored in black to emphasize overlapping membership.

Circle	**Number of Nodes**	**% of Nodes**
circle15	133	39.94%
circle16	32	9.61%
circle11	30	9.01%
circle0	20	6.01%
circle6	20	6.01%
circle4	17	5.11%
circleNA	107	32.13%

Table 4.1. Summary of selected social circles (with ≥ 5% of node population).

A total of 21 nodes belong to multiple significant circles, making the ground truth community structure inherently overlapping (multi-label scenario). Only connected nodes are considered in the analysis; isolated nodes are excluded.

Fig. 4.4. Facebook ego network of user 0 with nodes colored by circle. Nodes belonging to multiple circles appear in black.

4.2. Application of the Pipeline

We begin by simulating ARD responses from the Facebook social circles network and circle membership information. The ARD matrix is constructed as a 333×7 matrix, where each row corresponds to a node and each column to a circle, counting the number of links the node has with members of each circle. Since circles can overlap, row sums in the ARD matrix may exceed the actual node degree in the original graph. Additionally, node features (the 333×224 binary metadata matrix) and the total proportion of inter-group links (calculated as 3.1719) are included as inputs in the pipeline.

4.2.1. Graph Estimation from Simulated ARD

Graph estimation is carried out using the ARD-based Bayesian model described in Chapter 2. For this, we perform three independent MCMC chains, each with 3000 iterations and a thinning parameter of 10, saving one sample every 10 iterations, and apply the model to the ARD simulated matrix. This results in a total of 450 estimated graphs, meaning 450 adjacency matrices, all of size 333×333.

A visual example of one of the estimated graphs is shown in Figure 4.5. At first glance, it is evident that this graph contains more disconnected nodes and smaller components compared to the original Facebook ego network. This qualitative observation suggests a loss of overall connectivity and motivates a more systematic comparison through quantitative metrics.

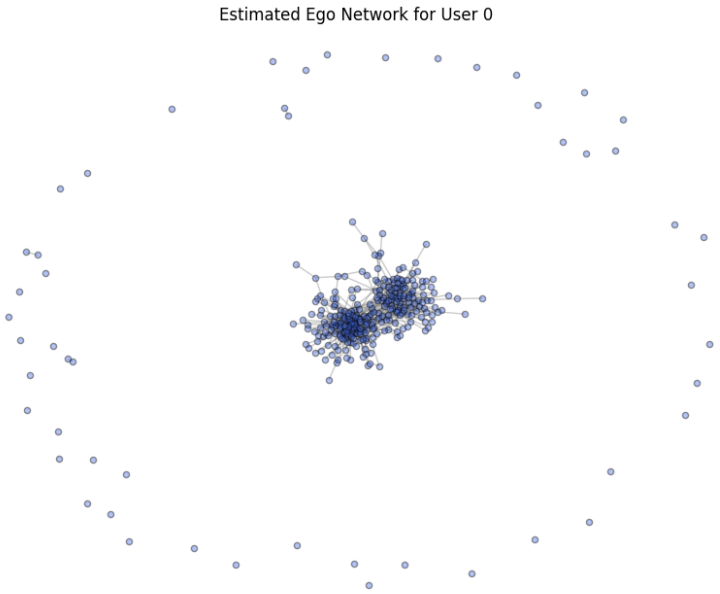


Fig. 4.5. Example of one estimated graph sampled from the posterior.

We evaluate the quality of graph estimation through both graph-level and node-level comparisons with the true Facebook ego graph.

At the **graph level**, we compute several global structural metrics (such as number of edges, average degree, size of the largest connected component, transitivity, and clustering coefficient) for each of the 450 estimated graphs. These values are then visualized using histograms, where each distribution reflects the variation of a specific metric across all sampled graphs. To facilitate comparison with the true Facebook ego network, the corresponding value of each metric in the original graph is marked with a dashed red line. These visualizations reveal that the estimated graphs tend to be significantly less connected and cohesive than the original, with the red line frequently lying at the far right of the distribution, especially in metrics related to connectivity and clustering.

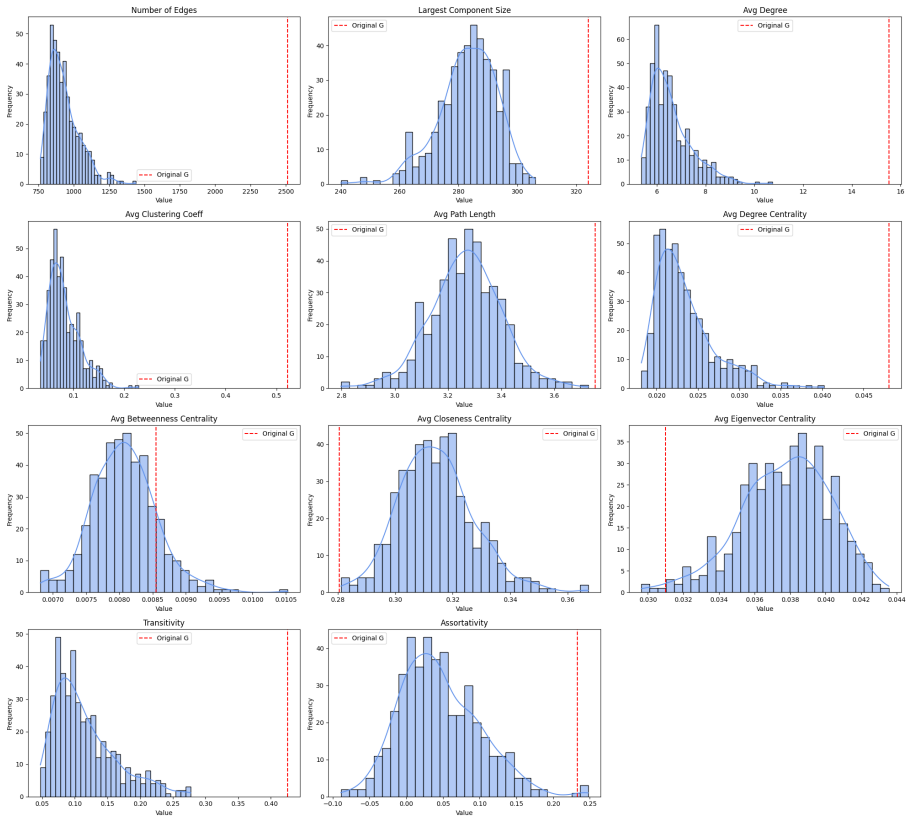


Fig. 4.6. Comparison of graph-level metrics: distribution across 450 estimated graphs vs. true graph values (red).

In contrast, metrics like betweenness centrality fall within the range of the simulations, though the real graph values remain higher, suggesting partial preservation of critical path structure. Interestingly, closeness and eigenvector centrality values tend to be higher in simulated graphs, likely because sparser and more fragmented graphs reduce average path lengths within components, artificially inflating these centralities.

At the **node-level**, we assess how well centrality measures are preserved for individual nodes. Specifically, we compare the value of each centrality metric (e.g., degree, betweenness, closeness, eigenvector) computed on the original graph with the mean of that metric across the 450 estimated graphs. In addition, we visualize the standard deviation of each estimate to assess the variability introduced by the inference process. Each scatter plot

includes the identity line $y = x$, which indicates perfect agreement between the original and estimated values. Points below this line suggest systematic underestimation, while deviations from it provide insights into the bias and dispersion of the estimates.

For example, in Figure 4.7 (top-left plot), we observe the relationship between each node's true degree and its average degree across all estimated graphs. There is a clear positive correlation, showing that nodes with higher degree in the original graph also tend to have higher estimated degrees. However, most points fall noticeably below the $y = x$ line, indicating a consistent underestimation of node connectivity. The plot also shows that this underestimation is relatively uniform across the network, aligning with earlier findings that the estimated graphs are systematically sparser than the original.

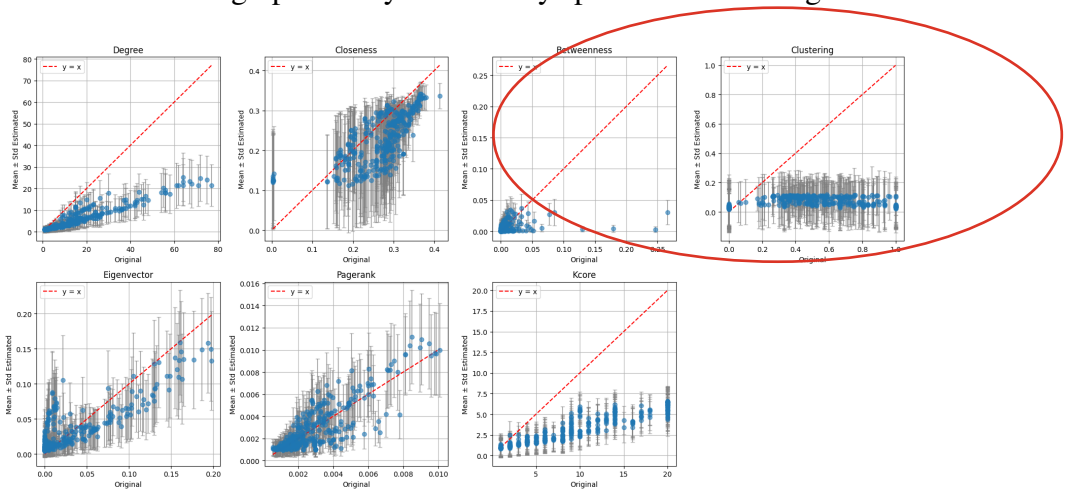


Fig. 4.7. Comparison of node-level metrics: scatter plots of true vs. estimated average node metrics with std from 450 estimated graphs.

This under-representation impacts higher-order metrics as well, and reflects the broader observation that the estimated graphs are sparser, less modular, and less hierarchical. Influential nodes and tight communities might be missed, which may limit the utility of these graphs for tasks like community detection or influence analysis directly.

4.2.2. Link Prediction

To address the under-representation problem observed in the estimated graphs, specifically their tendency to be less connected and to contain fewer edges compared to the original networks, we apply graph-theoretic link prediction methods to heuristically augment connectivity. As discussed in Section 2.3.2, this issue was already apparent in our earlier simulations and was particularly pronounced when the underlying group distributions (in this case, the distributions from which circle concentrations η_k are sampled) were more concentrated. Under such conditions, the ARD-based estimation model proposed by McCormick et al. [2] tended to produce overly sparse and fragmented networks. This is likely due to a loss of informative content in the ARD matrices: as group membership becomes more concentrated, fewer nodes contribute significantly to the aggregate counts,

leading to noisier and less representative inputs to the latent surface model. As a result, the model struggles to infer latent ties, especially weaker or less central connections, which results in underestimation of global connectivity.

This connection between group concentration and estimation sparsity, while hinted at in our experiments, has not been formally explored in the literature. Therefore, as a possible direction for future research, it would be valuable to develop technical metrics that quantify the extent to which concentration parameters affect ARD informativeness and, in turn, model performance. Such analysis would enable us to better understand the model's sensitivity to input structure and to potentially adjust estimation procedures accordingly. In our case study, lacking a formal mechanism, we decide to manually apply link prediction algorithms from graph theory to the estimated graphs, adding edges heuristically until their structural properties more closely resemble those of the original network. Although effective to some extent, this procedure highlights the need for a principled approach to compensate for sparsity arising from ARD concentration effects.

To mitigate this, we explored several classical link prediction algorithms widely used in network analysis, including Common Neighbors, Jaccard Coefficient, Adamic-Adar Index, Preferential Attachment, and Resource Allocation Index. These methods assign scores to unconnected node pairs based on different heuristics that estimate the likelihood of a future or missing link. Although several of them performed reasonably well, we ultimately selected the Adamic-Adar Index for its ability to produce structural metrics closer to those of the original graph, and because its weighting scheme, favoring rare shared neighbors, is well aligned and common with the dynamics of social networks.

The Adamic-Adar method assigns a score to each pair of unconnected nodes, and we added the top 1600 links with the highest scores to each estimated graph. This threshold was manually chosen based on the known ground truth (the Facebook ego network of user 0), as it most effectively reduces the structural discrepancy between the estimated and true graphs in terms of average degree.

Following this enhancement, we repeated the same evaluation procedure as before, analyzing the updated graphs at both the graph-level and the node-level using the same structural metrics. This allowed us to assess whether link prediction effectively compensates for the connectivity underestimation observed in the original ARD-based graph estimates.

The impact of adding 1600 edges using the Adamic-Adar Index can be clearly observed in the updated evaluation metrics. At the **graph-level**, the total number of edges increases significantly, as expected, and the average degree even slightly surpasses the value of the original Facebook graph. This suggests that the augmented graphs may be slightly denser than necessary, the added links appear to be distributed in a uniform manner across the graph instead of at key nodes. Clustering coefficient and transitivity (two metrics strongly associated with local cohesion) improve notably, becoming much closer to the original values, which confirms the intuition that Adamic-Adar tends to reinforce

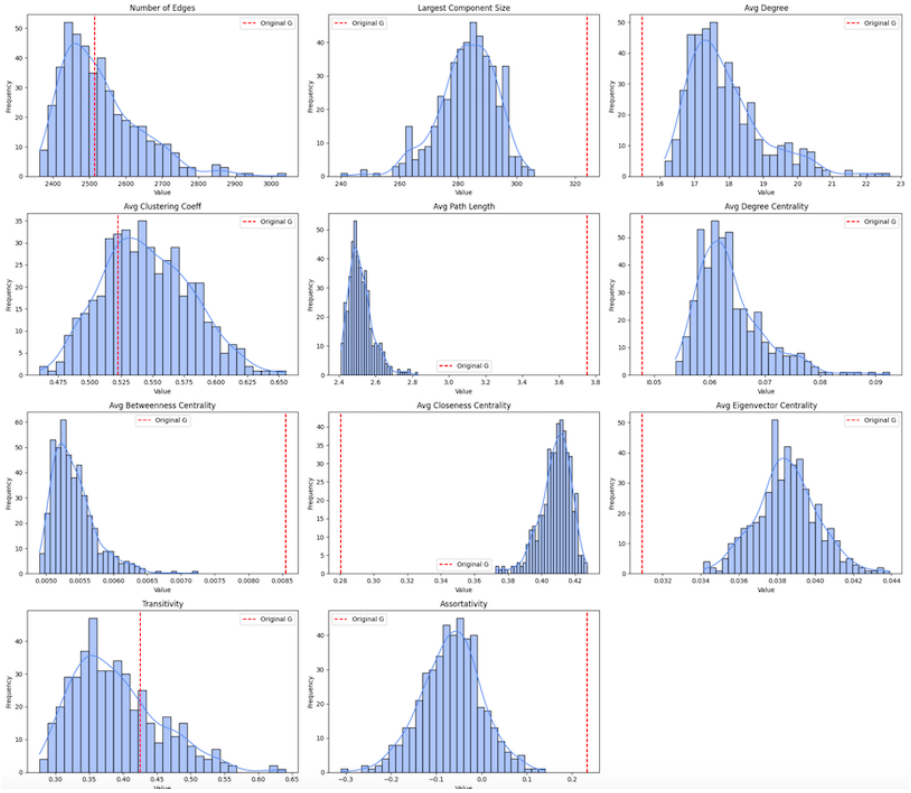


Fig. 4.8. Comparison of graph-level metrics: distribution across 450 estimated graphs with 1600 added links using Adamic-Adar Index vs. true graph values (red).

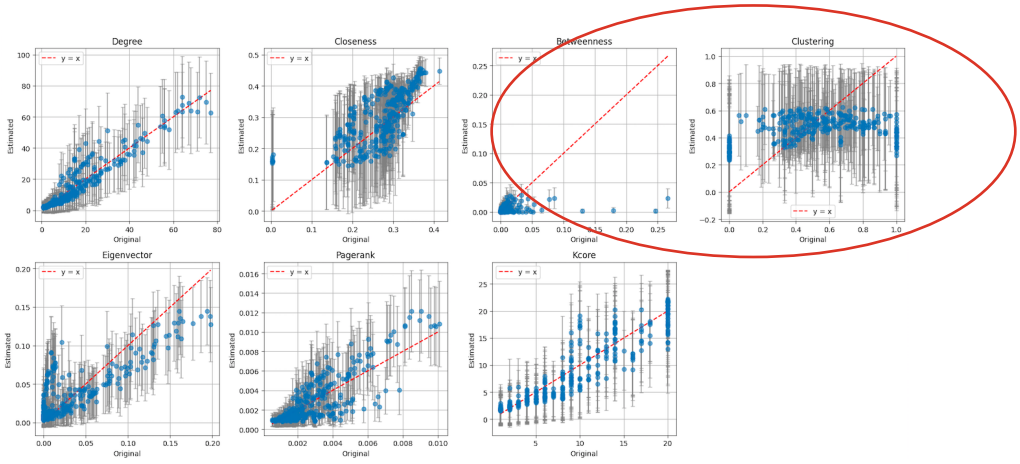


Fig. 4.9. Comparison of node-level metrics: scatter plots of true vs. estimated average node metrics with std from 450 estimated graphs with 1600 added links using Adamic-Adar Index.

local neighborhoods effectively.

However, some metrics diverge further from the reference values. For instance, the average shortest path length and average betweenness centrality both decrease, while closeness centrality increases. These shifts are expected: by adding many new connections, the graphs become more compact, reducing the average number of steps required to connect node pairs. Consequently, closeness increases and nodes are more central on average, but betweenness drops, as shortest paths are more evenly distributed across the network.

At the **node-level**, centrality metrics such as degree and Kcore index exhibit stronger alignment with the original values, with many points approaching the identity line $y = x$. Nevertheless, a slight increase in dispersion is observed, particularly in the clustering coefficient and Kcore values, suggesting that while global properties improve, fine-grained individual structures remain harder to recover. Still, the overall quality of the node embeddings benefits from the enhanced connectivity, as we will explore in the next section.

4.2.3. Application of GNN and Community Detection

To uncover community structure in the network, we apply the ~~Deep Graph Infomax~~ (DGI) model introduced in Section 3.2.1. The DGI architecture is composed of a GCN encoder (a single GCN~~Conv~~ layer with ReLU activation), a readout function that computes the mean embedding across all nodes followed by a sigmoid, and a corruption function that randomly permutes the node features. We train the model using a contrastive loss based on mutual information between the local node embeddings and the global summary vector.

Training is performed using the Adam optimizer with a learning rate of 0.001, and the model is trained for a maximum of 500 epochs. To avoid overfitting and reduce unnecessary computation, we implement early stopping with a patience of 20 epochs: if the loss does not improve for 20 consecutive epochs, training is halted and the best model so far is retained. At each epoch, the model updates its parameters by computing the contrastive loss based on the embeddings of real and corrupted nodes.

The encoder used in DGI is a single-layer GCN, named GCNConv. In our case, the number of input channels of the GCN layer corresponds to the number of input features per node, which is 224 (matching the columns of the binary metadata matrix). The number of output channels or hidden channels in the DGI model is set to 64, which defines the dimensionality of the embedding space for each node. That is, each node is ultimately represented by a 64-dimensional vector capturing both its features and structural context.

The training loop performs backpropagation at each epoch by minimizing the binary cross-entropy loss between the scores assigned to positive pairs (true nodes and graph summary) and negative pairs (corrupted nodes and the same summary vector). Every 50 epochs, the training loss is logged to monitor progress. The final model produces one embedding per node, which can be used for posterior tasks such as clustering or visualization.

Initially, we apply DGI to a single estimated graph from the ARD posterior. The results are unsatisfactory: the node embeddings exhibit poor clustering structure, and subsequent k-means clustering leads to a very large optimal number of clusters, as indicated by the within-cluster sum of squares (WCSS) criterion shown in Figure 4.10. Given that the graph has only 333 nodes, such fragmentation is unrealistic and indicative of overfitting or insufficient representation.

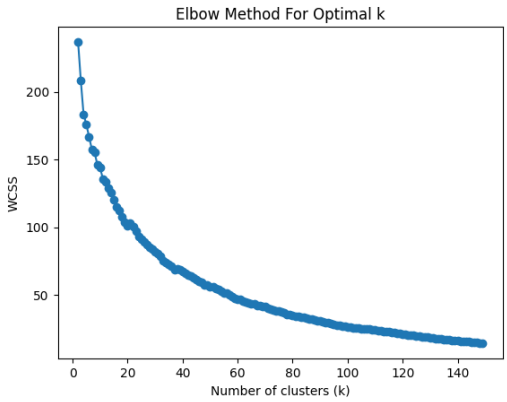


Fig. 4.10. WCSS elbow curve for a single estimated graph.

To overcome this, we apply DGI independently to each of the 450 estimated graphs. This results in 450 different embeddings for each node. We then compute the average embedding vector for each node across all runs, effectively forming an ensemble representation that is more robust to noise in individual samples. This ensemble embedding strategy significantly improves stability and performance.

With a single embedding per node, we apply k-means clustering to identify communities. Multiple values of k are tested, and the optimal number of clusters is selected based on the WCSS elbow method. Figure 4.11 shows the evolution of WCSS across number of clusters, and following the elbow method 4 clusters are selected. The ensemble embeddings lead to a much more coherent clustering structure, with a reasonable number of clusters and lower intra-cluster variance, suggesting that using all estimated graphs reflects meaningful node similarities.

To validate the robustness of the clustering solution, we also experimented with alternative algorithms: Gaussian Mixture Models (GMM) and Spectral Clustering, using the same number of clusters ($k = 4$) in each case. To quantitatively assess clustering quality, we computed the Silhouette Score for each method. This metric evaluates how well each point lies within its cluster by comparing its average distance to other points in the same cluster with its distance to points in the nearest different cluster. The score ranges from -1 to 1, with higher values indicating better-defined and more cohesive clusters. Among the tested algorithms, k-means obtained the highest Silhouette Score, so we selected the clusters defined by this method.

Finally, to visualize the embedding structure and assess the quality of the clustering, we apply t-distributed Stochastic Neighbor Embedding (t-SNE), a non-linear dimension-

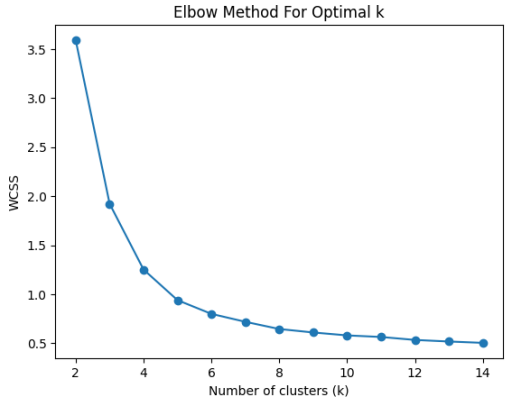


Fig. 4.11. WCSS elbow curve for the ensemble of 450 estimated graphs.

ality reduction technique that projects high-dimensional data into two dimensions. t-SNE works by preserving local similarities: points that are close together in the original space tend to remain close in the 2D projection. It does not use the clustering labels during computation, but provides a useful way to visually assess how well-separated the clusters are. The resulting projection, shown in Figure 4.12, displays the ensemble embeddings colored according to their k-means cluster assignments.

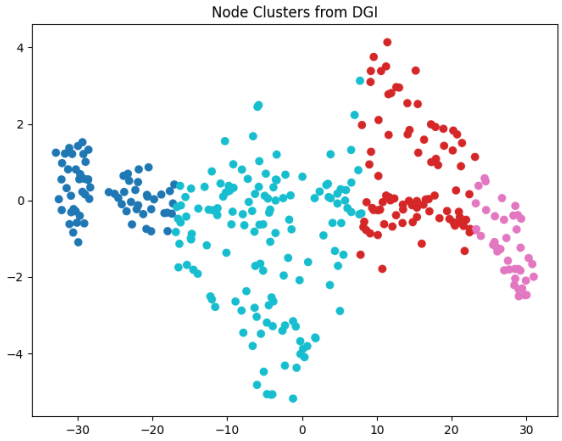


Fig. 4.12. 2D t-SNE projection of ensemble embeddings colored by k-means cluster labels (4 clusters).

4.2.4. Analysis of Communities

To understand the nature of the clusters detected by k-means, we analyze both their topological structure and their metadata characteristics.

Figure 4.13 shows the original Facebook ego graph (user 0), with nodes colored according to their assigned cluster, out of 4 total clusters. This visualization allows us to examine whether clusters correspond to identifiable topological regions within the graph. In this case, no clear spatial or neighborhood-based structure emerges: nodes from the same cluster appear scattered throughout the network, and do not form localized or densely

interconnected communities. This suggests that the clustering is not driven primarily by the graph's topology. It is worth noting that such analysis is only possible because, in our case study, we retain access to the original graph. In most real-world scenarios where only ARD responses, such visualization of community topology displayed on the real underlying graph is not possible and could be done, for example, on one of the estimated graphs. Figure 4.14 complements this view by showing the size of each cluster: Cluster 3 is the largest, containing around 140 nodes, while the others range between 40 and 80 nodes.

Fig. 4.13. Original Facebook graph (ego user 0) with nodes colored by cluster assignment (4 clusters).

We then analyze the role of node attributes in shaping the detected communities. The binary metadata matrix includes 224 features per node, representing socio-demographic characteristics such as education, gender, location, and languages. Figure 4.15 shows a histogram of the proportion of 1s across all features, revealing that most variables are extremely sparse, and the number of features active in more than 10% of nodes is very low. This high sparsity is typical for large binary metadata matrices and highlights the importance of focusing on features that are actually informative for distinguishing between clusters. Interestingly, the few features with the highest overall prevalence in the dataset are related to locale, education type and gender.

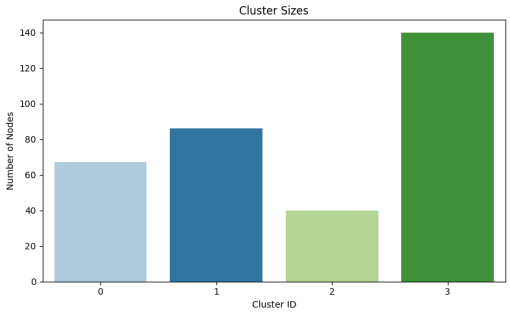


Fig. 4.14. Bar chart showing the number of nodes in each cluster.

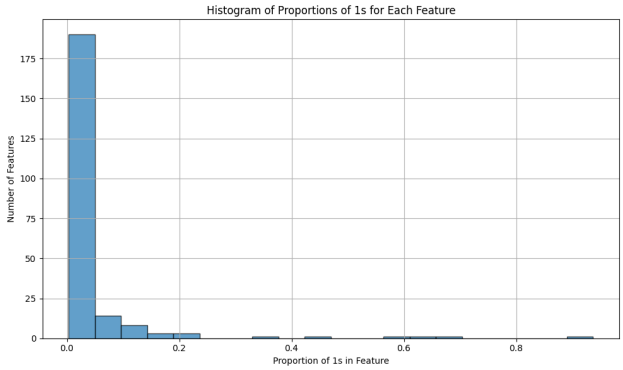


Fig. 4.15. Histogram of the proportion of 1s across all metadata binary features. Most features are extremely sparse.

To characterize each cluster, we compute the prevalence of each binary feature (i.e., the mean value, or proportion of 1s) per cluster. Figure 4.16 shows a heatmap of the prevalence of the features in each cluster. Features represented in darker colors are the most representative or active across that group. Most of them correspond to education-related variables, particularly education type, school, and graduation year. Other relevant features include gender and language. The heatmap reveals that Cluster 0 is the most distinct: it shows low prevalence across all features and lacks any dominant attributes. In contrast, Clusters 1, 2, and 3 share many of the same high-prevalence features, suggesting that they group individuals with similar socio-educational backgrounds. Among them, Clusters 1 and 2 show slightly more diversity in their metadata patterns.

To complement this analysis, we used mutual information to identify the top features that best distinguish between clusters. Mutual information is a measure of dependency between variables and is more appropriate than variance-based methods like ANOVA for binary or categorical data. Again, education-related features emerge as the most informative, confirming their central role in community structure.

Finally, Figure 4.17 displays a bar chart of all features with prevalence above 40% in each cluster, providing a more intuitive individual cluster-specific summary of which characteristics dominate within each group. Cluster 0 once again appears as the least defined, with fewer prevalent features which are also not informative. The remaining clusters share several key attributes with high frequency, consistent with earlier findings.

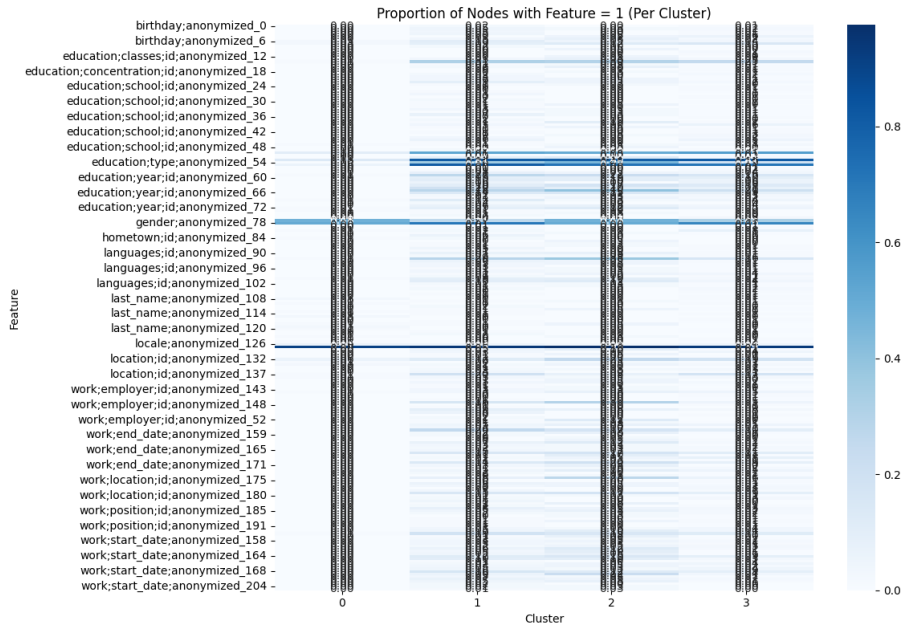


Fig. 4.16. Heatmap of prevalence of features in each cluster. Education-related features dominate the profile.

While these features, such as certain education types or gender, are also the most common in the dataset overall, they still play a discriminative role in differentiating clusters, as confirmed by the mutual information analysis and the cluster-specific distribution shown in Figure 4.16.

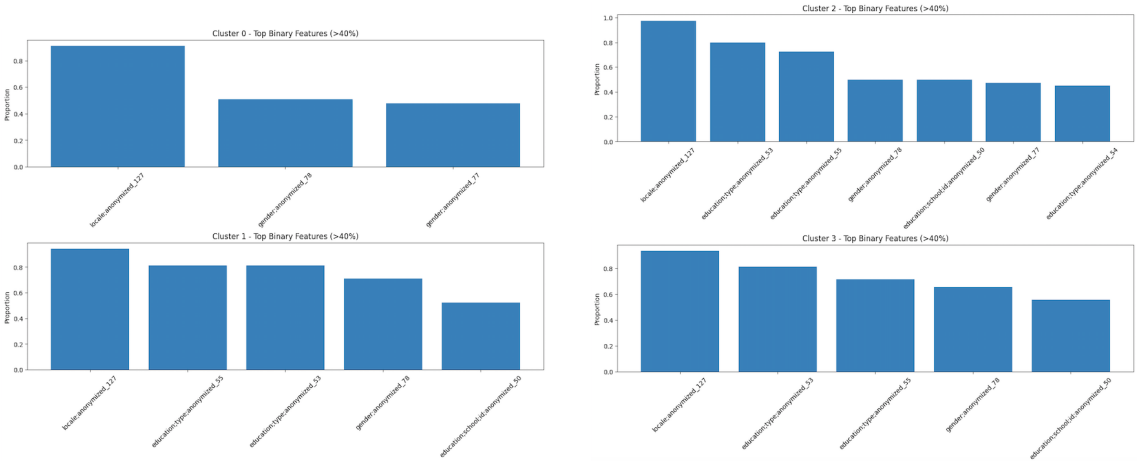


Fig. 4.17. Bar charts of features with prevalence >40% in each cluster, showing cluster-wise attribute distributions.

In summary, the detected communities appear to be driven primarily by socio-demographic similarity, especially in terms of education. While the graph topology may contribute weakly to the embeddings, the dominant signal is derived from the node metadata. The resulting clusters therefore reflect socio-educational groupings rather than purely structural communities, providing an interpretable outcome grounded in user characteristics.

4.2.5. Comparison with Ground Truth

Evaluating the quality of community detection is inherently challenging, particularly in unsupervised settings. In our context, this challenge is further complicated by the fact that the input graph used for clustering is not the original Facebook graph but an estimated version reconstructed from ARD data, and the network estimation step also introduces structural variation.

Nonetheless, we take advantage of the fact that we have access to the original Facebook ego network, including the user-defined social circles, to perform a comparative analysis. Although clustering algorithms are not expected to reproduce these predefined groupings, such a comparison provides valuable insight into the nature of the recovered communities. Specifically, we first attempt to study the correspondence between the 4 clusters obtained through DGI and k-means, and the original Facebook circles used to simulate the ARD responses.

It is important to emphasize that unsupervised clustering is performed without access to ground truth labels. The algorithm seeks to identify patterns based solely on data structure, in this case, node embeddings learned from metadata and local estimated graph context. A good clustering does not necessarily align with predefined groups; instead, it may reveal alternative, and potentially meaningful, partitions. Furthermore, in our specific case, the original dataset allowed nodes to belong to multiple circles simultaneously, whereas the clustering algorithm assigns each node to a single group. This fundamental mismatch limits the degree of alignment we can expect between circles and clusters.

We begin by analyzing how the 7 original Facebook circles are distributed across the 4 clusters obtained in our earlier ensemble-based k-means clustering. Figure 4.18 shows, for each cluster, the proportion of nodes that originally belonged to each circle (including overlapping circle memberships). As observed earlier in Figures 4.13 and 4.14, there is no clear visual indication that the clusters correspond to the original circles. The bar chart confirms this impression, although there are some conclusions we can draw. Nodes that belonged to multiple circles (denoted as `circlesoverlap`) are spread relatively evenly across all clusters, which is consistent with their ambiguous group identity. Nodes without clear circle assignment or belonging to very small circles (grouped as `circlesNA`) are also distributed throughout the clusters, but with a higher concentration in Cluster 0 (the least informative group in terms of metadata features), which is also consistent with their lack of identity. Interestingly, `circle15`, the largest circle in the dataset, is represented across all clusters but appears most concentrated in Cluster 2, which, despite being the smallest in size, may be the most representative of this circle. Other circles are more uniformly spread, with the exception of `circle16`, which is only found in Clusters 1 and 3.

To investigate further the possible relationship with ground social circles truth, we repeat the clustering procedure but now force k-means to produce exactly 7 clusters, matching the number of original Facebook circles. Although the WCSS analysis on ensemble

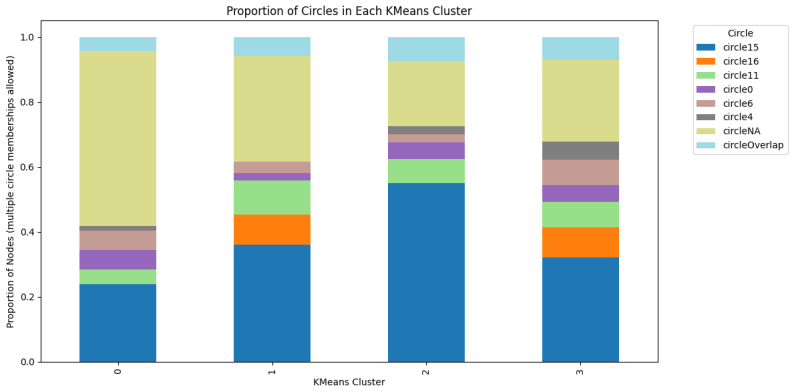


Fig. 4.18. Distribution of original circle memberships across the 4 clusters.

embeddings had previously suggested an optimal number closer to 4, this new configuration allows for a more direct comparison with the predefined groups.

Figure 4.20 shows the original Facebook graph with nodes colored according to their new cluster assignments. Visually, there is no clear alignment between clusters and topological regions, similar to our previous findings. There is also no clear alignment with the distribution of circles across the graph, seen in Figure 4.4. The cluster size distribution (Figure 4.19) also differs substantially from the ground truth circles, which were notably imbalanced: two large circles with more than 100 members and the rest with fewer than 40, as shown in Table 4.1.

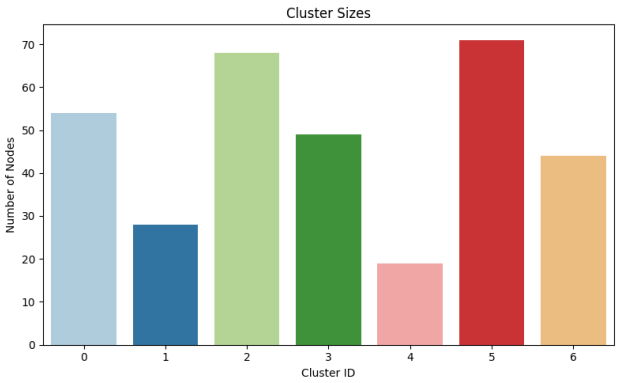


Fig. 4.19. Number of nodes in each of the 7 clusters.

We also examine how the original circles are distributed across these 7 clusters. Figure 4.21 presents a stacked bar chart for each cluster, showing the proportion of nodes belonging to each original circle. While there are hints of concentration for certain groups, such as `circle16` being more prominent in Cluster 5, the overall distribution remains diffuse. Most circles appear across several clusters, and no clear one-to-one correspondence emerges. This is expected, particularly as the number of clusters increases, making patterns harder to interpret.

This discrepancy likely arises from the different types of information used to construct each grouping. The original circles were manually defined by the ego user and likely re-

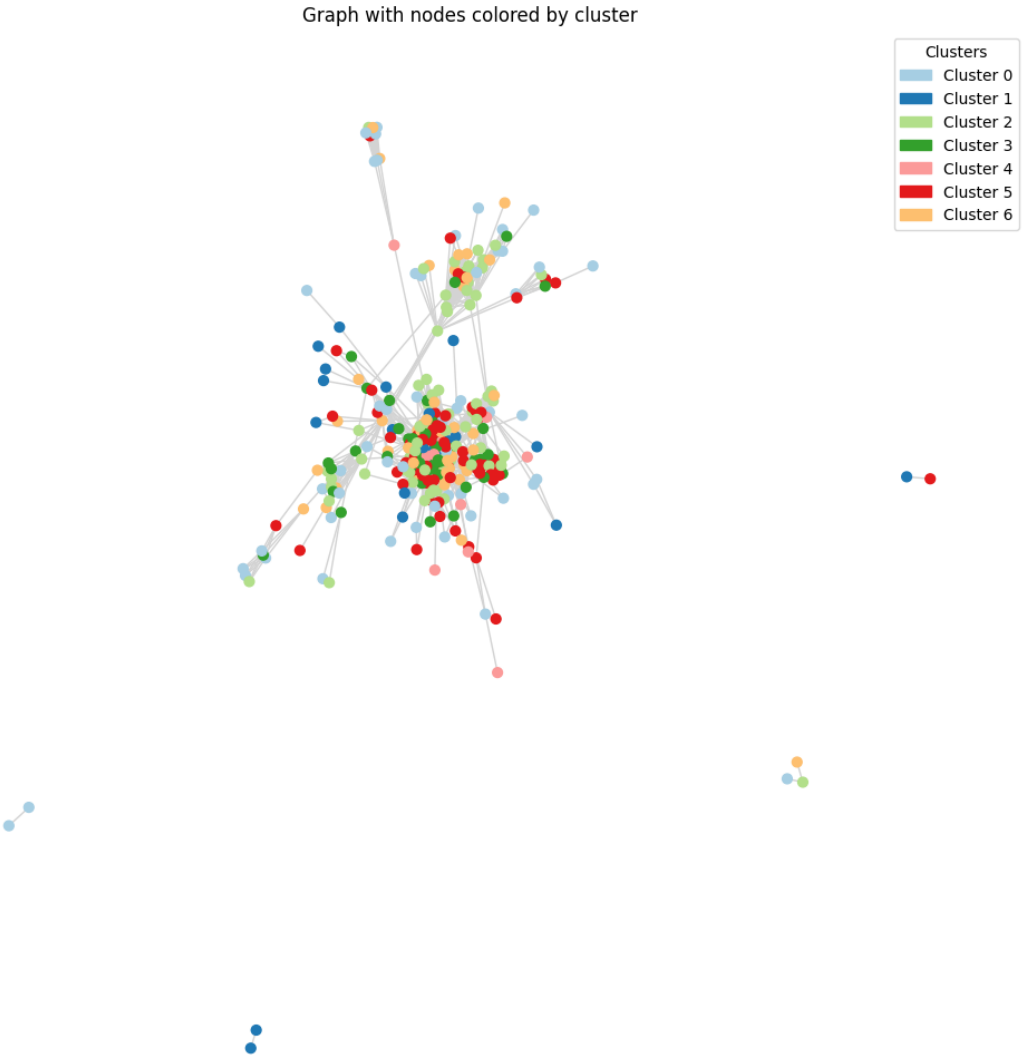


Fig. 4.20. Original Facebook graph with nodes colored by the forced 7 clusters.

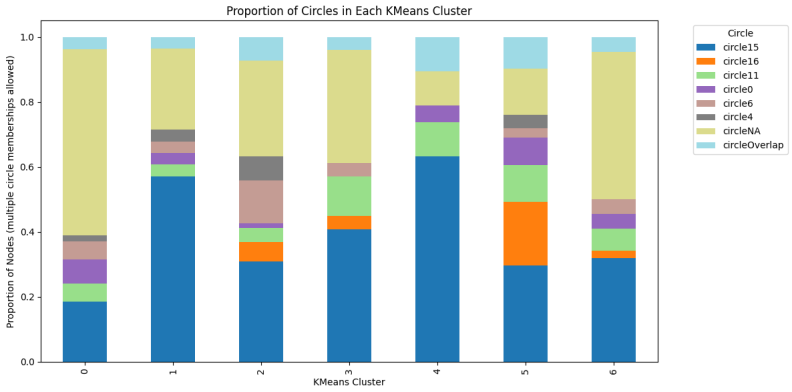


Fig. 4.21. Distribution of original circle membership across the 7 forced clusters.

flect explicit social ties and interactions, thus more closely aligned with graph topology. In contrast, the clusters learned via DGI and k-means are influenced primarily by node metadata, as encoded in the learned embeddings. These embeddings capture latent structural patterns and attribute similarities, but not necessarily the explicit connections or social interpretations used to define circles.

This highlights a fundamental distinction between structure-based and attribute-based clustering. Our method leverages node features and local graph context encoded in the embeddings, rather than the full topology and information of the original graph. As such, the learned communities reflect socio-demographic similarity rather than social circles as originally defined, but they remain valid and meaningful representations of the latent organization of the network.

5. CONCLUSION AND FUTURE WORK

In this project, we have developed and tested a complete pipeline for latent network estimation and community detection from ~~Aggregated Relational Data~~ (ARD). Starting from real-world data extracted from Facebook social circles, we simulated ARD matrices and applied the ARD-based Bayesian model proposed by McCormick et al. to infer the underlying social network. Our analysis revealed that the estimated graphs tend to systematically underestimate connectivity compared to the original network, a pattern observed both visually and through quantitative metrics. To address this, we applied link prediction techniques to enhance graph structure, followed by the use of ~~Deep Graph Infomax~~ (DGI) and node embeddings to uncover community structure. Although the clusters recovered did not align with the original Facebook circles, they revealed socio-demographic groupings primarily shaped by metadata features such as education. Overall, the results illustrate the value of combining statistical estimation, graph-theoretic augmentation, and GNN-based representation learning to reconstruct and analyze networks from partial information.

An important direction for future research is to better understand the relationship between group concentration parameters and graph sparsity in ARD-based estimation. As shown in our simulation studies, high group concentration tends to reduce the informativeness of the ARD matrix, leading to underconnected and fragmented graphs. A formal approach to measure this influence would allow practitioners to anticipate estimation bias and make principled decisions about graph correction. One possible line of research would be to derive a theoretical mapping between group concentration and connectivity loss, using synthetic simulations to calibrate this relationship. Then, in real ARD scenarios, one could project individuals and group centers onto the latent hypersphere and estimate the group concentration parameters. With this information, it would be possible to determine whether link augmentation is necessary and, if so, how many edges to add. This approach could provide a more grounded alternative to heuristic link prediction, enhancing both the interpretability and reliability of latent graph reconstruction.

BIBLIOGRAPHY

[1] M. E. J. Newman, *Networks: an introduction*. Oxford; New York: Oxford University Press, 2010. [Online]. Available: http://www.amazon.com/Networks-An-Introduction-Mark-Newman/dp/0199206651/ref=sr_1_5?ie=UTF8&qid=1352896678&sr=8-5&keywords=complex+networks.

[2] T. H. McCormick and T. Z. and, "Latent surface models for networks using aggregated relational data," *Journal of the American Statistical Association*, vol. 110, no. 512, pp. 1684–1695, 2015. DOI: 10.1080/01621459.2014.991395. eprint: https://doi.org/10.1080/01621459.2014.991395. [Online]. Available: https://doi.org/10.1080/01621459.2014.991395.

[3] D. S. Baum and P. V. Marsden, "Uses and limitations of dichotomous aggregate relational data," *Social Networks*, vol. 74, pp. 42–61, 2023. DOI: https://doi.org/10.1016/j.socnet.2023.02.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378873323000102.

[4] E. Breza, A. G. Chandrasekhar, T. H. McCormick, and M. Pan, *Using aggregated relational data to feasibly identify network structure without network data*, 2018. arXiv: 1703.04157 [stat.ME]. [Online]. Available: https://arxiv.org/abs/1703.04157.

[5] A. V. Banerjee, A. G. Chandrasekhar, E. Duflo, and M. O. Jackson, *Gossip: Identifying central individuals in a social network*. National Bureau of Economic Research Cambridge, MA, USA: 2014.

[6] A. Fernández-Anta, J. Aguilar, J. M. Ramírez, R. E. Lillo, and S. Díaz-Aranda, "Coronasurveys: Encuestas indirectas en línea para monitorizar la evolución del covid- 19," *Revista Española de Comunicación en Salud*, pp. 15–23, ene. 2024. DOI: 10.20318/recs.2024.7876. [Online]. Available: https://e-revistas.uc3m.es/index.php/RECS/article/view/7876.

[7] S. Sra, "Directional statistics in machine learning: A brief review," *arXiv preprint arXiv:1605.00316*, 2016.

[8] I. S. Dhillon and S. Sra, "Modeling data using directional distributions," Citeseer, Tech. Rep., 2003.

[9] C. Karras, A. Karras, M. Avlonitis, and S. Sioutas, "An overview of mcmc methods: From theory to applications," in *IFIP international conference on artificial intelligence applications and innovations*, Springer, 2022, pp. 319–332.

[10] Z. Liu and J. Zhou, *Introduction to graph neural networks*. Springer Nature, 2022.

[11] W. L. Hamilton, *Graph representation learning*. Morgan & Claypool Publishers, 2020.

[12] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

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

[14] Y. Ma and J. Tang, *Deep learning on graphs*. Cambridge University Press, 2021.

[15] L. Wu, P. Cui, J. Pei, L. Zhao, and X. Guo, "Graph neural networks: Foundation, frontiers and applications," in *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, 2022, pp. 4840–4841.

[16] J. Zhou *et al.*, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[17] M. Kim, J. Lee, and J. Kim, "Gmr-net: Gcn-based mesh refinement framework for elliptic pde problems," *Engineering with Computers*, vol. 39, no. 5, pp. 3721–3737, 2023.

[18] P. Veličković *et al.*, *Deep graph infomax*, 2018. arXiv: 1809.10341 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1809.10341.

[19] R. D. Hjelm *et al.*, "Learning deep representations by mutual information estimation and maximization," *arXiv preprint arXiv:1808.06670*, 2018.

[20] J. Leskovec and J. Mcauley, "Learning to discover social circles in ego networks," *Advances in neural information processing systems*, vol. 25, 2012.