Data Engineering 424 Miniproject

Author: Ethan John Joule
Student Number: 24838578

Declaration of Graduate Attributes (GA)

This report and the work presented herein are my own. No external assistance was received unless explicitly stated below.

ECSA Graduate Attributes Satisfied

- **GA 1: Problem Solving**
  The design, implementation, and evaluation of the probabilistic graphical model (PGM) for community detection demonstrate the identification, formulation, and analysis of a complex engineering problem. The solution creatively adapts Loopy Belief Propagation (LBP) and symmetry-breaking techniques to address convergence challenges, as detailed in the Model Design section. This section contains the formulation of the solution to the problem as well as how smaller issues like the symmetry and convergence were addressed.

- **GA 2: Application of Scientific and Engineering Knowledge**
  The solution applies PGM theory (e.g., Stochastic Block Models, cluster graph construction) and computational methods (e.g., inference algorithms, parameter estimation) from the module to solve the community detection problem. This is primary documented in the Implementation section, where the the algorithms and mathematical derivations applied in the code are explained and expanded on. The results of what has been implemented is displayed in the Results section, where our model makes predictions on real-life data.

Attestation

I confirm that the evidence for the above GAs is fully contained in this report and that the work complies with the academic integrity requirements of DE424.

Introduction

This report presents a probabilistic graphical model approach for community detection in networks, using stochastic block models to analyze both synthetic and real-world network data. The methodology combines theoretical foundations with practical implementation, developing a framework that processes adjacency matrices to identify latent community structures through probabilistic inference. Key aspects include model design, parameter estimation techniques, and evaluation using Zachary's karate club dataset, demonstrating how PGMs can effectively uncover network communities while handling uncertainty. The

implementation bridges theoretical concepts with applied network analysis, offering insights into community detection performance through both quantitative metrics and qualitative visualization.

Background

Network Graphs and Adjacency Matrices

A network graph represents entities (nodes) and their interactions (edges). The structure of the network is encoded using an *adjacency matrix*, which is an $N \times N$ matrix where each entry $A_{i,j}$ indicates whether an edge exists between nodes i and j. For undirected graphs, the adjacency matrix is symmetric ($A_{i,j} = A_{j,i}$).

Communities in Networks

A community is a subset of nodes within a network that are more densely connected among themselves than with nodes outside the subset. Identifying communities helps in analyzing relationships in social networks, citation networks, and biological systems. Community detection, also called graph clustering, groups nodes based on their connectivity patterns.

Stochastic Block Models (SBMs)

A Stochastic Block Model (SBM) is a probabilistic model used to generate graphs with community structures. It assigns nodes to communities and defines edge probabilities based on whether nodes belong to the same community (w_s) or different communities (w_d). SBMs serve both as synthetic data generators for testing algorithms and as the foundation for model-based community detection methods.

Core Assumptions

- **Undirected Graph:** Edges have no direction, meaning connections are reciprocal.
- **No Self-loops:** Nodes cannot be connected to themselves.
- **Simple Graph:** No multiple edges exist between the same pair of nodes.

Model Design

Definition

We define the following terms and variables used in the community detection problem:

- n – Total number of nodes in the graph.
- R – Number of communities in the graph.

- $z_i \in \{1, 2, \ldots, R\}$ – The community assignment of node i.
- $A \in \{0, 1\}^{n \times n}$ – The adjacency matrix representing the graph structure:
  - $A_{ij} = 1$ if there is an edge between nodes i and j.
  - $A_{ij} = 0$ otherwise.
- w_s – The probability of an edge between two nodes in the **same** community.
- w_d – The probability of an edge between two nodes in **different** communities.
- $\mathbf{z} = (z_1, z_2, \ldots, z_n)$ – Vector of community assignments for all nodes.

We assume the following:

- The graph is undirected: $A_{ij} = A_{ji}$.
- The graph contains no self-loops: $A_{ii} = 0$ for all i.
- The graph is simple: No multiple edges between the same pair of nodes.

Bayesian Network Representation

The Stochastic Block Model (SBM) can be represented as a Bayesian Network or more generally, a Probabilistic Graphical Model (PGM), which captures the dependencies among the variables in the system.

Each node i in the network has an associated hidden variable z_i representing its community assignment. The observed data is the adjacency matrix A, where A_{ij} indicates the presence or absence of an edge between nodes i and j. The generative process is defined by:

- Sampling each z_i from a prior distribution $\phi(z_i)$, typically uniform over R communities.
- For each pair of nodes (i, j), sampling A_{ij} based on z_i and z_j with probability:

$$P(A_{ij} = 1 \mid z_i, z_j) = \begin{cases} w_s & \text{if } z_i = z_j \\ w_d & \text{if } z_i \neq z_j \end{cases}$$

A simple diagram of the Bayesian Network is as follows:

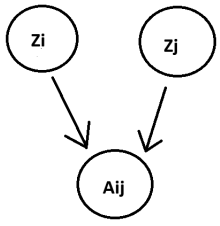

Figure 1: Bayesian network.

This bayesian network for every possible combination of nodes in the system.

Factorization

The joint distribution over all variables factorizes as:

$$P(\mathbf{z}, A) = \prod_i \phi(z_i) \prod_{i<j} \phi(z_i, z_j, A_{ij})$$

where:

- $\phi(z_i)$ is the unary factor encoding the prior distribution over communities.
- $\phi(z_i, z_j, A_{ij})$ is the pairwise factor encoding the likelihood of observing an edge given the community assignments.

Factor Tables

The specific values for the factors are given in the tables below.

Table 1: Unary Factor $\phi(z_i)$

z_i	$\phi(z_i)$
1	$\frac{1}{R}$
2	$\frac{1}{R}$
$\vdots$	$\vdots$
R	$\frac{1}{R}$

Table 2: Pairwise Factor $\phi(z_i, z_j, A_{ij})$

z_i	z_j	$A_{ij} = 1$	$A_{ij} = 0$
r	r	w_s	$1 - w_s$
r	r' $(r \neq r')$	w_d	$1 - w_d$

These factor tables are used in the Loopy Belief Propagation (LBP) algorithm to iteratively compute approximate marginal distributions over the hidden variables $\{z_i\}$, enabling both parameter estimation and community prediction.

Cluster Graph

The cluster graph for this solution contains all the pairwise factors being connected together and sending messages that contain shared variables. The cluster graph below is an example cluster graph for a sample with only 3 nodes.

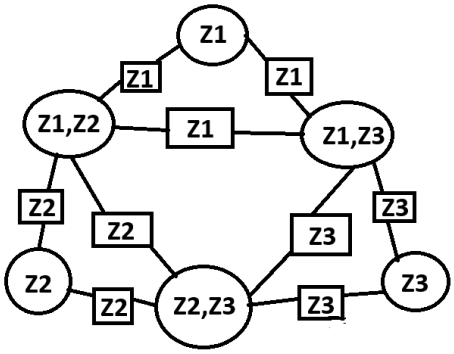

Figure 2: Cluster Graph.

The cluster graph contains all the possible pairwise and unary factors. They factors are then connected to factors that share the same variables. The messages sent between these connected factors consists of the shared variables.

Implementation

Inference Method

To perform inference on the probabilistic graphical model designed for community detection, we used **Loopy Belief Propagation (LBP)**, an iterative message-passing algorithm for approximate inference in graphical models with cycles. Given that the underlying model—derived from the **Stochastic Block Model (SBM)**—contains numerous loops due to the dense interconnections typical in network graphs, LBP was a well-suited approach.

The implementation leveraged the `ClusterGraph` and `lbp_cg` classes, simplifying the process by allowing iterative definition of unary and pairwise factors before passing them into a `ClusterGraph` object, which automatically handled the necessary connections.

In practice, **damping** was essential to ensure message convergence, particularly in graphs with dense connectivity or closely competing community structures. By combining new messages with previous ones via a weighted average, damping reduced oscillations in message updates, significantly improving numerical stability.

Noise and Damping

Symmetry Breaking via Noise: In the Stochastic Block Model (SBM), community labels are inherently symmetric—the identities of communities (e.g., "Community 0" vs. "Community 1") are arbitrary and interchangeable. This symmetry can cause inference algorithms like Loopy Belief Propagation (LBP) to stall, as initial uniform priors over community assignments lead to perfectly balanced beliefs with no gradient for differentiation. To break this symmetry, small noise ($\epsilon \sim 10^{-5}$) was added to the initial categorical distributions of node-community assignments. This perturbation preserves the near-uniformity of priors

while providing a slight bias that allows the inference to resolve communities effectively. The noise magnitude was chosen to be negligible in the final results but sufficient to disrupt symmetry during early iterations.

Estimating Parameters

w_s and w_d

For the cases where the parameters w_s and w_d are not given I have used the EM algorithm to estimate these values. The algorithm slightly differs from its standard notation however it functioned adequately. The EM algorithm consists of two steps. The first step is the E-step, where beliefs were obtained over the nodes namely Z_i. The second step, the M-step, consists of recalculating w_s and w_d. The formulas used in the M-step are shown below:

Once we have obtained new values for w_s and w_d:

1. **Return to the E-step:** Use the newly updated w_s and w_d to re-estimate the posterior probabilities (or "beliefs") of the hidden variables — in this case, the community assignments of each node. This is done using Loopy Belief Propagation (LBP) on the cluster graphs, incorporating the updated parameters into the pairwise factors.
2. **Check for convergence:** After each full E-M iteration, check whether the change in parameters w_s and w_d is below a small threshold ϵ. If this condition is satisfied, the algorithm is considered to have converged. If not, we repeat the process. For practical purposes, we impose a maximum of 20 iterations to prevent infinite loops.
3. **Final Prediction:** Once EM converges, we use the final node beliefs to assign each node to the community with the highest marginal probability.

Expectation-Maximisationn Derivation

Let the observed data be the adjacency matrix $A \in \{0,1\}^{N\times N}$, where $A_{ij} = 1$ indicates an edge between nodes i and j. The latent variables are $z_i \in \{1,\dots,R\}$ indicating the community assignment of node i. We assume:

$$P(A_{ij} \mid z_i, z_j) = \begin{cases} w_s, & \text{if } z_i = z_j \\ w_d, & \text{if } z_i \neq z_j \end{cases}$$

The goal is to maximize the marginal likelihood:

$$\log P(A \mid w_s, w_d) = \log \sum_{\mathbf{z}} P(A \mid \mathbf{z}; w_s, w_d) P(\mathbf{z})$$

Since this summation is intractable, we use the EM algorithm.

E-step We compute the expected complete log-likelihood using the posterior $Q(\mathbf{z}) = P(\mathbf{z} \mid A; w_s, w_d)$, approximated via Loopy Belief Propagation:

$$E_Q[\log P(A, \mathbf{z} \mid w_s, w_d)] = \sum_{i<j} \sum_{r,s} Q_{ij}(r, s) \log P(A_{ij} \mid z_i = r, z_j = s)$$

Assuming a factorized approximation:

$$Q_{ij}(r, s) \approx P(z_i = r) \cdot P(z_j = s)$$

M-step We update w_s and w_d by maximizing the expected log-likelihood.

First, compute the probability that nodes i and j belong to the same community:

$$P_{ij}^{\text{same}} = \sum_{r=1}^{R} P(z_i = r) \cdot P(z_j = r)$$

Then:

$$w_s = \frac{\sum_{i<j} A_{ij} \cdot P_{ij}^{\text{same}}}{\sum_{i<j} P_{ij}^{\text{same}}} \qquad w_d = \frac{\sum_{i<j} A_{ij} \cdot (1 - P_{ij}^{\text{same}})}{\sum_{i<j} (1 - P_{ij}^{\text{same}})}$$

These updates maximize the expected log-likelihood under the current beliefs.

Repeat E-step and M-step until convergence.

R

In the case where R was not provided, an alternative method was implemented to estimate R, since the EM algorithm alone is insufficient without a known number of communities. This was treated as a model selection problem, where the strategy is to evaluate a set of candidate values for R, and for each:

1. **Fix a candidate value of** R (e.g., $R = 2, 3, 4, 5, 6$).
2. **Construct a probabilistic graphical model (PGM)** using this R. This affects:
   - The domain size of each hidden variable (node community assignment).
   - The structure of the pairwise factors used in the model.
3. **Estimate the parameters** w_s **and** w_d **using the EM algorithm:**
   - Run the E-step using Loopy Belief Propagation to compute node beliefs.
   - Run the M-step to re-estimate w_s and w_d from these beliefs.
   - Iterate until convergence or until a maximum number of iterations is reached.
4. **Compute the log-likelihood** of the observed data given the current model and estimated beliefs.
5. **Select the best** R as the one that achieves the highest log-likelihood.

Optionally, model selection criteria such as AIC or BIC could be used to penalize model complexity. However, these were not implemented in this solution.

Log-Likelihood Formula

The log-likelihood of the data given the model can be expressed as:

$$\log P(\mathcal{D} \mid w_s, w_d, R) = \sum_{\text{edges } (i,j)} \log \left[\sum_{z_i, z_j} P(z_i) P(z_j) \phi_{ij}(z_i, z_j; w_s, w_d) \right]$$

Software Design and Engineering

Parameter flexibility

The code supports both manual configuration and automatic estimation of key parameters. This flexibility is achieved through a centralized configuration system defined at the top of the main script.

- The parameter **N** need not be specified in any cases. The program determines how many nodes there are from the adjacency matrix provided.
- Users can manually specify values in the `Config` section. They have two options:
  - Provide explicit values for R, w_s, and w_d.
  - Leave one or more of these parameters set to `-1`, which will trigger automatic estimation.
- Clear inline comments guide the user on how to configure these options.
- Any parameters not explicitly configured by the user will be estimated automatically by the program.

This design ensures ease of use and flexibility, allowing both novice users and advanced practitioners to interact with the system according to their needs.

Results

The evaluation consists of both quantitative and qualitative assessments:

Explanation of Output

Quantitative Evaluation

- Reports the final estimated parameters:
  - Number of communities (R)
  - Intra-community edge probability (w_s)
  - Inter-community edge probability (w_d)

- Compares predicted vs. ground truth communities for each node
- Computes accuracy metric with the following caveat: The reported accuracy may be affected by label switching - identical clusterings with permuted community labels will show artificially low accuracy. The numerical values should be interpreted in conjunction with the qualitative visualization.

Qualitative Evaluation

- Exports community predictions to `community_predictions.txt`
- Generates visualization via Python script (`visualize_communities.py`) showing:
  - Node layout preserving graph structure
  - Color-coded community assignments
  - Comparison with ground truth (when available)
- Output saved as `community_comparision.png` which compares the visualised community assignments of the model and the given ground truths

Network 1 Results

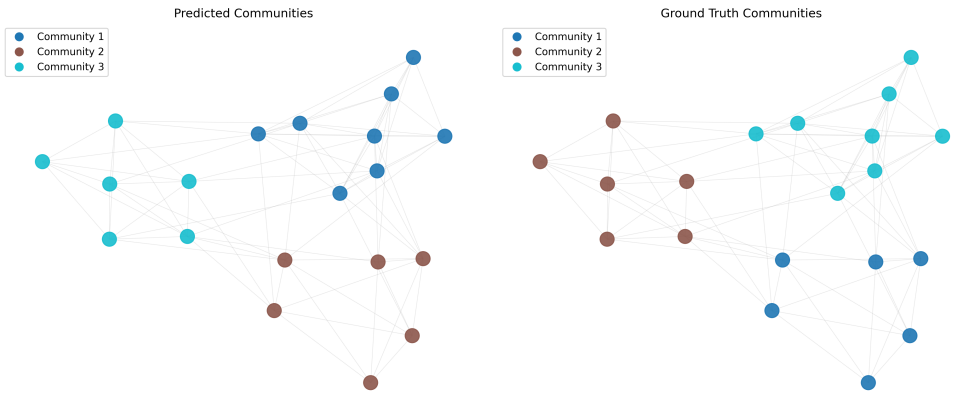

Figure 3: Qualitative results from A1 text

As can be seen in the above figure, the model predicted the communities of the nodes with an accuracy of 100%.

Network 2 Results

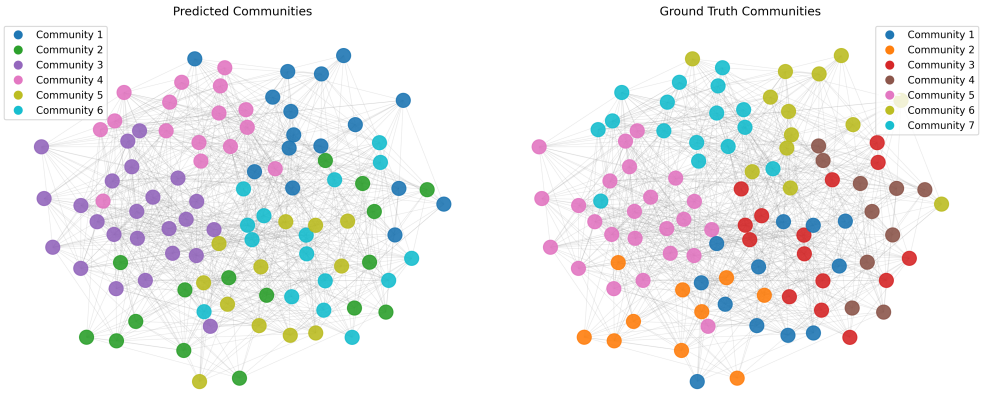

Figure 4: Qualitative results from A2 text

The program ran a bit longer than for Network 1 due to the large number of nodes, however the model still performed well with around 90% percent accuracy.

Network 3 Results

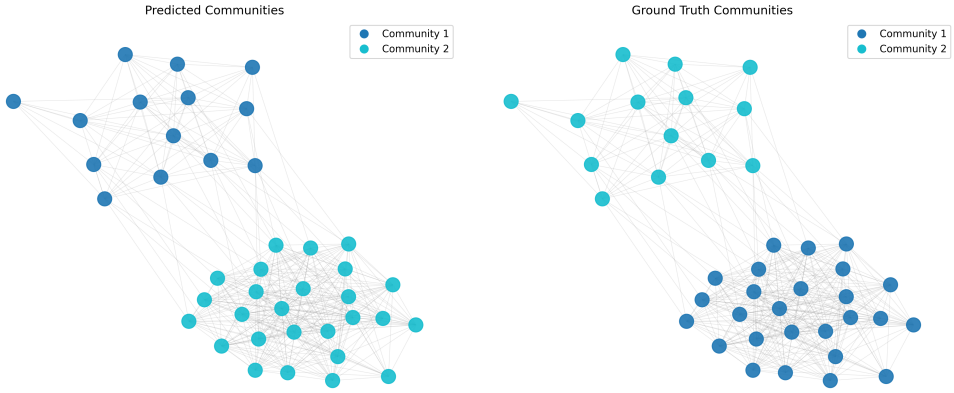

Figure 5: Qualitative results from A3 text

The model performed very well on Network 3, with 100% accuracy. The estimated values for w_s and w_d were 0.78125 and 0.0741758 respectively.

Network 4 Results

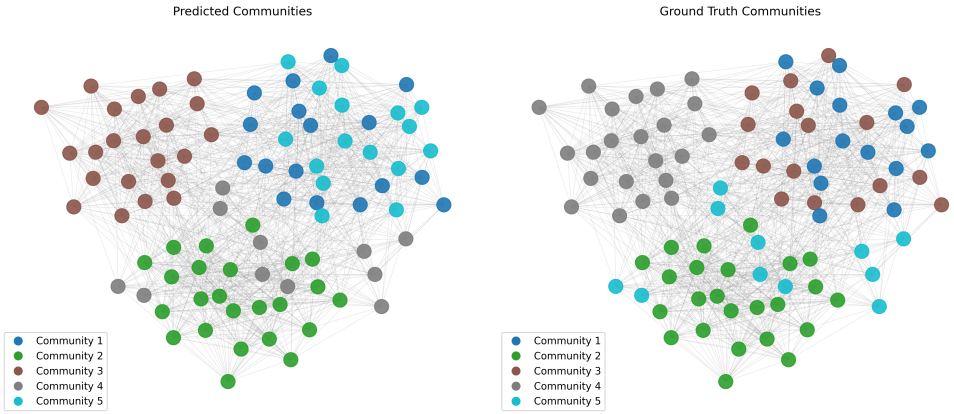

Figure 6: Qualitative results from A4 text

The model performed very well on Network 4, with a very high accuracy. The estimated values for w_s and w_d were 0.845966 and 0.151239 respectively.

Network 5 Results

The model did not perform well on Network 5. The loopy belief propagation did not converge and the program took about an hour to run. No results were obtained.

NOTE: The accuracy displayed by the quantitative evaluations for all the tests may be misleading due to label switching i.e my model refers to community 1 as 2 and vice versa.This metric should be supplemented with the qualitative evaluation.