Graph Neural Networks vs. Non-Learning-Based Methods for Community Detection Problem

Ben Dabush (316014760)
Arad Ben Menashe (207083353)
Naama Shiponi (318607314)

July 27, 2024

GitHub Repository

Abstract

Complex networks are ubiquitous in modern information systems, inherently containing community structures. These communities are sets of nodes that are densely connected internally and sparsely connected with the rest of the network. Community detection, the task of revealing these inherent structures, is crucial for understanding and analyzing complex networks.

This study compares traditional non-learning-based methods with modern machine learning approaches for community detection. We investigate label propagation and spectral clustering as non-learning-based methods, and focus on Graph Neural Networks (GNNs) for learning-based approaches. Our research implements and evaluates these methods on synthetic datasets generated using the Stochastic Block Model (SBM) to assess their effectiveness and efficiency in community detection.

We demonstrate that under specific conditions, our GNN model can outperform the spectral clustering algorithm. Our key contribution is a comprehensive comparison table[link to table here] indicating the optimal scenarios for each method's application. This study provides insights into the strengths and limitations of both traditional and modern approaches to community detection in complex networks.

1 Conclusion

Our study on community detection reveals distinct strengths and weaknesses of Graph Neural Networks (GNNs) compared to non-learning-based methods. GNNs demonstrate superior adaptability, effectively handling various graph structures and multiple communities. They excel particularly in dense graphs and complex network structures where traditional methods may falter. The learning-based nature of GNNs allows them to capture subtle patterns in the data that might be missed by predefined algorithms.

Non-learning-based methods, on the other hand, offer advantages in interpretability and often require less computational resources. They perform well on sparse graphs and in scenarios where the community structure aligns with their underlying assumptions. These

Table 1: GNNs vs. Non-Learning-Based Methods for Community Detection

Aspect	**Graph Neural Networks**	**Non-Learning-Based Methods**
Adaptability	Can adapt to various graph structures and learn complex patterns	Rely on predefined algorithms and may struggle with unconventional structures
Scalability	Generally scale well with large datasets, especially sparse graphs	May face scalability issues with very large networks (e.g., Spectral Clustering)
Performance on Dense Graphs	Excel in dense graph structures	May lose efficiency or accuracy as graph density increases
Performance on Sparse Graphs	Some models (e.g., LGNN) perform well on sparse graphs	Often perform well, especially methods like Label Propagation
Multiple Communities	Handle multiple communities effectively	Performance may degrade with increasing number of communities
Interpretability	Often considered "black box" models	Usually more interpretable due to their algorithmic nature
Computational Resources	Require significant computational power for training	Generally less computationally intensive, but may have high complexity for some algorithms
Prior Knowledge Requirement	Can learn from data without extensive domain knowledge	Often require some level of prior understanding of network structure

methods can be particularly effective when the network exhibits clear, well-defined communities.

The choice between GNNs and non-learning-based methods ultimately depends on the specific characteristics of the network, available computational resources, and the need for interpretability. For networks with complex or ambiguous community structures, GNNs provide a powerful tool. However, for networks with clear community divisions or when computational resources are limited, traditional non-learning-based methods can be equally effective and more efficient.

Our research suggests that the future of community detection may lie in hybrid approaches that combine the strengths of both paradigms. Such approaches could leverage the adaptability and pattern recognition capabilities of GNNs while incorporating the efficiency and interpretability of non-learning-based methods, potentially offering more robust solutions across a wider range of network scenarios.

2 Conclusion

Our study on community detection methods reveals that both learning-based and non-learning-based approaches have their strengths and ideal use cases. Graph Neural Networks

Table 2: Comparison of Community Detection Methods

Method	**Strengths**	**Weaknesses**	**Best Use Cases**
Label Propagation	• Simple and efficient • Linear time complexity	Less accurate for complex community structures	• Large sparse networks • When computational speed is crucial
Spectral Clustering	• Theoretically well-founded • Effective for various graph structures	High computational complexity ($O(n^3)$)	• Small to medium-sized networks • When accuracy is more important than speed
GCN	• Learns complex node representations • Effective for dense graphs	• Requires training data • May struggle with very sparse graphs	• Dense networks • When node features are available
LGNN	• Captures edge-level information • Effective for sparse graphs and multiple communities	• More complex implementation • May struggle with very dense graphs	• Sparse networks • Networks with many communities

(GNNs), particularly GCN and LGNN, show promise in handling complex network structures and multiple communities. GCN excels in dense graphs, while LGNN performs well in

sparse networks with many communities. Among non-learning methods, Spectral Clustering demonstrates high accuracy across various graph structures but faces scalability issues for large networks. Label Propagation, while less accurate for complex structures, offers efficiency for large sparse networks.

The choice between these methods depends on the specific network characteristics, computational resources, and the balance needed between accuracy and speed. Our results suggest that for networks with clear community structures, traditional methods like Spectral Clustering can be as effective as GNNs. However, for more complex or ambiguous community structures, GNNs provide a valuable tool, especially when node or edge features are available.

Future work could explore hybrid approaches that combine the strengths of both learning-based and non-learning-based methods to further improve community detection in diverse network scenarios.

3 Introduction

3.1 Community Detection in Graphs: Definition and Historical Context

Community detection in graphs is a fundamental problem in network analysis, with applications ranging from social networks to biological systems. It involves identifying groups of nodes more densely connected internally than with the rest of the network, often when true community labels are unknown [1]. This problem, crucial for understanding network structure and optimizing information flow, traces back to Rice's work in 1927 on social contact matrices but gained significant attention in the late 20th century with the advent of large-scale network data [2].

Approaches to community detection can be categorized into non-learning-based (model-based) methods and learning-based methods. Traditional model-based methods rely on simple mathematical models hand-designed from domain knowledge. These include Label Propagation, inspired by Zachary's study [3], and Spectral Clustering, which utilizes the graph Laplacian's eigenvalues [4]. Model-based methods carry out community detection based on underlying model knowledge, relying on simplifying assumptions that make the process tractable and computationally efficient. However, these simplifications often fail to capture nuances of high-dimensional complex data and dynamic variations.

3.2 Model-Based Methods vs. Learning-Based Approaches

The emergence of Graph Neural Networks (GNNs) has revolutionized learning-based approaches to community detection. The seminal work by Scarselli et al. in 2009 introduced the GNN model, extending neural network methods to process data represented in graph domains [5]. This model can handle various graph types (acyclic, cyclic, directed, and undirected) and maps a graph and its nodes into a Euclidean space, providing a foundation for subsequent developments in the field.

Building on this groundwork, recent advancements in GNNs have led to powerful tools capable of aggregating local and global structural information [6]. For instance, the work of Kipf and Welling on Graph Convolutional Networks (GCNs) has further enhanced the capabilities of GNNs in semi-supervised learning tasks [7]. In our study, we implement GNNs using modern libraries like PyTorch Geometric and explore advanced architectures such as the Line Graph Neural Network (LGNN) [8] using the Deep Graph Library (DGL).

While GNNs offer advantages in processing graph data and recovering complex features, they face challenges such as high computational burden, need for large datasets, and lack of interpretability [9]. Their model-agnostic nature results in complex detection rules, potentially less robust to topology changes. Conversely, model-based methods offer interpretability and efficiency but may oversimplify complex relationships.

The limitations associated with both model-based methods and black-box deep learning systems necessitate a comprehensive comparative analysis. Our study aims to evaluate the effectiveness and efficiency of traditional non-learning-based methods and modern machine learning approaches across various network structures, providing insights into their respective strengths and limitations in community detection tasks.

4 Problem setup

Many real-world graph datasets are composed of community structures. We focus on comparing different community detection methods, particularly emphasizing the comparison between model-based and deep learning approaches for graphs. In Subsection 1, we define the mathematical structures required to understand the community detection problem. In Subsection 2, we briefly present the methods we compare. In Subsection 3, we present the comparison criteria we have used, and finally, in Subsection 4, we present the dataset we have used.

4.1 Problem formulation

We consider the an undirected unweighted graph, $G = (V, E)$, consists of n nodes $V = \{v_1, v_2, \ldots, v_n\}$, m edges $E = e_{ij} \subseteq V \times V$. The topological structure of G can be defined by an $n \times n$ adjacency matrix $A = (a_{ij})_{n \times n}$, where $a_{ij} = 1$ if $e_{ij} \in E$, or 0, otherwise.

Definition 1: Community. The network G contains k communities $C = \{C_1, C_2, \ldots, C_k\}$, where C_i is a subgraph of G and the nodes within C_i are densely connected whereas the nodes across C_i and C_j are sparsely connected. The communities are non-overlapping when $C_i \cap C_j = \emptyset, \forall i, j$.

Definition 2: Community Detection. Given a network G, community detection is to design a mapping F to assign every node v_i of G into at least one of the k communities, i.e., to label v_i at least one community identity $c_i \in \{C_1, C_2, \ldots, C_k\}$. Equivalently, the problem is to derive a community assignment of nodes $C = (c_1, c_2, \ldots, c_n)$.

4.2 Methods

Approaches to solve the community detection problem can be categorized into model-based methods and deep learning methods. In our experiment, we use common known model-based methods: Label Propagation, an hierarchical clustering technique inspired by Zachary's study [3], and Spectral Clustering, which utilizes the graph Laplacian's eigenvalues to partition networks [4].

Among learning-based methods, Graph Neural Networks (GNNs) have emerged as powerful tools capable of aggregating local and global structural information [5]. In this paper, we build a GNN using modern libraries like PyTorch Geometric. We also implement the

Line Graph Neural Network (LGNN) created by Chen, Li, and Bruna in their paper "Supervised Community Detection with Line Graph Neural Networks" [6], using the Deep Graph Library (DGL), a library for deep learning on graphs.

4.3 Accuracy and loss calculation

Community detection models may correctly identify structures without matching arbitrary ground truth labels. To address this, we employ the Hungarian algorithm [7] as our primary evaluation metric. This algorithm solves the linear assignment problem, finding the optimal one-to-one mapping between predicted and true community labels while minimizing overall loss. Our evaluation process:

1. Apply the Hungarian algorithm to determine the best label mapping.
2. Align predicted labels with ground truth using this mapping.
3. Calculate accuracy or loss based on aligned predictions.

This approach ensures fair comparison between different community detection methods by focusing on structural accuracy rather than arbitrary label assignments.

4.4 Stochastic Block Model

When building the dataset for our project, we employed the Stochastic Block Model (SBM), a widely used probabilistic framework for generating synthetic graphs with community structures. SBM is particularly useful in modeling networks where nodes are organized into distinct groups or communities, and the likelihood of connections between nodes depends on their community affiliations.

In the SBM framework, the graph is partitioned into C blocks, each representing a distinct community. Two key probabilities govern edge formation: p, the probability of an edge between nodes within the same community, and q, the probability of an edge between nodes in different communities. Typically, $p > q$, reflecting denser intra-community connections compared to sparser inter-community connections.

Formally, the probability of an edge existing between nodes i and j is defined as:

$$P(a_{ij} = 1) = \begin{cases} p & \text{if } c_i = c_j \text{ (same community)} \\ q & \text{if } c_i \neq c_j \text{ (different communities)} \end{cases} \tag{1}$$

where a_{ij} is the adjacency matrix element indicating the presence of an edge, and c_i and c_j are the community assignments of nodes i and j, respectively.

By using SBM to generate our dataset, we were able to create a controlled environment that reflects various community structures, enabling us to evaluate and compare the performance of different community detection methods, including graph neural networks and non-learning-based approaches.

5 Algorithms

5.1 Traditional Algorithms

5.1.1 Label Propagation

An algorithm that has gained popularity for its efficiency, conceptual clarity, and ease of implementation is the Label Propagation Algorithm (LPA) [8]. This algorithm, inspired by epidemic spreading, detects a community by the iterative propagation of node labels until convergence (nodes with the same label constitute a community).

An advantage of the LPA is its asymptotic worst-case complexity: only $O(n + m)$ steps are needed per iteration, where n is the number of nodes and m the number of edges. If a constant number of T iterations are considered, the total time complexity of the algorithm is $O(T(n + m))$, quadratic with respect to the number of nodes and linear with respect to the number of edges, as $m \in O(n^2)$ in the worst case. For sparse graphs with few edges ($m \in O(n)$) and a constant T, LPA can be considered a linear-time algorithm. Raghavan et al. [8] report that after only five iterations, the labels of 95% of the nodes have converged to their final values.

5.1.2 Spectral clustering

Spectral clustering [9] is a popular and straightforward algorithm for clustering that consists of two main steps. First, it involves performing eigenvalue decomposition of either the adjacency matrix or the Laplacian matrix. Next, the k-means algorithm is applied to partition these points into k clusters, where k represents the number of communities or clusters we aim to identify. The k-means algorithm works by iteratively assigning points to clusters and updating the cluster centroids until convergence.

One significant advantage of spectral clustering is its solid theoretical foundation, which is rooted in linear algebra and spectral graph theory. This foundation makes the method not only intuitive but also highly effective in revealing the intrinsic community structure of the graph. However, it is important to note that full eigenvalue decomposition in the first step generally requires $O(n^3)$ time, where n denotes the number of nodes. This can be computationally demanding when n becomes large.

5.2 Graph Neural Networks Models

5.2.1 GCN Model

For our learning-based approach, we implemented a Graph Convolutional Network (GCN) using PyTorch Geometric. The GCN model is designed to learn node representations by aggregating information from neighboring nodes, making it well-suited for community detection tasks. Our GCN architecture consists of three layers:

1. An input GCN layer that takes the node features and applies a graph convolution operation.

2. A hidden GCN layer that further processes the node representations.

3. An output GCN layer that produces the final node classifications.

Between each layer, we apply a ReLU activation function and dropout regularization to prevent overfitting. The final layer's output is passed through a softmax function to obtain class probabilities. The mathematical formulation of our GCN model can be described as follows:

$$H^{(0)} = X \tag{2}$$

$$H^{(1)} = \sigma(f_1(H^{(0)}, A)) \tag{3}$$

$$H^{(2)} = \sigma(f_2(H^{(1)}, A)) \tag{4}$$

$$Y = \text{softmax}(f_3(H^{(2)}, A)) \tag{5}$$

Where:

- X is the input feature matrix
- A is the adjacency matrix
- $H^{(l)}$ is the node representation at layer l
- f_l is the graph convolution operation at layer l
- σ is the ReLU activation function

The graph convolution operation f_l is defined as:

$$f_l(H, A) = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}HW_l \tag{6}$$

Where:

- $\hat{A} = A + I_N$ is the adjacency matrix with self-loops
- $\hat{D}$ is the degree matrix of $\hat{A}$
- W_l is the learnable weight matrix for layer l

The dropout operation is applied after each ReLU activation:

$$H^{(l)} = \text{dropout}(\sigma(f_l(H^{(l-1)}, A))) \tag{7}$$

This GCN architecture allows the model to learn hierarchical node representations by aggregating information from local neighborhoods, making it effective for community detection in graph-structured data.

5.2.2 LGNN Model

The LGNN as shown in the article [10] is constructed by first using a standard GNN framework, which operates on local graph operators such as the adjacency matrix, degree matrix, and identity matrix to update node state vectors. The LGNN extends this by creating a line graph L(G), representing edge adjacencies of the original graph G. The line graph's vertices are the directed edges of G, and a non-backtracking operator B is used to ensure directed information flow.

In the LGNN architecture, a GNN is defined on L(G), with B and its degree matrix D_B acting as adjacency and degree matrices. Edge and node features are updated using edge indicator matrices, which enable communication between them. The LGNN leverages the non-backtracking operator to construct and propagate oriented edge features through the graph, capturing the local oriented structure and improving performance in tasks like community detection. This approach allows the model to learn and predict complex structures in graphs effectively.

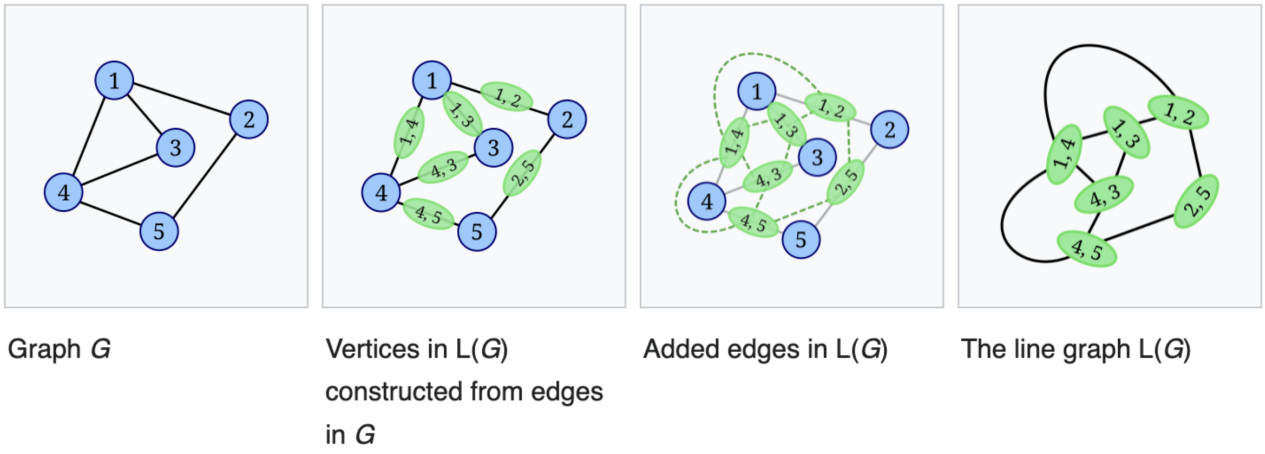

Figure 1: How a line graph is created

6 Results

6.1 First experiment

p	q	LABEL PROPAGATION	SPECTRAL CLUSTERING	GCN	LGNN
0.9	0.1	1	1	1	1
0.8	0.2	0.9567	1	1	1
0.7	0.3	0.6753	1	1	1
0.68	0.32	0.6018	0.9999	0.9997	0.9999
0.66	0.34	0.5640	0.9990	0.9986	0.9984
0.64	0.36	0.5201	0.9958	0.9947	0.9934
0.62	0.38	0.5039	0.9864	0.9861	0.9736
0.6	0.4	0.5	0.9651	0.9541	0.8725
0.58	0.42	0.5	0.8968	0.9052	0.5
0.56	0.44	0.5	0.7317	0.7958	0.5
0.54	0.46	0.5	0.7318	0.7900	0.5
0.52	0.48	0.5	0.7317	0.7792	0.5
0.5	0.5	0.5	0.5384	0.5347	0.5

Table 3: Comparison of the different models with different densities

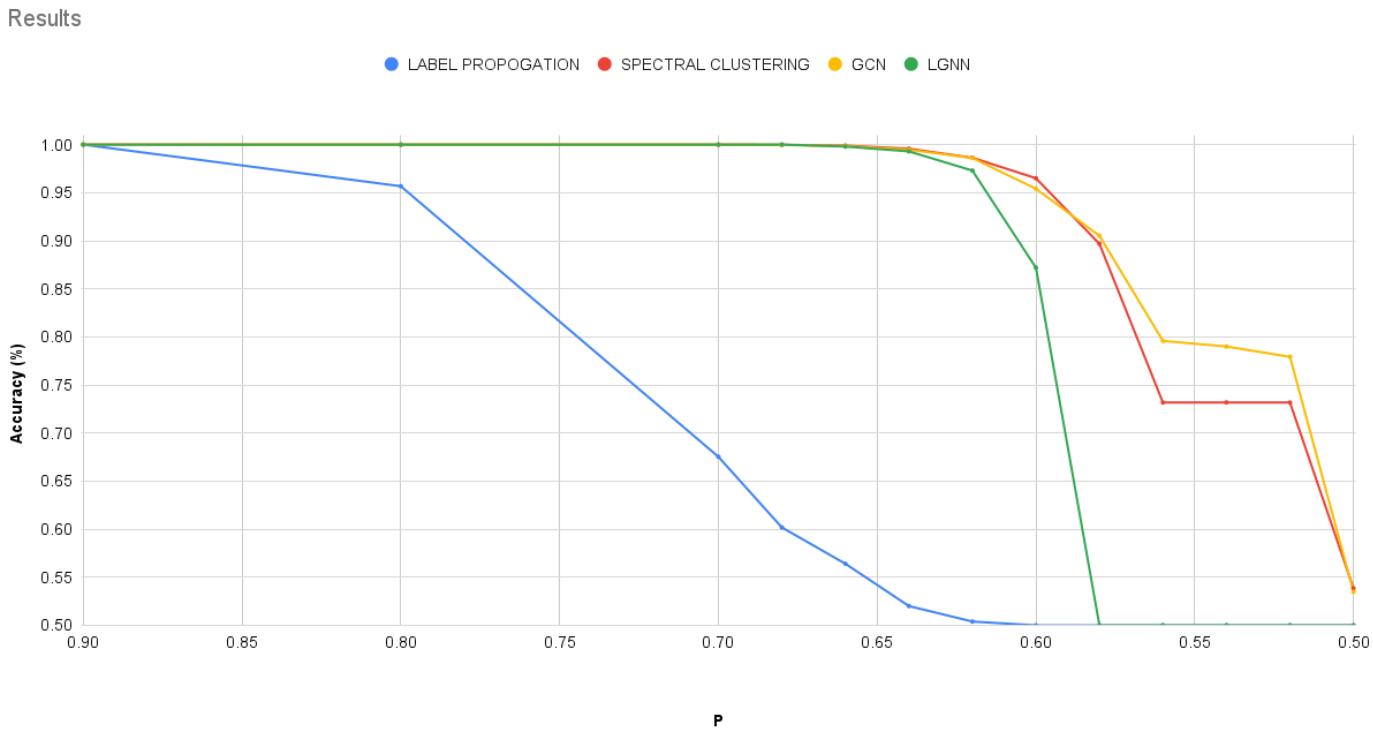

Figure 2: Comparison of the different models with different densities

6.2 Second experiment

CLASSES	LABEL PROPAGATION	SPECTRAL CLUSTERING	GCN	LGNN
2	0.9567	1	1	1
3	0.7789	1	0.754	0.921
4	0.6876	1	0.6088	0.9
5	0.5135	0.9996	0.4845	0.7623
6	0.3081	0.9971	0.2898	0.537

Table 4: Comparison between the different models when the number of communities changes

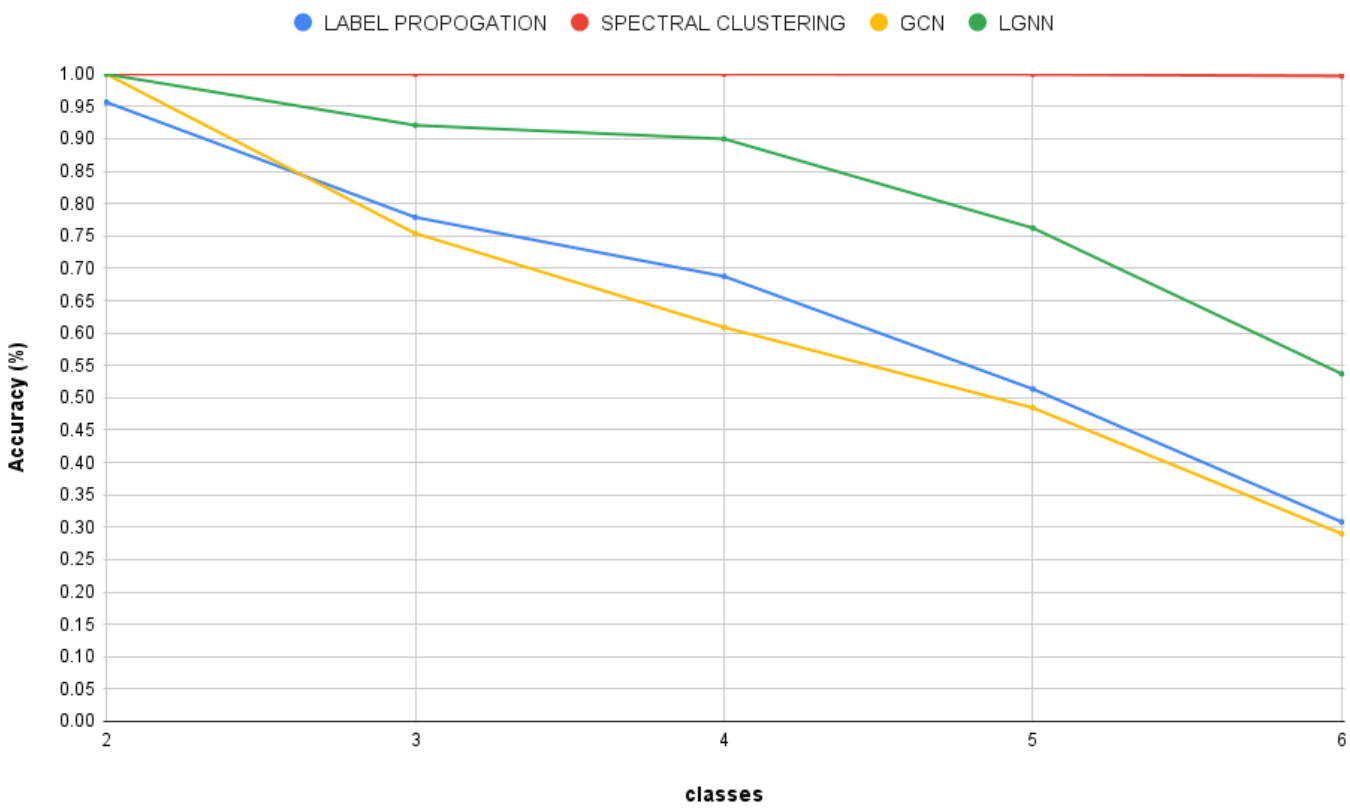

Figure 3: Comparison between the different models when the number of communities changes

7 Conclusion

References

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

[2] Stuart A Rice. The identification of blocs in small political bodies. *American Political Science Review*, 21(3):619–627, 1927.

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

[4] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14, 2001.

[5] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[6] Zhengdao Chen, Xiang Li, and Joan Bruna. Supervised community detection with line graph neural networks. *arXiv preprint arXiv:1705.08415*, 2017.

[7] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[8] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 76(3):036106, 2007.

[9] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17:395–416, 2007.

[10] Zhengdao Chen, Lisha Li, and Joan Bruna. Supervised community detection with line graph neural networks. *arXiv preprint arXiv:1901.01277*, 2019.