A Survey of GNN Based Movie Recommender Systems

Piyush Chauhan
Indiana University
Bloomington
pgchauha@iu.edu

Subhadra Mishra
Indiana University
Bloomington
sumishra@iu.edu

Divya Prasanth Paraman
Indiana University
Bloomington
dparaman@iu.edu

Abstract

Traditional recommendation systems often face data sparsity issues, limiting their ability to capture complex user-item interactions effectively. In this project, we analyse the existing graph-based algorithms that were designed to address these challenges. Following the standard procedure, we construct bipartite graphs from user-item interactions and implement Graph Neural Network (GNN) models such as Graph Convolution Matrix Completion, Light-GCN. We evaluate these models on the movie rating datasets, namely ml-100k and ml-1m since both of these datasets have side-information such as user demographic data and movie genre. We anticipate that these methods will generate more personalized movie recommendations. The models are evaluated with the help of metrics such as Precision@K and Recall@K. Our study helps enhance our perspective on the developments in this field and provide guidance for future research.

1. Introduction

Recommendation systems play an important role across various platforms in enhancing user experiences by providing personalized suggestions based on individual preferences. However, traditional recommendation systems often encounter challenges in capturing the implicit interactions between users and items. There are other issues such as data sparsity, cold-start problem and scalability. As the volume of available data continues to grow, there is a need for more robust, efficient systems to generate high-quality recommendations.

Graph-based algorithms have emerged as promising solutions for modeling complex data interactions. These techniques represent users, items as nodes and their interactions as edges, thus using a graph structure. Such type of representation helps to capturing *latent* relationships and patterns within the data. By utilising the graph structure and connectivity patterns, several graph-based approaches have proven their effectiveness by not only improving the accuracy and quality of recommendations but also overcoming the limitations of traditional methods. GNN is the popular choice for learning and in recommender systems processing data having a graph structure, resulting in substantial research [8]. It is also widely adopted for many use-cases in this field, thus we chose to particularly explore GNN-based recommender systems.

In this project, our objective is to explore the effectiveness of graph-based algorithms in collaborative filtering recommendation systems. Specifically, we focus on the stages (as described in [8]) that are generally used for implementing the user-item collaborative filtering type of recommender system. This includes constructing bipartite graphs from user-item interactions and applying Graph Neural Network (GNN) [9] models on the graph. We are limiting the scope of this study to just homogeneous (constructed) graphs. We also study the information propagation process in the neighbourhood aggregation step. GC-MC [1] uses mean-pooling as an aggregation operation, whereas Light-GCN [4] uses *degree-normalization* for giving weightage to nodes based on the structure of the graph. The information is then updated based on the aggregation information.

To evaluate the performance of the methods in consideration, we utilize movie datasets, the ones with user demographic information, namely ml-100k and ml-1m. These datasets provide side-information of users with attributes such as age, gender, occupation, and location. Using side-information such as demographic data into the recommendation model, has been advised by previous works such as [2]. It is evident that this helps in better modeling of the user preferences.

Graph Convolutional Matrix Completion" [1] is about predicting user-item interactions using a graph auto-encoder. "Light-GCN" [4] simplifies the architecture of Neural Graph Collaborative Filtering (NGCF) [7] by removing non-linear activations and linear transformations. "STAR-GCN" [11] uses a stack of GCN encoder-decoders to improve prediction performance. Additionally, we also briefly reviewed the following papers. "A General Graph-Based Framework for Top-N Recommendation Using Content, Temporal, and Trust Information" [5] combines different types of side information for top-N recommendation. "Knowledge-Enhanced Hierarchical Graph Transformer Network for Multi-Behavior Recommendation" [10] investigates multi-typed interactive patterns between users and items. Finally, "Graph-based Hybrid Recommender System" [2] uses a graph-based model with user ratings and demographic information for recommendations.

Through our study, we aim to understand the user-item collaborative filtering task in recommendation systems and how graph-based (GNN-based) algorithms are used to solve this. By analysing the working of the models considered for this study on the different stages of a recommender module, we also gain knowledge about the advantages and limitations of these techniques.

2. Methods:

We study two graph-based algorithms, namely Graph Convolutional Matrix Completion (GCMC) and Light Graph Convolutional Network (Light-GCN), for the movie recommendation task. Through the implementation of these methods, we understand how the connectivity and the structural patterns in the user-item interaction graph are used to generate personalized recommendations. The choice of these methods has been made by reviewing literature surveys [6] [8], taking into account several factors such as our domain expertise, the scope and feasibility of this project. This has also impacted our decision to evaluate the recommender models in the movies domain.

2.1. Graph Convolutional Matrix Completion (GCMC)

GCMC uses matrix completion techniques to recommend items or predict links within a bipartite user-item graph. The method begins by constructing a bipartite graph where nodes represent users and items, and edges denote observed ratings or interactions between users and items. Subsequently, a graph auto-encoder framework is implemented on this constructed graph, enabling the model to learn embeddings that capture the latent features of users and items.

The graph encoder takes the feature embeddings of size $N * D$ where D is the input feature size, along with the adjacency matrix of the graph, the movie rating matrix in this case. It generates an embedding matrix of size N*E where E denotes the size of the embeddings. The decoder, on the other hand, takes a pair of user-item embeddings from the output of the encoder and then predicts the rating in the corresponding adjacency matrix. Thus, it tries to *reconstruct* the adjacency matrix by adding the predicted ratings for the positions with no user-item interactions. At every node, the messages from the neighboring nodes are aggregated or summed together, then an activation function such as ReLU is used to get the final embedding for a user/item.

2.2. Light Graph Convolutional Network (Light-GCN)

Light-GCN [4] simplifies traditional Graph Convolutional Networks (GCNs) by removing feature transformations and nonlinear activation functions. This approach focuses on capturing higher-order connectivity within the user-item graph to learn more informative embeddings. Light-GCN employs a neighborhood aggregation mechanism, where the embedding of a node (user or item) at each layer is updated by averaging the embeddings of its neighboring nodes from the previous layer. This is mathematically represented as [4]:

$$\mathbf{e}_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(k)} \tag{1}$$

$$\mathbf{e}_i^{(k+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(k)} \tag{2}$$

where $e_u^{(k)}$, $e_i^{(k)}$ represent the user and item embedding representations after k layers of propagating information. By using this approach, Light-GCN is able to learn complex user-item relationships in the graph structure.

3. Experimental setup:

In this section, we detail the experimental setup used to evaluate the performance of LightGCN and Graph Convolutional Matrix Completion (GCMC) models on the MovieLens datasets.

3.1. Dataset

We utilized the MovieLens ml-100K [3] and ml-1m dataset [3], which contains a vast collection of movie ratings provided by users. The dataset comprises three main files: u.data, u.item, and u.user. The u.data file contains user-item interactions in the form of user IDs, movie IDs, ratings, and timestamps. Meanwhile, u.item includes information about movies, such as movie ID, title, release date, and genres. Lastly, u.user provides demographic details about users, including user ID, age, gender, occupation, and zip code.

3.2. Data Preprocessing

We performed the following preprocessing steps:

- We encoded user and movie IDs using label encoding to ensure numerical compatibility for the models.

- We split the dataset into training, validation, and test sets, with respective sizes of 80%, 10%, and 10%.

- Ratings with a threshold of 3.5 or higher were considered positive interactions, while lower ratings were considered negative.

3.3. Model Configuration

We implemented two recommendation models: LightGCN and GCMC. Both models were designed to predict user-item interactions based on graph neural networks.

LightGCN: We configured LightGCN with the following hyper-parameters:

- Number of Layers(K): The model uses a parameter 'K' for the number of layers in the LightGCN model, which is set to 3 initially and then adjusted to 1.
- Embedding Dimension: Set to 64, which indicates the size of the dimensions of the final embeddings generated.
- Dropout Rate: A dropout rate of 0.1 is defined but not explicitly used in the code.
- Learning Rate: Starts at 0.001 and is adjusted using a scheduler.
- Weight Decay: Set to 0.01 to help prevent over-fitting by penalizing large weights.
- Iterations: The main training loop runs for 10,000 iterations.
- Scheduler: An exponential decay on the learning rate with a gamma of 0.95, reducing the learning rate at every 200 iterations.

GCMC: Details of the GCMC model setup are provided in the following subsection.

- Number of Layers : The model uses 2 layers.
- Embedding Dimension: Set to 64.
- Dropout Rate: A dropout rate of 0.7 is used
- Learning Rate: Learning rate is set to 0.01.
- Iteration: The main training loop runs for 1,000 iterations.

3.4. Training Procedure

Both LightGCN and GCMC models were trained using the following procedure:

We performed training for 10,000 iterations with a batch size of 1024. We evaluated the models after every 200 iterations on the validation set. We used RMSELoss as the loss function for both models. We employed early stopping based on the validation loss to prevent overfitting.

3.5. Software and Hardware Requirements

- **Software Requirements**: The script is expected to run in an environment where libraries (torch:2.2.2 with cuda support, torch-geometric:1.0.1, pandas:2.0.3, numpy:1.25.2) are available, likely in Python 3.x.
- **Hardware Requirements**: We ran the script using the Luddy GPU servers. The script can also be run in a CPU environment.

3.6. Evaluation Metrics

For evaluation, we employed recall@k as the primary metric to measure the models' ability to recommend relevant items to users. We considered the top-k recommended items and calculated recall, which indicates the proportion of relevant items among the top-k recommendations. Additionally, precision@k was computed as a secondary metric to measure the proportion of correctly recommended items among the top-k predictions. Precision@K is defined as the measure of the items that the user interacts out of the recommended item set. It is mathematically represented with the formula [8]:

$$\textbf{Precision@K} = \frac{|R^K(u) \cap T(u)|}{K} \tag{3}$$

where, T(u) denotes the ground-truth, i.e the set of items that the user *u* has interacted with in past. $R^K(u)$ denotes the set of top-K recommendations. Similarly, Recall@K is defined as [8]:

$$\textbf{Recall@K} = \frac{|R^K(u) \cap T(u)|}{|T(u)|} \tag{4}$$

4. Results:

4.1. Results - Light-GCN:

For the experiment conducted on the Light-GCN model with a dataset size of 100k and 1m and a neighborhood size $k = 10$, the following performance metrics were obtained:

Dataset	Recall@10	Precision@10	RMSE Loss
ml-100k	0.24857	0.47359	1
ml-1m	0.24 at 4000 epoch	NA	1

Table 1. Model Performance

LightGCN implementation on the ml-1m dataset displayed a pattern in which the recall@k decreased to 0 during the training process. We believe this might be due to over-fitting of the data and including drop out layer in the architecture might lead to better results. In future, we intend to try out these methods.

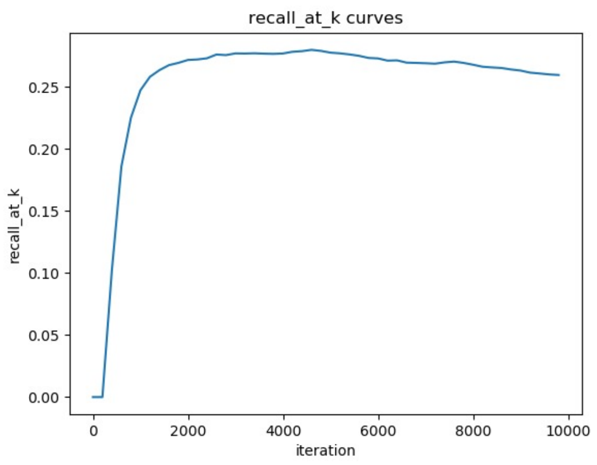

Figure 1. Recall@10 for ml-100k

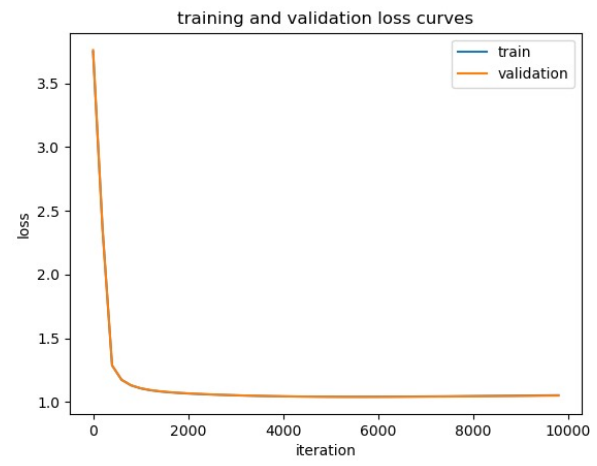

Figure 2. Loss for ml-100k

For the experiment conducted on the Light-GCN model with a dataset size of 1m and a neighborhood size $k = 10$, the following graphs were obtained:

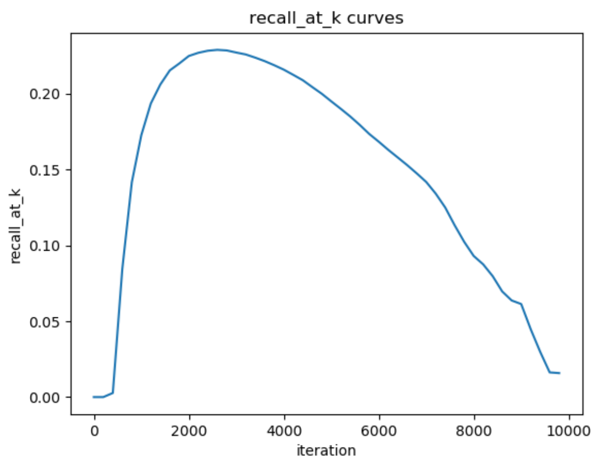

Figure 3. Recall@10 for ml-1m

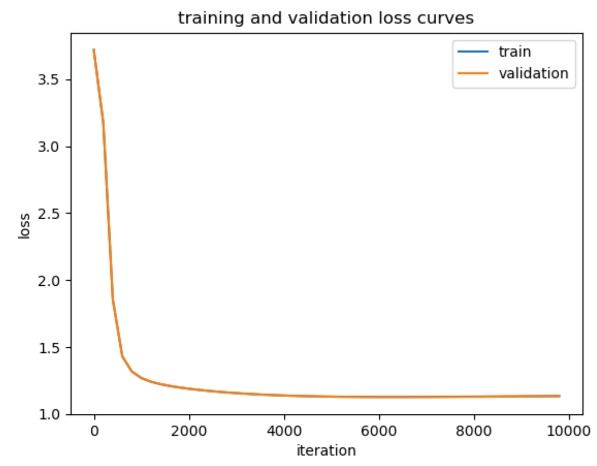

Figure 4. Loss for ml-1m

In future work, we'll work on improving the Light-GCN model's ability to recommend by adjusting how it looks at nearby items, adding more information about users and items, and using better methods for fine-tuning the model. By testing it on bigger sets of data and finding the best settings, we hope to make the recommendations even better, making users happier with the suggestions they receive.

4.2. Results - GCMC:

For the experiment conducted on the GCMC paper implementation with a dataset ml-100k, the following graph characteristics were obtained:

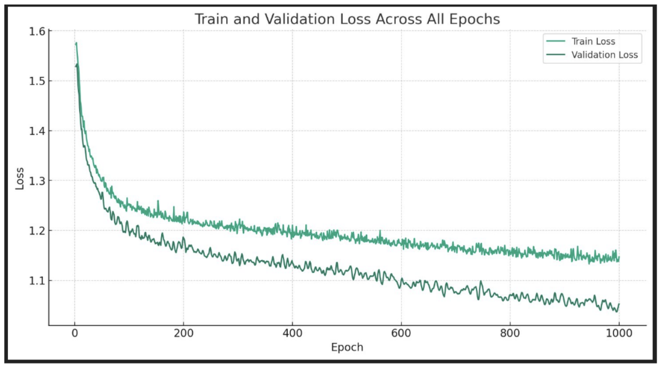

Figure 5. Loss for ml-100k

These results provide insights into the effectiveness of the graph based recommendation models under the specified conditions. The achieved recall and precision values highlight the model's ability to accurately predict relevant items within the top 10 recommendations. Additionally, the RMSE loss metric indicates the level of deviation between the predicted and actual ratings, with lower values indicating better performance in terms of rating prediction accuracy.

5. Conclusions:

In this project, we have investigated the working and performance of GNN-based techniques for the user-item collaborative type of recommendation systems on the movielens datasets ml-100k and ml-1m. Since, both of these datasets have a graph-like structure and are highly sparse, the GNN based techniques perform better, reducing the data sparsity problem. We also came across a common limitation of both of these methods. The information propagation or update process in both of the models gives higher importance the aggregated information than to the user/item's original representation. We would study the techniques solving this issue. We would like to extend our research towards exploring other graph learning approaches for recommender systems.

6. Team member contributions

Piyush reviewed the existing literature survey on Graph-based recommender systems. This helped him to formulate the project objectives and to finalize the datasets to be used considering the project constraints. He led discussions for selecting the methods to implement. He reproduced the results from the GC-MC model.
Subhadra worked on the implementation and analysis of LightGCN alongside with Divya Prasanth. Additionally, she contributed on writing both the presentation and the final report, to organise the project's findings.
Divya Prasanth helped in implementing the LightGCN and helped running the model on large datasets using the Luddy GPU servers. He also helped contribute to the ppt and the final report by generating necessary plots. Throughout the project, the team worked together closely, exchanging ideas and supporting one another in every phase of the project.

References

[1] Rianne van den Berg, Thomas N. Kipf, and Max Welling. *Graph Convolutional Matrix Completion*. 2017. DOI: 10.48550/ARXIV.1706.02263. URL: https://arxiv.org/abs/1706.02263.

[2] Zahra Zamanzadeh Darban and Mohammad Hadi Valipour. "GHRS: Graph-based Hybrid Recommendation System with Application to Movie Recommendation". In: (2021). DOI: 10.48550/ARXIV.2111.11293. URL: https://arxiv.org/abs/2111.11293.

[3] F. Maxwell Harper and Joseph A. Konstan. "The MovieLens Datasets: History and Context". In: *ACM Transactions on Interactive Intelligent Systems* 5.4 (Dec. 2015), pp. 1–19. ISSN: 2160-6463. DOI: 10.1145/2827872. URL: http://dx.doi.org/10.1145/2827872.

[4] Xiangnan He et al. *LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation*. 2020. DOI: 10.48550/ARXIV.2002.02126. URL: https://arxiv.org/abs/2002.02126.

[5] Armel Jacques Nzekon Nzeko'o, Maurice Tchuente, and Matthieu Latapy. "A general graph-based framework for top-N recommendation using content, temporal and trust information". In: (2019). DOI: 10.48550/ARXIV.1905.02681. URL: https://arxiv.org/abs/1905.02681.

[6] Shoujin Wang et al. *Graph Learning based Recommender Systems: A Review*. 2021. DOI: 10.48550/ARXIV.2105.06339. URL: https://arxiv.org/abs/2105.06339.

[7] Xiang Wang et al. "Neural Graph Collaborative Filtering". In: (2019). DOI: 10.48550/ARXIV.1905.08108. URL: https://arxiv.org/abs/1905.08108.

[8] Shiwen Wu et al. "Graph Neural Networks in Recommender Systems: A Survey". In: *ACM Computing Surveys* 55.5 (Dec. 2022), pp. 1–37. ISSN: 1557-7341. DOI: 10.1145/3535101. URL: http://dx.doi.org/10.1145/3535101.

[9] Zonghan Wu et al. "A Comprehensive Survey on Graph Neural Networks". In: (2019). DOI: 10.48550/ARXIV.1901.00596. URL: https://arxiv.org/abs/1901.00596.

[10] Lianghao Xia et al. *Knowledge-Enhanced Hierarchical Graph Transformer Network for Multi-Behavior Recommendation*. 2021. DOI: 10.48550/ARXIV.2110.04000. URL: https://arxiv.org/abs/2110.04000.

[11] Jiani Zhang et al. *STAR-GCN: Stacked and Reconstructed Graph Convolutional Networks for Recommender Systems*. 2019. DOI: 10.48550/ARXIV.1905.13129. URL: https://arxiv.org/abs/1905.13129.