Solving the Cold Start problem in Recommendation Systems - Case Study on MovieLens Dataset

Prateek Chanda (22D0362)[1], Sandarbh Yadav (22D0374)[1], Jimut Bahan Pal (22D1594)[2], and Goda Nagakalyani (21405001)[1]

[1]Department of Computer Science and Engineering, IIT Bombay
[2]Centre for Machine Intelligence and Data Science, IIT Bombay

November 2022

1 Introduction

Recommendation systems form the basis of many applications like Netflix movie recommendations, Amazon product recommendations etc. A recommendation model is built that can be used in a production setting for suitably recommending items (movies) to users and resulting in better user experience and user engagement. The project is based on a model proposed by a recent paper using Graph Convolution Neural Network (*LightGCN*). [1] A variation of original model, *LightGCN*++, is also proposed. Comparison of performance is done with traditional collaborative-based filtering like matrix factorization and state-of-the-art models like Alternative Least Squares (ALS). [2]

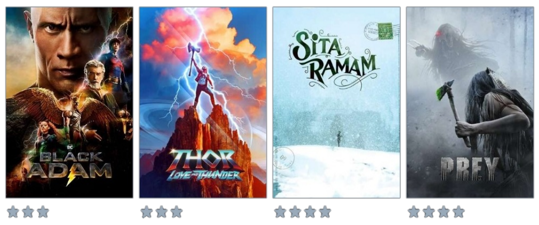


Figure 1: Recommendations are made based on the ratings provided by the users

2 Literature Review

Collaborative Filtering is a popular technique in modern recommendation systems. It parameterizes users and items as embeddings and learns the embedding parameters by reconstructing historical user-item interactions. SVD based approaches [3] use the weighted average of ID embeddings of historical items as the target user's embedding. Another type of CF methods consider historical items as the pre-existing features of a user, towards better user representations. Recent neural recommendation models like Neural Collaborative Filtering NCF [4] use the same embedding component while enhancing the interaction modeling with neural networks.

3 Motivation

Traditional methods make recommendations based on the rating history of user. However, this approach faces issues when dealing with new users. This problem of making recommendations to users without rating history is referred as cold start. Collaborative Filtering based methods which use the notion of K-nearest neighbours face problems when dealing with non rich nodes. A non rich node is one whose neighbours are quite far away from them and hence not representative of it. *LightGCN* deals with this issue by capturing the user-item interactions as a bipartite graph.

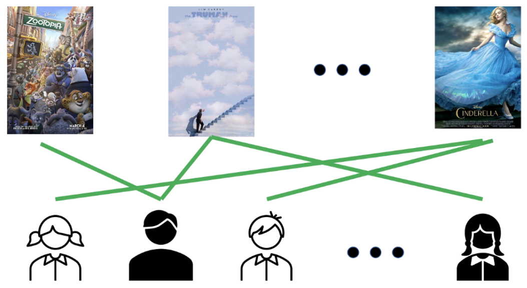
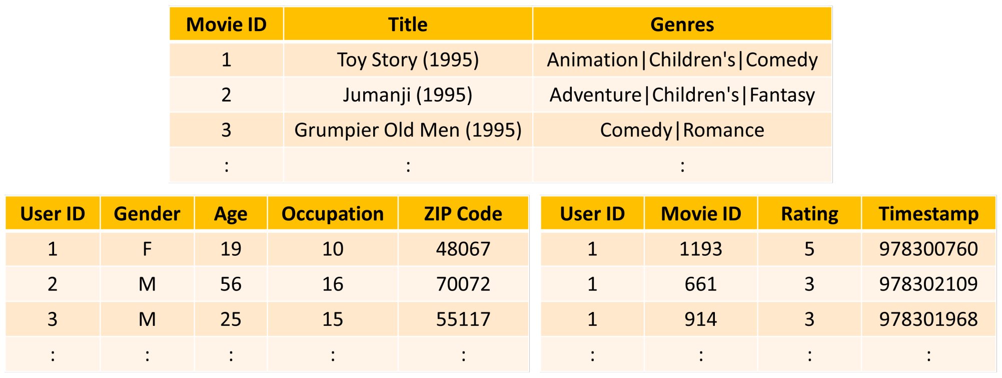


Figure 2: *LightGCN* considers user-item interaction as a bipartite graph

4 Dataset

For this project, **MovieLens** dataset is used. It contains integer movie ratings on a scale of 1 to 5 along with corresponding user features and movie features. It is commonly used as a benchmark for recommendation systems. Two variants of the dataset are used:

- Movielens 100K - It contains 100,000 ratings from 1000 users on 1700 movies. It was released in April 1998.
- Movielens 1M - It contains one million ratings from 6000 users on 4000 movies. It was released in February 2003.



Movie ID	Title	Genres
1	Toy Story (1995)	Animation|Children's|Comedy
2	Jumanji (1995)	Adventure|Children's|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
:	:	:

User ID	Gender	Age	Occupation	ZIP Code
1	F	19	10	48067
2	M	56	16	70072
3	M	25	15	55117
:	:	:	:	:

User ID	Movie ID	Rating	Timestamp
1	1193	5	978300760
1	661	3	978302109
1	914	3	978301968
:	:	:	:

Figure 3: MovieLens dataset contains data about movies, users and ratings

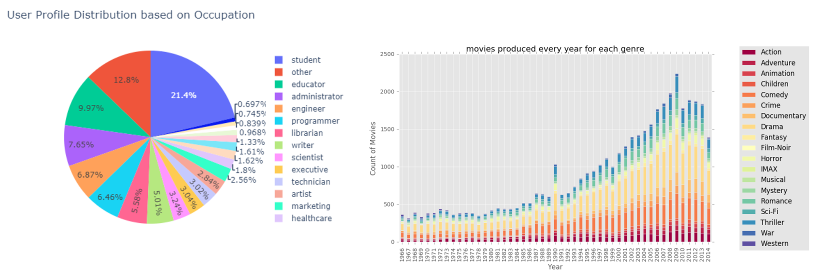


Figure 4: Distribution of users on the basis of age and movies on the basis of genres across years

5 Methodology

The graph neural network model proposed by the paper [1] is implemented. *LightGCN* captures the structural information present in the bipartite graph.

5.1 Message Aggregation for computing embeddings

Embeddings are computed via message aggregation using the following equations:

$$e_u^{k+1} = \sum_{i \in N_u} \frac{1}{\sqrt{|N_u|}\sqrt{N_i}} e_i^k \tag{1}$$

$$e_i^{k+1} = \sum_{i \in N_u} \frac{1}{\sqrt{|N_u|}\sqrt{N_i}} e_u^k \tag{2}$$

Here, e_u^k and e_i^k denote the user and movie embedding at the $k-th$ layer. $|N_u|$ and $|N_i|$ indicate the number of neighbors of user and item (movie) nodes. All in all, the model *LightGCN* removes any non-linearity, thereby simplifying the overall propagation rule. After K iterations over all the nodes, the Kth layer embedding is denoted as $E^{(K)}$.

5.2 Weighted Embeddings Average

For computing the final embedding, the model considers a weighted average with equal weights to all the previous layers. The following equations are used for calculation of final embedding:

$$\mathbf{e_u} = \sum_{k=0}^{K} \alpha_k \mathbf{e_u^{(k)}} \tag{3}$$

$$\mathbf{e_i} = \sum_{k=0}^{K} \alpha_k \mathbf{e_i^{(k)}} \tag{4}$$

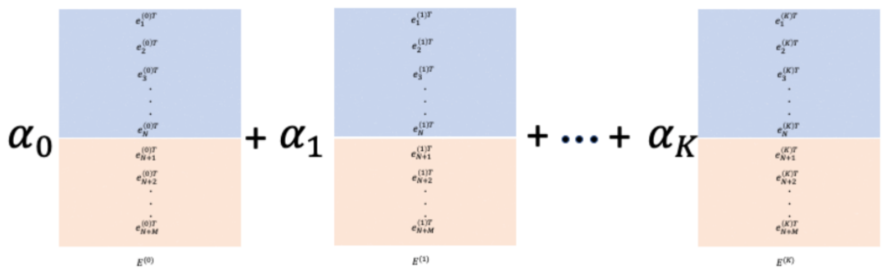


Figure 5: Weighted average of all the previous layers for computation of final embedding

5.3 Model architecture

The model architecture diagram is given in figure 6.

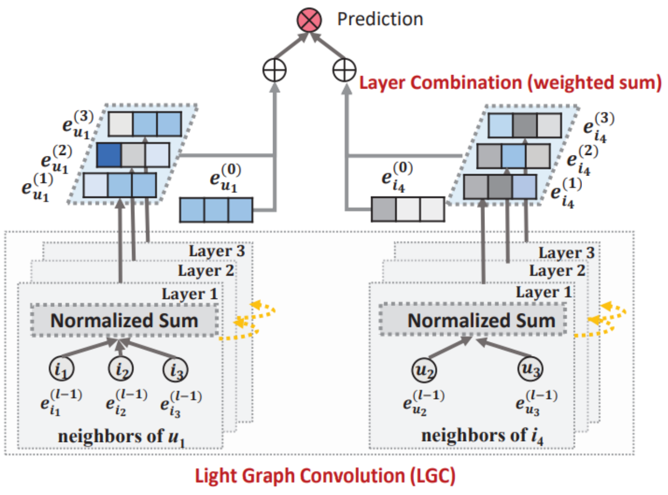


Figure 6: *LightGCN* Model Architecture

5.4 Loss Function

To evaluate the recommendation system, scores are computed using the final embeddings of user and items as follows:

$$\hat{y_{ui}} = e_u^T e_i \tag{5}$$

For training the model, **Bayesian Personalized Loss** is used. *BPR* loss is very popular loss function used for recommendation systems. It gives higher preference to the observed user-item predictions compared to the unobserved ones.

$$\mathcal{L}_{\mathcal{BPR}} = -\sum_{u=1}^{M} \sum_{i \in N_u} \sum_{j \notin N_u} ln\sigma(\hat{y_{ui}} - \hat{y_{uj}}) + \lambda ||E^{(0)}||^2 \tag{6}$$

where $\hat{y_{ui}}$ and $\hat{y_{uj}}$ are the predicted score of a positive sample and a negative sample respectively. The loss function uses regularization as well just like standard Ridge regression.

5.5 Training the Model

The problem thus boils down to minimizing the BPR loss using standard optimization techniques like *Gradient Descent* and training the model. In this project *Adam Optimizer* is used on top of Gradient Descent.

5.6 Evaluation Metrics

The scores computed at the output layer are used to determine the top K scoring movies for each user. Following evaluation metrics are used in the project:

- MAP: Mean Average Precision
- Top-K Precision : It denotes the fraction of K recommended movies that are liked by the user.
- Top-K Recall: It denotes the fraction of relevant movies that are recommended to the user in the K movie recommendations.
- Normalized Discounted Cumulative Gain (NDCG)

NDCG is widely used in recommendation systems and information retrieval-based literature. It takes into consideration the ordering of the retrieved responses from the recommendation. It takes the ratio of Discounted Cumulative Gain (DCG) of the predicted recommended order to the ideal order.

$$DCG_p = \sum_{i=1}^{p} \frac{2^rel_i - 1}{log_2(i+1)} \tag{7}$$

$$IDCG_p = \sum_{i=1}^{|REL_p|} \frac{2^rel_i - 1}{log_2(i+1)} \tag{8}$$

$$nDCG_p = \frac{DCG_p}{IDCG_p} \tag{9}$$

where $\mathbf{p}$ is a particular rank position, $\mathbf{rel_i} \in \mathbf{0}, \mathbf{1}$ is the graded relevance of the result at position $\mathbf{i}$ and $|\mathbf{REL}|_\mathbf{p}$ is the list of items ordered by their relevance upto position $\mathbf{p}$.

6 Algorithm

Algorithm 1: *LightGCN*

```
Input: MovieLens Dataset
       Hyperparameters Dictionary
Output: Movie recommendations to users
Initialise the embeddings at 0th layer, i.e., $e_u^0$ for all users and $e_i^0$ for all items (movies);
for $k$ = 1 to $K$ do
    Compute $k^{th}$ layer embeddings: $e_u^k$ for all users and $e_i^k$ for all items using Eqs. (1) & (2);
end
Combine the embeddings of k layers to determine the final embeddings $e_u$ for all users and
  $e_i$ for all items using Eqs. (3) & (4);
Compute the scores from final user and item embeddings using Eq. (5);
Recommend the items (movies) with highest scores to the users;
```

7 Overall Pipeline from Input to Output

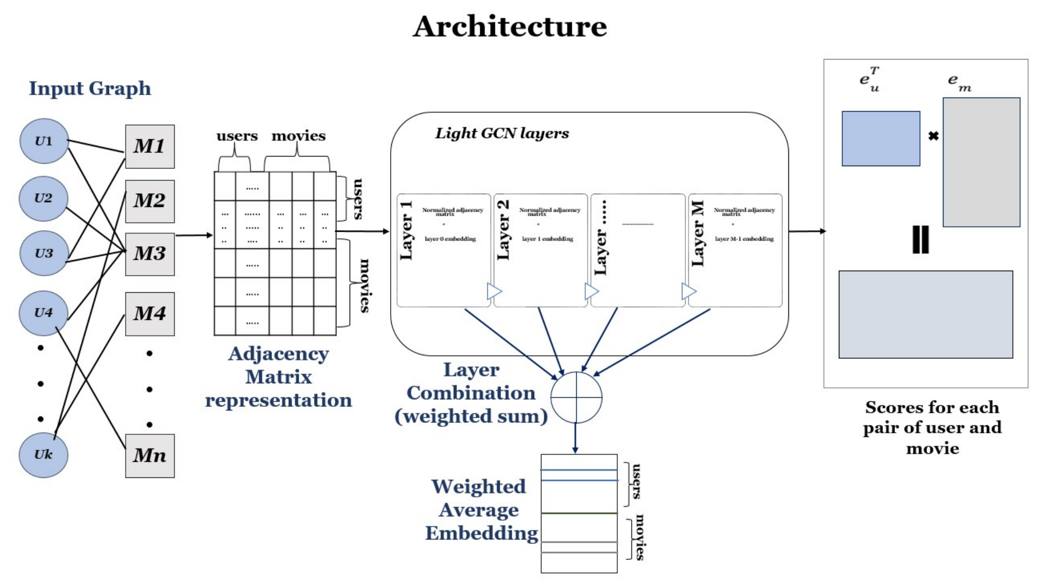

Architecture
Input Graph
U1
U2
U3
U4
Uk
M1
M2
M3
M4
Mn
users
movies
Adjacency Matrix representation
Light GCN layers
Layer 1
Layer 2
Layer ....
Layer M
Layer Combination (weighted sum)
Weighted Average Embedding
e_u^T
e_m
Scores for each pair of user and movie

Figure 7: Model Pipeline Architecure for *LightGCN*

8 Proposed Novel Modification

LightGCN++ is the proposed novel modification. For the final embedding computation, instead of equal weightage to each layer, more weightage is given to later layers. This is achieved by multiplying layer embeddings by $\alpha \epsilon (0,1)$ such that the initial layer embedding is multiplied K + 1 times by α and the last layer is multiplied only once by α. Thus, more weightage is given to the last layer embedding. The equations for computing final embedding get modified as follows:

$$\mathbf{e_u} = \sum_{k=0}^{K} \alpha^{K-k+1} \mathbf{e_u^{(k)}} \tag{10}$$

$$\mathbf{e_i} = \sum_{k=0}^{K} \alpha^{K-k+1} \mathbf{e_i^{(k)}} \tag{11}$$

9 Solving the Cold Start Problem

Given a new user with no past rating history, the embedding vector is computed for that user using its profile features. Next, the scores of this embedding are computed with all the movies and correspondingly the K movies with highest scores are recommended.

10 Results

Data is split into training, validation and test sets in 70:15:15 split ratio for both the 100K and 1M datasets. Following hyperparameter values are used:

Hyperparameter	**Value**
Embedding size	64
Number of layers	3
Learning rate	0.005
Batch size	1024
Number of epochs	100
Regularization parameter	0.0001
Top K recommendations	10

Table 1: Hyperparameter Values

Comparison is done with following baselines on the evaluation metrics discussed before:

- **Alternative Least Squares** [2] : It is a matrix factorization technique which minimizes two loss functions alternatively. Firstly, it fixes the user matrix and runs gradient descent using item matrix with L2 regularization and vice versa.
- **Singular Value Decomposition** [3] : This approach partitions the utility matrix A into 3 matrices: U - orthogonal left singular matrix, S - diagonal matrix, V - diagonal right singular matrix.
- **Neural Collaborative Filtering** [4] : It uses Feed Forward Neural Network to train a model for recommending items to users.

Results are summarized in table 2 and figure 8 and 9. It is observed that on evaluation metrics like MAP and Top-K Precision, *LightGCN++* performs better than all other algorithms while on evaluation metrics like NDGC and Top-K Recall, *LightGCN* performs better than the rest. Similar trend is observed for both the MovieLens 100K dataset and MovieLens 1M dataset.

Data	Algorithm	MAP	NDGC	Top-K Precision	Top-K Recall
100K	ALS	0.004697	0.046619	0.049629	0.016688
100K	SVD	0.012873	0.095930	0.091198	0.032783
100K	NCF	0.108609	0.398754	0.349735	0.181576
100K	LightGCN	0.121633	**0.417629**	0.360976	**0.196052**
1M	LightGCN++	**0.141294**	0.391641	**0.43819**	0.138974
1M	ALS	0.002683	0.030447	0.036707	0.011461
1M	SVD	0.008828	0.089320	0.082856	0.021582
1M	NCF	0.065931	0.357964	0.327249	0.111665
1M	LightGCN	0.089775	**0.423900**	0.385721	**0.147728**
1M	LightGCN++	**0.091297**	0.403426	**0.43819**	0.138974

Table 2: Results

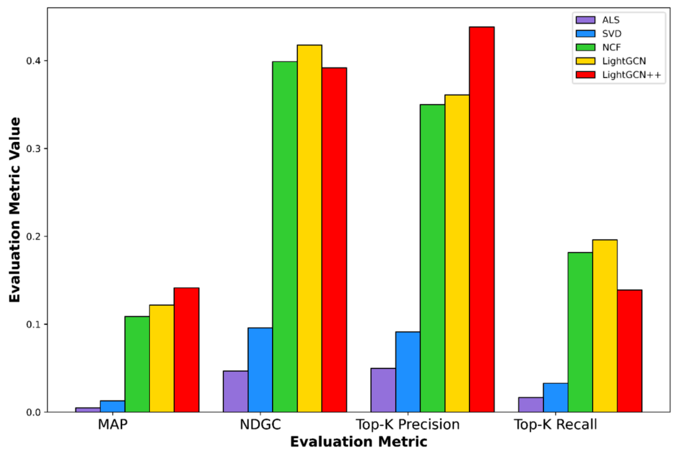


Figure 8: Results for MovieLens 100K dataset

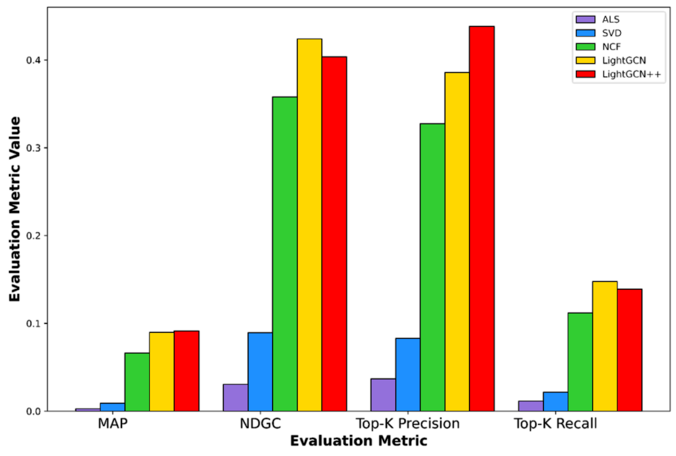


Figure 9: Results for MovieLens 1M dataset

11 Conclusion & Future Work

In this project, we have implemented LightGCN on 2 variants of MovieLens datasets. We have proposed a variant of original model, LightGCN++. We have compared the performance of LightGCN and LightGCN++ with 3 baselines (ALS, SVD NCF) on 4 evaluation metrics (MAP, NDGC, Top-K Precision Top-K Recall) and promising results are obtained. Cold start problem is also addressed. Demo of LightGCN working is built on Gradio and deployed on Huggingface. Here we are using order invariant convolutions for neighbor aggregration. Usage of permutation based convolutions can be explored in future.

12 Code Repository & Gradio Links

Code Repository: GitHub
Gradio Demonstration: Dataset Analysis & Top K Recommendations
Gradio Implementation: Dataset Analysis Code & Top K Recommendations Code

References

[1] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pp. 639–648, 2020.

[2] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the netflix prize," in *International conference on algorithmic applications in management*, pp. 337–348, Springer, 2008.

[3] X. Zhou, J. He, G. Huang, and Y. Zhang, "Svd-based incremental approaches for recommender systems," *Journal of Computer and System Sciences*, vol. 81, no. 4, pp. 717–733, 2015.

[4] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, pp. 173–182, 2017.