CS725- Foundations of Machine Learning Course Project - Preliminary Report

Prateek Chanda[1], Sandarbh Yadav[1], Jimut Bahan Pal[2], and Goda Nagakalyani[1]

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

November 2022

1 Project Idea

We are building a recommendation model that can be used in a production setting for suitable recommending items to users and resulting in better user experience and user engagement. For this course project, we use a model proposed by a recent paper using Graph Convolution Neural Network ($LightGCN$) [2] (using tensorflow from scratch). We compare with traditional collaborative-based filtering like matrix factorization and state-of-the-art models like Alternative Least Squares. We then propose a variation of the original paper based on motivations outlined below. Finally we try to solve the cold start problem in traditional recommendation systems by proposing suitable extensions to the model and implementing recent paper [1] (implementation is currently underway).

2 Related Works

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. Recent neural recommendation models like Neural Collaborative Filtering NCF [3] use the same embedding component while enhancing the interaction modeling with neural networks. Another type of CF methods consider historical items as the pre-existing features of a user, towards better user representations. SVD based approaches [5] use the weighted average of ID embeddings of historical items as the target user's embedding.

3 Dataset

For this project, we are using the **movielens** dataset. It contains integer movie ratings within the range of 1 to 5, collected from 6000 users and 4000 movies along with corresponding user features and movie features. It is commonly used as a benchmark for recommendation system-based publications.

We use two variants of the dataset.

- Movielens 100K - 100,000 ratings from 1000 users on 1700 movies. Released 4/1998.
- Movielens 1M - 1 million ratings from 6000 users on 4000 movies. Released 2/2003.

4 Methodology

We first implement the graph neural network model proposed by the paper [2] and briefly describe the architecture and methodology that we are using. Secondly we propose our modification to the above method and showcase through our experiments

Most recommendation systems use standard collaborative filtering based methods that take into account the notion of K-nearest neighbors to find another user in the system that has a similar embedding to that of the query user. However, for training purposes, these approaches fail to capture the essence of the interactions between users and items.

We thus use *LightGCN* which is a collaborative filtering based method but takes into account the concept of convolution used in Graph Convolution Neural Networks.

GCNs often capture the structural information present in a graph

4.0.1 Message Aggregation for computing embeddings

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

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

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 user and movie(item) nodes' number of neighbors.

All in all the model *LightGCN* removes any non-linearity, thereby simplifying the overall propagation rule.

After K iterations over all the nodes, we have the Kth layer embedding as $E^{(K)}$.

4.0.2 Weighted embeddings average

During the layer combination, the model considers a weighted average with equal weights to all the previous layers for computing the final embedding.

The final embeddings are as follows:

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

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

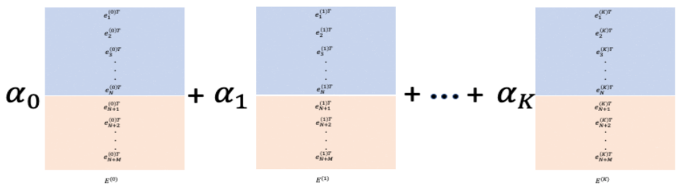

Figure 1: Weighted average of previous layers for final embedding computation

4.1 Model architecture

We provide the model architecture diagram based on what we discussed above. Figure. 2

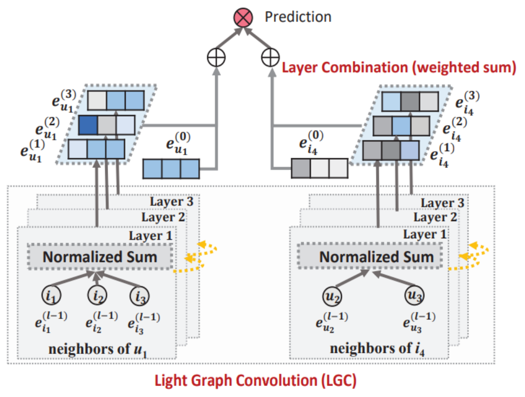

Figure 2: Model architecture from the paper

4.2 Loss Function

To evaluate our recommendation system, we compute the scores using the final embeddings of the user and items as follows.

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

In order to train this model, as proposed in the paper we use the **Bayesian Personalized Loss**. *BPR* loss is a pretty popular loss function used for optimization in recommendation scenarios. It gives higher preference to the observed user-item predictions compared to the unobserved ones.

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

where

$\hat{y_{ui}}$ and $\hat{y_{uj}}$ are the *predicted score of a positive sample and a negative sample respectively.*

As we can see the loss function uses regularization as well just like standard *Ridge Regression.*

4.3 Minimizing the loss and Training Model

The problem thus boils down to minimizing the BPR loss using standard optimization techniques like *Gradient Descent* and training the model. For our experiments, we also used *Adam Optimizer* on top of Gradient Descent.

4.4 Evaluation Metrics

We use the scores computed at the output layer : $\hat{y_{ui}}$. We then take the top K scoring movies for each user that were not in the training set and quantify the following metrics

- 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) : (Described below in detail)

NDCG is most 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 the Discounted Cumulative Gain (DCG) of the predicted recommended order the ideal order.

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

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

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

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 $\mathbf{p}$.

5 Overall Pipeline from Input to Output

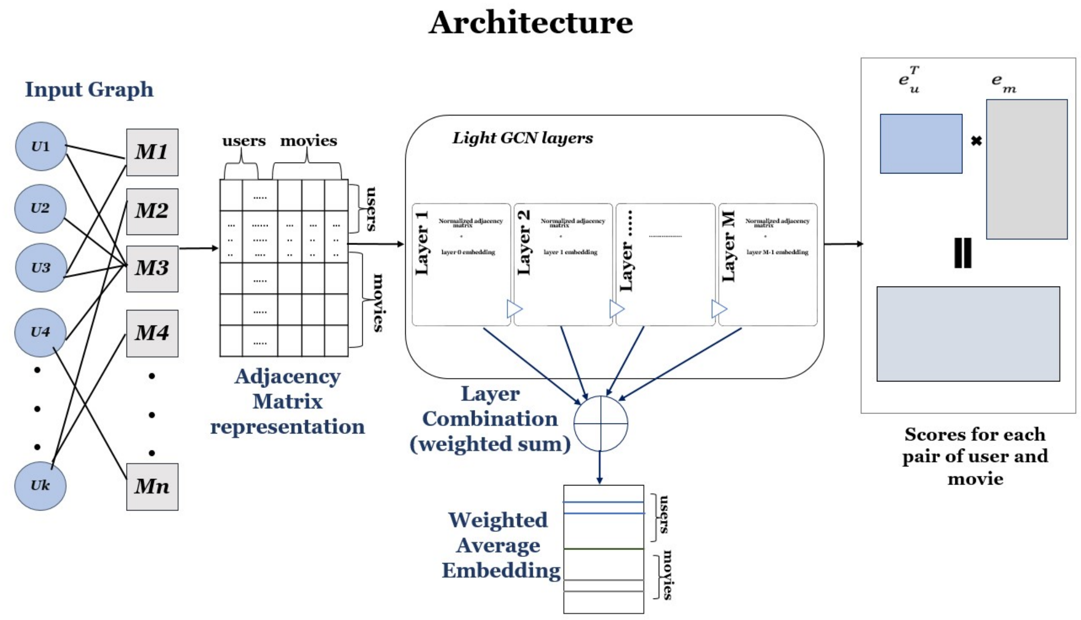

Figure 3: Model Pipeline Architecure for LightGCN

6 Novel Proposal for Model Extension

We present $LightGCN$++, where we propose the following change. For the final embedding computation, instead of uniform weightage, we take into consideration higher priority in the recent layers by multiplying some fractional constant

α with the corresponding Kth layer such that, the initial layer embedding has α multiplied $K-1$ times and the last layer has only α multiplied 1 time. Thus more weightage is given to the last layer embedding. We run our experiments below.

7 Experiments Done Till Now

The data is split into train and test sets using split ratio as 75-25% respectively. We do this for both the $100K$ and $10M$ datasets.

We then compare with respect to the following baselines:

- **Alternative Least Squares**: It applies block relaxation algorithm to a least squares loss function. Least squares loss functions are somewhat loosely defined here.
- **Singular Value Decomposition**: It refers to the factorization of a matrix into three matrices with interesting algebraic properties. It conveys important geometrical and theoretical insights about linear transformations.
- **Neural Collaborative Filtering** [3]: It is a neural matrix factorization model which combines Multi-layer perceptron and matrix factorization to learn the embeddings of users and items.

We showcase our results till now. Bold values are the best results till now.

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 1: Results

8 Experiments to be done

8.1 Solution to the cold start problem

For solving the cold start problem we are currently exploring two directions.

8.1.1 Use a suitable similarity embedding metric for a new user

When a new user arrives in the system, using a suitable similarity metric we compare the embedding of the new user to that of the existing users. The embeddings can be built by leveraging the existing user features already there in the dataset.

8.1.2 Pre-training the GCN Model

A recent 2021 WSDM paper "Graph Neural Pre-training for Enhancing Recommendations using Side Information " [1] mentions that pretraining the graph neural model before the recommendation can alleviate the cold start problem. Unlike the goal of recommendation, the pre-training GNN simulates the cold-start scenarios from the users/items with sufficient interactions and takes the embedding reconstruction as the pretext task, such that it can directly improve the embedding quality and can be easily adapted to the new cold-start users/items.

We are currently exploring the above paper and trying to implement it for our current scenario and designing suitable experiments for the same.

9 Possible Future Extensions

We are currently exploring some future directions such as in case of computing the embeddings for user and items, we perform order-invariant convolution operation as traditional GCN approaches follow. Recent paper from AAAI [4] discusses the concept of permutation GNNs where during the convolution operation they consider the role of order in which the convolution gets performed using the information from neighboring nodes. Owing to better performance in metrics as stated in the paper, this is a nice thing to explore further for our recommendation system if this can capture more complex interactions between the users and items.

Lastly, we also plan to explore the notion of interpretability for our neural network based recommendation system.

References

[1] Bowen Hao, Jing Zhang, Hongzhi Yin, Cuiping Li, and Hong Chen. Pre-training graph neural networks for cold-start users and items representation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 265–273, 2021.

[2] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng 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*, pages 639–648, 2020.

[3] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[4] Indradyumna Roy, Abir De, and Soumen Chakrabarti. Adversarial permutation guided node representations for link prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 9445–9453, 2021.

[5] Xun Zhou, Jing He, Guangyan Huang, and Yanchun Zhang. Svd-based incremental approaches for recommender systems. *Journal of Computer and System Sciences*, 81(4):717–733, 2015.