Solving the Cold Start problem in Recommendation Systems

Case Study on MovieLens Dataset

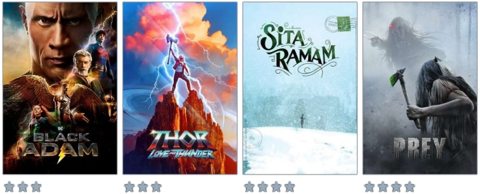
Prateek Chanda (22D0362), Sandarbh Yadav (22D0374),
Jimut Bahan Pal (22D1594) & Goda Nagakalyani (214050010)

under the guidance of
Prof. Preethi Jyothi

CS725 - Foundations of Machine Learning

Table of Contents

Introduction



Recommendation systems form the basis of many applications like Netflix movie recommendations, Amazon product recommendations etc. In this project:

- A recommendation model, *LightGCN* [1], is built using GCN (SIGIR 2020).
- A novel variant of original model, *LightGCN++*, is proposed.
- Comparison of performance is done with traditional and state of the art models.

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.
- *LightGCN* captures the user-item interactions as a bipartite graph.

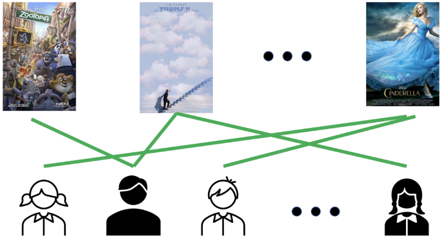


Table of Contents

MovieLens Dataset

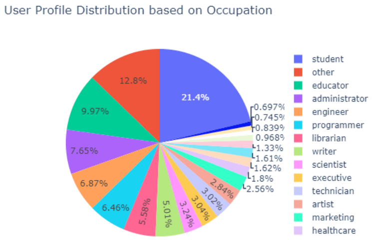
- MovieLens is a popular benchmark dataset for recommendation systems.
- It contains data about movies, users and ratings (on a scale of 1 to 5).

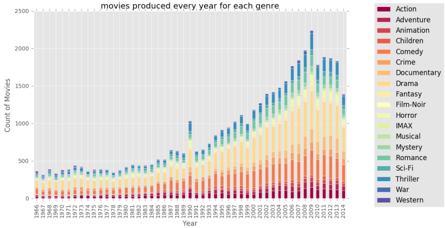
Movie ID	Title	Genres
1	Toy Story (1995)	Animation|Children's|Comedy
2	Jumanji (1995)	Adventure|Children's|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
:	:	:

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
:	:	:	:

MovieLens Dataset





Two variants of MovieLens dataset are used:

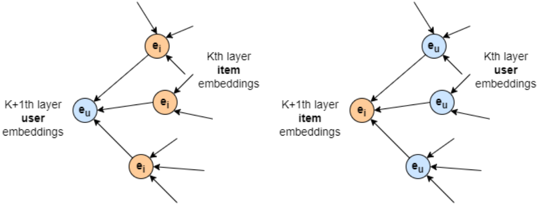
- **MovieLens 100K :** 100,000 ratings from 1000 users on 1700 movies
- **MovieLens 1M :** 1,000,000 ratings from 6000 users on 4000 movies

Table of Contents

Graph Convolution Neural Networks

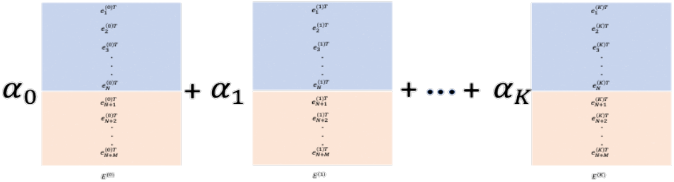
- *LightGCN* is based on Graph Convolution Neural Networks (GCN) which captures the structural information present in the bipartite graph. It simplifies the overall propagation rule by removing non-linearity.
- Embeddings are computed via message aggregation using the following equations:

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



Weighted Embeddings Average

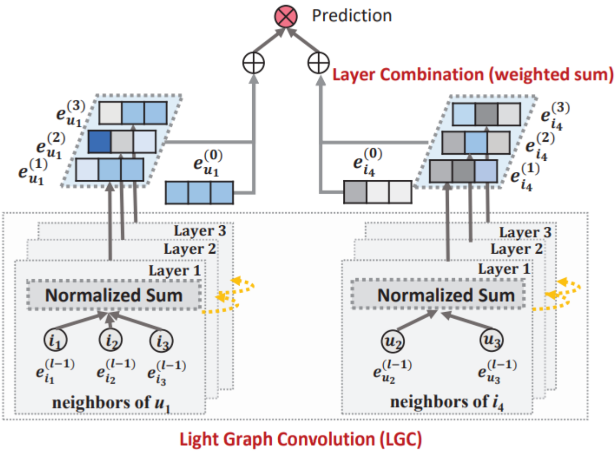
- For computing the final embedding, the model considers a weighted average with equal weights to all the previous layers.



- The final embeddings are computed as follows for $\alpha_k = \frac{1}{K+1}$:

$$\mathbf{e_u} = \sum_{k=0}^{K} \alpha_k \mathbf{e_u^{(k)}} \qquad \textit{and} \qquad \mathbf{e_i} = \sum_{k=0}^{K} \alpha_k \mathbf{e_i^{(k)}} \tag{2}$$

Model Architecture



Loss Function

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

$$\hat{y_{ui}} = e_u^T e_i \tag{3}$$

- **Bayesian Personalized Loss** (*BPR*) loss is a popular loss function in recommendation systems. It gives higher preference to observed user-item predictions compared to the unobserved ones. BPR loss is used in this project.

$$\mathcal{L}_{\mathcal{BPR}} = -\sum_{u=1}^{M} \sum_{i \in N_u} \sum_{j \notin N_u} ln\sigma(\hat{y_{ui}} - \hat{y_{uj}}) + \lambda ||E^{(0)}||^2 \tag{4}$$

- The problem reduces to minimizing the BPR loss and training the model. *Adam Optimizer* is used on top of Gradient Descent.

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 K movie recommendations.
- **Normalized Discounted Cumulative Gain (NDCG):** It considers the ordering of retrieved responses from the recommendation. It is widely used in recommendation systems.

$$nDCG_p = \frac{DCG_p}{IDCG_p} \tag{5}$$

LightGCN++ : A Novel Contribution

- *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.

$$\mathbf{e_u} = \sum_{k=0}^{K} \alpha^{K-k+1} \mathbf{e_u^{(k)}} \quad \textit{and} \quad \mathbf{e_i} = \sum_{k=0}^{K} \alpha^{K-k+1} \mathbf{e_i^{(k)}} \tag{6}$$

Table of Contents

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, we compute the scores of this embedding with all the movies and correspondingly recommend the K movies with highest scores.

Table of Contents

Experimental Setup

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	15
Regularization parameter	0.0001
Top K recommendations	10

Baselines

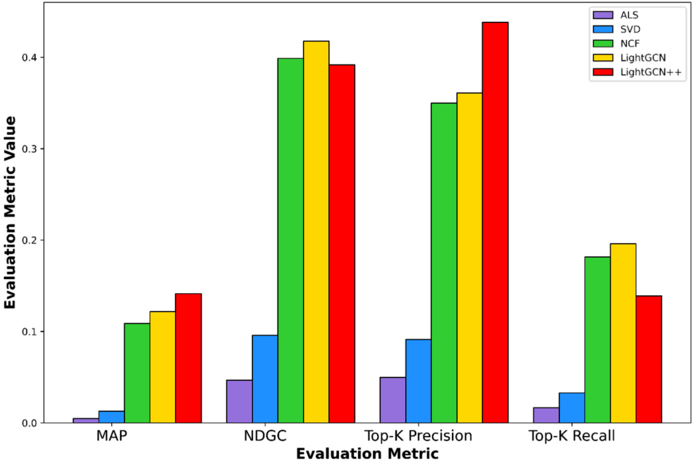
Comparison is done with following baselines on the evaluation metrics discussed before:

- **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

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**
100K	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	**0.47371**	0.138974

Results for MovieLens 100K dataset



Results for MovieLens 1M dataset

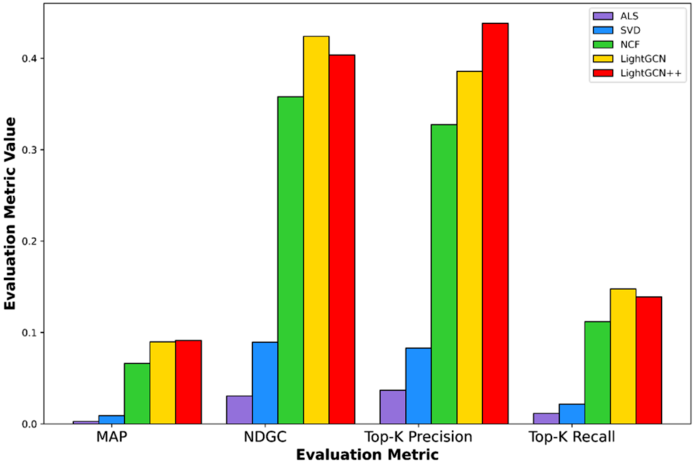


Table of Contents

Conclusion

- In this project, we have implemented *LightGCN* on 2 variants of MovieLens datasets using TensorFlow.
- We have proposed a novel variant of the 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 built on Gradio, deployed on *Huggingface*.
- **Future Work**: Here we are using order invariant convolutions for neighbor aggregration, can we use permutation based convolutions if they give better results?
- Code repository: GitHub
- Gradio: Dataset Analysis Top K Recommendations

Contributions

- Problem Statement Formulation: Sandarbh
- Literature Review: Jimut
- Dataset Exploration: Sandarbh
- *LightGCN* Implementation: Prateek
- *LightGCN++* Implementation: Prateek
- Diagrams: Nagakalyani
- Evaluation Metrics: Prateek
- Baselines Implementation: Jimut
- Experiments: Sandarbh
- Plots: Nagakalyani
- Gradio: Prateek
- Presentation: Sandarbh
- Report: Nagakalyani

References

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.

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.

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.

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.

Thank You