Recommender Systems Project

Movie Recommendations with Graph Neural Network

Group:
22111013 – Atul Kumar
22111054 – Shubham Rathore
22111069 – Vivek Kumar Gautam

Problem Statement

Yield recommendations on the MovieLens-100K dataset

About the dataset

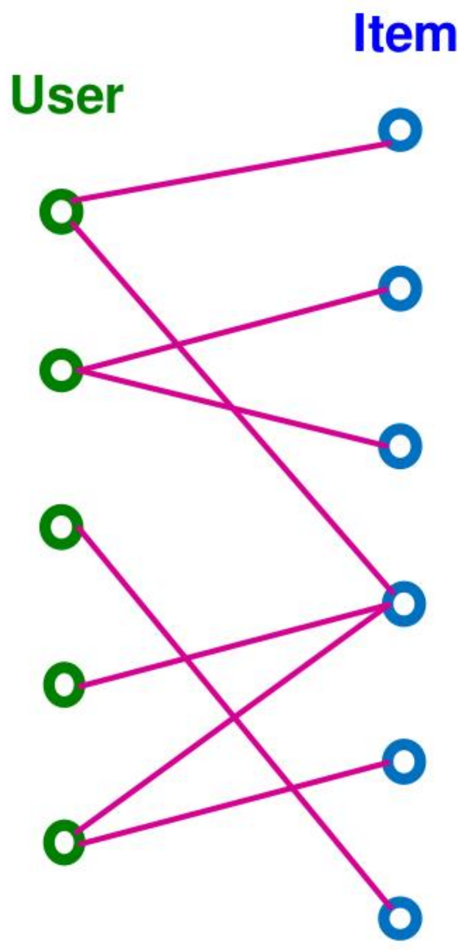
- 100,000 ratings (1-5) from 983 users on 1682 movies
- Each user has rated at least 20 movies
- Some demographic information is present about the users
- Collected on the MovieLens website during Sept '97 through Apr '98
- Citation: F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872

Recommender Systems as a Graph

- Any recommendation system can be naturally modeled as a **bipartite graph**

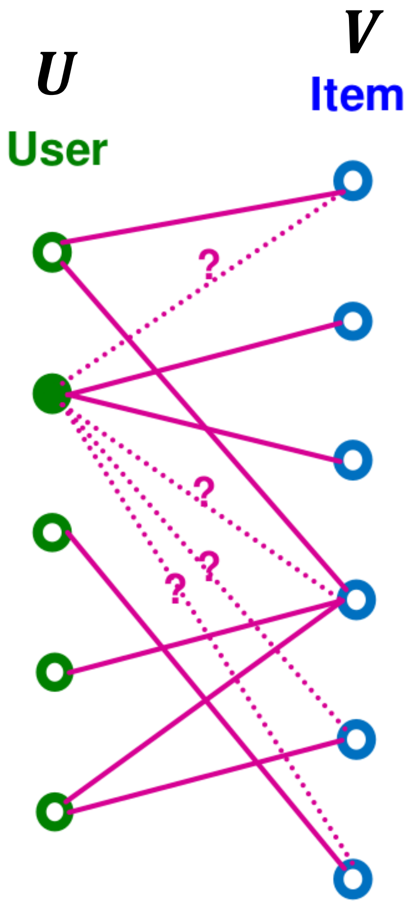
- Nodes:
  - Users
  - Movies

- Edges:
  - An edge denotes an **interaction** between a user and an item
  - User interacted with a movie by giving it a good (or a bad?) rating



Recommendation Task

- **Given:** Ratings from users on movies
- Predict which movie the user will like in future **i.e., predict an edge**
- We need to get a score $f\ (u, v)$, for $u \in \boldsymbol{U}$, $v \in \boldsymbol{V}$
- This score will indicate the probability of a user liking a movie



Top-K Recommendation

- For each user, we recommend K items

- For recommendation to be effective, $\boldsymbol{K}$ needs to be much smaller than the total number of items

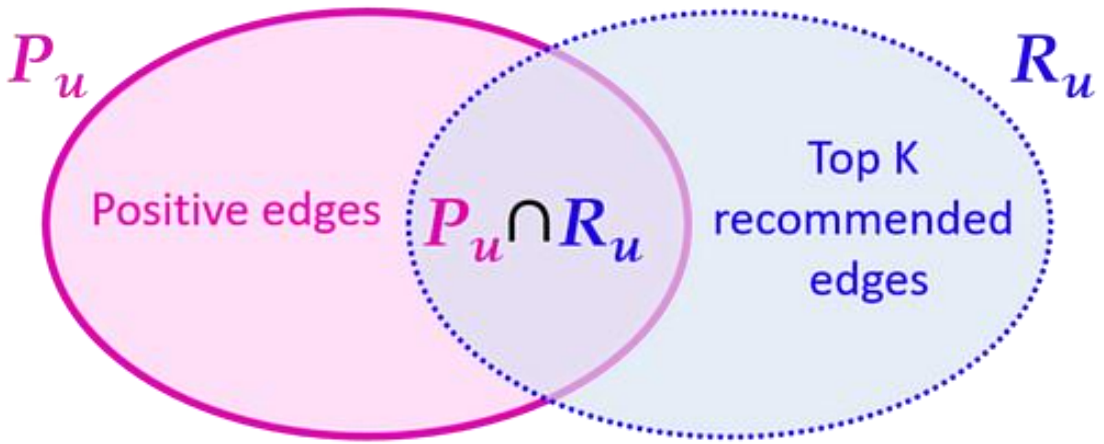
- K is typically in the order of 10 – 100

- Include as many **positive items** as possible in the top-K recommended items

- Positive items = Items that the user will interact with in the future

Evaluation Metric: Recall@K

- For each user $\boldsymbol{u}$,
  - Let Pu = set of positive items the user will interact with in the future,
  - Let Ru = set of items recommended by the model
  - $|Ru|$ = K, since the model recommends only the top-K items
  - Recall@$\boldsymbol{K}$ for user $\boldsymbol{u}$ is $(\boldsymbol{Pu} \cap \boldsymbol{Ru})$ / $\boldsymbol{Pu}$



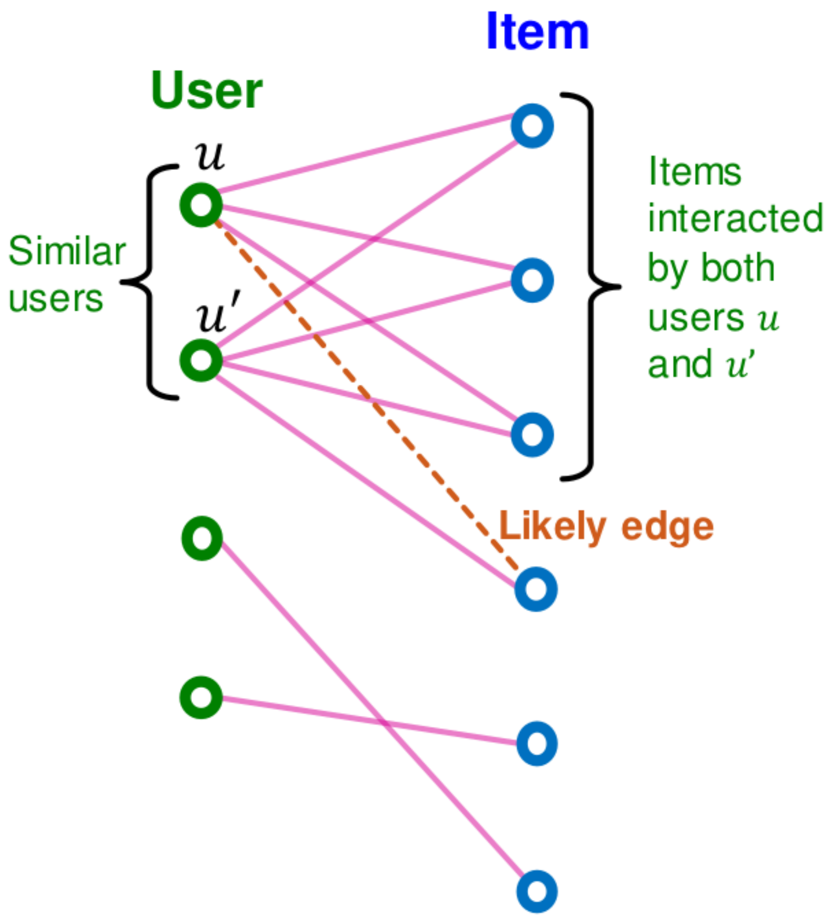
$$\text{Recall@K} = \frac{1}{|\text{Users}|} \sum_{u \in \text{Users}} \left(\frac{P_u \cap R_u}{P_u} \right)$$

Collaborative Filtering

- Recommend movies to a user based on the preferences of similar users

- Similar users tend to prefer similar movies – collaborative filtering idea

- **Task**: Capture similarity between users/movies



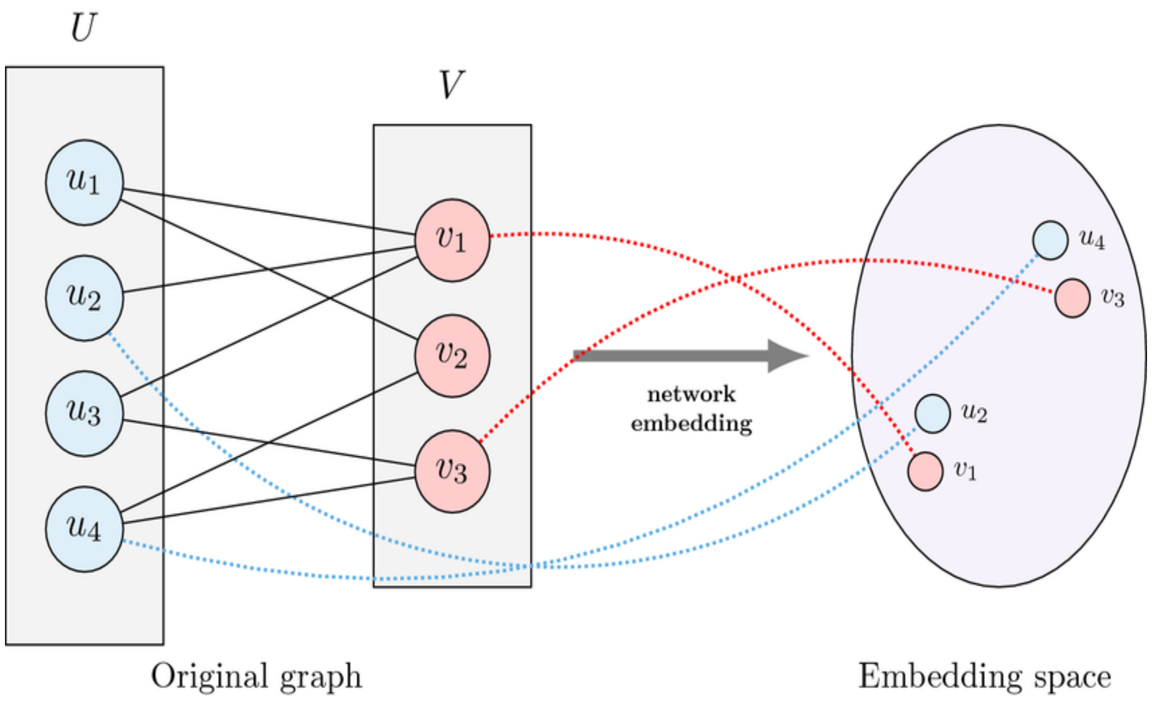
Which model to use?

- A complex feature-based model could simply memorize the user-item interactions without learning similarity

- Solution: **Embedding-based models**

- Low-dimensional embeddings are forced to capture similarities to fit the data

- Better predictions on **unseen** user-item interactions



Training Objective

- Optimize the model to **achieve high recall@K on seen items** i.e., the training items

- We hope that this will lead to high recall@K on unseen (test) items

Problem with recall@K

- It is not differentiable – so we **cannot apply a gradient-based optimization**

- However, another loss function called the Bayesian Personalized Ranking (BPR) loss can be used

- It aligns with the recall metric and is differentiable

So Far

- Use Recall@K as a metric for personalized recommendation

- Use an embedding-based model and learn -
  - User encoder (to generate user embeddings)
  - Item encoder (to generate item embeddings)
  - Score function (to predict user-item interaction likelihood)

- BPR loss for achieving the optimization objective

Choosing a GNN architecture

- GNNs* have been widely successful in recommendation tasks

- **NGCF** [Wang et al. 2019]
  - Neural Graph Collaborative Filtering (NGCF)
  - Incorporates non-linear activations and complex feature transformations
  - Learns graph-aware user-item embeddings

- **LightGCN** [He et al. 2020]
  - Light Graph Convolutional Network (LightGCN)
  - Empirically proves that NGCF's complex methods are not required for good performance
  - Explicitly models the graph structures
  - Assumes no user/item features

*Sanchez-Lengeling, et al., "A Gentle Introduction to Graph Neural Networks", Distill, 2021.

LightGCN Overview

- LightGCN has the following convolution/propagation rule between each layer:

Embedding of user u_m at layer k+1

$$\boxed{u_m^{(k+1)}} = \sum_{i_n \in N(u_m)} \frac{1}{\sqrt{d_{u_m}} \cdot \sqrt{d_{i_n}}} \cdot \boxed{i_n^{(k)}}$$

$$\boxed{i_n^{(k+1)}} = \sum_{u_m \in N(i_n)} \frac{1}{\sqrt{\underbrace{d_{i_n}}_{}} \cdot \sqrt{\underbrace{d_{u_m}}_{}}} \cdot \boxed{u_m^{(k)}}$$

Embedding of item i_n at layer k+1

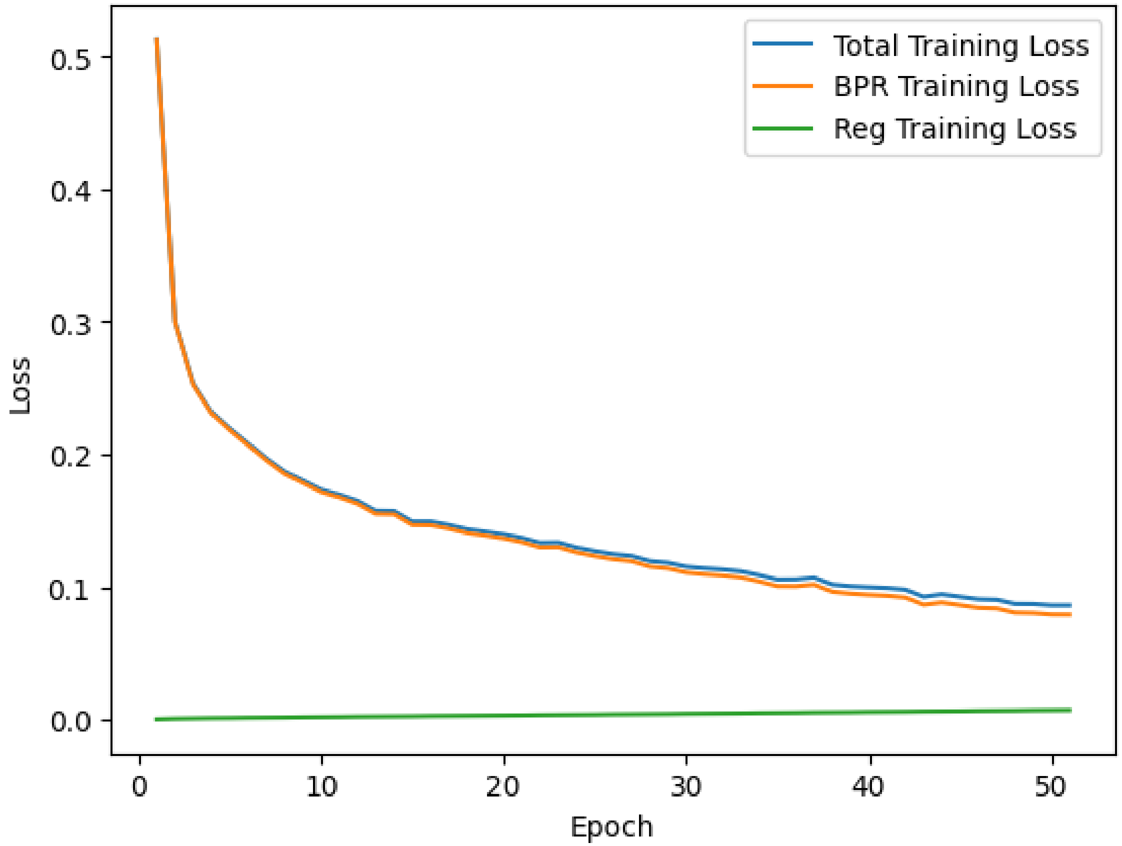
Degree of node u_m

Degree of node i_n

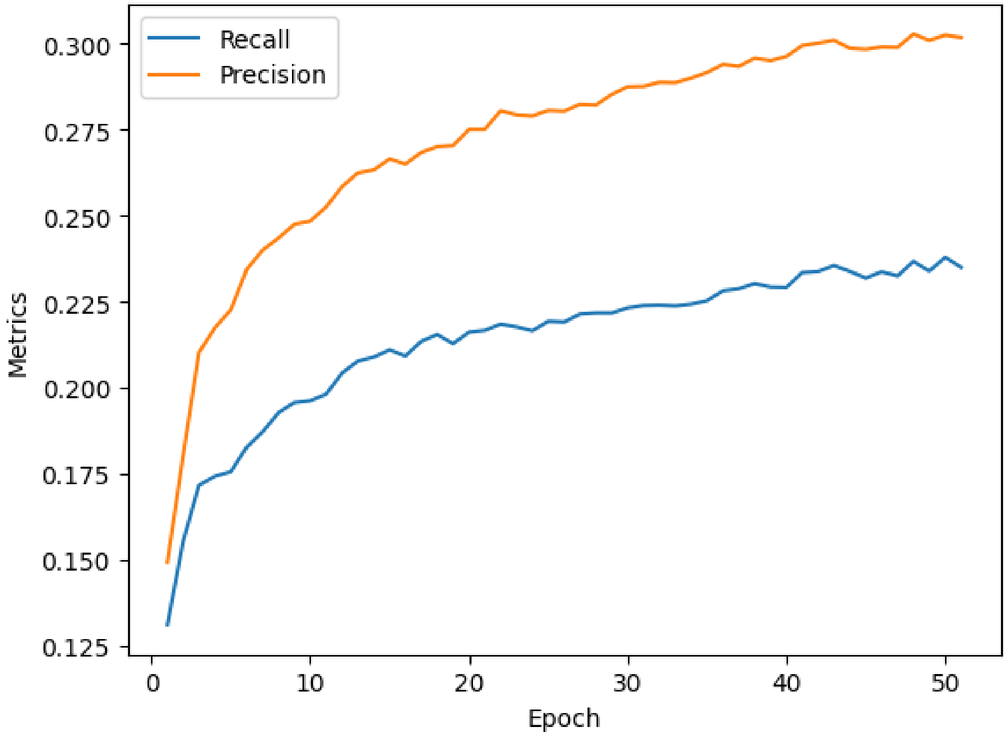
- The only trainable model parameters in LightGCN are the 0-th layer embeddings of all users and all items

Training and Evaluation

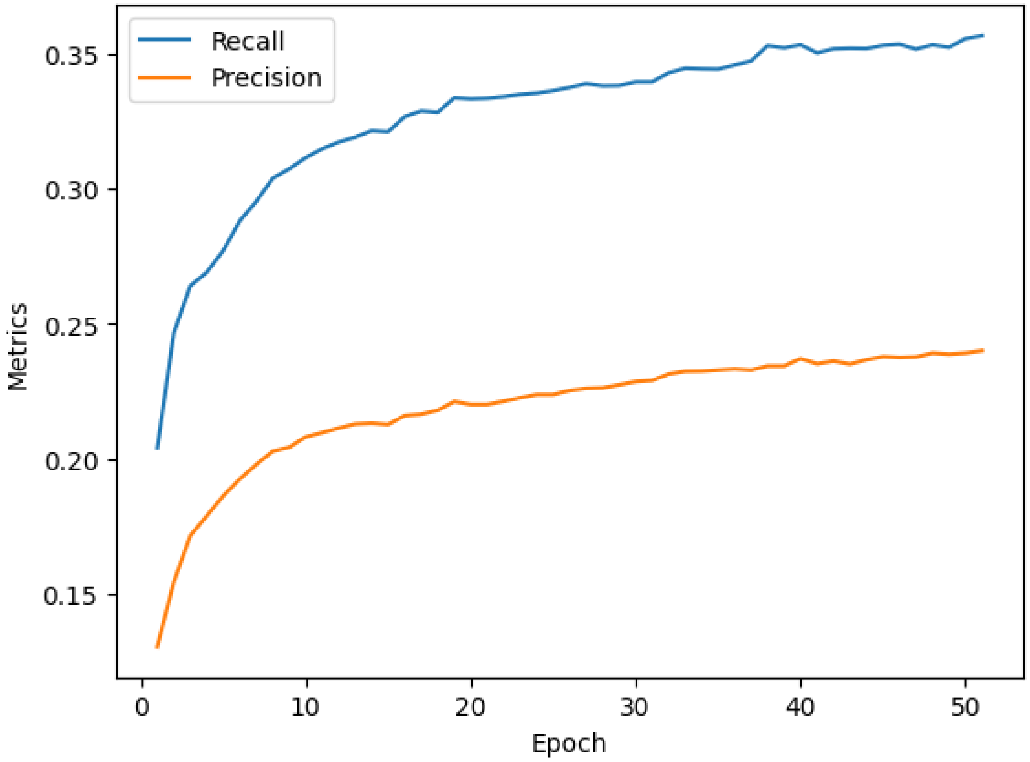
- A 3-layer LightGCN model takes ~3 minutes to train for 50 epochs (on Nvidia T4 GPU)



Training and Evaluation



Recall@10 and Precision@10
~0.22 and ~0.31 respectively



Recall@20 and Precision@20
~0.35 and ~0.24 respectively

Generating Recommendations

```
some movies that the user rated highly:

Toy Story (1995)
Twelve Monkeys (1995)
Seven (Se7en) (1995)
Apollo 13 (1995)
Star Wars (1977)
Fugitive, The (1993)
Terminator 2: Judgment Day (1991)
Silence of the Lambs, The (1991)
Rock, The (1996)
Die Hard (1988)


top 10 recommended movies for the user:

Father of the Bride (1950)
Supercop (1992)
Full Metal Jacket (1987)
Things to Do in Denver when You're Dead (1995)
Clockwork Orange, A (1971)
Crow, The (1994)
Pink Floyd - The Wall (1982)
Strange Days (1995)
Bonnie and Clyde (1967)
Wings of Desire (1987)
```

THANK YOU