Project Proposal:
LightGCN for Movie Recommendation

Le Hoang Long
Hanoi University of Science and Technology (HUST)
No. 1 Dai Co Viet, Hai Ba Trung
long.lh232099m@sis.hust.edu.vn, hoanglong1712@gmail.com

1 Application domain

MovieLens is an excellent dataset for training movie recommendation systems. The MovieLens 1M dataset is particularly well-suited for smaller projects, containing 1 million movie ratings, 4,000 movies, and 6,000 users.

The dataset is well-maintained and thoroughly validated, facilitating quick training. It has been referenced in numerous studies. All features, including those related to users and movies, are easy to interpret. Movie features include titles and genres, while user features encompass gender, age, occupation, and zip code.

We are going to build a LightGCN model for movie recommender based on the paper: "LightGCN: Simplifying and powering graph convolution network for recommendation" https://arxiv.org/pdf/2002.02126

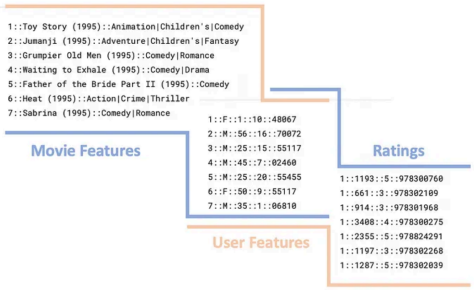

Figure 1: Features & Ratings

2 Graph ML techniques

The core concept of Graph Convolutional Networks (GCNs) is to learn node representations by smoothing features across the graph. This is accomplished through iterative graph convolution, where the features of neighboring nodes are aggregated to create a new representation for a target node. Such aggregation can be abstracted as:

$$e_u^{k+1} = AGG(e_u^k, \{e_i^k : i \in N_u\})$$

The AGG is an aggregation function that considers the k-th layer's representation of the target node and its neighbor nodes.

The fundamental concept of Graph Convolutional Networks (GCNs) is to iteratively gather feature information from local graph neighborhoods using neural networks. Each "convolution" operation

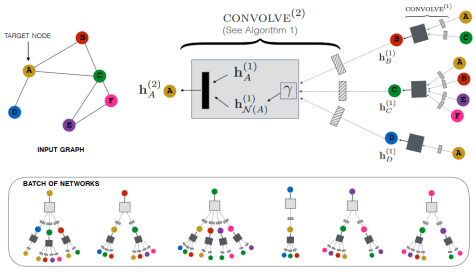

Figure 2: GCN

processes and aggregates feature data from a node's immediate one-hop neighborhood. By stacking multiple convolution layers, information can be transmitted across broader areas of the graph. In contrast to traditional content-based deep models, GCNs utilize both content information and the underlying graph structure.

The above model architecture uses depth-2 convolutions (best viewed in color). Left: A small example input graph. Right: The 2-layer neural network that computes the embedding $h_A^{(2)}$ of node A using the previous-layer representation, $h_A^{(1)}$, of node A and that of its neighborhood $N(A)$ (nodes B, C, D).

In LightGCN, we apply a simple weighted sum aggregator and abandon the use of feature transformation and nonlinear activation. The propagation rule is defined as:

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

$$e_i^{k+1} = \sum_{u \in N_i} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} e_u^{(k)}$$

$\frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}}$ is a symmetric normalization term, it helps to avoid the scale of embeddings increasing with graph convolution operations.

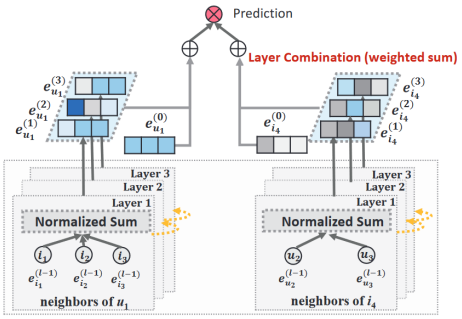

Figure 3: LightGCN

In LGC, only the normalized sum of neighbor embeddings is performed towards next layer; other operations like self-connection, feature transformation, and nonlinear activation are all removed, which largely simplifies GCNs. In Layer Combination, we sum over the embeddings at each layer to obtain the final representations.

In collaborative filtering tasks, the lack of rich node features often hinders inductive GNNs from effectively transforming features into a latent space. However, LightGCN takes a transductive

approach, directly learning embeddings for users and movies. This makes LightGCN particularly well-suited for the MovieLens dataset and other recommender systems that utilize simple user and item features.