PMLDL report on assignment 2

by Abdulayev Damir DSBS-02

Introduction

In the era of information overload, recommender systems play a crucial role in helping users discover content that aligns with their preferences. This report details the design, implementation, and evaluation of a movie recommender system model lightGCN, developed as part of the Practical Machine Learning and Deep Learning course's Assignment 2.

Data analysis

The dataset used for this movie recommender system is the MovieLens 100K dataset, a widely used benchmark in collaborative filtering and recommendation system research. The dataset consists of 100,000 ratings provided by 943 users for 1682 movies. Each rating ranges from 1 to 5, reflecting the user's preference for a particular movie. The dataset also includes demographic information for users, such as age, gender, occupation, and zip code.

To facilitate the training and evaluation of the recommender system, the dataset is split into training and test sets using the train_test_split function from the scikit-learn library. The split is performed with 80% of the data used for training and 20% for testing, ensuring a diverse representation of user-item interactions in both sets.

In order to prepare the data for machine learning models, label encoding is applied to 'user_id' and 'item_id' using LabelEncoder from the scikit-learn library. This step ensures that user and item IDs are represented as numerical indices, facilitating the model's understanding of categorical information.

Dataset Characteristics
The MovieLens 100K dataset exhibits the following characteristics:

Number of Users: 943
Number of Movies: 1682
Ratings Range: 1 to 5
Minimum Ratings per User: 20
The demographic information for users, including age, gender, occupation, and zip code, provides additional context for understanding user preferences and behavior.

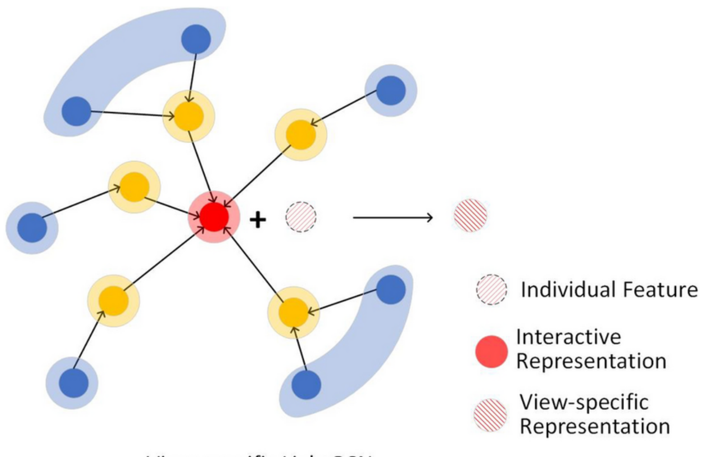
The next sections will delve into data exploration, model implementation, training processes, and evaluation, shedding light on the entire workflow of creating and assessing the movie recommender system.

Model Implementation

Graph Neural Network (GNN) Architecture
The recommendation system is implemented using a Graph Neural Network (GNN) architecture, specifically tailored for collaborative filtering tasks. The GNN model is designed to capture latent embeddings of users and items, facilitating effective recommendations. The core components of the model are the LightGCNConv layer and the RecSysGNN module.

LightGCNConv: Message Passing Layer
The LightGCNConv is a custom message passing layer derived from the PyTorch Geometric library. This layer is fundamental for aggregating information between connected nodes in the user-item interaction graph.

The LightGCNConv layer calculates the degree of nodes and performs normalization for effective message passing. This enables the model to understand the importance of each neighbor node's contribution to the final embeddings.



View-specific LightGCN

RecSysGNN: Graph Neural Network for Recommendation
The RecSysGNN module is the overarching structure that integrates the LightGCNConv layer to create a GNN tailored for recommendation systems.

The RecSysGNN module consists of an embedding layer for users and items, multiple LightGCNConv layers for message passing, and an initialization function for model parameters. The forward pass aggregates embeddings through the

LightGCNConv layers, producing both initial and final embeddings for users and items.

Model Advantages and Disadvantages

Advantages

Simplicity and Efficiency:
The LightGCNConv layer and RecSysGNN module are designed with simplicity in mind, making the model easy to understand and efficient to train.

Collaborative Filtering Capability:
The model leverages collaborative filtering by effectively capturing user-item interactions through graph-based message passing. This enables the model to make recommendations based on the preferences and behaviors of similar users.

Scalability:
With its lightweight design, the model exhibits scalability, allowing it to handle larger datasets and graphs without sacrificing performance.
Interpretability:

Disadvantages

Limited Expressiveness:
The simplicity of the LightGCNConv layer might limit the model's ability to capture complex relationships in the data. More sophisticated convolutional layers could be explored for improved expressiveness.

Cold Start Problem:
The model may face challenges in handling the cold start problem, where new users or items have limited interaction history. Enhancements, such as content-based features or hybrid models, may be required to address this issue.

Over-smoothing:
In scenarios with deep GNN architectures, there is a risk of oversmoothing, where node embeddings become indistinguishable. The aggregation strategy used in the model might contribute to this issue, and alternative aggregation methods could be investigated.

Limited User and Item Features:
The model relies solely on user and item indices for learning embeddings. Integrating additional user and item features, such as demographic information or movie genres, could potentially enhance recommendation quality.

Single Graph Structure:
The model assumes a single graph structure for user-item interactions. In scenarios with diverse interaction types or evolving user behaviors, a more dynamic graph approach might be beneficial.

Training Process

Hyperparameters
The training process for the movie recommender system involves defining several hyperparameters:

latent_dim: Dimensionality of the latent embeddings.
n_layers: Number of layers in the Graph Neural Network (GNN).
EPOCHS: Number of training epochs.
BATCH_SIZE: Size of each training batch.
DECAY: Weight decay for regularization.
LR: Learning rate for the optimizer.
K: Number of top recommendations to consider.

Training and Evaluation Function
The training and evaluation function, train_and_eval, orchestrates the model training and evaluation process. The function takes the recommendation model, optimizer, and training data as input and returns lists containing epoch-wise losses, BPR losses, regularization losses, recall scores, and precision scores.

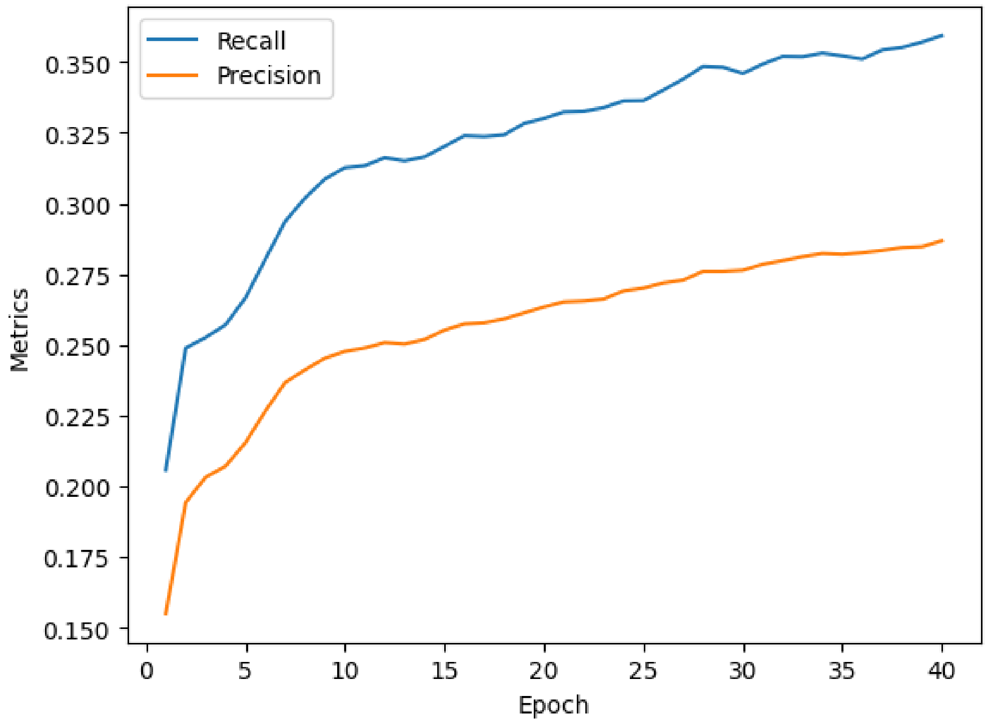
Training Loop
The training loop spans multiple epochs, with each epoch comprising iterations over batches of training data. The model is trained using the BPR (Bayesian Personalized Ranking) loss, a common loss function for recommendation systems. The regularization loss is applied to prevent overfitting.

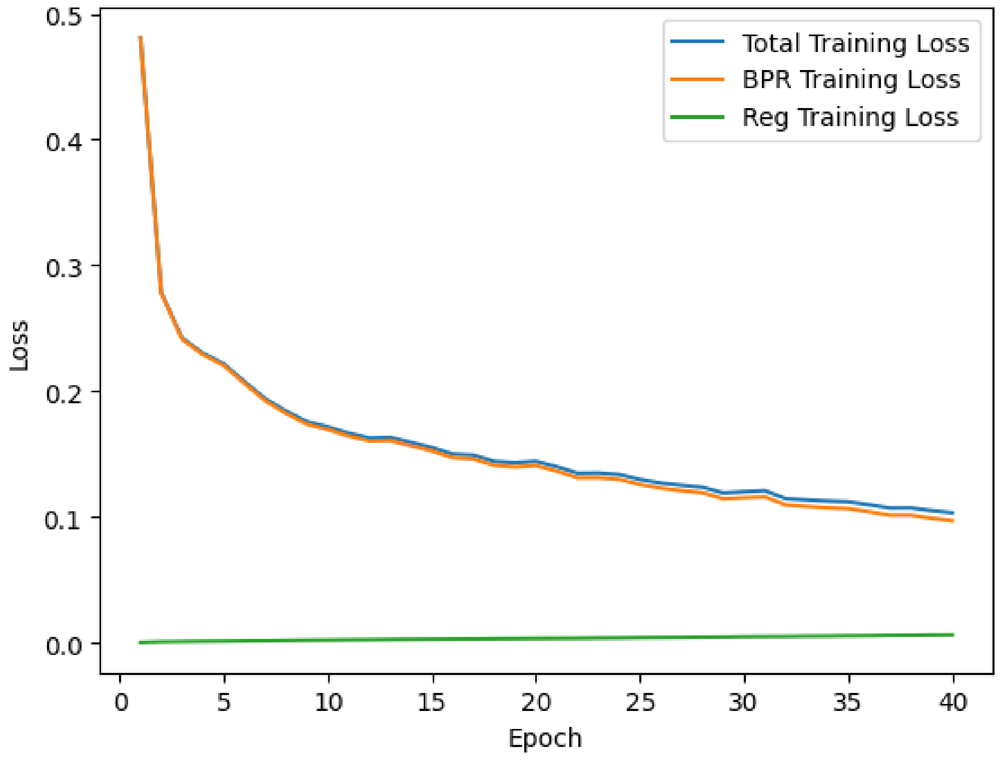
Evaluation

Evaluation Function: get_metrics
The evaluation of the movie recommender system is performed using the get_metrics function, calculating mean recall and mean precision. Recall represents the proportion of relevant items successfully recommended, while precision indicates the accuracy of the recommendations.

Results:

You can see metric here on the graphs:


Recall
Precision
Metrics
Epoch


Total Training Loss
BPR Training Loss
Reg Training Loss
Loss
Epoch