Movie Recommender System

Igor Alentev
i.alentev@innopolis.university

December 3, 2023

1 Introduction

In this work we will consider the development of a movie recommender system. Given the information about the user and his movie preferences, the neural network should suggest good movies to watch.

Generally, the work should provide a good overview of *Graph Neural Networks* and *Graph Convolutional Networks* in particular.

Moreover, we will briefly cover several questions regarding recommender systems:

- Negative sampling
- Loss functions
- Recommeder system metrics

2 Data Analysis

During the solution of this assignment I will not consider any additional data sources, or external datasets. Consequently, I will use only the provided MovieLens-100k dataset.

While the dataset contains a lot of additional metadata, I will use only the most important fracture of it to build the analysis and solution for the given task.

First and foremost, we can consider the genres of the movies in the dataset as depicted on figure 1.

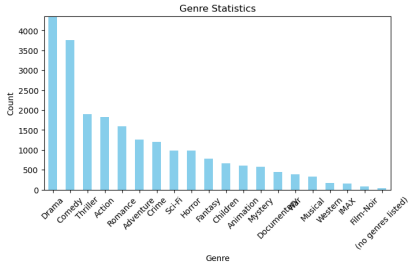

Figure 1: Genres distribution

We can conclude, that the dataset is pretty imbalanced, which is itself a problem. It is safe to assume that this distribution of the movies will affect the final predictions of the solution. For instance, most probably the model will try to propose dramas and it will be difficult to derive useful Noir recommendations. For the sake of simplicity, I will ignore the imbalance.

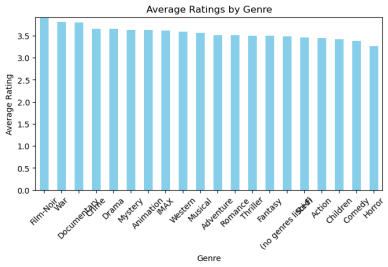

Figure 2: Average raiting by genre

Moreover, from figure 2 we can derive, that popularity of the genre does not necessarily mean it's quality. For instance, if we will consider comedy, which is the second popular genre, it's overall rating is the second from the bottom. It represents the difficulty the recommender system has to overcome. Learning the preferences is not enough as general genre recommendation will lead to poor results.

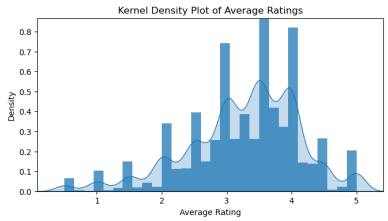

Figure 3: Average movie ratings

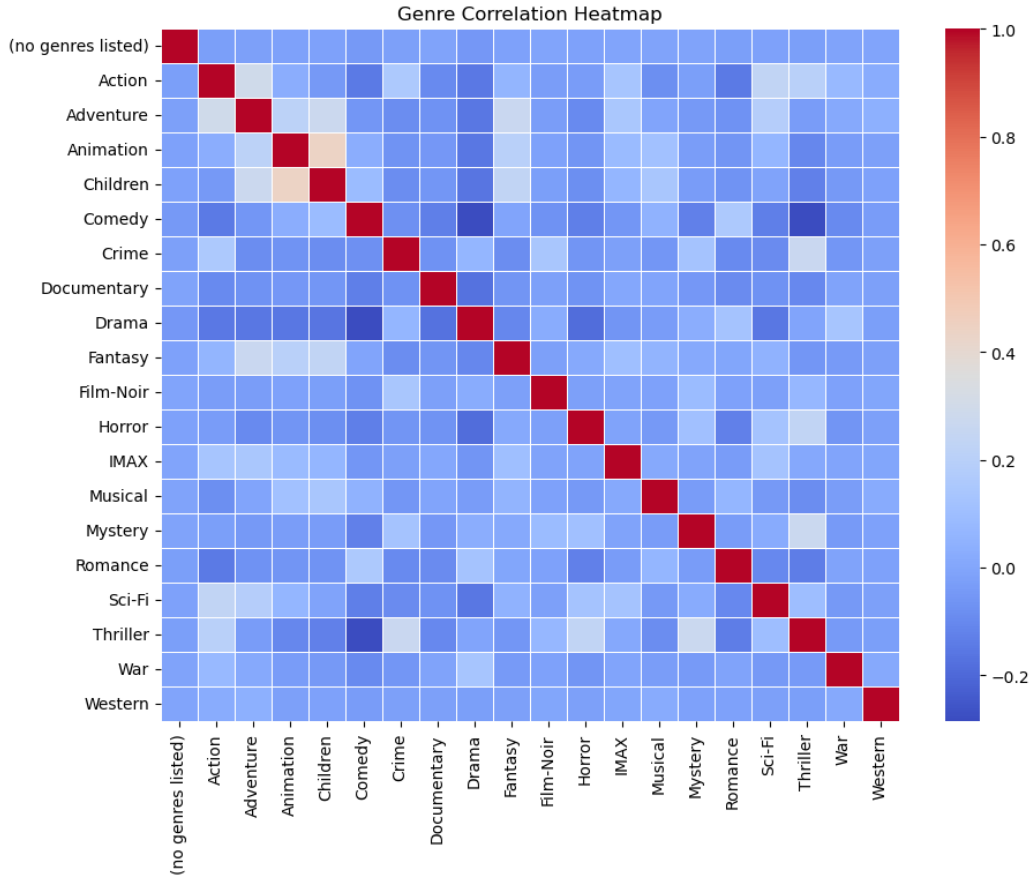

Figure 4: Average genre ratings

From user rating on figure 3 we can learn interesting properties as well. For instance, the rating distribution seem to be normal except some outliers. But these outlieres are not random as well. They tend to reflect that users prefer to give *pretty* numbers as ratings. For instance, integer numbers, or half-numbers like 1.5 or 2.5.

Moreover, candlesticks plots show that while Film-Noir tends to be very rare genre, even lowest ratings are high. However, the horror movies tend to have very large ratings gap. Even though most of the grades are relatively low, leading to low average.

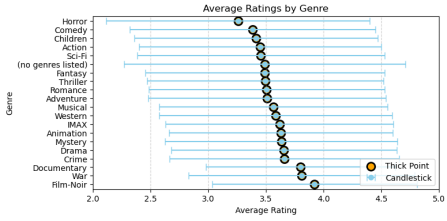

Figure 5: Genres rating candles

Finally, we can consider the correlation data of the genres. This data mostly reflects how the genres work along with each other. For instance, if a movie is a comedy, it is most prob-

ably not a thriller, as well as children movies most often tend to be animation.

The correlation data may seem redundant. However, carefully incorporating these properties into the model, we can try to learn the correlation. For instance, if a user prefers mostly comedy, probably model should not recommend thrillers a lot.

Finally, successfully learning the properties we discovered in the data, we can properly build the model, learning these patterns and improving overall performance. Consequently, we can proceed to the model implementation in the next chapter.

3 Implementation

3.1 General Idea

The structure of the dataset, as well as, to be honest, recent labs of the **Practical Machine Learning and Deep Learning** course inspire to use the graph representation of the data.

We need to address the problem with corresponding tool. For instance, Graph Neural Network [1] seems to be pretty obvious and good solution.

3.2 Stack

For such kind of task, especially the **GNN** we can use the corresponding tools. For instance *PyTorch* [2] and *PyG* [3]. Most of the testing was performed using *Google Colab* capabilities, since I am ~~proud~~ owner of the **AMD** graphics card.

To properly address the properties of the data we may need the special architecture. For example, *Graph Convolutional Network*. Convolutional layers may solve these questions. One of the possible solutions is *LightGCN* [4]. We may be particularly interested in the structure of the convolutional layer architecture and it is presented on the figure 6.

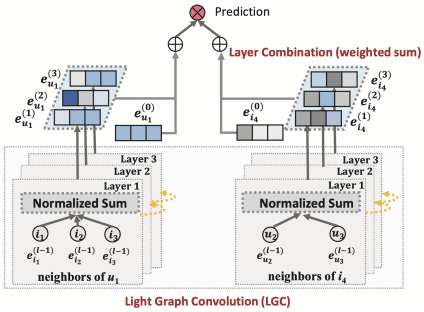

Figure 6: LGC Layer

Between each layer, LightGCN uses the following propagation rule for user and item embeddings.

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

$$e_i^{(k+1)} = \sum_{u \in N_i} \frac{1}{\sqrt{|N_i|}\sqrt{|N_u|}} e_u^{(k)} \quad (2)$$

N_u: the set of all neighbors of user u (items liked by u)

N_i: the set of all neighbors of item i (users who liked i)

$e_u^{(k)}$: k-th layer user embedding

$e_i^{(k)}$: k-th layer item embedding

3.3 Loss Function

I have decided to use Bayesian Personalized Ranking (BPR) loss [5], as I think that pairwise objective which encourages the predictions of positive samples to be higher than negative samples for each user will properly reflect the given task.

$$L_{BPR} = -\sum_{u=1}^{M} \sum_{i \in N_u} \sum_{j \notin N_u} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda ||E^{(0)}||^2 \quad (3)$$

$\hat{y}_u$: predicted score of a positive sample

$\hat{y}_{uj}$: predicted score of a negative sample

λ: hyperparameter which controls the L2 regularization strength

4 Considerations

4.1 Disadvantages

- Firstly, I would like to point out the limitation of the data used. Even though it is not the problem with the overall approach, it is worth mentioning. In particular, the dataset has the genre imbalance as can be seen on figure 1, which is pretty important issue, but is not addressed in this work (in data preprocession, but consider section 5).

- One of the main limitations of *GNN* is the inability of distinguishing the different structures of the graphs. This issue is related to the graph isomorphism. It is hard to say whether this concrete limitation affecting the problem we are trying to solve. However, since we are considering graph structures, we need to take into consideration every possible issue. Moreover, one of the possible solutions if *Graph Isomorphism Network* [6]

- I do not see any point in considering noise vulnerability of the *GNN*s, since this issue is clearly not related to the problem considered.

- Furthermore, *GNN*s suffer from time-space complexity issues. It is very difficult to scale such a network. Number of edges in dense graph can reach up to $O(n^2)$. Therefore, all the computations difficulty and space requirenments raise drastically.

4.2 Advantages

- If the data represented in the natural for it form (i.e. graphs), it allows to capture difficult connections between the data nodes. Learning from not the data itself, but the connections as well, allows for better accuracy and overally better results.

- Moreover, natural representation of the data increases the interpretability of the processes behind the scenes, as well as, the interpretability of the final results.

- The overall generality of the approach allows to apply the theory to a wide variety of the problems. Since many real-world structures can be respresented as graphs, different interpretations are possible.

    1. Convolution
    2. Attention
    3. LSTM
    4. RNN
    5. CV
    6. and many others

5 Training Process

Since we have already discussed the model architecture, loss, there not much left. Evaluation will be considered in further chapters, so here we will rather focus on the process of training itself.

Firstly, we need to consider the learning constants which were used during the training process.

```
epochs = 1e4
batch_size = 2^10
lr = 1e-3
lambda = 1e-6
```

Moreover, I am using mini-batch sampling for feeding the training data to the *GNN*. Sampling is very important for proper learning process.

In detail, sampling should be done in a very accurate way. The role of negative sampling [7] is particularly important. For instalnce, the idea behind is pretty similar. Remembering the class imbalance depicted on figure 1, we have to deal somehow with it. One of the ideas is to build *anti-graph* — graph on *anti-edges* (non-existent edges). Sampling from both graphs we can try to bridge the gap in the data.

This tricky sampling strategy allows to build a proper recommender solution.

The split proportions are 80/10/10 for training, validation, test, correspondingly.

In general these are all specific implementation details of the training process.

6 Evaluation

$$\text{Recall} = \frac{TP}{TP + FP} \tag{4}$$

$$\text{Precision} = \frac{TP}{TP + FN} \tag{5}$$

These are standard metrics for models, but we need a particular modification *Recall@K* and *Precision@K*. It means, that we are interested in proportion of relevant items in the top-K predictions (recommendations) given by our solution.

Another important metric is Discounted Cumulative Gain (DCG). The main purpose of the metric is to consider the order of recommendations. For instance, how relevant and how high in the recommendation list the result is.

$$\text{DCG}_\text{p} = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2 (i + 1)} \tag{6}$$

p: a particular rank position

$rel_i \in \{0, 1\}$: graded relevance of the result at position i

Further extension is Idealised Dicounted Cumulative Gain (IDCG), namely the maximum possible DCG, at rank position p.

$$\text{IDCG}_\text{p} = \sum_{i=1}^{|REL_p|} \frac{2^{rel_i} - 1}{\log_2 (i + 1)} \tag{7}$$

$|REL_p|$: list of items ordered by their relevance up to position p

Finally, the actual metric we are going to use, concludes the DCP extension and is formulated as Normalized DCG [8].

$$\text{nDCG}_\text{p} = \frac{\text{DCG}_p}{\text{iDCG}_p} \tag{8}$$

This concludes the evaluation information of the proposed solution.

7 Results

Loss function expectedly shows convergence and decreases quickly.

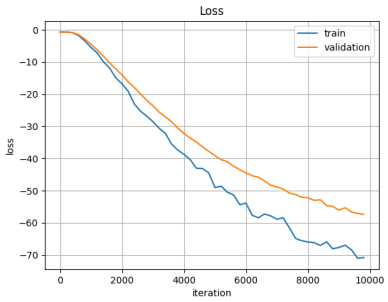

Figure 7: Loss (BPR)

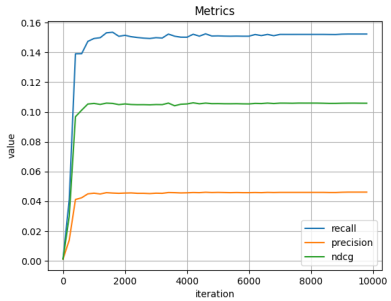

Figure 8: Metrics

However, loss function is negative, which can be considered pretty strange. I believe, that the issue is with the implementation, but not sure where. It is fine, since the general behaviour of loss is as expected.

Metrics behave pretty good as well. All three metrics converge to the certain value within some time interval.

Even though the overall loss continuous to degrease on both train and validation, metrics do not tend to further increase, so there is no point in continuing the training. Without any meaningful changes to the model, it would be difficult to improve the results.

Bibliography cited

[1] J. Zhou, G. Cui, S. Hu, *et al.*, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020, ISSN: 26666510. DOI: 10.1016/j.aiopen.2021.01.001. (visited on 12/03/2023).

[2] *PyTorch*.

[3] *PyG*.

[4] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, *LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation*, Jul. 2020. arXiv: 2002.02126 [cs]. (visited on 12/03/2023).

[5] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, *BPR: Bayesian Personalized Ranking from Implicit Feedback*, May 2012. arXiv: 1205.2618 [cs, stat]. (visited on 12/03/2023).

[6] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, *How Powerful are Graph Neural Networks?* Feb. 2019. arXiv: 1810.00826 [cs, stat]. (visited on 12/03/2023).

[7] Z. Yang, M. Ding, C. Zhou, H. Yang, J. Zhou, and J. Tang, *Understanding Negative Sampling in Graph Representation Learning*, Jun. 2020. arXiv: 2005.09863 [cs, stat]. (visited on 12/03/2023).

[8] O. Jeunen, I. Potapov, and A. Ustimenko, *On (Normalised) Discounted Cumulative Gain as an Off-Policy Evaluation Metric for Top-n Recommendation*, Nov. 2023. arXiv: 2307.15053 [cs]. (visited on 12/03/2023).