LightGCN+
A Graph Convolutional Network for Matrix Completion

Alain Joss
Solomon Thiessen
Antoine Suter
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—Personalized recommendation has become essential for businesses to enhance user experience and satisfaction. In this context, we explore the task of collaborative filtering within the framework of matrix completion. While the primary goal of collaborative filtering is to rank preferences for unobserved user-item interactions, predicting ratings can provide deeper insights into user preferences, thus improving decision-making across various use cases.**

In this work, we propose LightGCN+, a graph convolutional encoder-decoder network designed to predict ratings for unobserved user-item interactions, extending the original LightGCN model. LightGCN+ incorporates the message passing logic of LightGCN in its hidden layer, adapting it to handle a set of real valued user-item ratings, rather than binary user-item interactions. The decoder then concatenates these representations and passes them through a multi-layer perceptron (MLP) to predict ratings, rather than just ranking items. This approach enhances the model's ability to deliver accurate and personalized recommendations.

I. Introduction

Collaborative filtering refers to the task of predicting user ratings of items by leveraging past relationships between users and items. Traditional methods of accomplishing this task include matrix factorization (Alternating Least Squares) and Neural Network methods. Recent surveys [**?**] have highlighted the effectiveness of matrix completion methods in recommendation systems, outlining the mathematical models and computational algorithms involved, as well as their applications beyond traditional recommendations However, we believe that a graph-based approach may perform better than these standards as it models the interaction between users and items more explicitly.

Recent advancements in graph-based models have made the utility of graph-based methods apparent. Graph Convolutional Networks (GCNs), in particular, have been applied to collaborative filtering and recommendation systems with great success. By modeling user-item interactions as a bipartite graph, it is possible to apply GCN methodology to this problem. Previous work in [1] showed state of the art performance using a graph auto-encoder and bilinear decoder on popular movie rating datasets.

In this paper, we propose a novel approach to collaborative filtering using GCNs. Our model is inspired by the LightGCN [2] architecture that saw commendable success when applied to recommendation systems. We adapt the previously proposed model by considering user-item rating when computing user and item embeddings. We then employ MLP architecture to get output from high-dimensional concatenated embeddings. Finally, we ensemble multiple trained models with different hyperparameters and stochastic instantiations via weighted combination to decrease prediction variance and increase performance on unseen data.

In short: we represent user-item interactions in a bipartite graph and introduce a GCN-based model adapted for matrix completion tasks in collaborative filtering, we perform rigorous experiments to show the effectiveness of our model compared with standard collaborative filtering baselines, and we give a thorough analysis of the model's performance by discussing the significance of hyperparameter selection and architectural design decisions.

II. Model and Methods

A. Problem Formulation

Given a partially observed (sparse) user-item rating matrix $R \in \mathbb{R}^{m \times n}$, where m is the number of users and n is the number of items, the task of matrix completion consists of predicting the missing ratings in R. To tackle the problem using graph-based methods, the first step is to represent the user-item interactions as a bipartite graph. we construct a bipartite graph $G = (U, I, E)$, where U and I are the sets of users and items, respectively, and E is the set of edges connecting users to items. The adjacency matrix $A \in \mathbb{R}^{(m+n)\times(m+n)}$ of the graph G is defined as:

$$A = \begin{bmatrix} 0 & R \\ R^T & 0 \end{bmatrix} \tag{1}$$

B. Preprocessing

As a preprocessing step, we standardize the rating matrix R using z-scores, applied separately over columns and rows to obtain Z_{col} and Z_{row}, having zero mean and unit variance along their respective dimensions. This ensures that ratings are comparable across items in Z_{col} and users in Z_{row}. Subsequently, we define the standardized adjacency matrix $\hat{A}$ as:

$$\hat{A} = \begin{bmatrix} 0 & Z_{\text{col}} \\ Z_{\text{row}}^T & 0 \end{bmatrix}$$

We find that standardizing ratings, specifically by computing Z_{col} and Z_{row}, greatly enhances the model's generalization to unobserved user-item interactions.

The rationale behind this preprocessing step, is that standardizing across columns makes items comparable, while standardization across rows makes users comparable. For example, if an item is highly rated by all users, it indicates that the item is generally well-received, and this information is captured uniformly across the dataset through Z_{col}. Conversely, if a particular user consistently rates items highly, Z_{row} adjusts for this user's rating tendency, allowing the model to focus on the relative preferences of the user rather than their absolute rating scale. This dual standardization process helps in mitigating biases and variances inherent in user and item ratings, leading to more robust and accurate predictions.

This insight is key for the effectiveness of the convolution operation defined in the next section, and more broadly to the overall model's performance.

C. Encoder

1) Graph Convolution: The encoder of LightGCN+ inherits the graph convolution (message passing logic) from the LightGCN model. The graph convolution operation propagates information across the graph, updating the hidden representation of each node based on its neighbors. Using multiple layers of graph convolution allows the model to capture higher-order relationships between users and items. We adapt the graph convolution operation to handle real valued user-item ratings, such that for a given node v, the hidden representation $h_v^{(l+1)}$ at layer $l+1$ is computed as:

$$h_u^{(l+1)} = \sum_{i \in N(u)} \frac{z_{\text{col}_{u,i}}}{\sqrt{|N(u)||N(i)|}} h_i^{(l)} \quad \forall u \in U \tag{2}$$

$$h_i^{(l+1)} = \sum_{u \in N(i)} \frac{z_{\text{row}_{u,i}}}{\sqrt{|N(u)||N(i)|}} h_u^{(l)} \quad \forall i \in I \tag{3}$$

where $N(v)$ is the set of neighbors of node v, $z_{u,i}$ is the edge weight (standardized rating) between user u and item i, and $h_v^{(l)}$ is the hidden representation of node v at layer l. The hidden representations $h_v^{(0)}$ at the input layer represent the learnable embeddings of node v. Finally, the symmetrical normalization term $\sqrt{|N(u)||N(i)|}$ ensures that the hidden representations are scaled appropriately.

The L-layer graph convolution operation can be expressed more compactly in terms of matrix multiplications as:

$$H^{(l+1)} = \tilde{A} H^{(l)} \tag{4}$$

where $\tilde{A} = D^{-\frac{1}{2}} \hat{A} D^{-\frac{1}{2}}$ is the normalized standardized-adjacency matrix, D is the diagonal degree matrix of $\hat{A}$, and $H^{(l)}$ is the matrix of hidden representations at layer l.

2) Aggregator: After L layers of graph convolution, the hidden representations $h_v^{(l)}$ of node v are aggregagated to obtain the final representations. To maximize the expressiveness of the model, we aggregate the hidden representations by concatenating them along the feature dimension, resulting in the final representation h_v:

$$h_v = \text{concat}(h_v^{(0)}, h^{(1)}, \ldots, h_v^{(L)}) \tag{5}$$

This choice is justified by the use of a multi-layer perceptron (MLP) in the decoder, which can take advantage of the concatenation aggregation, in contrast to other aggregation methods such as summation or averaging.

D. Decoder

The decoder of LightGCN+ is a multi-layer perceptron (MLP) that takes the concatenated hidden representations h_u and h_i of user u and item i couples as input, and outputs the predicted rating $\hat{r}_{u,i}$:

$$\hat{r}_{u,i} = \text{MLP}(\text{concat}(h_u, h_i)) \tag{6}$$

The specific choice of the MLP's architecture is a hyperparameter of the overall model.

E. Training

The training objective of LightGCN+ is to predict the ratings of the observed user-item interactions in the training set Ω. The learnable parameters of the model W comprise the embeddings of users and items and the weights of the MLP, and are trained end-to-end using the Adam optimizer. We initialize the embeddings using a normal distribution, with tunable standard deviation σ_{init}, and the weights of the MLP using the Xavier initialization. As loss function we use the mean squared error (MSE) between the predicted ratings $\hat{R}$ and the true ratings R:

$$\mathcal{L} = \frac{1}{|\Omega|} \sum_{(u,i) \in \Omega} (\hat{r}_{u,i} - r_{u,i})^2 + \lambda ||W||_2^2 \tag{7}$$

where λ is the weight decay regularization strength. In addition to weight decay, to regularize the model we apply dropout to the MLP. Finally, we avoid mini-batching to ensure that the model can learn the global structure of the graph, concurrently speeding up convergence.

F. Tuning

The hyperparameters of LightGCN+ include the number of layers L, the dimensionality of the hidden representations K, the learning rate η, the weight decay regularization strength λ, the dropout rate p, and the hyperparameters of the MLP. The MLP's hyperparameters consists of σ_{init}, the number of hidden layers, the number of units per hidden layer, and the activation function f.

Here we start speaking about the specific experiments on the data

After performing an initial grid search, we narrow the search-space by fixing following hyperparameters: $K = 28$, $\eta = 0.01$, $p = 0.5$, and $\lambda = 0.00005$, $\sigma_{\text{init}} = 0.075$, and $f = \text{GELU}$.

We tune the remaining hyperparameters, the number of layers in the graph convolution L, the dimension of the hidden representations K and the MLP's architecture, by performing a simple grid search.

G. Inference

1) Ensembling: In the tuning phase, we find that multiple models with different hyperparameters can achieve similar performance. We take advantage of this insight by ensembling the top M models, decreasing the variance of the predictions. We find that this additional step makes predictions more robust and improves the overall performance of the model.

The ensembled prediction matrix is obtained by computing a weighted average of the predictions of the individual models. The weights are determined by splitting the validation set into a set for fitting the ensemble weights and a set for evaluating the ensemble's performance. We cap the number of models in the ensemble at $M = 10$ to prevent overfitting.

2) Postprocessing: To ensure that the predictions are within the valid rating range, we clip the predictions to the interval $[1, 5]$.

III. Experiments

A. Dataset

We evaluate the performance of LightGCN+ on the MovieLens 100K dataset, a widely used benchmark dataset for collaborative filtering tasks. The dataset consists of 100,000 ratings from 943 users on 1682 movies, with ratings ranging from 1 to 5. We split the dataset into training, validation, and test sets, with 80 perc of the ratings used for training, 10 perc for validation, and 10 perc for testing.

B. Design Choices

We experiment with different design choices to evaluate the impact of each on the model's performance.

1) Matrix Standardization:

2) Graph Convolution: In the multi-layer graph convolution, different normalization techniques can be applied to the standardized adjacency matrix $\hat{A}$. For example, it would be possible to normalize using the inverse degree matrix D^{-1}, or learn normalization parameters $\alpha_{u,i}$ and $\beta_{u,i}$ for each edge. To this end, the GAT [3] attention mechanism could be used. We decide to apply symmetric normalization, following the simplicity philosophy of the original LightGCN.

3) Layer Aggregation:

4) MLP Architecture: In the graph convolution we experiment with different normalization techniques, including symmetric normalization, row normalization, and column normalization.

C. Model Discovery Process

IV. Results

Hyper params, rmse, different methods attempted, graphics showing loss etc.

At first, we achieved the best results with the following hyperparameters:

include them in a list, give the relevant score

Following this initial breakthrough, we searched over a parameter grid to find the best set, which led us to increase the number of layers L. This was initially unsuccessful for us as we didn't allow the model to train for enough epochs. However, following intuition, more layers allow for better aggregation of information by each node and therefore a more accurate prediction of the rating a user will give to an item. and other stuff

A. Baselines

To assess the value our solution adds to the field of collaborative filtering, we compare it's performance against that of some standard models.

1) Alternating Least Squares: Alternating Least Squares (ALS) turns an otherwise quartic matrix factorization problem into one that is only quadratic by alternating between optimizing the user matrix while holding the item matrix constant and vice versa. When applied to the public test set, it achieved a RMSE score of 1.426

2) Neural Collaborative Filtering: Our Neural Collaborative Filtering (NCF) model is fairly vanilla. We used two embedding layers followed by a feed forward neural network. The ReLU activation function was used at all layers besides the final layer. We trained using the Adam optimizer with a learning rate of 0.001, MSE loss, and embedding dimension 16. After 25 epochs, this model achieved a RMSE score of 1.094 on the public test set.

V. Discussion

Strengths and weaknesses.

VI. Summary

Our model expands on the work done in [1] for recommender systems by adapting the LightGCN architecture for use in collaborative filtering tasks. We accomplish this by including the standardized user-item rating in the aggregation step (i.e. when updating node embeddings) and passing the resulting pairs of concatenated user-item embeddings through a MLP to generate a prediction. This model significantly outperforms standard baseline models for the same task and achieves impressive results on real data.

REFERENCES

[1] R. van den Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *arXiv preprint arXiv:1706.02263*, 2017.

[2] 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 (SIGIR '20)*. New York, NY, USA: ACM, 2020, pp. 639–648.

[3] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ