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 sparse user-item rating matrix $R \in \mathcal{R}^{m \times n}$, with observed ratings $r_{u,i}$ for observed interaction pairs $(u, i) \in \Omega$, the task of matrix completion consists of predicting the missing ratings in R.

Graph based methods approach matrix completion by representing the user-item interactions as a bipartite graph $G = (U, I, E)$, where U and I are the sets of users and items (nodes), respectively, and E is the set interactions (edges) with weights $r_{u,i}$. The adjacency matrix $A \in \mathcal{R}^{(m+n)\times(m+n)}$ of the bipartite graph G is defined as:

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

The adjacency matrix may be preprocessed using different normalization techniques. We define the standardized adjacency matrix $\hat{A}$ as:

$$\hat{A} = \begin{bmatrix} 0 & Z_{\text{col}} \\ Z_{\text{row}}^T & 0 \end{bmatrix} \tag{2}$$

where Z_{col} and Z_{row} represent the standardized user-item ratings, derived using z-score normalization on the columns and rows of R, respectively.

B. Model Architecture

The LightGCN+ model consists of an encoder-decoder architecture, with the encoder comprising a multi-layer graph convolutional network and an aggregator function, and the decoder consisting of a multi-layer perceptron (MLP).

Nodes v in the graph (users and items) are represented by embeddings $e_v \in \mathbb{R}^K$, for all $v \in U \cup I$, and are learned end-to-end during training.

C. Encoder

The encoder of LightGCN+ consists of a multi-layer graph convolutional network and an aggregator function. It takes the embedding matrix $E \in R^{(m+n)xK}$ and the standardized adjacency matrix $\hat{A}$ as inputs, and generates hidden representations H, which are then passed to the decoder.

1) Graph Convolution: Graph convolution is a form of message passing over the graph, where each node updates its representation by aggregating information from its neighbors, reflecting the local graph structure. Applying this operation multiple times, in a layered fashion, enables the model to capture higher-order relationships between nodes. For a given node v, the hidden representation $h_v^{(l+1)}$ at layer $l+1$ is computed as:

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

$$h_i^{(l+1)} = \sum_{u \in N(i)} \frac{z_{\text{row}_{u,i}}}{\sqrt{|N(u)||N(i)|}} h_u^{(l)} \quad \forall i \in I \tag{4}$$

where $N(v)$ is the set of neighbors of v. Finally, the symmetrical normalization term $\sqrt{|N(u)||N(i)|}$ ensures that the hidden representations are scaled appropriately. Note that the hidden representations $h_v^{(0)}$ at the input layer correspond to the learnable embeddings e_v.

The L-layered convolution operation can be expressed more compactly as:

$$H^{(l+1)} = \tilde{A} H^{(l)} \tag{5}$$

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 aggregegagated to obtain the final representations. In contrast to the original LightGCN model, instead of averaging we use concatentation as aggregation function:

$$h_v = \oplus(h_v^{(0)}, h^{(1)}, \ldots, h_v^{(L)}) \tag{6}$$

where $\oplus$ denotes concatenation operator. Concatentating layered representations is a natural choice of aggregation for LightGCN+, given the use of a multi-layer perceptron (MLP) as the decoder.

D. Decoder

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

$$\hat{r}_{u,i} = \text{MLP}(\oplus(h_u, h_i)) \tag{7}$$

During training, user-item pairs $(u, i) \in \Omega$ (the set of observed interactions), while during inference, $(u, i) \notin \Omega$.

E. Objective Function

A natural choice of loss function for rating prediction is the mean squared error (MSE) between the predicted ratings $\hat{R}$ and the true ratings R:

$$\mathcal{L} = \frac{1}{|\Omega|} \sum_{(u,i) \in \Omega} (\hat{r}_{u,i} - r_{u,i})^2 \tag{8}$$

Other loss functions could be used, depending on the specific use case. Learnable parameters of the model include user and item embeddings and the parameters of the MLP.

III. Experiments

A. Dataset

We evaluate the performance of LightGCN+ on the ETHZ-CIL-Collaborative-Filtering-2024 dataset. The dataset consists of $10,000$ users and $1,000$ items, with 1176952 observed ratings, which corresponds to a sparsity of 88.23%. The ratings are integers in the range $[1, 5]$, with a mean rating of 3.86 and a standard deviation of 1.12.

B. Preprocessing

We find that normalizing the $\hat{A}$ using Z_{col} and Z_{row}, as shown in Equation 2, significantly enhances the model's generalization ability. The rationale behind this dual standardization is that it reduces biases and variances from individual user or item rating patterns. This makes them comparable across the graph, which is crucial for the effectiveness of the convolution operation.

C. Training

The model is trained end-to-end using the Adam optimizer, with learning rate $\eta = 0.1$, weight decay $\lambda = 0.00005$, and dropout rate $p = 0.5$. We avoid mini-batching to ensure that the model can learn the global structure of the graph, while also speeding up convergence. To prevent overfitting we tuned λ and p accordingly.

D. Inference

To predict ratings of unobserved user-item interactions, a simple forward pass through the learned model is performed. To ensure that the predictions are within the valid rating range we clip them to the interval $[1, 5]$.

E. Model Design Choices

During the design phase of the model we conduct several experiments to determine the best architecture for LightGCN+.

1) Matrix Standardization: We train the model without standardizing the ratings, with only column standardization, with only row standardization, and with both column and row standardization, finding that the latter yields the best performance.

2) Graph Convolution: In the multi-layer graph convolution, different normalization techniques can be applied to the message passing operation. 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 find that attention performs best with a single layer, but that a simple symmetric normalization enables to perform message passing across multiple layers, which can be effectively leveraged by the MLP in the decoder.

3) Layer Aggregation: Because at tghe output layer we use a MLP, aggregating by concatenation is a natural choice. We verify our hypothesis, trying out even averaging and weighted averaging, with $L+1$ learnable weights, verifying the initial intuition.

4) MLP Architecture: We perform extensive hyperparameter tuning on the MLP architecture, finding that a simple two-layer MLP with 64 units per layer and GELU activation function performs best.

F. Hyperparameter Tuning

After performing an initial investigation of the hyperparameters on isolation, we decide narrow the search-space by fixing the 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. We find that multiple models with different hyperparameters can achieve similar performance.

G. Ensembling

We take advantage of the insights gained in the tuning of the model by ensembling the top M performing models, with the aim of 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. Following the oiginal simplicity principle of the model and to avoid overfitting, we cap the number of models in the ensemble at $M = 10$.

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