18127039 - Lâm Ngọc Phương Anh
18127046 - Lư Ngọc Liên
18127272 - Nguyễn Thị Anh Đào

Subject: Big Data Application
CSC14114

Topic 1 - Recommender Systems

Question 01:

Name of the paper*: "LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation"*
Author: Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, Meng Wang
Release: 2 Feb 2020
Link: https://paperswithcode.com/paper/lightgcn-simplifying-and-powering-graph

I. General ideal

The common solution for collaborative filtering nowadays is applying Graph Convolution Network (GCN) with two most common designs: feature transformation and nonlinear activation. However, GCN has redundant neural network operations, which is used for graph classification. According to the empirical result, these operations degrade recommendation performance. Therefore, LightGCN is created and designed based on GCN but keeps only the most vital components (neighborhood aggregation) that are appropriate for a recommendation, and it helps to improve the performance of the model.

The main idea of LightGCN is: linear propagates user and item embeddings on the user-item interaction graph, then sum the weights of the embeddings have learnt at all layers for the final embedding.

NGCF gets inspired and directly inherits many operations from GCN but does not justify their necessity. To simplify the design of GCN for the recommendation, LightGCN is introduced. LightGCN ablates two main operations: feature transformation and nonlinear activation on NGCF (state-of-the-art model for CF).

II. Algorithms

LightGCN is inspired by GCN and ablation on NGCF. LightGCN architecture is shown in the figure below.

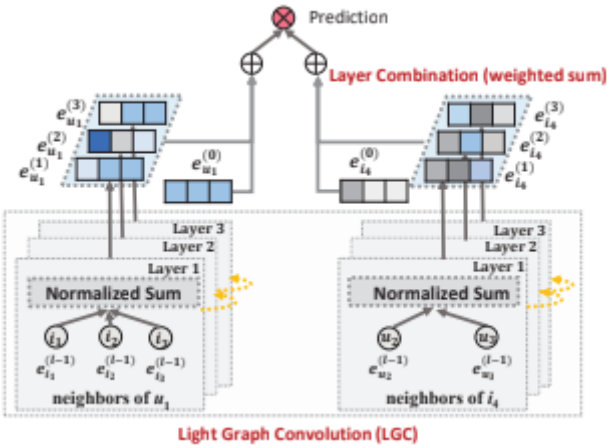

Figure 1. LightGCN architecture

Self-connection, feature transformation and linear activation are removed in LGC, just keeping the sum of neighbor embeddings which is normalized to pass to the next layer. The final representation will be the sum of embedding of all layers (Layer Combination) and return the prediction.

- **Light Graph Convolution (LGC)**: by removing feature transformation and nonlinear activation, focus only on connected neighbors, the propagation rule will be defined as:

$$e_u^{k+1} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} e_i^k$$

$$e_i^{k+1} = \sum_{i \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} e_u^k$$

+ e_u is the ID embedding of user u and e_i is ID embedding of item i
+ e_u^k and e_i^k is delicate embedding of user u and item i after k layer propagation
+ $\mathcal{N}_i$ is the set of user interact with item i, $\mathcal{N}_u$ is the set of items user u interact with.

The term $\frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}}$ is designed based on the standard GCN to avoid embedded scale ascending with graph convolution operations. Moreover we can also apply the L_1 norm to get the same effect.

- **Layer Combination and Model Prediction**: The embeddings at the 0-th layer are the only trainable parameters. The embeddings of higher layers are computed by the equation above (LGC). After getting the embeddings of K layers, the final representations (user and item) are obtained by summing all layers' embeddings.

$$e_u = \sum_{k=0}^{K} \alpha_k e_u^k$$

$$e_i = \sum_{k=0}^{K} \alpha_k e_i^k$$

+ α_k: the layer combination coefficient of layer K.

To make a prediction, LightGCN uses the inner product of the user and the item final representations:

$$\widehat{y_{ui}} = e_u^T e_i$$

- **Matrix Form**: Giving a more intuitive view to discuss and compare with others models.

The adjacency matrix of user-item graph as below:

$$A = \begin{pmatrix} 0 & R \\ R^T & 0 \end{pmatrix}$$

+ R is the user-item interaction matrix ($R \in \mathbb{R}^{M \times N}$ - M, N are the number of users, items respectively). $R_{ui} = 1$ if user u interacted with item i otherwise = 0.

In the 0-th layer, the matrix will be $E^{(0)} \in \mathbb{R}^{(M+N) \times T}$ (T is the embedding size).

Therefore, at layer k+1 the matrix will form as

$$E^{(k+1)} = \left(D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right) E^{(k)}$$

+ D: a diagonal matrix $(M + N) \times (M + N)$ where D_{ii} represents the number of non-zero elements in the i-th row of the matrix A.

Eventually, we obtain the final matrix which is used to predict as using the equation below:

$$E = \alpha_0 E^{(0)} + \alpha_1 E^{(1)} + \cdots + \alpha_K E^{(K)} = \alpha_0 E^{(0)} + \alpha_1 \widetilde{A} E^{(0)} + \cdots + \alpha_K \widetilde{A}^K E^{(0)}$$

+ $\widetilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ (symmetrically normalized matrix)

III. Data

The authors evaluate the model on 3 different datasets: Gowalla, Yelp2018 and Amazon-Book.

- Gowalla: a location-based social networking website where users share their locations by checking-in. The friendship network is undirected and was collected using their public API, and consists of 196,591 nodes and 950,327 edges. We have collected a total of 6,442,890 check-ins of these users over the period of Feb. 2009 - Oct. 2010.
  (from E. Cho, S. A. Myers, J. Leskovec. Friendship and Mobility: Friendship and Mobility: User Movement in Location-Based Social Networks ACM

SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2011
Stanford University, SNAP Gowalla, https://snap.stanford.edu/data/loc-gowalla.html).

- Yelp2018: a subset of Yelp's businesses, reviews, and user data for use in personal, educational, and academic purposes. 8,635,403 reviews, 160,585 businesses, 200,000 pictures, 8 metropolitan areas.
  (from https://www.yelp.com/dataset)
- Amazon-Book: contains 697.053 total records over the period 1 Jan 2020 to 31 Jan 2020 from PromptCloud and DataStock (https://datastock.shop/). Can download the whole dataset in (https://app.datastock.shop/?site_name=Amazon)

IV. Experimental

Recall and ndcg are the evaluation metrics used to compare among methods. Besides NGCF, they compare with Mult-VAE (item-based CF use variational autoencoder) and GRMF (use Laplacian regularizer to smooth matrix factorization).

In terms of hyperparameters:

+ For all methods, the size of embedding is 64.
+ Xavier method is used to initialize parameters for embedding.
+ Adam optimization is used for LightGCN.
+ Set $\frac{1}{K+1}$ to the coefficient for combining layers, where K is the number of layers.
+ The number of epochs used is 1000.
+ With LightGCN, the most essential hyperparameter is the coefficient Lambda of L2 regularization. Empirically, the authors show that if the value of this hyperparameter is over 1e-3, the performance degrades swiftly.

Comparison between LightGCN and NGCF:

+ LightGCN surpasses NGCF in all cases, the details are in the table below.

Dataset		Gowalla		Yelp2018		Amazon-Book	
Layer #	Method	recall	ndcg	recall	ndcg	recall	ndcg
1 Layer	NGCF	0.1556	0.1315	0.0543	0.0442	0.0313	0.0241
	LightGCN	0.1755(+12.79%)	0.1492(+13.46%)	0.0631(+16.20%)	0.0515(+16.51%)	0.0384(+22.68%)	0.0298(+23.65%)
2 Layers	NGCF	0.1547	0.1307	0.0566	0.0465	0.0330	0.0254
	LightGCN	0.1777(+14.84%)	0.1524(+16.60%)	0.0622(+9.89%)	0.0504(+8.38%)	0.0411(+24.54%)	0.0315(+24.02%)
3 Layers	NGCF	0.1569	0.1327	0.0579	0.0477	0.0337	0.0261
	LightGCN	0.1823(+16.19%)	0.1555(+17.18%)	0.0639(+10.38%)	0.0525(+10.06%)	0.0410(+21.66%)	0.0318(+21.84%)
4 Layers	NGCF	0.1570	0.1327	0.0566	0.0461	0.0344	0.0263
	LightGCN	0.1830(+16.56%)	0.1550(+16.80%)	0.0649(+14.58%)	0.0530(+15.02%)	0.0406(+17.92%)	0.0313(+18.92%)

*The scores of NGCF on Gowalla and Amazon-Book are directly copied from Table 3 of the NGCF paper (https://arxiv.org/abs/1905.08108)

Figure 2. Performance comparison between NGCF and LightGCN at different layers

Comparison between LightCGN and state-of-the-arts:

+ The performance of LightGCN on all three datasets is completely outperforming other methods, which is figured in the table below.

Dataset	Gowalla		Yelp2018		Amazon-Book	
Method	recall	ndcg	recall	ndcg	recall	ndcg
NGCF	0.1570	0.1327	0.0579	0.0477	0.0344	0.0263
Mult-VAE	0.1641	0.1335	0.0584	0.0450	0.0407	0.0315
GRMF	0.1477	0.1205	0.0571	0.0462	0.0354	0.0270
GRMF-norm	0.1557	0.1261	0.0561	0.0454	0.0352	0.0269
LightGCN	**0.1830**	**0.1554**	**0.0649**	**0.0530**	**0.0411**	**0.0315**

Figure 3. The comparison of overall performance among LightGCn and competing methods

+ According to the information given from the table, Mult-VAE has the most effective performance among the baselines.

The effect of embedding smoothness is the main reason for the effectiveness of LightGCN.

V. Pros and Cons

1. PROS

- LightGCN is more interpretable, basically simple to train and maintain, technically easy to analyze the model behavior and adjust it towards more adequate directions.
- Rejecting many unnecessary GCN's operations such as feature transformation and nonlinear activation. Since each node in the user-item interact graph of CF is only described by a one-hot ID and without any concrete semantics besides. Feeding this ID embedding to multiple layers of nonlinear feature transformation makes the model difficult for training (the numerical evidence from section 2.2 of the authors' paper).
- Only the connected neighbors are aggregated without self-connection because the layer combination operation captures the same effect.
- Combining layers helps the algorithm to prevent being over smoothing when increasing the number of layers. Due to the embedding capturing different semantics at each layer, combining them makes the final representation more comprehensive. The users overlap on the interacted items are smoothed by the second layer, which makes the embeddings evolve smoother and more appropriate for the recommendation (inspired by APPNP).
- LightGCN does not use dropout mechanisms for preventing overfitting because they do not have feature transformation weight matrices,

alternatively using L2 regularization on the embedding layer to avoid overfitting.

- Because of simplicity and accuracy, LightGCN is suitable for use in online industrial scenarios.

2. CONS

- LightGCN does not tune the alpha k (the importance of the k-th layer embedding - the layer combination coefficient) and sets it uniformly as 1/(K+1).

Question 2:

1. Dataset

The dataset we use is MovieLens 1M (https://files.grouplens.org/datasets/movielens/ml-1m.zip) contains 1,000,209 ratings of film from 6040 users on 3900 movies, which was released on 2/2003. The dataset has 3 files:

- user.dat: UserID::Gender::Age::Occupation::Zip-code
- movies.dat: MovieID::Title::Genres
- ratings.dat: UserID::MovieID::Rating::Timestamp

The ratings.dat file only contains explicit feedback of users (expressed by 1-5).

2. Model

To deal with collaborative filtering which is very common in recommendation system, we user ALS (alternating least squares) provided by sparkML with following parameters:

```
ALS( userCol = "UserID", itemCol = "MovieID", ratingCol = "Ratings",
coldStartStrategy = "drop", nonnegative = True)
```

3. Train model

- We split ratings.dat into 2 parts: train (80%) and test (20%)
- Train model with train dataset
- Predict with test dataset

4. Evaluation

RMSE	Hit rate	NDCG
0.87	**0.01**	**0.956**

The model works well if the training dataset has more explicit feedback and few implicit responses, the user-item interaction dataset is not very sparse.

Question 3:

1. Dataset

Anime Recommendations Database contains 7,8 millions ratings from 73,516 users on 12,294 anime.

You can download the whole dataset at: [Anime Recommendations Database]

The dataset has 2 files:

- Anime.csv: anime_id;name;genre;type;episodes;rating;members
- Rating.csv: user_id;anime_id;rating

2. Model

```
ALS( userCol = "user_id", itemCol = "anime_id", ratingCol = "rating",
coldStartStrategy = "drop", nonnegative = True)
```

3. Train Model

+ We split Rating.csv into 2 parts: train (80%) and test (20%)
+ Train model with train dataset
+ Predict with test dataset

4. Evaluation

RMSE	Hit rate	NDCG
1.168	**0.003**	**1.171**