Group 21: MixGCF

Shih-yang LIU
20383290
sliuau@connect.ust.hk

Park Chan Ho
20835621
chparkaa@connect.ust.hk

Abstract

The task of recommendation systems is to produce personalized recommendations for users. Recently, graph based collaborative filtering (Schafer et al., 2007) approaches have been widely considered as the state-of-the-art solutions. However, most of the recommendation system data-sets doesn't includes negative example or simply confront serious data sparsity problems, this create a fundamental challenge of distilling quality negative signals form the implicit feedback for graph recommendation system algorithm, as they all follows the setting of BPR loss function(Rendle et al., 2012) which requires negative sampling. As negative sampling hasn't been largely explored especially in Graph neural networks (GNN) based model, MixGCF proposed to utilize the hop mixing technique to synthesize hard negatives, rather than sample raw negatives from data. In this research project, we come up with two possible direction to improve MixGCF, one is to incorporate co-training framework into the negative synthesizing segment, so that the two classifiers from co-training can jointly decide whether the synthesized hard negatives is a quality negatives or not. And second is to add in fair regularization to remove strong bias to account for sparsity in the data.

[1] [2]

[1]Shih-yang LIU (20383290) Supervisor: Dik Lun Lee, Research Topic: Machine learning, Information Retrieval and Computational Finance. I declare that this project is done solely within this course but not other courses.

[2]Park Chan Ho (20835621) Supervisor: Qifeng Chen, Research Topic: Deep learning and Computer Vision. I declare this project is done solely within this course but not other courses.

1 Introduction of Recommendation System and the problem

Recommender systems have been developed to mitigate the problem of information overload and have been applied in many real-world scenarios. These systems are now used by almost every online content provider and e-commerce platform. Typically, users on such platforms interact with provided items in multiple ways; for example, users on online streaming services such as Netflix and Spotify click, play, share or rate videos and songs, and users on e-commerce platforms such as Amazon and Shopify view, purchase, or leave reviews for items. To extract the information encoded in such complicated user-item interactions, Collaborative Filtering has been a fundamental building block toward exploiting the past user-item interactions to achieve the prediction and effective personalized recommendation. Currently, the most prevalent approach for utilizing CF is to map users and items into high dimensional latent space (embeddings) and conduct the preference prediction based on the similarity of the user and item vectors. Matrix factorization is an early model that works by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices. Later models incorporate different techniques into the recommendation systems. For example, Collaborative Deep Learning integrates the deep representations learned from rich side information of items, Neural Collaborative Filtering replaces the original Matrix Factorization interaction function with neural networks to introduce nonlinearity into the model, Deep item based collaborative filtering uses deep neural networks to model the high order interactions among different items to provide better representation and LightGCN simplifies the Graph Convolution Network by discarding the nonlinear activation while improving the efficiency and effectiveness of the model and is currently considered as the state of the art.

Though Graph neural networks (GNNs) have recently been widely seen as state-of-the-art collaborative filtering (CF) solutions, as such GNN models possess the ability to model high order information in the user-item interactions graph, the exploration of how to conduct proper negative sampling remains untouched. Therefore, in this work, we would like to explore different ideas and try to improve the traditional negative sampling methods.

2 Related works

Negative sampling in graph mining can be traced back to skip-gram (Mikolov et al., 2013) training, which aims to reduce the computation complexity and prevent the model from degeneracy. There are several approaches to find the negative samples in the recommendation systems (Huang et al., 2021), static sampling, hard negative sampling, GAN-based sampling, and graph-based sampler. Static sampling just takes the samples randomly from the uniform distribution, such as Bayesian Personalized Ranking (Rendle et al., 2012). Hard negative sampling (Zhang et al., 2013) aims to find the negative samples which are hardest to distinguish from the positive samples. It can adjust the decision boundary of the neuron network for better generalization ability. GAN-based sampling (Wang et al., 2017) utilizes the generative adversarial network to play a min-max game in the recommendation, where the generator can generate negative samples. Graph-based sampler leverages the metrics from graphs for sampling. For instance, PinSage (Ying et al., 2018) obtains the negative samples based on the self-defined PageRank score.

Collaborative filtering (CF) is a widely-used method in modern recommender systems.The Matrix Factorization(MF)(Koren et al., 2009) tries to discover the shared factors hidden behind the user-item matrix and transforms them into embedding vectors. HOP-Rec (Yang et al., 2018) collects the user-item interaction matrix while doing the random walk on the graph and incorporates the factorization method on it to get the high-order proximity. With the rapid development of graph convolution network techniques, many works are proposed to improve the recommendation system. Under the assumptions of collaborative filtering(CF), GC-MC (van den Berg et al., 2017) first applies the GCN in this problem and they treat users and items as the same nodes in the graph and tries to figure out the relationship between them.

Regularization and data augmentation are modes of making the model training more robust to new or unseen data. In the context of improving the embedding space representation, Mix-up(Zhang et al., 2018) makes use of linearly interpolating between inputs to create inputs that represent cases that are in between examples. By doing so it lets the model learn a clearer decision boundary. Manifold Mixup (Verma et al., 2019) is another work that does so in the embedding space rather than input space. While fair mixup (Chuang and Mroueh, 2021) regularizes the decision boundary through reducing the disparity between different distribution of inputs. Both lines of work can be seen as an effort to regularize the embedding space structure so that the trajectory between one input to another is smooth so that the decision boundary is clearer.

3 Contribution

In this work, we aim to improve the negative sampling and decision boundary to enhance the model performance in collaborative filtering of lightGCN based on the MixGCF framework. We proposed three direction to improve the MixGCF framework, and our main contributions can be listed as follow,

- LIU, Shih-yang adopts a co-training framework to improve the negative sampling for a harder sampling strategy.

- Park Chan Ho adopts other modes of mix-up in the process of BPR loss calculation.

- Park Chan Ho apply fair regularization method to improve the decision boundary of the embedding space.

- They both evaluate and compare their proposed frameworks and additional modules with the MixGCF framework.

This work helps the general community compare different approaches of negative sampling with some intuition on the capability of different frameworks and modules.

4 Problem Definition

General recommendation system problem can be mathematically formulated as follows:

Training objective

$$\text{Min. } \Sigma_{(u,v)\in(U\times V)} \text{ Loss(Recommendator } ((u,v),Y)) \quad (1)$$

Here Y is the set of binary ground truth interaction between user-item. To measure the effectiveness of personal score predicted by the model, metrics include Recall@N and NDCG@N which are rank-aware evaluation metrics.

User-Item interaction = (U, V)

$$\text{U} = \{\text{u}^i : \text{ith user embedding}\}$$

$$\text{V} = \{\text{v}^i : \text{ith item embedding}\}$$

$$\|U\| = I, \|V\| = J$$

$$\text{P} = \{(\text{j}_+, j_-) : v_{j_+} >_u v_{j_-}\}$$

Here $\geq_u$ implies user u's preference of v_{j_+} over v_{j_-}

As the main idea of MixGCF lies on the negative sample sampling method, the base models used were the existing state of the art methods NGCF and LightGCN.

$$\text{Total loss} = \Sigma^I_{u_i\in U}\Sigma_{(j_+,j_-\in P)}L(u_i, v_{j_+}, v_{j_-})$$

$$\text{L} = \text{BPR Loss} = \ln\sigma(\hat{x}_{u_ij+j_-}) + \ln p(\Theta)(2)$$

Here each of the notations are

$$\hat{x}_{u,j_+,j_-} = \text{Prob. of user } u_i\text{'s preference of}$$

$$\text{j}_1 \text{over } j_2$$

$$= \text{u}_i \cdot v_{j_+} - u_i \cdot v_{j_-}$$

Sigmoid function $= \sigma = \frac{1}{1+e^{-x}}$

$\text{p}(\Theta)$ = model specific regularization.

The major contribution of the paper MixGCF can be seen as how they generate positive-negative item pair. They introduce two methods Hop Mixing and Positive Mixing inspired by [13],[14] and use them together to generate negative item embedding. By employing these pseudo-negative item samples during the training, it has been proven to learn more informative embeddings and draws clearer decision boundaries between negative/positve items.

First, LightGCN need a triplet of a user u, positive item v^+ and negative item v^- to learn user preference with BPR loss function. In the original negative sampling setting, a group of negative items will be randomly picked to form a triplet with u and v^+ where negative indicate that user u has never interact with those items. The intuition behind MixGCF is that such negative sampling approach will not necessarily pick out the items that user dislike, in fact, the items picked might be an item that user would like, but not yet interacted. Therefore, MixGCF proposed to use Positive Mixing alongside with Hop Mixing for generating sythetic negatives which is hard enough compare to random sample item and can thus improve the decision boundary learning of the model.

The framework of MixGCF can be formulated as follow:

1. To construct the fake negative e_{v^-}, it first select M negative items given u to form the candidate set $\mathcal{M}$. This M negative items can form a negative embedding set $\mathcal{E} = \{e^l_{v_m}\}$ where l is the l th layer of the embedding e, in other words, the l hop neighbors embedding aggregation.

2. Then, for each candidate embedding $e^l_{v_m} \in \mathcal{E}$, the positive mixing operation can be formulated as:
$$e^l_{v_m}{}' = \alpha^l e^l_{v^+}{}' + (1-\alpha)e^l_{v_m} \qquad (3)$$
where α is a float uniformly sampled form (0,1)

3. Then, the hop mixing is done by fusing all candidate embeddings by the pooling operation:
$$e_{v^-} = f_{pool}(e^l_{v_x}{}', ..., e^l_{v_y}{}') \qquad (4)$$

where ${e^l_{v_x}}'$ denotes the l-th layer embedding of v_x that is sampled at layer l and f_{pool} is the same pooling function that is used in LightGCN which is averaging.

4. Then, we have the triplet of e_u, e_{v^+} and e_{v^-} and the lightGCN parameters θ are update using BPR loss function.

$$loss = BPRloss(e_u, e_{v^+}, e_{v^-}|\theta) \tag{5}$$

As done in MixGCF, we will further extend this work by introducing new methods of generating pseudo-negative items through various methods like co-training and manifold mixup.

5 Methodology

In this section, we will formulate the detail of our frameworks and discuss the intuition behind the methods.

5.1 Co-Training Framework

Originally, co-training framework (Blum and Mitchell, 1998) is proposed to solve semi-supervised learning problem. Considering we have a dataset $D = S \cup U$ where S are labeled data and U are unlabeled data. The task is to build classifiers on the categories C in S using D. The assumption made by the paper is that each data x in D has two views, $v1$ and $v2$, and each view is sufficient for learning an effective model. Assuming that f_1 is trained on $v1$, f_2 is trained on $v2$ and f is trained on x. The assumption can be formulated as follow:

$$f(x) = f_1(v1) = f_2(v2) \tag{6}$$

Then, with this assumption, (Blum and Mitchell, 1998) proposed to train two classifiers for each view on S, and the trained classifiers should posses good understanding of the similar target from two different perspective. Then, to give U the pseudo label, the prediction result of the two classifiers on U have to reach consensus, for example, they should both agree that the current unlabeled data z belongs to the same label.

To train a deep co-training network, we follows the setting of (Qiao et al., 2018), which applies co-training on a deep learning semi-supervised image recognition task. Following the notation in the previous paragraph, Let $D = S \cup U$ denote all the provided data. Let $v1(x)$ and $v2(x)$ denote the two views of data x. In their settings, $v1(x)$ and $v2(x)$ are convolutional representations of x before the final fully-connected layer $fi(\Delta)$ that classifies $vi(x)$ to one of the categories in S. On the supervised dataset S, standard cross entropy loss is used:

$$L_{sup}(x,y)=H(y,f_1(v1(x)))+H(y,f_2(v2(x))) \tag{7}$$

where y is the label for $x \in S$ and $H(q,p)$ is the cross entropy function.

To model the previous Co-Training assumption, which assume that $f1(v1(x))$ and $f2(v2(x))$ should agree on their predictions. We want the classifier to have close prediction on unlabeled dataset U. Thus, the loss function to minimize the Jensen-Shannon divergence of two classifiers can be formulated as:

$$L_{cot}(z)=E(\tfrac{1}{2}(f_1(z)+f_2(z))-\tfrac{1}{2}(E(f_1(z))+E(f_2(z))) \tag{8}$$

where $z \in U$ and $E(p)$ is the entropy of p. There is no need to minimize L_{cot} on S as S are already trained with labels and should therefore produce similar result.

Inspired by Co-training and its application on classifying unlabeled data(Qiao et al., 2018), we propose to also incorporate the co-training framework into the synthetic negatives generating part of MixGCF. As the intuition of synthesizing negatives comes from data augmentation and metric learning which indicates that the harder the examples the better the model will learn, by adding co-training classifiers, those two classifiers can be used to identified whether the generated negatives are hard or boundary enough, in other words, can the generated negatives fool those two classifiers at the same time to be a positive example. However, we can not directly incorporate Co-training into the MixGCF framework, as they both requires some tweak to work seamlessly.

Conventionally, the input of recommendation systems includes a set of users $\mathcal{U} = \{u\}$, items $\mathcal{V} = \{v\}$, and users' implicit feedback $\mathcal{O}^+ = \{(u, v^+)|u \in \mathcal{U}, v^+ \in \mathcal{V}\}$, where each pair indicates an interaction between user u and item v^+.

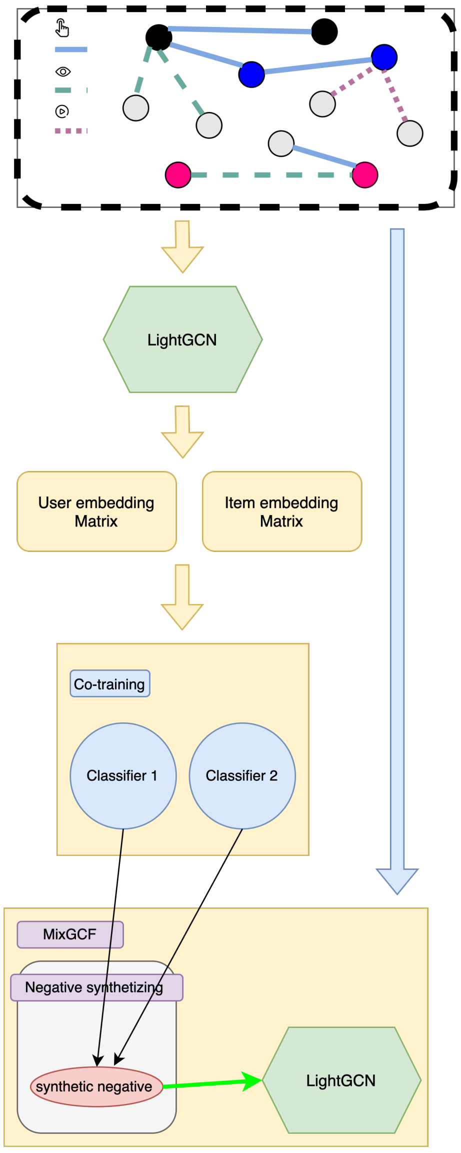

Figure 1: An overall framework of co-training MixGCX

To incoporate Co-training framework, we have to make the following changes:

1. First, we pre-train LightGCN on the dataset with ordinary negative sampling method, that is given a user randomly pick a not yet interacted item, and retrieved the user embedding matrix, M_u of size $n \times d$, and item embedding matrix, M_v of size $m \times d$, to construct the input for co-training classifiers where d is the dimension of each user and item vector and n is the total numbers of users and m is the total number of items. Denote e_{u_i} as the i^{th} row in the M_u which represent the embedding for user u_i where $u_i \in \mathcal{U}$ and e_{v_i} as the i^{th} row in the M_v which represent the embedding for item v_i where $v_i \in \mathcal{V}$

2. A single training input for co-training classifiers is uv which is the concatenation of e_{u_i} and e_{v_j} given a user item interactions. Note that v_j dose not have to be positive item to u_i, it could also be a negative example. And the label is simply 0 and 1 for which 0 indicate that uv is a negative interaction while 1 indicate uv that v is positive to user u.

3. Recall that for co-training to work, two classifier has to be trained on different view of the same data, in our setting, we modify such setting to different view of the same user, so that the classifiers can learn the preference characteristic of each user and can later on be used to test if the quality of synthetic negatives. Thus the above equation can be rewritten as follow:

$$L_{sup}(y_i,y_j,uv_i,uv_j)=H(y_i,f_1(uv_i))+H(y_j,f_2(uv_j)) \tag{9}$$

where y_i and y_j is the corresponding label for uv_i and uv_j. Note that, in order to provide more diverse view point for each user, we construct two supervised dataset $S1$ and $S2$ with negative sampling as it is likely to pick different negatives for each user when generating the inputs example. Thus, uv_i is retrieved from $S1$ and uv_j is from $S2$. Also, it is worth mentioning that since, lightGCN is a GNN, so the retrieved embedding for each user and item is the aggregation of all its k-hop neighbors' embedding, and this help the classifiers to incorporate both the idea of collaborative filtering and underlying relationships among users and items when doing the classification.

4. To model the Co-Training assumption, which assume that both classifiers should agree on their predictions. We need to construct the unsupervised dataset U which dose not have any label. In our scenario, unsupervised dataset can be constructed simply using negative sampling as the not-yet-interacted items are not necessarily a negative item to a user, it is possible that it is positive. Thus, the above equation 5

can be rewritten as:

$$L_{cot}(uv)=E(\tfrac{1}{2}(f_1(uv)+f_2(uv))-\tfrac{1}{2}(E(f_1(uv))+E(f_2(uv)))\quad(10)$$

where uv is from U.

5. After the training of two classifiers, they should be able to tell that whether a input uv is a positive or negative user-item interaction. Then, we move back to the mixgcf and apply those two classifiers on top of it. MixGCF synthesize a negative example $\hat{v}^-$ give a user u, we can concatenate the user embedding e_u and the synthetic negative $e_{\hat{v}^-}$ and use the two classifiers to test the difficulty of $\hat{v}^-$. If $\hat{v}^-$ is hard enough, both classifiers should predict $u\hat{v}^-$ as positive example, and if classifiers do not agree on the prediction, we simply abandon the generated synthetic negative. As a result, we only utilize the generated synthetic negatives that is hard enough and can help LiughtGCN better learns the boundary cases. Since BPR(Rendle et al., 2012) loss function is used to train LightGCN, we need a triplet of (u, v^+, v^-). Then, the whole LightGCN MixGCF training procedure with co-training can be formulated as in algorithm 1:

Algorithm 1 An algorithm with caption

```
Require: $TrainingSet\{(u, v^+)\}$
  for $(u, v^+) \in TrainingSet$ do
    $e_{\hat{v}^-} = MixGCF(u, v^+)$
    if $f_1(e_{\hat{v}^-}) == f_2(e_{\hat{v}^-}) == 1$ then
      $v^- = e_{\hat{v}^-}$
    else
      $v^- = RandomNegativeSample(u)$
    end if
    BPRloss($u$,$v^+$,$v^-$)
  end for
```

where $e_{\hat{v}^-}$ is the synthetic negative generated using MixGCF give a positive user-item interaction pair.

In conclusion, the overall co-training pipeline can be visualized as on fig 4.

5.2 Fair Regularization

As mentioned above, one important aspect of recommendation system is that negative signals are sampled from the sparse user-item interaction matrix. However there is a chance where these negative samples could possibly have a positive interaction but because it is unknown. Although we have proposed Co-training framework to selectively make use of quality negative samples, we propose to make use of fair regularization to remove any strong bias on the predictions.

Fair regularization is a method proposed by (Chuang and Mroueh, 2021) where the aim is to improve the generalizability of classifiers. The concept of "fairness" is introduced from this work where fairness implies less correlation between the particular input feature and the output, in our case the preference prediction. A clear example would be given that one of the input for the model is gender, which can be an sensitive input feature for some cases, we ideally do not want the model to form any strong bias on the gender feature. Given this setting, fair regularization aims to smooth the path between two distribution clusters(P_0, P_1) of input. This implies that the model has less bias on this feature.

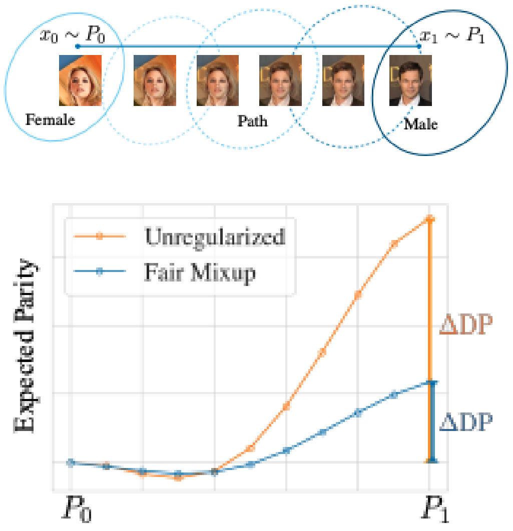

Figure 2: Fair regularization

$$\Delta DP(f) = \|E_{x\sim P_0} f(x) - E_{x\sim P_1} f(x)\|$$

$$\Delta DP(f\circ g) = \|\int_0^1 \frac{d}{dt}\int f(t(g(x_0)) + (1-t)g(x_1))dP_0(x_0)dP_1 x_1 dt\|$$

(11)

About the details of the regularization, demographic parity (DP) is a metric representing the fairness of the output distribution with regard to a particular input. And combining this with manifold mixup or any other mixup like input mixup, the equation for demographic parity can be written as equation (11). The first one will represent the parity between two distributions and second will represent the area under the curve along the path shown in Figure 1.

To apply this idea to our setting, that is given a set of item embeddings (V), we want the distribution of model prediction to be less extreme. The implication is to let the model draw less biased conclusions about the preference. This well suits the recommendation system setting because our "negative examples" are not really negative examples but user-item pairs that has not been interacted.

5.3 Modification of BPR loss through mix-up

We have also tried incorporating mix-up on the input of BPR loss function which is the output of MixGCF so that the model learns a clearer decision boundary. The intuition is quite clear as output of MixGCF represent the feature of the item and its neighborhood. Hence mix-up on the output of MixGCF will be representative of the neighborhood in the user-item embedding space. Performing mix-up on these features will generate features representing new neighborhoods. Depending on what label we give to these features it will be representative of decision boundaries between positive and negative or simply negative neighbors. In the experiment we have tested mixing up one of the options: two negative neighborhoods, two positive neighborhoods and one positive-one negative neighborhood. And for cases where there is a mix-up with positive neighborhood, we assign ground truth label as the mix-up coefficient. That is

$$Loss = label - (e_u, Mixup(e_{v^+}, e_{v^+} \text{ or } e_{v^-})) \tag{12}$$

In summary we further investigate the effect of mix-up on the following settings specified in Table 1.

Symbols imply such meaning: N2 (mix-up on two Negative Neighborhoods), P2 (mix-up on two Positive Neighborhoods), N1P1(One Positive, One Negative Neighborhoods). And ground truth is what value is regarded as ground truth when there is at least one positive neighbor involved in the mix-up. Lastly sample ratio is how many from the item neighborhoods pool is used to perform mix-up.

6 Experiment

This section first introduces the two datasets used. We then briefly describe the baseline models and their settings, following which we detail the experimental settings and evaluation metrics. Finally, we report the results of our two proposed method.

6.1 Dataset Baseline

In this paper, we evaluate our method on two datasets: Alibaba, Yelp2018 which are publicly available. Each dataset consists of the user-item interactions solely, as summarized in Table 2. We follow the same settings described in MixGCF to split the datasets into training, validation, and testing sets. One point to notice is that the density of Alibaba is much lower compared to Yelp2018, making it a more challenging task to solve.

6.1.1 Baseline

To verify the effectiveness of our proposed methods, we have two baseline approaches: 1. LightGCN with RNS and 2. LightGCN with MixGCF. To include as much experiments and tuned hyper-parameters, the reported results include performance after training for 100 epochs. Due to time and resource limit, we have chosen to train for only 100 epochs as training shows a tendency of starting to converge and the relative ordering of performance does not change drastically. For the same reason, we did not have enough time to include result of co-training framework on yelp2018. However since alibaba dataset, which has more user/item but less number of interactions, is a more challenging dataset than yelp hence we believe the improvement in result should be consistent.

- LightGCN: This is a state-of-the-art CF

Neighborhoods	Ground Truth	Alpha	Sample ratio	Weight on mix-up loss
N2, P2, N1P1	0, Alpha	Beta(1), uniform(0,1)	0.2, 0.6	0.1, 0.001

Table 1: Hyperparameter tuning setting for mix-up on input embedding

Dataset	Users	Items	Interactions	Density
Alibaba	106,042	53,591	907,407	0.00016
Yelp2018	31,668	38,048	1,561,406	0.00130

Table 2: Datasets information

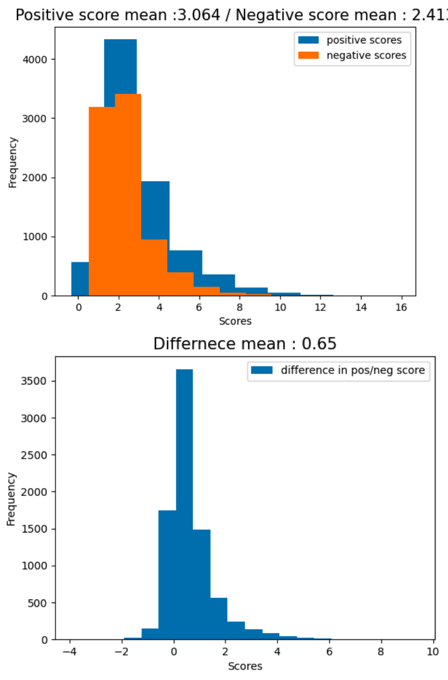

Figure 3: Positive/Negative score distribution of baseline

method. LightGCN claims that by removing the non-linear activation function within NGCF(Wang et al., 2019) and simply applies the sum-based pooling module, it can achieve better empirical performance.

- RNS: Random negative sampling (RNS) strategy applies uniform distribution to sample negative items.

- MixGCF: unlike RNS, MixGCF synthesize negative item with positive mixing and hop mixing, and thus help model performs better on boundary decision.

6.2 Parameter Settings

We have fine tune the LightGCN and find that the following parameters set yield the best result, and the same set of parameters were applied for all of the following experiments.

Dimension of embedding= 64, Number of layer = 2, Pooling = mean, Learning rate = 1e-4, L2 regularization= 1e-4, messDropoutRate = 0.1, messDropoutRate=0.1,

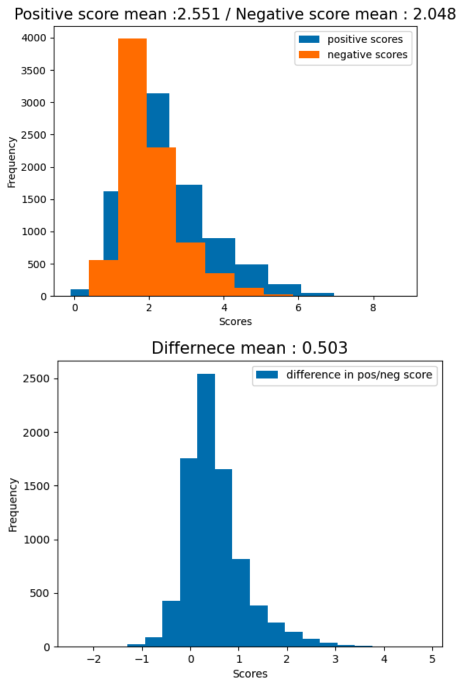

Figure 4: Positive/Negative score distribution after applying fair regularization

6.3 Running Time

The running time for LightGCN + RNS is approximately 3 hours, and 5 hours for MixGCF and Co-training + MixGCF. And for using MBPR, it takes approxmiately 6 hours for training.

6.4 Evaluation

The overall evalution results for both datasets can be found in Table 3 Table 4. CT,MBPR and FR each represent making use of co-training framework, modified BPR loss function and incorporating fair regularization during training.

Looking at the result, we can notice that there is an improvement in performance for

Model Type	**NDCG**	**Hit Ratio**	**Precision**	**Recall**
	@20 @40 @60	**@20 @40 @60**	**@20 @40 @60**	**@20 @40 @60**
LightGCN + RNS	0.0162 0.0211 0.0242	0.0453 0.0726 0.0922	0.0023 0.0018 0.0016	0.0373 0.0602 0.0767
MixGCF	0.0338 0.0405 0.0447	0.0834 0.1200 0.1462	0.0042 0.0031 0.0025	0.0717 0.1030 0.1257
MixGCF + FR	0.0292 0.0354 0.0394	0.074 0.108 0.1329	0.0038 0.00281 0.00234	0.0632 0.092 0.114
MixGCF + MBPR + FR	0.0296 0.0360 0.040	0.075 0.109 0.1341	0.0038 0.0284 0.00236	0.0637 0.0928 0.115
MixGCF + CT	**0.0351 0.0415 0.0453**	**0.0861 0.1205 0.1441**	**0.0044 0.0031 0.0025**	**0.0743 0.1039 0.1244**

Table 3: Overall performance on Alibaba dataset

Model Type	**NDCG**	**Hit Ratio**	**Precision**	**Recall**
	@20 @40 @60	**@20 @40 @60**	**@20 @40 @60**	**@20 @40 @60**
LightGCN + RNS	0.03463 0.04533 0.0531	0.2844 0.4088 0.4871	0.01970 0.01653 0.0146	0.04244 0.0708 0.09311
MixGCF	0.0490 0.0644 0.0760	0.3753 0.5242 0.6145	0.0273 0.0231 0.0206	0.0598 0.1006 0.1339
MixGCF + FR	0.0499 0.0654 0.0771	0.3796 0.5259 0.6167	0.0276 0.0233 0.0208	0.0609 0.1019 0.1353
MixGCF + MBPR + FR	**0.0503 0.0656 0.0774**	**0.3799 0.5251 0.6173**	**0.0277 0.0233 0.0208**	**0.0612 0.1018 0.1355**

Table 4: Overall performance on Yelp2018 dataset

using co-training framework(CT) in alibaba dataset while using modified BPR loss and fair regularization does not seem to help. However for yelp2018 dataset, modified BPR loss and fair regularization. This can be because of the large difference in the sparsity of the dataset. Since alibaba dataset is approximately ten times sparser than yelp2018 dataset, mix-up on pairs can lead to misleading pseudo neighborhoods while in yelp dataset it can help with constructing a more effective decision boundary.

Furthermore we have analyzed the change in distribution of positive and negative scores (Figure 3) and distribution of difference between positive and negative score (Figure 4) after applying fair regularization. We can notice the shift in distribution of both positive and negative scores to the left through its means while the difference between positive and negative scores became smaller. Along with this aspect, we can the model making less extreme predictions on the preference of the user from the narrower range of scores.

7 Conclusion

In this work, we have shown constructing negative signals and regularizing the user-item embedding space plays an important role in the performance of recommendation system. Even though regularization may not be effective when the training data is too sparse, one can first train based on negative signals generated by MixGCF and co-training framework and perhaps can further train by applying regularization after a certain point of training with this framework.

For future works, we would like to conduct more experiments by comparing our proposed methods to more baselines. Due to the computing resource limit, we couldn't fine-tune the co-training framework on Yelp2018 and would like to finish related experiments given more time. We would like to also integrate a more complicated Tri-training framework into our framework and we will also consider exploring the direction of adversarial data creation for the creation of realistic hard examples.

References

Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph convolutional matrix completion.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, page 92–100, New York, NY, USA. Association for Computing Machinery.

Ching-Yao Chuang and Youssef Mroueh. 2021. Fair mixup: Fairness via interpolation.

Tinglin Huang, Yuxiao Dong, Ming Ding, Zhen Yang, Wenzheng Feng, Xinyu Wang, and Jie Tang. 2021. Mixgcf: An improved training method for graph neural network-based recommender systems. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery amp; Data Mining*, KDD '21, page 665–674, New York, NY, USA. Association for Computing Machinery.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality.

Siyuan Qiao, Wei Shen, Zhishuai Zhang, Bo Wang, and Alan Yuille. 2018. Deep co-training for semi-supervised image recognition.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*.

Ben Schafer, Ben J, Dan Frankowski, Dan, Herlocker, Jon, Shilad, and Shilad Sen. 2007. Collaborative filtering recommender systems.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, Aaron Courville, David Lopez-Paz, and Yoshua Bengio. 2019. Manifold mixup: Better representations by interpolating hidden states.

Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan. *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, and Ming-Feng Tsai. 2018. Hop-rec: High-order proximity for implicit recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 140–144, New York, NY, USA. Association for Computing Machinery.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization.

Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, page 785–788, New York, NY, USA. Association for Computing Machinery.