One-class recommendation systems with the hinge pairwise distance loss and orthogonal representations

Ramin Raziperchikolaei
Rakuten Group, Inc.
San Mateo, CA, USA
ramin.raziperchikola@rakuten.com

Young-joo Chung
Rakuten Group, Inc.
San Mateo, CA, USA
youngjoo.chung@rakuten.com

Abstract

In one-class recommendation systems, the goal is to learn a model from a small set of interacted users and items and then identify the positively-related (i.e., similar) user-item pairs among a large number of pairs with unknown interactions. Most loss functions in the literature rely on dissimilar pairs of users and items, which are selected from the ones with unknown interactions, to obtain better prediction performance. The main issue with this strategy is that it needs a large number of dissimilar pairs, which increases the training time significantly. In this paper, our goal is to only use the similar set to train the models and discard the dissimilar set. We highlight three trivial solutions that the recommendation system models converge to when they are trained only on similar pairs: collapsed and dimensional collapsed solutions. We propose a hinge pairwise loss and an orthogonality term that can be added to the objective functions in the literature to avoid these trivial solutions. We conduct experiments on various tasks on public and real-world datasets, which show that our approach using only similar pairs can be trained several times faster than the state-of-the-art methods while achieving competitive results.

CCS Concepts

• **Information systems** → **Recommender systems**.

Keywords

Recommendation systems, Training efficiency

ACM Reference Format:
Ramin Raziperchikolaei and Young-joo Chung. 2024. One-class recommendation systems with the hinge pairwise distance loss and orthogonal representations. In *18th ACM Conference on Recommender Systems (RecSys '24), October 14–18, 2024, Bari, Italy*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3640457.3688189

1 Introduction

In implicit feedback prediction [10, 14, 20], we only know whether a user has interacted (e.g,. purchased or clicked) with an item, without knowing their satisfaction level. This problem setting is also called a one-class recommendation system (RS) problem since we only have access to the "positive interactions." The absence of an interaction between a user and an item in the training set does not indicate a negative interaction, but rather a lack of preference information, caused by the huge number of items in the system that users cannot fully see.

Different types of loss functions have been used in the literature to learn an RS model, such as mean squared error (MSE) loss [4, 10, 11, 14, 17, 18, 21, 26], binary cross-entropy (BCE) [5, 9, 24], Bayesian personalized rank (BPR) loss [7, 8, 19], and contrastive loss [13, 18, 23]. These loss functions need both similar and dissimilar pairs of users and items to learn a model. If we train these loss functions only using similar pairs, we get a collapsed solution: all representations will be mapped to the same point in the latent space, and the model predicts the same interaction score for all the pairs. The performance of the collapsed solution is as bad as assigning random representations to the users and items. Therefore, the dissimilar sets are essential in RS models to avoid the collapsed solution.

In one-class recommendation systems, we only have access to the implicit (known) interactions, and the rest of the interactions are unknown. To create a dissimilar set, the common approach is random negative sampling, where a random set of user and item pairs with unknown interactions are considered dissimilar [4, 5, 9, 18, 24, 26]. Another approach is non-sampling, where all pairs with unknown interactions are considered dissimilar [1, 10]. The final approach is hard-negative sampling, where the pairs with the unknown interactions that the model has difficulty classifying are considered dissimilar [2, 3, 27].

The main issue with the random negative sampling strategy is that to achieve reasonable results we need a large set of dissimilar pairs. This will increase the training time significantly. Also, as mentioned in [12], another disadvantage of this approach is that it increases the chance of converting a "similar pair with an unknown interaction" to a dissimilar pair, which hurts the performance.

The non-sampling approach makes a unrealistic assumption that all missing interactions are negative. This makes the labels in the training dataset noisy and hurts the performance.

The hard negative sampling approach has two main issues. The first one is that "similar pairs with unknown interactions" are by definition difficult to classify as dissimilar, and will be mistakenly taken as hard negatives, which hurts the performance. The second issue is that selecting hard negatives usually needs model evaluation on a large number of pairs, which increases the training time.

In this paper, we propose a new objective function that only needs a similar set of users and items to achieve comparable results to the state-of-the-art methods. To avoid the collapsed solution, we propose a hinge pairwise distance loss and to avoid the dimensional

RecSys '24, October 14–18, 2024, Bari, Italy

ACM ISBN 979-8-4007-0505-2/24/10
https://doi.org/10.1145/3640457.3688189

collapsed solution, we propose to minimize the correlation between the dimensions of representations by making them orthogonal.

We conduct extensive experiments on several real-world and public datasets in different settings. The experimental results in Section 4 show that our approach achieves competitive results while being trained faster than the state-of-the-art methods.

Notations. We denote the interaction matrix by $\mathbf{R} \in \mathbb{R}^{m \times n}$, where m and n are the number of users and items, respectively, $R_{jk} > 0$ is the interaction value of the user j on the item k, and $R_{jk} = 0$ means the interaction is unknown. The goal is to predict the unknown interactions in $\mathbf{R}$. The ith row of a matrix $\mathbf{H}$ is shown by $\mathbf{H}_{i,:}$ and the jth column is shown by $\mathbf{H}_{:,j}$. The d-dimensional representations of all users and all items are denoted by $\mathbf{Z}^u \in \mathbb{R}^{m \times d}$ and $\mathbf{Z}^i \in \mathbb{R}^{n \times d}$, respectively. The representation of the jth user and kth item are denoted by $\mathbf{z}_j^u = \mathbf{Z}_{j,:}^u$ and $\mathbf{z}_k^i = \mathbf{Z}_{k,:}^i$, respectively.

2 Related works

Since our main contribution is proposing a new objective function, we review the different loss functions used in the RS literature. To define the loss functions in RSs, previous works use sets of similar and dissimilar pairs of users and items. The similar set S^+ contains all the users and items that interacted with each other, i.e., $(j,k) \in S^+$ if $R_{jk} = 1$. The dissimilar set S^- contains a subset of the users and items with unknown interactions, i.e., $R_{jk} = 0$.

Two popular loss functions in the literature are mean squared error (MSE) [4, 10, 11, 14, 17, 18, 21, 26] and binary cross-entropy (BCE) [5, 9, 15, 24], which directly minimize the difference between the predicted and actual interactions:

$$l_{\text{BCE}}(\mathbf{z}_j^u, \mathbf{z}_k^i) = -(R_{jk} \log \hat{R}_{jk} + (1 - R_{jk}) \log(1 - \hat{R}_{jk})),$$
$$l_{\text{MSE}}(\mathbf{z}_j^u, \mathbf{z}_k^i) = (\hat{R}_{jk} - R_{jk})^2, \tag{1}$$

where $j,k \in S^+ \cup S^-$ and $\hat{R}_{jk}$ is the predicted interaction by the RS model. On the other hand, the BPR loss [7, 8, 19] is defined based on the difference between the predicted interactions of the similar and dissimilar pairs. The Contrastive loss [6] is also used in recommendation systems [13, 18, 23], where the idea is to map representations of similar pairs of users and items close to each other and the dissimilar ones far away.

Note that the above loss functions are optimized over the user and item representations, $\mathbf{z}_j^u$ and $\mathbf{z}_k^i$, which are used to generate the predicted interaction. The most common mappings from representations to predicted interactions are dot product [4, 7, 10, 11, 14, 18, 21, 26], cosine similarity [24, 25], and neural networks [5, 8, 9, 16, 17].

3 Proposed method

In this section, we first explain how the previous objective functions converge to a collapsed solution when they only use similar pairs. Then, we explain our proposed method and we show how we avoid the collapsed and dimensional collapsed solutions by proposing new terms.

3.1 Collapsed solution without dissimilar set

The loss functions introduced in Section 2 use both similar and dissimilar pairs to predict the actual representations. Let us explain what happens if we only use the similar pairs and discard the dissimilar ones using the contrastive loss defined in [6]:

$$E_{\text{cont}}(\mathbf{Z}^u, \mathbf{Z}^i) = \sum_{j,k \in S^+} ||\mathbf{z}_j^u - \mathbf{z}_k^i||^2. \tag{2}$$

Optimizing this objective function will lead to a collapsed solution, where all the user and item representations are mapped to any d-dimensional vector in the latent space. At the collapsed solution, which is the result of removing dissimilar pairs from the loss functions, the model always returns the same prediction, no matter what the input pairs are. This works as poorly as a random model.

As another example, consider the following MSE loss function with only similar pairs S^+ and no dissimilar pairs:

$$E_{\text{MSE}}(\mathbf{Z}^u, \mathbf{Z}^i) = \sum_{j,k \in S^+} ((\mathbf{z}_j^u)^T \mathbf{z}_k^i - 1)^2, \tag{3}$$

The optimal solution of the E_{MSE} is achieved by mapping all the user and item representations to any d-dimensional vector with a unit L2 norm. That's because the dot product of all pairs becomes 1, i.e., $\hat{R}_{jk} = (\mathbf{z}_j^u)^T \mathbf{z}_k^i = 1$, which makes the loss value 0 for all the terms.

Different combinations of mapping functions and loss functions can give us different objective functions. In all cases, without dissimilar pairs, the result is a collapsed solution.

3.2 Avoiding the collapsed solution: hinge pairwise distance loss

Let us assume the d-dimensional representations of the users and items are denoted by $\mathbf{Z}^u \in \mathbb{R}^{m \times d}$ and $\mathbf{Z}^i \in \mathbb{R}^{n \times d}$, respectively. The joint user-item representation is achieved by vertically concatenating the user and item representations, $\mathbf{Z} = [\mathbf{Z}^u, \mathbf{Z}^i] \in \mathbb{R}^{(m+n) \times d}$. The pairwise distance between all the representations in $\mathbf{Z}$ is computed as:

$$d_p = E_{\text{cont}}(\mathbf{Z}, \mathbf{Z}) = \frac{1}{(m+n)^2} \sum_{l=1}^{m+n} \sum_{s=1}^{m+n} ||\mathbf{z}_l - \mathbf{z}_s||^2. \tag{4}$$

d_p computes the distance between all the user-user, item-item, and user-item representations, which is different from $E_{\text{cont}}(\mathbf{Z}^u, \mathbf{Z}^i)$ that computes the distance between similar pairs of the users and items.

Mathematically, at the collapsed solution, we have $d_p = 0$. To avoid the collapsed solution, we propose a hinge pairwise distance loss that keeps the average pairwise distance d_p greater than a margin m_p. The new objective function can be written as:

$$E = E_{\text{cont}}(\mathbf{Z}^u, \mathbf{Z}^i) + E_{d_p}(\mathbf{Z}) = \sum_{j,k \in S^+} \left(||\mathbf{z}_j^u - \mathbf{z}_k^i||^2\right) + \max(0, m_p - d_p)^2. \tag{5}$$

Note that d_p involves computing the distances between all the pairs, which could be very time-consuming. Next, we show that d_p is equivalent to twice the summation of the variance of each dimension, which can be computed significantly faster. Let us denote the qth dimension of the lth representation as $z_{l,q}$, and the pairwise distance of the qth dimension as d_p^q. Then, d_p in Eq. (4) can be separated over the d dimensions as $d_p = \sum_{q=1}^{d} d_p^q$.

Below, we show that d_p^q is equivalent to twice the variance of the qth dimension:

$$d_p^q = \frac{1}{(m+n)^2}\sum_{l=1}^{m+n}\sum_{s=1}^{m+n}(z_{lq}-z_{sq})^2 = \frac{2}{(m+n)^2}\sum_{l=1,s=1}^{m+n} {z_{lq}}^2 - \frac{2}{(m+n)^2}\sum_{l=1}^{m+n} z_{lq}\sum_{s=1}^{m+n} z_{sq} = \frac{2}{(m+n)}\sum_{l=1}^{m+n} {z_{lq}}^2 - 2\bar{\mathbf{Z}}_{:,q}^2 = 2\text{var}(\mathbf{Z}_{:,q}). \quad (6)$$

This means that at the collapsed solution, the variance of each dimension is 0, and to avoid this solution, we need the summation of the variance of the dimensions to be greater than a margin.

3.3 Avoiding dimensional collapsed solution: orthogonality term

While the objective function of Eq. (5) avoids the collapsed solution, it gives us low-quality representations by converging to a dimensional collapsed solution. In this solution, each user or item has the same value across its dimensions, but it's distinct from other users and items. To understand this solution, consider a simpler version of our objective function, where we use $E_{d_p} = -d_p$:

$$E = E_{\text{cont}}(\mathbf{Z}^u, \mathbf{Z}^i) + E_{d_p}(\mathbf{Z}) = E_{\text{cont}}(\mathbf{Z}^u, \mathbf{Z}^i) - d_p = \sum_{q=1}^{d}\Big(\sum_{j,k\in S^+} ||\mathbf{z}_{jq}^u - \mathbf{z}_{kq}^i||^2 - 2\text{var}(\mathbf{Z}_{:,q})\Big) = \sum_{q=1}^{d} E_q. \quad (7)$$

Since the objective function E separates over each dimension, optimizing E is equivalent to optimizing the 1-dimensional objective function E_q separately for $q = 1, \ldots, d$. Since the 1-dimensional objective E_q is the same for any pair of dimensions, we get d identical solutions.

Although the mathematical proof does not extend to the case when the hinge pairwise loss is used as in Eq. (5), our empirical results still confirm that Eq. (5) converges to a dimensional collapsed solution as shown in the third plot of Fig. 2.

To avoid the dimensional collapsed solution, we add an orthogonality term to the objective function to minimize the correlation between the dimensions:

$$E_{ours} = \lambda_1 E_{\text{cont}}(\mathbf{Z}^u, \mathbf{Z}^i) + \lambda_2 E_{d_p}(\mathbf{Z}) + \lambda_3 E_{\text{orth}}(\mathbf{Z}), \quad (8)$$

where $E_{\text{cont}}(\mathbf{Z}^u, \mathbf{Z}^i)$ and $E_{d_p}(\mathbf{Z})$ are defined in Eq. (5), $E_{\text{orth}}(\mathbf{Z}) = \sum_{q=1}^{d}\sum_{s=q+1}^{d}\hat{\mathbf{Z}}_{:,q}^T\hat{\mathbf{Z}}_{:,s}$, $\hat{\mathbf{Z}}$ is achieved by subtracting the mean of each dimension from $\mathbf{Z}$, and $\hat{\mathbf{Z}}_{:,q}$ is the qth column of $\hat{\mathbf{Z}}$. The orthogonality term lets the objective avoid the dimensional collapsed solution by making the off-diagonal values of the covariance matrix small, which makes the correlation of the dimensions of the representations small.

3.4 Why do we need both terms?

As explained above, if we only optimize the hinge pairwise distance loss $E_{d_p}(\mathbf{Z})$ and $E_{\text{cont}}(\mathbf{Z}^u, \mathbf{Z}^i)$ together without using the orthogonality term $E_{\text{orth}}(\mathbf{Z})$, it will converge to a dimensional collapsed solution.

Table 1: Details of Datasets.

Dataset	User	Item	Interaction	Sparsity
AMusic	1 700	13 000	46 000	99.8%
Lastfm	1 741	2 665	69 149	98.5%
Ichiba10m	1.4M	844 000	10M	99.9991%

Let us now ignore the term $E_{d_p}(\mathbf{Z})$ and consider $E_{\text{orth}}(\mathbf{Z}) + E_{\text{cont}}(\mathbf{Z}^u, \mathbf{Z}^i)$ as the objective function. We show here that a constant matrix $\mathbf{Z}$, where all its elements are equal to a scalar $\alpha \in \mathbb{R}$, is an optimal solution for this objective function. The constant matrix $\mathbf{Z}$ makes $E_{\text{cont}}(\mathbf{Z}^u, \mathbf{Z}^i) = 0$ since the pairwise distance between any two points is 0. It also makes $E_{\text{orth}}(\mathbf{Z}) = 0$, since the covariance between any pair of dimensions is 0. As a result, any constant matrix is the optimal solution to this objective function, which gives us a collapsed solution. For this reason, it's important to keep all the terms to avoid the collapsed and dimensional collapsed solutions.

3.5 Computational complexity and batch-wise training

Here, we analyze the computational complexity of each term in our objective function in Eq. (8). The time complexity of E_{cont} and E_{orth} are $O(|S^+|d)$ and $O((m+n)d^2)$, respectively. For E_{d_p}, if we use Eq. (4), the complexity is $O((m+n)^2 d)$. If we use Eq. (6) to compute the variance, then the time complexity of computing E_{d_p} will decrease to $O((m+n)d)$, which is linear in the total number of users and items.

Since we use batches to train the model and update the representations, the three terms are computed on a batch and will have a much smaller time complexity, which depends on the number of users and items in the batch. We create batches based on the number of interactions: a batch of size B contains B interactions. We compute the representations of the users and items that exist in the batch and then compute the loss function.

4 Experiments

Our proposed objective function uses **Sim**ilar pairs, **P**airwise **D**istance loss, and **O**rthogonality loss, and is denoted by **SimPDO**.

4.1 Experimental setup

Datasets and evaluation metrics. We conducted experiments on two public benchmark datasets and one real-world dataset. Details of the datasets are in Table 1. For public benchmark datasets, we used Amazon Music (AMusic) and Lastfm[1] datasets. These datasets are available online[2]. The goal is to predict unknown interactions between existing users and items. We follow [5, 9] and report NDCG and Hit Ratio (HR) to evaluate the implicit feedback task. For the real-world dataset, we used Ichiba10m[3], where the goal is to retrieve potential buyers for the new items based on the items' side information. The train/test split is done based on the time period,

[1] http://ocelma.net/MusicRecommendationDataset
[2] https://github.com/familyld/DeepCF
[3] https://www.rakuten.co.jp

Table 2: Comparison with the state-of-the-art methods. Our method achieves competitive results using smaller number of training pairs. We report the mean of three runs here.

	AMusic				Lastfm				Negative
method	HR@5	HR@10	NDCG@5	NDCG@10	HR@5	HR@10	NDCG@5	NDCG@10	sampling
SimPDO	**0.326**	**0.434**	**0.219**	**0.256**	0.754	**0.896**	0.518	0.571	Positive only
SRNS	0.298	0.377	0.213	0.238	**0.776**	0.877	0.583	**0.615**	Hard negative
BUIR-ID	0.257	0.365	0.171	0.195	0.701	0.841	0.510	0.565	Positive only
DirectAU	0.281	0.373	0.194	0.236	0.741	0.863	**0.584**	0.550	Positive only
CFNet	0.297	0.384	0.210	0.242	0.771	0.886	0.556	0.583	Random
DMF	0.285	0.381	0.197	0.224	0.734	0.859	0.532	0.572	Random
WMF	0.278	0.343	0.199	0.220	0.748	0.881	0.528	0.566	None-sampling

Table 3: Performance of different RS models improves when Integrated with SimPDO.

	AMusic		Lastfm	
method	HR@5	HR@10	HR@5	HR@10
DMF	0.285	0.381	0.734	0.859
SimPDO + DMF	**0.326**	**0.434**	**0.754**	**0.896**
MF	0.284	0.357	0.742	0.881
SimPDO + MF	**0.314**	**0.393**	**0.748**	**0.890**

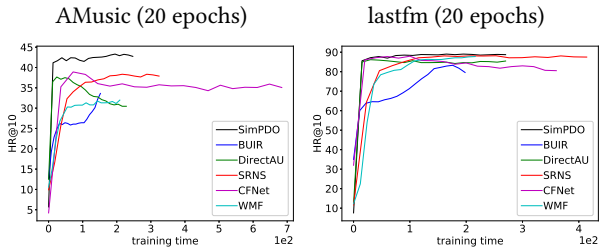

Figure 1: We report the performance of the methods as a function of training time. SimPDO achieves a higher performance with a faster convergence.

where the test items are the items that didn't appear during the training period. We follow [18] and report recall@50 to evaluate the performance of the methods, which shows what portion of the retrieved items/articles is relevant to the users.

We compare our method with a set of baseline and state-of-the-art methods. BUIR-ID [12] only uses positive pairs and uses the momentum mechanism to update the parameters and to avoid the collapse solution. DirectAU [22] only uses similar pairs of users and items and avoids the collapsed solution by using the uniformity term. SRNS [3] uses hard negative sampling, CFNet [5] and DMF [24] use random negative sampling, and WMF [10] uses all unknown samples as negative samples.

Implementation details. We implemented our method using Keras with TensorFlow 2 backend. We used one Nvidia Tesla V100-SXM2 32GB GPU in the internal cluster. We used grid search to set the hyperparameters using a subset of the training set and a small validation set. Here is the range of the hyperparameter we searched for: learning rate in $\{0.1, 0.01, 0.5\}$, batch size in $\{32, 64, 128\}$, λs in $\{0.01, 0.1, 1\}$, m_p in $\{0.01, 0.1\}$, embedding size in $\{100, 500, 1000\}$. We set the maximum number of epochs to 50. We set the batch size to 128 in all datasets. We set $\lambda_1 = 0.01$, $\lambda_2 = 1$, and $\lambda_3 = 1$ in all datasets. The margin m_p is 0.01 in all datasets. The dimension of the user and item embeddings is 1 000 in AMusic and Lastfm datasets and 100 in Ichiba10m dataset. For WMF, we found that the weight of negative samples plays an important role and we tuned it carefully. In Lastfm and AMusic, we set it to 0.1 and 0.5, respectively.

4.2 Experimental results on public datasets

SimPDO achieves competitive performance compared to the state-of-the-art methods. In Table 2, we compare the methods on AMusic and Lastfm datasets. We use the publicly available code for all baseline methods. We integrated SimPDO into DMF, referred to as SimPDO, and compared its performance to other baselines in Table 2. The results show that SimPDO outperforms most state-of-the-art methods and achieves competitive results. It even outperformed SRNS, which utilizes a complicated hard negative sampling strategy.

SimPDO improves the performance of different RS models. To verify whether SimPDO can improve different RS models that utilize random negative sampling, we combined SimPDO with Matrix Factorization (MF) and Deep Matrix Factorization (DMF); we replaced their MSE and BCE objective functions with SimPDO. As the results show in Table 3, SimPDO boosted the performance of DMF and MF. For example, combining SimPDO with DMF and MF yields a 14% and 10% improvement in the AMusic dataset for HR@5, respectively.

SimPDO achieves higher performance with a faster convergence. In Fig. 1 we train each of the methods for 20 epochs and show how their performance on the test set changes during the training. SimPDO achieves higher performance faster than the other methods.

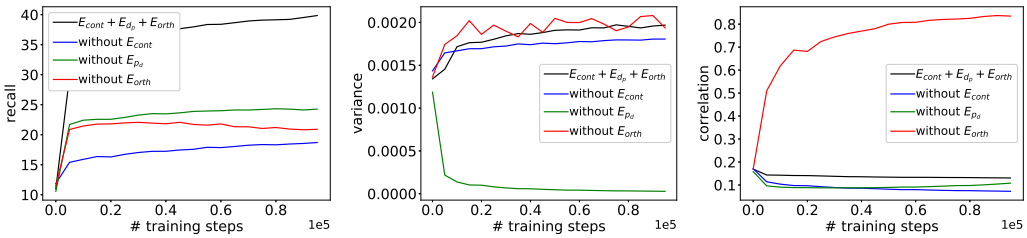

Figure 2: Impact of each term of our objective function in Ichiba10m datasets. Using all three terms leads to the best results.

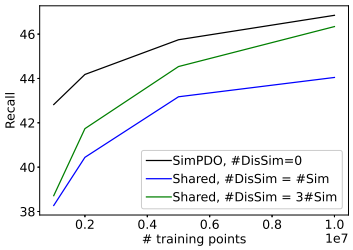

Figure 3: SimPDO vs Shared as we change the number of training pairs from 1M to 10M in Ichiba10m dataset. SimPDO achieves significantly better results when we use a smaller number of training pairs.

SimPDO, BUIR, and DirectAU only use positive pairs, while the rest of the methods use both positive and negative pairs. As a result, these three methods use a significantly smaller number of training pairs compared to the other methods, such as CFNet [5] and can be trained faster than most of the previous works.

4.3 Experimental results on the large-scale real-world dataset

The strength of SimPDO becomes particularly clear when applied to large-scale datasets. The Ichiba10m dataset is 200 times larger than the benchmark dataset discussed in the previous section. For this dataset, training one epoch takes one hour, and it takes more than 10 epochs to converge. For this dataset, we combined SimPDO with the Shared [18] model, as it is the state-of-the-art model for this dataset and task. Shared model utilizes random negative sampling to optimize its contrastive loss. In this section, we refer to SimPDO + Shared as SimPDO for simplicity, and compare its performance with the original Shared model.

SimPDO achieves better results with fewer training points. In the Ichiba10m dataset, we select 2 000 test items and report recall@50 as we change the number of training pairs from 1M to 10M. To train the Shared model, we divide the training pairs between similar and dissimilar pairs in different proportions. In Fig. 3, we show two scenarios, where the number of dissimilar pairs is 1) the same as and 2) three times greater than the number of similar pairs.

There are three remarkable points about the results of Fig. 3. First, SimPDO performs better than the original Shared model, no matter how many training pairs are used, which shows the advantage of training only on similar pairs. Second, the performance gap between SimPDO and Shared model becomes smaller as we increase the portion of the dissimilar pairs compared to the similar ones, which shows the importance of using a large number of dissimilar pairs. Third, SimPDO is significantly better than Shared using a smaller number of training pairs. This is a big advantage of SimPDO when the datasets have billions of pairs and it's time-consuming to train on all of them: SimPDO can be trained on a smaller training set and still achieve reasonable results.

Impact of each term of SimPDO. In Fig. 2, we investigate the impact of each term of our objective function. We report three metrics: 1) the recall as we train the models, 2) the average variance of the dimensions of all the representations, and 3) the average correlation between the dimensions of the representations.

We show the results on Ichiba10m In Fig. 2. We can see that the method with all three terms achieves the maximum recall. 1) without E_{cont}, as we can see in the first column, the performance drops significantly since the similarity between the similar pairs will not be preserved, 2) without E_{d_p}, as we can see in the second column, the variance drops significantly towards 0, which is a sign of the collapsed solution, and 3) without E_{orth}, as we can see in the third column, the average correlation increases, which is a sign of the dimensional collapsed solution.

5 Conclusion

In this paper, we proposed SimPDO, a new objective function that enables the training of one-class recommendation system models without dissimilar pairs. We showed that by only using similar pairs, the optimal solution of existing objective functions becomes a collapsed solution, where every representation is mapped to the same point in the latent space. We avoided the collapsed solution by providing a hinge loss for pairwise distances. We proved that this loss is equivalent to the summation of the variance of each dimension of representations. We also showed that we need an orthogonality term to avoid dimensional collapsed solutions, which can be computed much faster than the original hinge loss. Finally, we showed that both terms are necessary to learn meaningful representations. The results demonstrated that SimPDO outperformed the existing RS objective functions without using dissimilar pairs. Also, SimPDO can be trained more efficiently with a smaller number of training pairs. Our ablation study showed it is important to keep all terms in our objective function to achieve the best results.

References

[1] Chong Chen, Min Zhang, Yongfeng Zhang, Yiqun Liu, and Shaoping Ma. 2020. Efficient Neural Matrix Factorization without Sampling for Recommendation. *ACM Transactions on Information Systems* 38 (2020).

[2] Jingtao Ding, Yuhan Quan, Xiangnan He, Yong Li, and Depeng Jin. 2019. Reinforced negative sampling for recommendation with exposure data. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*.

[3] Jingtao Ding, Yuhan Quan, Quanming Yao, Yong Li, and Depeng Jin. 2020. Simplify and Robustify Negative Sampling for Implicit Collaborative Filtering. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS)*.

[4] Xin Dong, Lei Yu, Zhonghuo Wu, Yuxia Sun, Lingfeng Yuan, and Fangxi Zhang. 2017. A hybrid collaborative filtering model with deep structure for recommender systems. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*.

[5] Zhi-Hong Dong, Ling Huang, Chang-Dong Wang, Jian-Huang Lai, and Philip S Yu. 2019. DeepCF: A Unified Framework of Representation Learning and Matching Function Learning in Recommender System. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*.

[6] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality Reduction by Learning an Invariant Mapping. In *Proceedings of the 200614 IEEE Computer Society Conference Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2006.100

[7] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. 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*.

[8] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. 2018. Outer Product-Based Neural Collaborative Filtering. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*.

[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*.

[10] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *Proceedings of the 8th IEEE International Conference Data Mining (ICDM)*.

[11] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37. https://doi.org/10.1109/mc.2009.263

[12] Dongha Lee, SeongKu Kang, Hyunjun Ju, Chanyoung Park, and Hwanjo Yu. 2021. Bootstrapping User and Item Representations for One-Class Collaborative Filtering. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM. https://doi.org/10.1145/3404835.3462935

[13] Zihan Lin, Changxin Tian, Yupeng Hou2, and Wayne Xin Zhao. 2022. Improving Graph Collaborative Filtering withNeighborhood-enriched Contrastive Learning. In *Proceedings of the 31st International Conference on World Wide Web (WWW)*.

[14] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N. Liu, and Rajan Lukose. 2008. One-Class Collaborative Filtering. In *Proceedings of the 8th IEEE International Conference Data Mining (ICDM)*.

[15] Ramin Raziperchikolaei and Young-Joo Chung. 2022. A Recommendation System Framework to Generalize AutoRec and Neural Collaborative Filtering. In *IEEE International Conference on Data Mining Workshops (ICDMW)*.

[16] Ramin Raziperchikolaei and Young joo Chung. 2022. Simultaneous learning of the inputs and parameters in neural collaborative filtering. *arXiv preprint arXiv:2010.06070* (2022).

[17] Ramin Raziperchikolaei, Tianyu Li, and Young joo Chung. 2021. Neural Representations in Hybrid Recommender Systems: Prediction versus Regularization. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[18] Ramin Raziperchikolaei, Guannan Liang, and Young joo Chung. 2021. Shared Neural Item Representations for Completely Cold Start Problem. In *Proceedings of the 15th ACM Conference on Recommender Systems (RecSys)*.

[19] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*.

[20] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor (Eds.). 2011. *Recommender Systems Handbook*. Springer US. https://doi.org/10.1007/978-0-387-85820-3

[21] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. 2008. Matrix factorization and neighbor based algorithms for the netflix prize problem. In *Proceedings of the 2nd ACM Conference on Recommender Systems (RecSys)*. https://doi.org/10.1145/1454008.1454049

[22] Chenyang Wang, Yuanqing Yu, Weizhi M, Min Zhang, Chong Chen, Yiqun Liu, and Shaoping Ma. 2022. Towards Representation Alignment and Uniformity in Collaborative Filtering. In *Proceedings of the 28th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

[23] Ga Wu, Maksims Volkovs, and Chee Loong Soon. 2019. Noise Contrastive Estimation forOne-Class Collaborative Filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[24] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*. https://doi.org/10.24963/ijcai.2017/447

[25] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. https://doi.org/10.1145/3219819.3219890

[26] Jiani Zhang, Xingjian Shi, and Shenglin Zhaoand Irwin King. 2019. STAR-GCN: Stacked and Reconstructed Graph Convolutional Networks for Recommender Systems. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*.

[27] 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*.