Bootstrapping User and Item Representations for One-Class Collaborative Filtering

Dongha Lee[1], SeongKu Kang[1], Hyunjun Ju[1], Chanyoung Park[2], Hwanjo Yu[1*]

[1]Pohang University of Science and Technology (POSTECH), South Korea
[2]Korea Advanced Institute of Science and Technology (KAIST), South Korea
{dongha.lee, seongku, hyunjunju, hwanjoyu}@postech.ac.kr, cy.park@kaist.ac.kr

ABSTRACT

The goal of one-class collaborative filtering (OCCF) is to identify the user-item pairs that are positively-related but have not been interacted yet, where only a small portion of positive user-item interactions (e.g., users' implicit feedback) are observed. For discriminative modeling between positive and negative interactions, most previous work relied on negative sampling to some extent, which refers to considering unobserved user-item pairs as negative, as actual negative ones are unknown. However, the negative sampling scheme has critical limitations because it may choose "positive but unobserved" pairs as negative. This paper proposes a novel OCCF framework, named as BUIR, which does not require negative sampling. To make the representations of positively-related users and items similar to each other while avoiding a collapsed solution, BUIR adopts two distinct encoder networks that learn from each other; the first encoder is trained to predict the output of the second encoder as its target, while the second encoder provides the consistent targets by slowly approximating the first encoder. In addition, BUIR effectively alleviates the data sparsity issue of OCCF, by applying stochastic data augmentation to encoder inputs. Based on the neighborhood information of users and items, BUIR randomly generates the augmented views of each positive interaction each time it encodes, then further trains the model by this self-supervision. Our extensive experiments demonstrate that BUIR consistently and significantly outperforms all baseline methods by a large margin especially for much sparse datasets in which any assumptions about negative interactions are less valid.

CCS CONCEPTS

• **Information systems** → **Collaborative filtering**; • **Computing methodologies** → *Learning from implicit feedback*; *Unsupervised learning*.

KEYWORDS

One-class collaborative filtering, Bootstrapping-based representation learning, Self-supervised learning, Recommender systems

*corresponding author

1 INTRODUCTION

Over the past decade, one-class collaborative filtering (OCCF) problems [13, 24] have been extensively researched to accurately infer a user's preferred items, particularly for the recommender systems where only the users' implicit feedback on items are observed (e.g., click, purchase, or browsing history). This problem has remained challenging due to an extreme sparseness of such implicit feedback (i.e., most users have interacted with only a few items among numerous items), and also the non-existence of the negative labels for user-item interactions (i.e., observed feedback is expressions of positive interactions). Precisely, the goal of OCCF is to identify the most likely positive user-item interactions among a huge amount of unobserved interactions, by using only a small number of observed (positively-labeled) interactions.

The most dominant approach to the OCCF problem is discriminative modeling [11, 12, 15, 17, 27, 32], which explicitly aims to distinguish positive user-item interactions from the negative counterparts. They define the *interaction score* indicating how likely each user interacts with each item, based on the similarity (e.g., inner product) between the representation of a user and an item. From matrix factorization [13, 27] to deep neural networks [11, 32], a variety of techniques have been studied to effectively model this score. Then, they optimize the scores by using the pointwise prediction loss [11, 13] or the pairwise ranking loss [12, 27] to discriminate between positive and negative interactions.

However, since the negative interactions are not available in the OCCF problem, previous discriminative methods assume that all unobserved interactions are negative. In other words, for each user, the items that have not been interacted yet are regarded to be less preferred to positive items. In this sense, they either use all unobserved user-item interactions as negative or adopt a *negative sampling*, which randomly samples unobserved user-item interactions in a stochastic manner to alleviate the computational burden. For better recommendation performance and faster convergence, advanced negative sampling strategies [5, 26] are also proposed to sample from non-uniform distributions.

Nevertheless, the negative sampling approach has critical limitations in the following aspects. First, the underlying assumption

about negative interactions becomes less valid as user-item interactions get sparser. This is because as fewer positive interactions are observed, the number of "positive but unobserved" interactions increases, which consequently makes it even harder to sample correct negative ones. Such uncertainty of supervision eventually degrades the performance for top-K recommendation. Second, the convergence speed and the final performance depend on the specific choice of distributions for negative sampling. For example, sampling negative pairs from a non-uniform distribution [5, 26] (e.g., the multinomial distribution which models the probability of each interaction being actually negative) can improve the final performance, but inevitably incurs high computational costs, especially when a lot of users and items should be considered.

As a solution to the aforementioned limitations, this paper proposes a novel OCCF framework, named as BUIR, which does not require the negative sampling at all for training the model. The main idea is, given a positive user-item interaction (u, v), to make representations for u and v similar to each other, in order to encode the preference information into the representations. However, a naive end-to-end learning framework that guides positive user-item pairs to be similar to each other without any negative supervision can easily converge to a *collapsed solution* – the encoder network outputs the same representations for all the users and items.

We argue that the above collapsed solution is incurred by the simultaneous optimization of u and v within the end-to-end learning framework of a single encoder. Hence, we instead adopt the student-teacher-like network [6, 29] in which only the student's output u (and v) is optimized to predict the target v (and u) presented by the teacher. Specifically, BUIR directly bootstraps[1] the representations of users and items by employing two distinct encoder networks, referred to as *online* encoder and *target* encoder. The high-level idea is training only the online encoder for the prediction task between u and v, where the target for its prediction is provided by the target encoder. That is, the online encoder is optimized so that its user (and item) vectors get closer to the item (and user) vectors computed by the target encoder. At the same time, the target encoder is updated based on momentum-based moving average [6, 8, 29] to slowly approximate the online encoder, which encourages to provide enhanced representations as the target for the online encoder. By doing so, the online encoder can capture the positive relationship between u and v into the representations, while preventing the model from collapsing to the trivial solution without explicitly using any negative interactions for the optimization.

Furthermore, we introduce a stochastic data augmentation technique to relieve the data sparsity problem in our framework. Motivated by the recent success of self-supervised learning in various domains [2, 4], we exploit *augmented views* of an input interaction, which are generated based on the neighborhood information of each user and item (i.e., the set of the items interacted with a user, and the users interacted with an item). The stochastic augmentation is applied to positive user-item pairs when they are passed to the encoder, so as to produce the different views of the pairs. To be

precise, by making our encoder use a random subset of a user's (and item's) neighbors for the input features, it produces a similar effect to increasing the number of positive pairs from the data itself without any human intervention. In the end, BUIR is allowed to learn various views of each positive user-item pair.

Our extensive evaluation on real-world implicit feedback datasets shows that BUIR consistently performs the best for top-K recommendation among a wide range of OCCF methods. In particular, the performance improvement becomes more significant in sparser datasets, with the help of utilizing augmented views of positive interactions as well as eliminating the effect of uncertain negative interactions. In addition, comparison results on a downstream task, which classifies the items into their category, support that BUIR learns more effective representations than other OCCF baselines.

2 RELATED WORK

2.1 One-Class Collaborative Filtering

One-class collaborative filtering (OCCF) was firstly introduced to handle the real-world recommendation scenario where only positive user-item interaction can be labeled [13, 24] as a form of users' implicit feedback on items. That is, only the set of positive user-item pairs, denoted by $\mathcal{R}$, is given for training the model. The main challenge of OCCF is to find out the most likely positive interactions among a large number of unobserved user-item pairs in which both positive and negative interactions are mixed together. To handle the absence of negatively-labeled interactions, most existing methods have either treated all unobserved user-item pairs as negative, or sampled some of them [11], assuming that the items that have not been interacted yet are less preferred to positive items.

To be specific, discriminative methods [11, 12, 15, 17, 27, 32, 33] train their model so that it can differentiate the scores between positive and negative interactions. Pairwise learning, which is the most popular approach to personalized ranking, explicitly utilizes the pairs of positive and negative interactions for training. Formally, the pairwise ranking loss optimizes the similarity for a positive interaction to become larger than that for a negative one as follows.

$$\mathcal{L} = - \sum_{(u,v^p,v^n) \in O} \phi(sim(u, v^p) > sim(u, v^n)), \qquad (1)$$

where $O = \{(u, v^p, v^n) | (u, v^p) \in \mathcal{R}, (u, v^n) \notin \mathcal{R}\}$, and ϕ is a scoring function to facilitate the optimization. For example, Bayesian personalized ranking [9, 15, 32] defines the similarity of a user and an item by the inner product of their representations, and collaborative metric learning [12, 17, 25] directly learns the latent space by modeling their similarity as the Euclidean distance. However, all these methods obtain the negative interactions from unobserved user-item pairs, thus the convergence speed and final performance largely depend on the negative sampling distribution [26].

On the other hand, generative methods [1, 18, 19, 31] aim to learn the underlying latent distribution of users, usually represented by binary vectors indicating their interacted items. They employ the architecture of variational autoencoder (VAE) [18] or generative adversarial networks (GAN) [1, 19, 31], in order to infer the users' preference on each item based on the reconstructed (or generated) user vectors. Rather than exploiting the negative sampling, most of

---

[1]In this paper, the term "bootstrapping" is not used in the statistical meaning, but in the idiomatic meaning [6]. Strictly speaking, it refers to using estimated values (i.e., the output of networks) for estimating its target values, which serve as supervision for the update. For instance, semi-supervised learning based on predicted pseudo-labels [29] also can be thought as a bootstrapping method.

the generative methods implicitly assume that all unobserved user-item pairs are negative in that they learn the partially-observed binary vectors as their inputs. We remark that this assumption is not strictly valid, which eventually leads to limited performance.

2.2 Self-supervised Contrastive Learning

Recently, a self-supervised learning approach has achieved a great success in computer vision and natural language understanding [2, 4, 8]. Most of them basically adopt contrastive learning, which optimizes the representations of positively-related (similar) instances to be close, while those of negatively-related (dissimilar) ones far from each other. Given an unlabeled dataset $\mathcal{D} = \{x_1, \ldots, x_N\}$, positive pairs for each instance (x, x^p) is usually obtained from the data itself (i.e., data augmentation), such as geometric transformations on a target image. Note that it does not require any human annotations or additional labels, thus this approach falls into the category of self-supervised learning. The noise contrastive estimator (NCE) loss [7, 23] mainly used for contrastive learning is defined by using all the other instances except for x as negative:

$$\mathcal{L} = - \sum_{x \in \mathcal{D}} \log \frac{\exp(sim(x, x^p))}{\exp(sim(x, x^p)) + \sum_{x^n \in \mathcal{D} \setminus \{x\}} \exp(sim(x, x^n))}. \tag{2}$$

In case of large-scale datasets, the predefined number of negative instances can be selected (i.e., negative sampling). For contrastive learning, negative pairs must be considered for its optimization so as to prevent the representations of all instances from being similar, which is known as the problem of collapsed solutions.

Pointing out that the contrastive methods need to carefully treat the negative instances during the training for effectiveness and efficiency, the most recent work proposed a bootstrapping-based self-supervised learning framework [3, 6], which is capable of avoiding the collapsed solution without the help of negative instances. Inspired by bootstrapping methods in deep reinforcement learning [21, 22], it directly bootstraps the representation of images by using two neural networks that iteratively learn from each other. This approach achieves the state-of-the-art performance for various downstream tasks in computer vision, and also shows better robustness to the choice of data augmentations used for self-supervision.

3 BUIR: PROPOSED FRAMEWORK

In this section, we present our OCCF framework, named as BUIR, which learns the representations of users and items without any assumptions about negative interactions. We first describe the overall learning process with a simple encoder that takes the user-id and item-id as its input (Section 3.2) and how to infer the interaction score using the representations (Section 3.3). We also introduce a stochastic data augmentation technique with an extended encoder to further exploit the neighborhood information (Section 3.4).

3.1 Problem Formulation

Let $\mathcal{U} = \{u_1, \ldots, u_M\}$ and $\mathcal{V} = \{v_1, \ldots, v_N\}$ be the set of M users and N items, respectively. Given a set of observed user-item interactions $\mathcal{R} = \{(u, v) | \text{user } u \text{ is interacted with item } v\}$, the goal of OCCF is to obtain the interaction (or preference) score $s(u, v) \in \mathbb{R}$ indicating how likely the user u interacts with (or prefers to) the
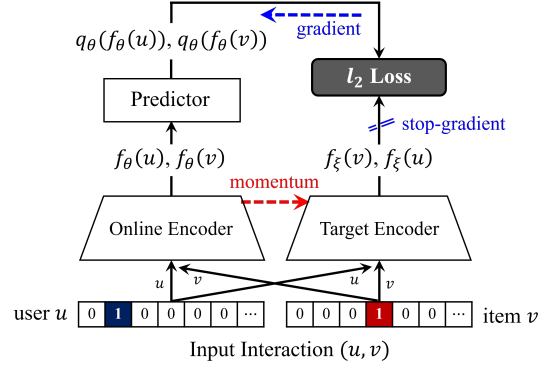

Figure 1: The overall BUIR framework.

item v. Based on the interaction scores, we can recommend K items with the highest scores for each user, called as top-K recommendation. To define the interaction score by using the representations of users and items, we focus on training the encoder network that maps each user and item into a low-dimensional latent space where the users' preferences on the items are effectively captured.

3.2 Bootstrapping the Representations

Let f be the encoder network to produce the representations of users and items. The simplest architecture of the encoder is a single embedding layer (i.e., embedding matrix); this maps each user-id (or item-id) into a D-dimensional embedding vector that represents the latent factors of the user (or item). Specifically, each encoder consists of a user encoder and an item encoder, and they take a one-hot vector indicating the user-id and item-id as their input.

BUIR makes use of two distinct encoder networks that have the same structure: *online encoder* f_θ and *target encoder* f_ξ. They are parameterized by θ and ξ, respectively. The key idea of BUIR is to train the online encoder by using outputs of the target encoder as its target, while gradually improving the target encoder as well. The main difference of BUIR from existing end-to-end learning frameworks is that f_θ and f_ξ are updated in different ways. The online encoder is trained to minimize the error between its output and the target, whereas the target network is slowly updated based on the momentum update [8] so as to keep its output consistent.

To be precise, for each observed interaction $(u, v) \in \mathcal{R}$, the BUIR loss is defined based on the mean squared error of the prediction against each other (i.e., representations of u and v) using the predictor $q_\theta : \mathbb{R}^D \rightarrow \mathbb{R}^D$ on top of the online encoder. It includes two error terms: one is for updating the *online* user vector $f_\theta(u)$ to accurately predict the *target* item vector $f_\xi(v)$, and the other is for updating the *online* item vector $f_\theta(v)$ to make its prediction as the *target* user vector $f_\xi(u)$. Finally, the loss is described as follows:

$$\mathcal{L}_{\theta, \xi}(u, v) = l_2 \left[q_\theta \left(f_\theta(u) \right), f_\xi(v) \right] + l_2 \left[q_\theta \left(f_\theta(v) \right), f_\xi(u) \right]$$
$$\approx - \frac{q_\theta \left(f_\theta(u) \right)^\top f_\xi(v)}{\| q_\theta(f_\theta(u)) \|_2 \| f_\xi(v) \|_2} - \frac{q_\theta \left(f_\theta(v) \right)^\top f_\xi(u)}{\| q_\theta(f_\theta(v)) \|_2 \| f_\xi(u) \|_2}, \tag{3}$$

where $l_2[\mathbf{x}, \mathbf{y}]$ is the l_2 distance between two normalized vectors $\overline{\mathbf{x}}$ and $\overline{\mathbf{y}}$; i.e., $\overline{\mathbf{x}} = \mathbf{x}/\|\mathbf{x}\|_2$ and $\overline{\mathbf{y}} = \mathbf{y}/\|\mathbf{y}\|_2$. Since the mean squared errors between two normalized vectors are equivalent to the negative value of their inner product (Equation (3)), we simply use the inner product for the optimization. Note that BUIR updates $f_\theta(u)$ to be similar with $f_\xi(v)$ instead of $f_\theta(v)$ through the predictor, and vice versa. This is because directly reducing the error between $f_\theta(u)$ and $f_\theta(v)$ leads to the collapsed representations when negative interactions are not considered at all for training the encoder.

To sum up, the parameters of the online encoder and target encoder are optimized by

$$\theta \leftarrow \theta - \eta \cdot \nabla_\theta \mathcal{L}_{\theta, \xi}$$
$$\xi \leftarrow \tau \cdot \xi + (1 - \tau) \cdot \theta. \tag{4}$$

η is the learning rate for stochastic optimization, and $\tau \in [0, 1]$ is a momentum coefficient (also called as target decay) for momentum-based moving average. The online encoder f_θ (and the predictor q_θ) is effectively optimized by the gradients back-propagated from the loss (Equation (3)), while the target encoder f_ξ is updated as the moving average of the online encoder. By taking a large value of τ, the target encoder slowly approximates the online encoder. This momentum-based update makes ξ evolve more slowly than θ, which enables to *bootstrap* the representations by providing enhanced but consistent targets to the online encoders [6, 8]. Figure 1 illustrates the overall framework of BUIR with the simple one-hot encoders.

Bypassing the collapsed solution. It is obvious that the loss in Equation (3) admits the collapsed solution with respect to θ and ξ, which means both the encoders generate the same representations for all users and items. For this reason, the conventional end-to-end learning strategy, which optimizes both f_θ and f_ξ to minimize the loss (i.e., cross-prediction error), may easily lead to such collapsed solution. In contrast, our proposed framework updates each of the encoders in different ways. From Equation (4), the online encoder is optimized to minimize the loss, while the target encoder is updated to slowly approximate the online encoder. That is, the direction of updating the target encoder ($\theta - \xi$) totally differs from that of updating the online encoder ($-\nabla_\theta \mathcal{L}_{\theta, \xi}$), and this effectively keeps both the encoders from converging to the collapsed solution. Several recent work on bootstrapping-based representation learning [3, 6] empirically demonstrated that this kind of dynamics (i.e., updating two networks differently) allows to avoid the collapsed solution without any explicit term to prevent it.

3.3 Top-K Preferred Item Prediction

To retrieve K most preferred items for each user (i.e., user-item interactions that are most likely to happen), we define the interaction score $s(u, v)$ by using the representations of users and items. As we minimize the prediction error between u and v for each positive interaction (u, v), their positive relationship is encoded into the l_2 distance between their representations (Equation (3)). In other words, a smaller value of $\mathcal{L}_{\theta, \xi}(u, v)$ indicates that the user-item pair (u, v) is more likely to be interacted, which means the loss becomes inversely proportional to the interaction score. To consider the symmetric relationship between u and v, the interaction score is defined based on the cross-prediction task; the prediction of v by
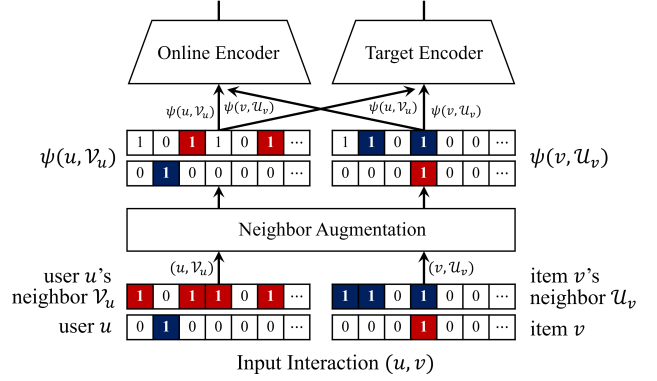

Figure 2: The stochastic data augmentation technique of BUIR based on the neighborhood information.

u, and the prediction of u by v.[2]

$$s(u, v) = q_\theta \left(f_\theta(u) \right)^\top f_\theta(v) + f_\theta(u)^\top q_\theta \left(f_\theta(v) \right). \tag{5}$$

For the computation of the interaction scores, we use only the representations obtained from the online encoder, with the target encoder discarded. Since the online encoder and the target encoder finally converge to equilibrium by the slow-moving average, it is possible to effectively infer the interaction score only with the online encoder. Considering the purpose of the target network, which generates the target for training the online network, it does make sense to leave the online encoder in the end.

Existing discriminative OCCF methods [12, 27] have tried to optimize the latent space where the user-item interactions are directly encoded into their inner product (or Euclidean distance). On the contrary, BUIR additionally uses the predictor to model their interaction, which results in the capability of encoding the high-level relationship between users and items into the representations. In conclusion, with the help of the predictor, BUIR accurately computes the user-item interaction scores as well as optimizes the representation without explicitly using negative samples.

3.4 Neighbor-based Data Augmentation

The another available source for OCCF is the neighborhood information of users and items. The neighbors of user u and item v, denoted by $\mathcal{V}_u$ and $\mathcal{U}_v$, refer to the set of the items interacted with u, and the users interacted with v, respectively. From the perspective that user-item interactions can be considered as a bipartite graph between user nodes and item nodes, each node's neighbors (or its local graph structure) can be a good feature to encode the similarity among the nodes. To take advantage of these neighbors as input features of users and items, we use a neighbor-based encoder [10, 15, 32] which additionally takes a given set of users (or items) as its input. Namely, this encoder is able to learn such set-featured inputs, represented as multi-hot vectors, capturing both the co-occurrence of users (or items) and their relationship. Adding the multi-hot inputs $\mathcal{V}_u$ and $\mathcal{U}_v$ to the one-hot inputs u and v within our framework, the neighbor-based user/item representations, denoted by $f_\theta(u, \mathcal{V}_u)$ and $f_\theta(v, \mathcal{U}_v)$, can be effectively

---

[2]We empirically found that the normalized representations cannot take into account the popularity of users and items, thus simply use the output of the online encoder.

optimized and utilized, instead of $f_\theta(u)$ and $f_\theta(v)$. In this case, the online encoder parameters related to user u (or item v) are shared for computing $f_\theta(u, \mathcal{V}_u)$ and $f_\theta(v, \mathcal{U}_v)$, thus they are updated by two types of supervision (i.e., optimized not only as a target but also as one of the neighbors), which brings an effect of regularization.

For acquisition and exploitation of richer supervision, we extend our framework to consider much more user-item interactions that are augmented based on their neighborhood information in a self-supervised manner. To this end, we introduce a new augmentation technique specifically designed for positive user-item interactions; it does not statically increase the number of interactions as a pre-processing step, rather be *stochastically* applied to each input interaction during the training. This stochastic data augmentation allows the encoder to learn slightly perturbed interactions, referred to as *augmented views* of an interaction. By doing so, BUIR can effectively learn the representations even in the case that only a few positive user-item interactions are available for training (i.e., highly sparse dataset). To this end, we first represent each user and item as the pair of its identity and neighbors: $(u, \mathcal{V}_u)$ and $(v, \mathcal{U}_v)$. Then, we apply the following augmentation function ψ to the user and item before passing them to the neighbor encoder.

$$\psi(u, \mathcal{V}_u) = (u, \mathcal{V}_u'), \text{ where } \mathcal{V}_u' \sim \{\mathcal{S} | \mathcal{S} \subseteq \mathcal{V}_u\},$$
$$\psi(v, \mathcal{U}_v) = (v, \mathcal{U}_v'), \text{ where } \mathcal{U}_v' \sim \{\mathcal{S} | \mathcal{S} \subseteq \mathcal{U}_v\}. \quad (6)$$

This augmentation function chooses one of the subsets of the user's neighbors (i.e., $\mathcal{V}_u'$) for an input user, and works in a similar way for an input item. For each input interaction (u, v), we can make a variety of interactions containing small perturbations $(\psi(u, \mathcal{V}_u), \psi(v, \mathcal{U}_v))$, and they produce a similar effect to increasing the number of positive pairs from the data itself.

Similarly to Section 3.2, the online encoder is trained by minimizing $\mathcal{L}_{\theta,\xi}(\psi(u, \mathcal{V}_u), \psi(v, \mathcal{U}_v))$, and the target encoder is slowly updated by the momentum mechanism. After the optimization is finished, the interaction score is inferred by $f_\theta(u, \mathcal{V}_u)$ and $f_\theta(v, \mathcal{U}_v)$ (Equation (5)). Figure 2 shows an example of our data augmentation which injects a certain level of perturbations to the neighbors.

4 EXPERIMENTS

In this section, we describe the experimental results that support the superiority of our proposed framework. We first present comparison results with other OCCF methods for top-K recommendation (Section 4.2), then validate the effectiveness of each component through an ablation study (Section 4.3 and 4.4). We also evaluate the quality of obtained representations for a downstream task (Section 4.5) and finally provide the hyperparameter analysis (Section 4.6).

4.1 Experimental Settings

Datasets. In our experiments, we use three real-world datasets: CiteULike [30], Ciao [28], and FourSquare [20]. For preprocessing the datasets, we follow previous work [11, 14, 27, 32] which provide the minimum count of user-item interactions for filtering long-tail users/items, considering the property of each dataset (e.g., the statistics or the domain where the implicit feedback is collected).[3] Table 1 summarizes the statistics of the datasets.

---

[3]We remove users having fewer than 5 (CiteULike, Ciao) & 20 interactions (FourSquare), and remove items having fewer than 5 (Ciao) & 10 interactions (FourSquare).

Table 1: The statistics of the datasets.

Dataset	CiteULike	Ciao	FourSquare
#Users	5,219	7,265	19,465
#Items	25,181	11,211	28,593
#Interactions	125,580	149,141	1,115,108
Density	0.096%	0.183%	0.200%

Baselines. We compare the performance of BUIR with that of baseline OCCF methods, including both discriminative and generative methods. They are re-categorized as either 1) the methods using only the user-id/item-id or 2) the ones additionally using the neighborhood information. Most of the methods in the first category directly optimize the embedding vectors of users and items.

- **BPR** [27]: The Bayesian personalized ranking method for OCCF. It optimizes matrix factorization (MF) based on the pairwise ranking loss.
- **NeuMF** [11]: The neural network-based method that uses the pointwise prediction loss. It combines MF and multi-layer perceptron (MLP) to model the user-item interaction.
- **CML** [12]: A metric learning approach to the OCCF problem. It optimizes the Euclidean distance between a user and an item based on the pairwise hinge loss.
- **SML** [17]: The state-of-the-art OCCF method based on metric learning. For symmetrization, it considers the Euclidean distance among items as well as between a user and an item.

Next, the neighbor-based OCCF methods exploit the neighborhood information of users and items to compute the representations.

- **NGCF** [32]: A neighbor-based method which encodes a user's (and item's) neighbors by using graph convolutional networks (GCN). It can consider multi-hop neighbors as well based on a stack of GCN layers.
- **LGCN** [10]: The state-of-the-art method that further tailors the GCN-based user (and item) encoder for the OCCF task. It simplifies the GCN by using the light graph convolution.
- **M-VAE** [18]: The OCCF method based on a variational autoencoder that reconstructs partially-observed user vectors. It enforces the latent distribution to approximate the prior, assumed to be the normal distribution.
- **CFGAN** [1]: The state-of-the-art GAN-based OCCF method. The discriminator is trained to distinguish between input (real) user vectors and generated (fake) ones, while the generator is optimized to deceive the discriminator.

Among them, NGCF and LGCN are the discriminative methods that optimize their model by using the pairwise loss based on the BPR framework. On the contrary, M-VAE and CFGAN are the generative methods that focus on learning the latent distribution of users, represented by binary vectors indicating their interacted items. We build two variants of BUIR using different encoder networks.

- **BUIR_id**: The BUIR framework using a single embedding layer as its encoder. It simply takes the user/item vectors from the embedding matrix (Section 3.2).

- **BUIR$_{nb}$**: The BUIR framework based on the LGCN encoder. It computes the user/item representations by using the light-weight GCN [10] that adopts the proposed neighbor augmentation technique (Section 3.4).

Note that any types of user/item encoder networks, which are originally optimized in a discriminative framework (e.g., BPR), can be easily embedded into our framework.

Evaluation Protocols. For each dataset, we randomly split each user's interaction history into training/validation/test sets, with various split ratios. In detail, to verify the effectiveness of BUIR with varying levels of data sparsity, we build three training sets that include a certain proportion of interactions for each user, i.e., $\beta \in \{10\%, 20\%, 50\%\}$,[4] then equally divide the rest into the validation set and the test set. We report the average value of five independent runs, each of which uses different random seeds for the split.

As we focus on the top-K recommendation task for implicit feedback, we evaluate the performance of each method by using two widely-used ranking metrics [1, 17, 18]: Precision (P@K) and Normalized Discounted Cumulative Gain (N@K).[5] P@K measures how many test items are included in the list of top-K items and N@K assigns higher scores on the upper-ranked test items.

Implementation Details. We implement the proposed framework and all the baselines by using PyTorch, and use the Adam optimizer to train them. For BUIR, we fix the momentum coefficient τ to 0.995, and adopt a single linear layer for the predictor q_θ.[6] The augmentation function ψ simply uses a uniform distribution for drawing a drop probability $p \sim \mathcal{U}(0, 1)$, where each user's (item's) neighbor is independently deleted with the probability p.

For each dataset and baseline, we tune the hyperparameters using a grid search, which finds their optimal values that achieve the best performance on the validation set: the dimension size of representations $D \in \{50, 100, 150, 200, 250\}$, the weight decay (i.e., coefficient for L_2 regularization) $\lambda \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$, the initial learning rate $\eta \in \{10^{-1}, 10^{-2}, 10^{-3}10^{-4}\}$, and the number of negative pairs for each positive pair (particularly for discriminative baselines) $n \in \{1, 2, 5, 10, 20\}$. In case of baseline-specific hyperparameters, we tune them in the ranges suggested by their original papers. We set the maximum number of epochs to 500 and adopt the early stopping strategy; it terminates when P@10 on the validation set does not increase for 50 successive epochs.

4.2 Comparison with OCCF Methods

We first measure the top-K recommendation performance of BUIR and the baseline methods. Table 2 presents the comparison results on three different sparsity levels of datasets. In summary, BUIR achieves the best performance among all the baselines, and especially shows the significant improvements in highly sparse datasets. We analyze the results from various perspectives.

---

[4] This setting (high sparsity) is more difficult and practical than the traditional setting.
[5] As pointed out in [16], a sampled metric where only a smaller set of random items and the relevant items are ranked (e.g., leave-one-out evaluation protocol [11]) cannot correctly indicate the true performance of recommender systems. For this reason, we instead consider the ranked list of all the items with no interaction.
[6] We empirically found that these hyperparameters hardly affect the final performance of BUIR, and the sensitivity analysis on the parameters is provided in Section 4.6.
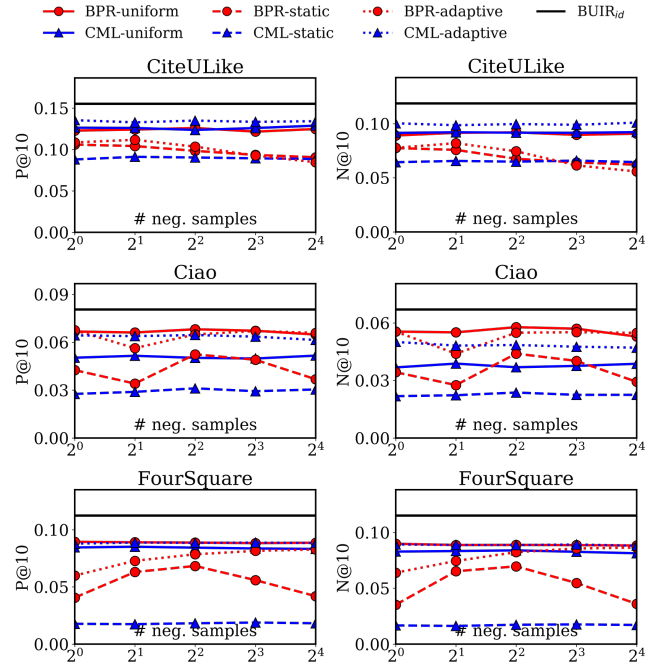

Figure 3: Comparison with discriminative methods (BPR and CML) using various negative sampling strategies.

4.2.1 Effectiveness of BUIR$_{id}$. For all the datasets, BUIR$_{id}$ shows the substantially higher performance than the discriminative methods taking only user-id/item-id (i.e., BPR, NeuMF, CML, and SML). In particular, the sparser the training set becomes, the larger the performance improvement of BUIR$_{id}$ is achieved over the best baseline (denoted by *Improv$_{id}$*). It is obvious that BUIR$_{id}$ is more robust to the extreme sparsity compared to the other baselines that are more likely to explicitly use "positive but unobserved" interactions as negative interactions when positive user-item interactions are more rarely observed. BUIR$_{id}$ is not affected by such inconsistent supervision from uncertain negative interactions because it directly optimizes the representations of users and items by using only positive interactions.

Furthermore, in terms of the number of retrieved items (denoted by K), BUIR shows much larger performance improvements for P@10 and N@10 compared to P@50 and N@50, respectively. In other words, BUIR performs much better at predicting the top-ranked items than the other baselines, which makes it practically advantageous for real-world recommender systems that aim to accurately provide the most preferred items to their users.

4.2.2 Effectiveness of BUIR$_{nb}$. We also observe that BUIR$_{nb}$ significantly outperforms all the other neighbor-based competitors, including discriminative (i.e., NGCF and LGCN) and generative methods (i.e., M-VAE and CFGAN). Similar to Section 4.2.1, there exist a consistent trend on its performance gain (denoted by *Improv$_{nb}$*), which becomes more significant as fewer interactions are given for training. Specifically, the neighbor-based baselines improve the recommendation performance over the methods not using the neighborhood information, as they are able to cope with the

Table 2: The recommendation performances of a wide range of OCCF methods, varying the sparsity of the datasets. *Improv*$_{id}$ and *Improv*$_{nb}$ respectively denote the improvement of BUIR over the best id/neighbor-based baseline. The superscripts *, **, and *** indicate $p \leq 0.05$, $p \leq 0.005$, and $p \leq 0.0005$ for the paired t-test of BUIR$_{nb}$ vs. the best baseline on P@10.

	Setting		User/Item ID						User/Item ID + Neighbor					
Data	β	Metric	BPR	NeuMF	CML	SML	BUIR$_{id}$	*Improv*$_{id}$	NGCF	LGCN	M-VAE	CFGAN	BUIR$_{nb}$	*Improv*$_{nb}$
CiteULike	10%***	P@10	0.0369	0.0350	0.0327	0.0279	**0.0542**	46.88%	0.0387	0.0518	0.0330	0.0437	**0.0637**	22.80%
		P@20	0.0484	0.0474	0.0451	0.0409	**0.0708**	46.20%	0.0506	0.0676	0.0444	0.0589	**0.0814**	20.38%
		P@50	0.0729	0.0785	0.0790	0.0685	**0.1050**	32.93%	0.0762	0.1010	0.0740	0.0968	**0.1202**	19.08%
		N@10	0.0310	0.0311	0.0272	0.0222	**0.0480**	54.60%	0.0337	0.0465	0.0289	0.0382	**0.0568**	22.11%
		N@20	0.0351	0.0349	0.0316	0.0266	**0.0533**	51.88%	0.0376	0.0516	0.0327	0.0433	**0.0623**	20.78%
		N@50	0.0429	0.0441	0.0421	0.0350	**0.0636**	44.18%	0.0456	0.0619	0.0417	0.0548	**0.0742**	19.91%
	20%***	P@10	0.0634	0.0422	0.0696	0.0515	**0.0903**	29.78%	0.0684	0.0835	0.0433	0.0730	**0.0956**	14.48%
		P@20	0.0862	0.0565	0.0964	0.0717	**0.1210**	25.41%	0.0915	0.1097	0.0601	0.0979	**0.1243**	13.37%
		P@50	0.1298	0.0847	0.1506	0.1145	**0.1775**	17.83%	0.1356	0.1607	0.0973	0.1237	**0.1807**	12.45%
		N@10	0.0510	0.0358	0.0576	0.0424	**0.0795**	37.93%	0.0580	0.0727	0.0377	0.0566	**0.0831**	14.22%
		N@20	0.0591	0.0407	0.0668	0.0494	**0.0880**	31.74%	0.0657	0.0812	0.0435	0.0631	**0.0912**	12.40%
		N@50	0.0726	0.0493	0.0833	0.0627	**0.1050**	25.96%	0.0793	0.0966	0.0548	0.0754	**0.1071**	10.82%
	50%**	P@10	0.1229	0.1138	0.1310	0.1195	**0.1555**	18.73%	0.1470	0.1561	0.1116	0.1389	**0.1624**	4.05%
		P@20	0.1719	0.1512	0.1845	0.1690	**0.2065**	11.91%	0.1978	0.2110	0.1513	0.1863	**0.2170**	2.84%
		P@50	0.2566	0.2162	0.2794	0.2545	**0.2993**	7.12%	0.2862	0.3056	0.2243	0.2677	**0.3120**	2.10%
		N@10	0.0891	0.0877	0.0950	0.0899	**0.1189**	25.23%	0.1122	0.1189	0.0843	0.1052	**0.1240**	4.29%
		N@20	0.1046	0.0994	0.1121	0.1055	**0.1348**	20.25%	0.1283	0.1360	0.0968	0.1201	**0.1405**	3.29%
		N@50	0.1276	0.1174	0.1379	0.1287	**0.1600**	16.06%	0.1525	0.1617	0.1169	0.1425	**0.1656**	2.40%
Ciao	10%***	P@10	0.0289	0.0302	0.0422	0.0461	**0.0598**	29.67%	0.0336	0.0582	0.0434	0.0521	**0.0664**	14.05%
		P@20	0.0346	0.0404	0.0603	0.0697	**0.0787**	12.97%	0.0430	0.0748	0.0573	0.0679	**0.0831**	11.07%
		P@50	0.0508	0.0627	0.1021	0.1043	**0.1123**	7.59%	0.0669	0.1095	0.0843	0.0972	**0.1177**	7.47%
		N@10	0.0278	0.0269	0.0369	0.0418	**0.0535**	27.96%	0.0313	0.0557	0.0391	0.0443	**0.0628**	12.75%
		N@20	0.0289	0.0301	0.0433	0.0506	**0.0588**	16.16%	0.0339	0.0597	0.0434	0.0479	**0.0675**	12.99%
		N@50	0.0337	0.0371	0.0566	0.0643	**0.0695**	8.02%	0.0415	0.0705	0.0519	0.0572	**0.0784**	11.17%
	20%**	P@10	0.0478	0.0361	0.0505	0.0517	**0.0608**	17.62%	0.0440	0.0662	0.0501	0.0535	**0.0724**	9.37%
		P@20	0.0623	0.0469	0.0728	0.0725	**0.0817**	12.19%	0.0580	0.0849	0.0656	0.0705	**0.0911**	7.30%
		P@50	0.0940	0.0711	0.1126	0.1089	**0.1210**	7.44%	0.0886	0.1247	0.0993	0.1090	**0.1322**	5.98%
		N@10	0.0436	0.0322	0.0432	0.0419	**0.0539**	23.68%	0.0403	0.0606	0.0445	0.0480	**0.0670**	10.52%
		N@20	0.0481	0.0354	0.0514	0.0492	**0.0598**	16.17%	0.0447	0.0659	0.0491	0.0525	**0.0725**	10.08%
		N@50	0.0578	0.0429	0.0666	0.0605	**0.0726**	8.98%	0.0546	0.0786	0.0596	0.0619	**0.0854**	8.71%
	50%**	P@10	0.0679	0.0426	0.0533	0.0639	**0.0786**	15.83%	0.0576	0.0746	0.0481	0.0705	**0.0812**	8.79%
		P@20	0.0909	0.0602	0.0875	0.0873	**0.1005**	10.53%	0.0831	0.1029	0.0676	0.0982	**0.1092**	6.10%
		P@50	0.1370	0.0924	0.1579	0.1279	**0.1642**	3.99%	0.1302	0.1578	0.1000	0.1502	**0.1673**	6.02%
		N@10	0.0563	0.0337	0.0394	0.0530	**0.0641**	13.86%	0.0463	0.0612	0.0391	0.0570	**0.0679**	10.98%
		N@20	0.0633	0.0393	0.0516	0.0601	**0.0706**	11.43%	0.0546	0.0702	0.0456	0.0659	**0.0766**	9.18%
		N@50	0.0767	0.0485	0.0725	0.0718	**0.0824**	7.54%	0.0685	0.0862	0.0551	0.0810	**0.0932**	8.17%
FourSquare	10%***	P@10	0.0561	0.0441	0.0451	0.0317	**0.0890**	58.70%	0.0658	0.0926	0.0451	0.0519	**0.0999**	7.84%
		P@20	0.0696	0.0528	0.0623	0.0421	**0.1020**	46.65%	0.0793	0.1054	0.0534	0.0612	**0.1127**	6.97%
		P@50	0.1173	0.0758	0.1180	0.0698	**0.1573**	33.32%	0.1305	0.1613	0.0746	0.0981	**0.1695**	5.06%
		N@10	0.0619	0.0476	0.0451	0.0311	**0.1023**	65.27%	0.0732	0.1080	0.0483	0.0579	**0.1161**	7.44%
		N@20	0.0665	0.0503	0.0531	0.0360	**0.1045**	57.27%	0.0771	0.1104	0.0508	0.0610	**0.1171**	6.05%
		N@50	0.0865	0.0600	0.0769	0.0480	**0.1277**	47.52%	0.0985	0.1336	0.0598	0.0770	**0.1409**	5.46%
	20%***	P@10	0.0752	0.0489	0.0754	0.0820	**0.1099**	34.11%	0.0872	0.1063	0.0658	0.0856	**0.1142**	7.43%
		P@20	0.0941	0.0633	0.0988	0.0985	**0.1281**	29.75%	0.1065	0.1239	0.0779	0.1035	**0.1323**	6.82%
		P@50	0.1573	0.1099	0.1714	0.1515	**0.1997**	16.47%	0.1733	0.1905	0.1220	0.1643	**0.2043**	7.24%
		N@10	0.0829	0.0532	0.0785	0.0833	**0.1273**	52.77%	0.0973	0.1247	0.0725	0.1003	**0.1328**	6.47%
		N@20	0.0898	0.0588	0.0888	0.0904	**0.1312**	45.07%	0.1036	0.1281	0.0760	0.1060	**0.1363**	6.42%
		N@50	0.1161	0.0782	0.1195	0.1129	**0.1607**	34.48%	0.1313	0.1564	0.0949	0.1321	**0.1660**	6.17%
	50%***	P@10	0.0894	0.0900	0.0843	0.1005	**0.1125**	12.00%	0.1064	0.1123	0.0838	0.0965	**0.1204**	7.25%
		P@20	0.1249	0.1237	0.1202	0.1404	**0.1546**	10.16%	0.1469	0.1509	0.1226	0.1325	**0.1595**	5.71%
		P@50	0.2059	0.2024	0.2025	0.2273	**0.2502**	10.07%	0.2373	0.2386	0.2086	0.2162	**0.2531**	6.09%
		N@10	0.0898	0.0919	0.0830	0.1002	**0.1152**	14.96%	0.1071	0.1153	0.0796	0.1007	**0.1211**	5.01%
		N@20	0.1046	0.1058	0.0979	0.1166	**0.1324**	13.55%	0.1241	0.1307	0.0962	0.1152	**0.1361**	4.12%
		N@50	0.1341	0.1344	0.1279	0.1484	**0.1672**	12.70%	0.1571	0.1657	0.1274	0.1456	**0.1709**	3.14%

Table 3: Performances of BUIR that ablates each component.

Method	Framework	Predictor	Neighbor	Augment	P@10
BPR	BPR				0.1229 ± 0.0035
	BPR	✓			0.0752 ± 0.0027
LGCN	BPR		✓		0.1561 ± 0.0038
BUIR_{id}	BUIR	✓			0.1555 ± 0.0029
	BUIR	✓	✓		0.1592 ± 0.0028
BUIR_{nb}	BUIR	✓	✓	✓	$\mathbf{0.1624} \pm 0.0032$

high sparsity to some degree by leveraging the neighbors of users and items. Nevertheless, most of them, except for LGCN, perform worse than even BUIR_{id}; this strongly indicates that their imperfect assumption on negative interactions severely limits the capability of capturing users' preference on items even though they utilize rich information sources as well as employ advanced neural architectures. In short, for the OCCF problem where only a small number of positive interactions are given, our BUIR framework is effective regardless of the information sources used for training, in that any assumption on negative interactions is not required.

In addition, the critical drawback of the generative methods is the difficulty of stable optimization. For example, M-VAE should carefully treat the annealing technique for minimizing Kullback-Leibler (KL) divergence, and CFGAN needs to balance the adversarial updates between the discriminator and generator for their convergence to the equilibrium. In contrast, BUIR can easily train the encoder without any advanced techniques for stable optimization, which makes our framework much practical.

4.2.3 Comparison of different negative sampling strategies. To examine how much the choice of a negative sampling strategy affects the recommendation performance, we measure P@10 and N@10 of two discriminative methods (i.e., BPR and CML) that adopt different strategies. We vary the number of negative pairs (sampled for each positive pair) in the range of $\{2^0, 2^1, 2^2, 2^3, 2^4\}$, and consider three different distributions for negative sampling [26]: 1) *uniform* sampling, 2) *static-and-global* sampling which draws a pair based on the item popularity, and 3) *adaptive-and-contextual* sampling that uses the probability proportional to the interaction score.

In Figure 3, we observe that the performance of the discriminative methods largely depends on the sampling strategy, whereas BUIR_{id} consistently performs the best. To be specific, the sampling strategies show different tendencies or have different optimal hyperparameter values, depending on each dataset or each method. For instance, CML achieves marginal performance gains from the adaptive-and-contextual sampling compared to the uniform sampling, whereas BPR does not take any benefits from it. This is because CML optimizes its model by the hinge loss, which cannot produce the gradients to update the model parameters for too easily-distinguishable negative pairs. In this case, the adaptive-and-contextual sampling strategy can effectively select the hard-negative pairs for training, which accelerates the convergence and its final performance. We remark that this kind of sampling techniques can improve the performance of the discriminative methods to some extent, but the sampling operation requires a high computational cost itself as well as the process of hyperparameter tuning for each dataset (and method) takes huge efforts. On the contrary,
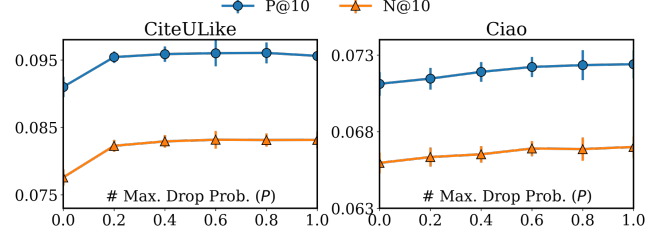
Figure 4: Performance changes of BUIR_{nb} with respect to the maximum drop probability for the augmentation.

as BUIR_{id} does not rely on negative sampling, it always shows the greater performance (plotted as a solid black line) compared to any of the discriminative methods using various sampling techniques. This result clearly validates the superiority of BUIR in that it is not affected by the choice of the negative sampling strategy any longer.

4.3 Ablation Study

To validate the effectiveness of each component in our framework, we measure the performance of the methods that ablate the following components: 1) modeling the interaction score based on the predictor (i.e., cross-prediction score defined in Equation (5)), 2) the neighbor-based encoder that is able to capture the user's (item's) neighborhood information, and 3) the stochastic neighbor augmentation that produces various views of an input interaction. In Table 3, we report P@10 on the CiteULike dataset (β=50%).

First of all, the BPR framework that optimizes the cross-prediction score, $q(f(u))^{\top} f(v) + f(u)^{\top} q(f(v))$, is not as effective as ours; it is even worse compared to the conventional BPR, which optimizes the inner-product score $f(u)^{\top} f(v)$. This implies that the performance improvement of BUIR is mainly caused by our learning framework rather than its score modeling based on the predictor. In addition, even without the stochastic augmentation, the neighbor-based encoder (i.e., LGCN) based on the BUIR framework beats LGCN based on the BPR framework, which demonstrates that BUIR successfully addresses the issue of incorrect negative sampling. Lastly, our framework with the stochastic neighbor augmentation further improves the performance by taking benefits from various views of the positive user-item interactions for the optimization.

4.4 Effect of Neighbor Augmentation

For an in-depth analysis on the effect of our stochastic data augmentation function ψ, we measure the performance of BUIR_{nb} on the CiteULike and Ciao datasets (β=20%), with various magnitudes of the perturbation added to the neighbors of users and items. We modify the augmentation function to randomly select the drop probability from a predefined interval, i.e., $p \sim \mathcal{U}(0, P)$ where P is the maximum drop probability, then increase P from 0.0 to 1.0.

In Figure 4, our stochastic data augmentation (i.e., $P > 0$) brings a significant improvement compared to the case of using the fixed neighborhood information (i.e., $P = 0$) as encoder inputs. This result shows that the augmented views of positive interactions encourage BUIR to effectively learn users' preference on items even in much sparse dataset. Interestingly, in case of the Ciao dataset which is less sparse than CiteULike, the benefit of our augmentation linearly
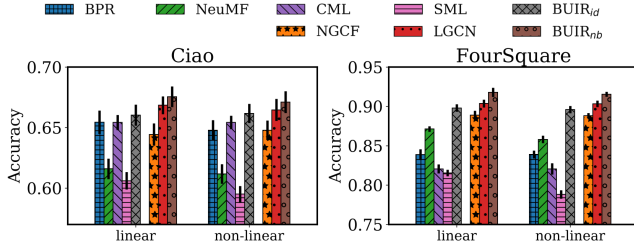
Figure 5: Evaluation on the quality of representations, by using a linear/non-linear classifier.

increases with the maximum drop probability. This is because there is room for producing more various views (i.e., larger perturbation) based on a relatively more number of neighbors, and it eventually helps to boost the recommendation performance. To sum up, our framework that adopts the neighbor augmentation function successfully relieves the data sparsity issue of the OCCF problem, by leveraging the augmented views of few positive interactions.

4.5 Evaluation on Representation Quality

To evaluate the quality of the obtained representations, we compare the performance for a downstream task by using the representations optimized by BUIR and the other baselines.[7] We consider an item classification task to evaluate how well each method encodes the items' characteristics or latent semantics into the representations. We choose two datasets that offer the side information on items, which are Ciao and FourSquare. Ciao provides the 28-category label of each item (i.e., the products), and FourSquare contains the GPS coordinates for each item (i.e., point-of-interest). In case of FourSquare, we first perform k-means clustering on the coordinates with $k=100$, and use the clustering results as the class labels. We train a linear and non-linear classifier (i.e., a single-layer perceptron and three-layer perceptron, respectively) to predict the class label of each item by using the fixed item representations as the input. Finally, we perform 10-fold cross-validation and report the average result and standard deviation.

In Figure 5, BUIR$_{id}$ and BUIR$_{nb}$ achieve significantly higher classification accuracy than the others in each category. This shows that the latent space induced by BUIR more accurately captures the item's characteristics (or their relationship) compared to the space induced by the baseline methods. Another observation is that the rank of each method for the downstream tasks is consistent with that for top-K recommendation (in Table 2). It implies that the observed user-item interactions are positively-correlated with the latent semantic of the items, for this reason, effectively learning the users' implicit feedback eventually results in a good performance in the downstream tasks as well.

4.6 Sensitivity Analysis

For the guidance of hyperparameter selection, we provide analyses on the sensitivity of BUIR to its several hyperparameters. We investigate the performance changes of BUIR$_{id}$ on the FourSquare

---

[7]In this comparison, we exclude the generative OCCF methods as our baselines, because they do not explicitly output the item representations.
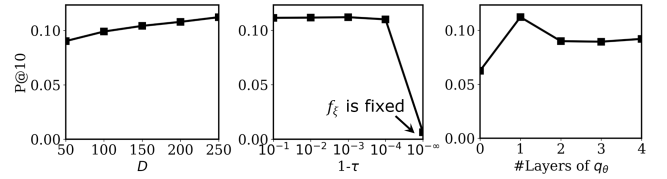

Figure 6: Sensitivity analyses on the BUIR hyperparameters.

dataset (β=50%) with respect to the dimension size D, the momentum coefficient τ,[8] and the number of layers in the predictor.

Figure 6 clearly shows that the performance is hardly affected by τ in the range of [0.9, 1.0). In other words, any values of τ larger than 0.9 allow the target encoder to successfully provide the target representations to the online encoder, by slowly approximating the online encoder; on the contrary, BUIR cannot learn the effective representations at all in case that the target encoder is fixed (i.e., $\tau = 1$). This observation is consistent with previous work on momentum-based moving average [6, 8, 29] that showed all values of τ between 0.9 and 0.999 can yield the best performance. Furthermore, BUIR performs the best with a single-layer predictor, because a multi-layer predictor makes it difficult to optimize the relationship between outputs of the two encoder networks. In conclusion, BUIR is more powerful even with fewer hyperparameters, compared to existing OCCF methods that include a variety of regularization terms or modeling components.

5 CONCLUSION

This paper proposes a novel framework for learning the representations of users and items, termed as BUIR, to address the main challenges of the OCCF problem: the implicit assumption about negative interactions, and high sparsity of observed (positively-labeled) interactions. First, BUIR directly bootstraps the representations of users and items by minimizing their cross-prediction error. This makes BUIR use only partially-observed positive interactions for training the model, and accordingly, it can eliminate the need for negative sampling. In addition, BUIR is able to learn the augmented views of each positive interaction obtained from the neighborhood information, which further relieves the data sparsity issue of the OCCF problem. Through the extensive comparison with a wide range of OCCF methods, we demonstrate that BUIR consistently outperforms all the other baselines in terms of top-K recommendation. In particular, the effectiveness of BUIR becomes more significant for much sparse datasets in which the positively-labeled interactions are not enough to optimize the model as well as the assumption about negative interactions becomes less valid. Based on its great compatibility with existing user/item encoder networks, we expect that our BUIR framework can be a major solution for the OCCF problem, replacing the conventional BPR framework.

---

[8]Considering that the target encoder should be slowly approximate the online encoder, we investigate τ in the range of [0.9, 1.0], as done in previous work [6, 8].

REFERENCES

[1] Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jung-Tae Lee. 2018. Cfgan: A generic collaborative filtering framework based on generative adversarial networks. In *CIKM*. 137–146.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *ICML*.

[3] Xinlei Chen and Kaiming He. 2021. Exploring Simple Siamese Representation Learning. In *CVPR*.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*. 4171–4186.

[5] Jingtao Ding, Guanghui Yu, Xiangnan He, Fuli Feng, Yong Li, and Depeng Jin. 2019. Sampler design for bayesian personalized ranking by leveraging view data. *TKDE* (2019).

[6] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. 2020. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS*. 21271–21284.

[7] Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*.

[8] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *CVPR*.

[9] Ruining He and Julian McAuley. 2016. VBPR: visual Bayesian Personalized Ranking from implicit feedback. In *AAAI*. 144–150.

[10] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*. 639–648.

[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.

[12] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *WWW*. 193–201.

[13] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM*. 263–272.

[14] SeongKu Kang, Junyoung Hwang, Wonbin Kweon, and Hwanjo Yu. 2020. DE-RRD: A Knowledge Distillation Framework for Recommender System. In *CIKM*. 605–614.

[15] Seunghyeon Kim, Jongwuk Lee, and Hyunjung Shim. 2019. Dual neural personalized ranking. In *WWW*. 863–873.

[16] Walid Krichene and Steffen Rendle. 2020. On Sampled Metrics for Item Recommendation. In *KDD*. 1748–1757.

[17] Mingming Li, Shuai Zhang, Fuqing Zhu, Wanhui Qian, Liangjun Zang, Jizhong Han, and Songlin Hu. 2020. Symmetric Metric Learning with Adaptive Margin for Recommendation. In *AAAI*. 4634–4641.

[18] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *WWW*. 689–698.

[19] Huafeng Liu, Jingxuan Wen, Liping Jing, and Jian Yu. 2019. Deep generative ranking for personalized recommendation. In *RecSys*. 34–42.

[20] Yiding Liu, Tuan-Anh Nguyen Pham, Gao Cong, and Quan Yuan. 2017. An Experimental Evaluation of Point-of-Interest Recommendation in Location-Based Social Networks. *PVLDB* 10, 10 (jun 2017), 1010–1021.

[21] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *ICML*. 1928–1937.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[23] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[24] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *ICDM*. 502–511.

[25] Chanyoung Park, Donghyun Kim, Xing Xie, and Hwanjo Yu. 2018. Collaborative translational metric learning. In *ICDM*. 367–376.

[26] Steffen Rendle and Christoph Freudenthaler. 2014. Improving pairwise learning for item recommendation from implicit feedback. In *WSDM*. 273–282.

[27] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*.

[28] Jiliang Tang, Huiji Gao, and Huan Liu. 2012. mTrust: discerning multi-faceted trust in a connected world. In *WSDM*. 93–102.

[29] Antti Tarvainen and Harri Valpola. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*. 1195–1204.

[30] Hao Wang, Binyi Chen, and Wu-Jun Li. 2013. Collaborative topic regression with social regularization for tag recommendation. In *IJCAI*.

[31] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *SIGIR*. 515–524.

[32] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*. 165–174.

[33] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *WSDM*.