Improving Top-K Recommendation via Joint Collaborative Autoencoders

Ziwei Zhu, Jianling Wang, and James Caverlee
Texas A&M University
{zhuziwei,jlwang,caverlee}@tamu.edu

ABSTRACT

In this paper, we propose a Joint Collaborative Autoencoder framework that learns both user-user and item-item correlations simultaneously, leading to a more robust model and improved top-K recommendation performance. More specifically, we show how to model these user-item correlations and demonstrate the importance of careful normalization to alleviate the influence of feedback heterogeneity. Further, we adopt a pairwise hinge-based objective function to maximize the top-K precision and recall directly for top-K recommenders. Finally, a mini-batch optimization algorithm is proposed to train the proposed model. Extensive experiments on three public datasets show the effectiveness of the proposed framework over state-of-the-art non-neural and neural alternatives.

CCS CONCEPTS

• **Information systems → Recommender systems**;

KEYWORDS

recommender systems; Autoencoder; hinge-based loss function

ACM Reference Format:
Ziwei Zhu, Jianling Wang, and James Caverlee. 2019. Improving Top-K Recommendation via Joint Collaborative Autoencoders. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3308558.3313678

1 INTRODUCTION

Since feedback like user clicks, purchases, and views is much more widespread than explicit ratings, implicit top-K recommenders offer great opportunities for far-reaching impact [6, 7, 17]. One of the most influential approaches is collaborative filtering (CF) [19], which learns the preference coherence among users (user-user CF) or the rating correlations among items (item-item CF). Recently, we have seen a series of works that aim to extend CF to new neural network frameworks that offer the promise of flexibility, non-linearity, and structural complexity, leading to more expressiveness and state-of-the-art recommendation performance [4, 5, 18, 20].

While promising, one limitation of these new neural neighborhood-based models is that they typically attempt to learn low-dimension embeddings of only users or only items, without careful consideration of the important interactions between users and items. As a result, the quality of recommendation may be restricted. In contrast, effective modeling of user-item interactions could lead to improved recommendation, as has been demonstrated in earlier non-neural approaches [16, 22, 25, 26]. We experimentally confirm this intuition in Section 3.3, where we find that while user-based and item-based Autoencoders do agree on many recommendations, there is a significant amount of disagreement between the two.

Hence, we propose in this paper a new Joint Collaborative Autoencoder (JCA) model that *jointly learns* user-user and item-item correlations simultaneously so that recommenders can take advantage of as much hidden information as possible for better recommendation quality. We show how to model these user-item interactions in JCA and demonstrate the importance of careful normalization to alleviate the influence of feedback heterogeneity (without which, naive methods may recommend items a user dislikes). Further, JCA adopts a pairwise hinge-based objective function for implicit top-K recommenders, which is especially designed for maximizing precision and recall for top-K recommendation. Through extensive experiments over three public datasets, we demonstrate how the proposed JCA model improves upon both non-neural and neural methods alike.

In sum, the contributions of this paper are as follows: (i) we propose a neural neighborhood-based CF approach called Joint Collaborative Autoencoder (JCA) that captures both user-user and item-item correlations simultaneously. (ii) We adopt a pairwise hinge-based objective function to optimize top-K precision and recall directly. (iii) We present a mini-batch optimization algorithm so that JCA can be trained on large datasets, without the (impractical) need to load the entire rating matrix as in a naive batch learning approach. (iv) Finally, through extensive experiments, we show that the proposed framework outperforms state-of-the-art neural and non-neural baselines.

2 RELATED WORK

Due to the flourishing development of neural networks, instead of focusing on the traditional non-neural algorithms [6, 7, 7, 12, 15, 17], more recent efforts have been dedicated to building neural based methods for implicit top-K recommendation. For example, Wu et al. [24] proposed a Collaborative Denoising Autoencoder (CDAE) model, which implements a neural version of the influential non-neural method SVD++ [9]. He el at. [5] presented a two-pathway architecture named NCF, ensembling a neural-based implementation of matrix factorization named Generalized Matrix Factorization (GMF) and a Multi-Layer Perceptron (MLP). Furthermore, there are many neural models for contextual recommenders [2, 3, 11, 14, 21] because neural models can be naturally extended to integrate auxiliary information. For example, Niu et al. [14] proposed a pairwise neural model integrating two GMF structures to implement BPR for

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313678

image recommendation, and considered visual, topic and geography information as contextual features. Similar to these existing neural-based recommenders, the model proposed in this paper is designed to take advantage of the flexibility, non-linearity, and structural complexity of neural architectures. Besides, we also demonstrate how to jointly learn from both user-user and item-item correlations for improved recommendation quality.

Previous works [16, 22, 25, 26] have identified the potential of combining user-based and item-based collaborative filtering models to improve recommendation. Although their motivations are similar to our proposed model, there are three main differences: (i) we focus on implicit top-K recommenders; (ii) we adopt an Autoencoder-based framework as the model foundation which is a neural-based algorithm; and (iii) we fuse the user-based and item-based models by jointly training them in a unified learning process, while previous work typically combines the two models via heuristics.

3 PRELIMINARIES

In this section, we define the implicit top-K recommendation problem, introduce the basics of an Autoencoder-based recommender, and empirically show that traditional user-based and item-based Autoencoder-based recommenders are complementary, which motivates the design of our proposed model.

3.1 Implicit top-K Recommendation

Formally, we denote the set of users as $\mathbb{U} = \{1, 2, \ldots, N\}$, the set of items as $\mathbb{I} = \{1, 2, \ldots, M\}$, and the set of historical feedback from the users to the items as $\mathbb{O} = \{u, i, r_{u,i}\}$ where u indexes one user, i indexes one item, and $r_{u,i}$ represents the rating from the user u to the item i.[1] For implicit top-K recommenders, there are usually only positive feedback signals, which means we only know which items user u likes but have no idea which items user u dislikes, i.e. all $r_{u,i}$ are 1. Based on this historical feedback, our goal is to predict the preference of each user u to the items they have not interacted with and recommend the items with the highest predicted scores to user u. Furthermore, we denote $\mathbb{O}_u^+$ as the set of items user u has given positive feedback to, and $\mathbb{O}_u^-$ denotes the set of items user u has not given feedback to. In practice, we will randomly sample a subset of the unrated items to train the model, hence in this paper we use $\mathbb{O}_u^-$ to denote the randomly selected negative samples.

3.2 Autoencoders for Recommendation

Toward tackling the implicit top-K recommendation problem, Autoencoder based models [20, 24] have shown promising performance in comparison with non-neural and other neural models. We input to an Autoencoder the rating vectors of users (or items) with 1 denoting a positive interaction, and 0 denoting a missing (or unknown) interaction. Then the Autoencoder outputs the corresponding completed rating vectors. Such an Autoencoder-based recommender called AutoRec has been proposed in [20], whose model can be written as:

$$\widehat{\mathbf{R}} = f(g(\mathbf{RV} + \mathbf{b}_1)\mathbf{W} + \mathbf{b}_2),$$

where the binary implicit feedback matrix $\mathbf{R}$ is input into AutoRec, and $\widehat{\mathbf{R}}$ is the predicted preference matrix. If we set both $f(\cdot)$ and $g(\cdot)$ as identity functions, and we also ignore the bias nodes, then AutoRec becomes: $\widehat{\mathbf{R}} = \mathbf{RVW}$, which is equivalent to the user-based neighborhood model with two factorized matrices as the similarity measurement (similar to the FISM model proposed in [7]). As a result, when we feed user rating vectors $\mathbf{R}^U \in \mathbb{R}^{N \times M}$ as the input (that is, user rating vectors are the rows), AutoRec learns the similarity among items. We refer to this model as user-based AutoRec (or U-AutoRec). In a similar fashion, we can input item rating vectors $\mathbf{R}^I \in \mathbb{R}^{M \times N}$ and learn similarity among users. We refer to this model as item-based AutoRec (or I-AutoRec).

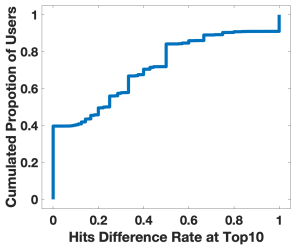

Figure 1: The CDF of the corresponding Hits Difference Rate between user-based AutoRec and item-based AutoRec.

3.3 User-based AutoRec and Item-based AutoRec Are Complementary

Sarwar showed in [19] that traditional (non-neural) user and item-based CF models are fundamentally different. Here, we investigate the difference between a neural neighborhood-based CF methods (U-AutoRec and I-AutoRec). More specifically, we explore whether different information from the same dataset captured by different models can produce complementary recommendations.

In order to answer this question, we run U-AutoRec and I-AutoRec on the same MovieLens 1M dataset (the details of this dataset and the model parameter settings are introduced in Section 5.1 and 5.2). For each user in this dataset, these two models will recommend two lists of ranked movies; we focus here on the top-10 ranked movies in the two lists. We consider the *hits* of a model as the set of movies that are in the recommended 10 movies and also in the test set, i.e. the ground truth of the user's preference. We denote the hits of U-AutoRec as $\mathbb{H}_U$ and of I-AutoRec as $\mathbb{H}_I$. Then we define the *hits difference rate* of these two model for one user as:

$$\text{HitsDifferenceRate} = \frac{size(\mathbb{H}_U \cup \mathbb{H}_I) - size(\mathbb{H}_U \cap \mathbb{H}_I)}{size(\mathbb{H}_U \cup \mathbb{H}_I)}.$$

where $size(\cdot)$ calculates the number of elements in a set, $\cap$ and $\cup$ are set intersection and union operators. The hits difference rate demonstrates the difference in correct recommendations between two models: a large difference rate indicates the two models provide complementary recommendations. The CDF of the hits difference rate for the users in MovieLens is presented in Figure 1. We can observe that more than 60% of users have a hits difference rate larger than 20%, with some users (around 30%) having a hits difference rate larger than 50%. Therefore, based on the observations from Figure 1

[1] In this paper, matrices are denoted by boldface uppercase letters like $\mathbf{A}$, and vectors are denoted by boldface lowercase letters like $\mathbf{a}$. $\mathbf{A}_i$ represents the row vector of the i-th row of matrix $\mathbf{A}$, and $(\mathbf{A}^\top)_j$ represents the row vector of the j-th column of matrix $\mathbf{A}$. We denote a user-item-rating matrix with N users and M items as $\mathbf{R} \in \mathbb{R}^{N \times M}$. $\mathbf{R}_u \in \mathbb{R}^{1 \times M}$ is the rating vector of user u, $(\mathbf{R}^\top)_i \in \mathbb{R}^{1 \times N}$ is the rating vector of item i. Besides, we use the syntax similar to Python to denote the matrix slicing operation, for example $\mathbf{A}[:, [1, 2]]$ denotes the first two columns of matrix $\mathbf{A}$.

	ML1M			Yelp			Games		
	P@10	R@10	F@10	P@10	R@10	F@10	P@10	R@10	F@10
U-AutoRec	**.2343**	.1698	.1969	.0230	.0608	.0333	.0152	.1022	.0265
I-AutoRec	.1782	.1454	.1602	.0202	.0556	.0296	.0138	.0979	.0241
UI-AutoRec	.2209	**.1801**	**.1984**	**.0240**	**.0646**	**.0350**	**.0171**	**.1192**	**.0298**

Table 1: Comparison between U-AutoRec, I-AutoRec and the average model of them – UI-AutoRec, where the best results are marked in bold. P represents precision, R represents recall, and F represent F1 score.

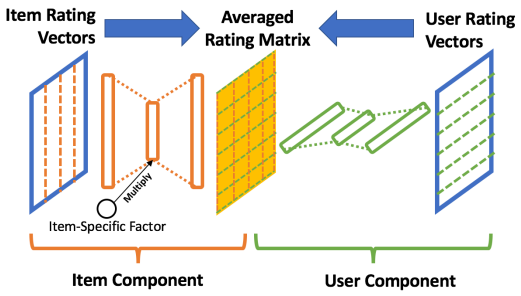

Figure 2: Overview of JCA: The Joint Collaborative Autoencoder takes both the Item Component with item rating vectors (left) and the User Component with user rating vectors (right) to recover the complete rating matrix (middle).

we can conclude that U-AutoRec and I-AutoRec generate different recommendations which are complementary to each other.

We further verify the conclusion that U-AutoRec and I-AutoRec are complementary by comparing the recommendation quality of U-AutoRec, I-AutoRec, and a simple averaging model of them (named UI-AutoRec, which averages the outputs of separately trained U-AutoRec and I-AutoRec to be the final result). In experiments over three datasets (details introduced in Section 5.1 and 5.2), we observe in Table 1 that the simple average of U-AutoRec and I-AutoRec can improve the top-10 recommendation quality, compared to either of the solo models, further indicating that U-AutoRec and I-AutoRec are complementary to each other. Therefore, combining the two models together to simultaneously explore both sides of information is promising for improved top-K recommendation.

4 PROPOSED MODEL

In this section, we present the design of a new **J**oint **C**ollaborative **A**utoencoder (**JCA**) model with a pairwise hinge-based objective function. In addition, we present a mini-batch optimization algorithm to optimize the JCA model.

4.1 Joint Collaborative Autoencoder

The intuition of JCA is to directly model both users and items simultaneously, rather than treating users and items in isolation. Further, naive combination of users and items without consideration of the heterogeneity of feedback can lead to overestimation of predicted scores in the output layer even for items a user dislikes. Hence, we describe in the following the design of the JCA structure with a special item normalization factor (to handle feedback heterogeneity), leading to the model shown in Figure 2.

Joint User-Item Correlations Modeling. Given N users and M items, the proposed JCA takes the whole rating matrix $\mathbf{R} \in \mathbb{R}^{N \times M}$ as a dual input. As illustrated in Figure 2, the rating matrix is input to the model on both the lefthand and righthand sides. The key difference is in how the matrix is sliced: the lefthand side takes as input one item rating vector $(\mathbf{R}^\top)_i \in \mathbb{R}^{1 \times N}$ as one sample (i.e. one column of $\mathbf{R}$), while the righthand side takes as input one user rating vector $\mathbf{R}_u \in \mathbb{R}^{1 \times M}$ as one sample (i.e. one row of $\mathbf{R}$). In essence, there are two components to the model, the first taking $\mathbf{R}_u$ as one input sample, which we name the *User Component*, and the other one with $(\mathbf{R}^\top)_i$ as one input sample, which we name the *Item Component*. The two components are two Autoencoder structures, which predict and recover the rating matrix independently. The first-layer weights of the user and item components are denoted as $\mathbf{V}^U \in \mathbb{R}^{M \times H^U}$ and $\mathbf{V}^I \in \mathbb{R}^{N \times H^I}$ respectively, where H^U and H^I represent the dimensions of the hidden layers of the two components. In the same way, we denote the second-layer weights as $\mathbf{W}^U \in \mathbb{R}^{H^U \times M}$ and $\mathbf{W}^I \in \mathbb{R}^{H^I \times N}$. The biases for the hidden layers and output layers are denoted as $\mathbf{b}_1^U$ and $\mathbf{b}_2^U$ for the user component, $\mathbf{b}_1^I$ and $\mathbf{b}_2^I$ for the item component. We set the sigmoid function as the activation function for hidden layers and output layers in both components.

Unlike the traditional user-based or item-based AutoRec, which inputs one or several incomplete rating vectors of a user or item and outputs the completed rating vectors, this Joint Collaborative Autoencoder takes the whole rating matrix as input, and outputs the whole completed rating matrix. The user component and item component output two completed rating matrices independently, and we combine the two outputs to be the final output as:

$$\widehat{\mathbf{R}} = \frac{1}{2}[h_{item}(\mathbf{R}) + h_{user}(\mathbf{R})], \tag{1}$$

where $\widehat{\mathbf{R}}$ is the predicted rating matrix, $h_{item}(\cdot)$ is the item component output, and $h_{user}(\cdot)$ is the user component output.

Learnable Item Normalization Factor. One challenge with traditional Autoencoder-based recommenders is that there is no normalization for the hidden layer. As a result, the heterogeneity of feedback can create problems. Take the user-based AutoRec as an example, inputting a user rating vector with more positive feedback into the AutoRec model results in a hidden layer with a larger norm, which leads to higher predicted scores in the output layer even for the items the user dislikes. A similar situation arises for the item-based AutoRec, where an item with more positive feedback has overall higher predicted scores even for the users who dislike it. Inspired by FISM [7] which adopts a user specific normalization constant, we add a learnable item normalization factor to alleviate the influence of this item feedback heterogeneity.

Specifically, we add a learnable *Item Normalization Factor* $\mathbf{f} \in \mathbb{R}^{M \times 1}$ multiplied to the hidden layer of the item component. This parameter is to normalize the norm of the latent factor for each item to solve the problem that items with more feedback will have higher predicted ratings. The user component does not need a normalization factor because the recommender ranks the predicted preference scores and recommends the top-K items for each user independently. Hence the norm of the rating vector of each user has no influence to the recommendation result.

The JCA Model. Combining the joint user-item correlations model with this learnable item normalization factor leads to the full Joint

Collaborative Autoencoder model:

$$\widehat{\mathbf{R}} = \frac{1}{2}[\sigma(\sigma(\mathbf{R}\mathbf{V}^U + \mathbf{b}_1^U)\mathbf{W}^U + \mathbf{b}_2^U) + \sigma(\sigma((\mathbf{R}^\top\mathbf{V}^I + \mathbf{b}_1^I) \circ \mathbf{f})\mathbf{W}^I + \mathbf{b}_2^I)^\top], \tag{2}$$

where $\sigma(x) = 1/(1+e^{-x})$ is the sigmoid function, $\circ$ is the element-wise product (because $\mathbf{f}$ is a column vector, $\circ$ multiplies each entry of $\mathbf{f}$ to each corresponding row of $(\mathbf{R}^\top\mathbf{V}^I + \mathbf{b}_1^I)$).

4.2 Pairwise Hinge-based Objective Function for Implicit Top-K Recommenders

We could learn the described model parameters by optimizing a traditional objective function for recommenders. Nevertheless, many of these conventional objective functions attempt to optimize MSE or AUC [17, 20, 24], which are not aligned with top-K recommendation metrics. As a result, we introduce a pairwise hinge-based objective function to directly optimize implicit top-K recommendation performance. In addition, we will show how the hinge-based objective function links to top-K precision and recall.

Inspired by [23] and [13], which proposed new objective functions to optimize position sensitive ranking loss functions such as ADG and NDCG, we adopt a pairwise hinge-based objective function to optimize top-K ranking result, shown in Equation (3):

$$\underset{\Theta}{\text{minimize}} \quad \mathcal{L} = \sum_{u\in\mathbb{U}} \sum_{\substack{i\in\mathbb{O}_u^+ \\ j\in\mathbb{O}_u^-}} max(0, \widehat{r}_{u,j} - \widehat{r}_{u,i} + d) + \frac{\lambda}{2}\|\Theta\|_F^2, \tag{3}$$

where $\widehat{r}_{u,i}$ and $\widehat{r}_{u,j}$ are the predicted ratings for user u to item i with positive feedback and j from negative sampling; λ is the trade-off parameter of the L2-norm regularization term; Θ represents all the parameters; d is a hyper-parameter determining how large the margin between positive feedback and random negative samples can be; and $\|\cdot\|_F$ is the Frobenius norm.

Because in practice we can not use all unobserved entries as the negative samples, we set a constant margin d for the random negative samples and positive feedback to counteract the influence from random negative sampling strategy.

Analogies to Precision@k and Recall@k. *Precision@k* (*P@k*) and *Recall@k* (*R@k*) are two most commonly used metrics to evaluate implicit top-K recommenders. They are defined as:

$$P@k = |\mathbb{O}_u^k \cap \mathbb{O}_u^+|/k, \quad |\mathbb{O}_u^k \cap \mathbb{O}_u^+|/|\mathbb{O}_u^+|, \tag{4}$$

where $\mathbb{O}_u^k$ is the set of items with the top k predicted preference scores for user u, and $|\cdot|$ calculates the number of elements in a set.

If we set $k = |\mathbb{O}_u^+|$, then *Precision@k* and *Recall@k* will be the same, and we can rewrite the formula as:

$$P@k = R@k = 1 - (\sum_{i\in\mathbb{O}_u^+} \delta((\sum_{j\in\mathbb{O}_u^-} (\widehat{r}_{u,j} - \widehat{r}_{u,i})) > 0))/|\mathbb{O}_u^+|, \tag{5}$$

where $\delta(x)$ is the Heaviside function, which returns 1 if x is true, otherwise 0.

The analogy between (5) and (3) is clear. To maximize the top-K precision and recall shown in Equation (5) is equivalent to minimizing $\sum_{i\in\mathbb{O}_u^+} \delta((\sum_{j\in\mathbb{O}_u^-}(\widehat{r}_{u,j} - \widehat{r}_{u,i})) > 0)$, which is equivalent to minimizing $\sum_{i\in\mathbb{O}_u^+, j\in\mathbb{O}_u^-} max(0, \widehat{r}_{u,j} - \widehat{r}_{u,i} + d)$ in Equation (3). Here, we use the differentiable Hinge loss function to replace the non-differentiable Heaviside function.

4.3 Mini-batch Optimization Algorithm

We can use any popular optimization algorithm to train the proposed JCA. However, one challenge is that in each training iteration, the whole user-item-rating matrix is used for training, which is not practical especially for large datasets. As a result, we propose a mini-batch optimization algorithm for handling large datasets.

For the mini-batch optimization algorithm, in each training iteration, we need to feed a mini-batch consisting of n user rating vectors denoted as $\mathbf{B}^U = \mathbf{R}[\mathbf{p}, :] \in \mathbb{R}^{n\times M}$ into the user component, where $\mathbf{p}$ is a list of n randomly sampled row indexes; and another mini-batch consisting of m item rating vectors denoted as $\mathbf{B}^I = \mathbf{R}[:, \mathbf{q}]^\top \in \mathbb{R}^{m\times N}$ into the item component, where $\mathbf{q}$ is a list of m randomly sampled column indexes. The user component recovers and outputs $\widehat{\mathbf{B}}^U$, and item component outputs $\widehat{\mathbf{B}}^I$. Due to the distinct shapes of $\widehat{\mathbf{B}}^U$ and $\widehat{\mathbf{B}}^I$, we can not simply average the them as Equation (2). Nevertheless, there are $n \times m$ common entries covered by both of $\widehat{\mathbf{B}}^U$ and $\widehat{\mathbf{B}}^I$, and we denote this commonly covered part as a matrix $\mathbf{C} \in \mathbb{R}^{n\times m}$. In this way, the whole rating matrix will be decomposed into several small matrices. We can consider one small matrix as one mini-batch. Then during each training iteration, one mini-batch $\mathbf{C}$ will be used for one time of updating, where only the weights connecting the training samples in $\mathbf{C}$ will be updated. Hence, the model formulation in (2) can be rewritten in the mini-batch format:

$$\begin{aligned}
\widehat{\mathbf{C}} &= \frac{1}{2}(\widehat{\mathbf{B}}^U[:, \mathbf{q}] + \widehat{\mathbf{B}}^I[\mathbf{p}, :]),\\
\widehat{\mathbf{B}}^U &= \sigma(\sigma(\mathbf{B}^U\mathbf{V}^U + \mathbf{b}_1^U)\mathbf{W}^U + \mathbf{b}_2^U),\\
\widehat{\mathbf{B}}^I &= \sigma(\sigma(((\mathbf{B}^I)^\top\mathbf{V}^I + \mathbf{b}_1^I) \circ \mathbf{f})\mathbf{W}^I + \mathbf{b}_2^I)^\top,
\end{aligned}$$

where $\widehat{\mathbf{B}}^U$, $\widehat{\mathbf{B}}^I$, and $\widehat{\mathbf{C}}$ are the predicted user mini-batch matrix, item mini-batch matrix, and the combined predicted mini-batch matrix, respectively; $\widehat{\mathbf{B}}^U[:, \mathbf{q}]$ means the columns indexed by $\mathbf{q}$ in $\widehat{\mathbf{B}}^U$, and $\widehat{\mathbf{B}}^I[\mathbf{p}, :]$ represents the rows indexed by $\mathbf{p}$ in $\widehat{\mathbf{B}}^I$.

The objective function (3) can be rewritten as:

$$\underset{\Theta}{\text{minimize}} \quad \mathcal{L} = \sum_{u\in{}^{\mathbf{C}}\mathbb{U}} \sum_{\substack{i\in{}^{\mathbf{C}}\mathbb{O}_u^+ \\ j\in{}^{\mathbf{C}}\mathbb{O}_u^-}} max(0, \widehat{r}_{u,j} - \widehat{r}_{u,i} + d) + \frac{\lambda}{2}\|\Theta\|_F^2,$$

where ${}^{\mathbf{C}}\mathbb{U}$ is the users set in mini-batch matrix $\mathbf{C}$, ${}^{\mathbf{C}}\mathbb{O}_u^+$ is the items set that user u gives positive feedback to in $\mathbf{C}$, and ${}^{\mathbf{C}}\mathbb{O}_u^-$ is the negative samples of user u in $\mathbf{C}$. This approach has the benefit of only needing to load the sampled user and item rating vectors into memory rather than storing the whole rating matrix as the batch learning algorithm does.

5 EXPERIMENTS

In this section, we empirically evaluate the proposed model in comparison with state-of-the-art non-neural and neural approaches over three datasets. We aim to answer two key questions: (i) How does the proposed JCA framework perform compared with other approaches for implicit top-K recommendation? and (ii) Are the proposed Joint Collaborative Autoencoder structure JCA and the pairwise hinge-based objective function effective?

	Users	Items	Ratings	Sparsity(%)
MovieLens 1M (ML1M)	6,027	3,062	574,026	3.11
Yelp	12,705	9,245	318,314	0.271
Video Games (Games)	19,056	9,073	184,609	0.107

Table 2: Characteristics of the three datasets.

5.1 Datasets

We use three public datasets from different domains for our experiments: MovieLens 1M[2] (ML1M), Yelp[3], and Video Games[4] (customer reviews from the Video Games category on Amazon). For ML1M and Yelp, we treat ratings higher or equal to 4 as positive feedback, and consider all other ratings as missing entries. For Game, we consider all purchase behaviors as positive feedback. These strategies are widely used for evaluating implicit recommenders [7, 17, 27]. Then we iteratively remove users and items with fewer than 5 positive feedback. Finally, we split the datasets into 20% for testing and 80% for training. In the training set, we further hold 10% as a validation set. The datasets are summarized in Table 2.

5.2 Experimental Setup

Metrics. We adopt *Precision@k* and *Recall@k* averaged across all users as the evaluation metrics for personalized ranking (recall Equation (4)). We also report the *F1@k*, which is defined as:

$$F1@k = (2 \cdot P@k \cdot R@k)/(P@k + R@k).$$

Baselines. To evaluate the proposed model, we consider eight state-of-the-art methods including neural and traditional models:

MF [10]. The conventional matrix factorization model with MSE objective function and Alternating Least Squares (ALS) optimizer.

BPR [17]. BPR learns an MF model by the pairwise ranking objective function. We use Gradient Descent to optimize the objective function.

CDAE [24]. This is an Autoencoder-based model, which implements the neural version of SVD++. We adopt the sigmoid function as the activation functions in both the hidden layer and output layer, and use MSE as the objective function.

U-AutoRec [20]. This is a user-based AutoRec model, which takes user rating vectors as inputs and learns item-item correlations. It uses the MSE loss function.

I-AutoRec [20]. Similar to U-AutoRec, but takes item rating vectors as inputs and learns user-user correlations.

UI-AutoRec. This model averages the outputs of U-AutoRec and I-AutoRec.

NCF [5]. This model combines a neural version of MF model (named GMF) and a multi-layer perceptron (MLP) model. The model structure configuration is similar according to the original paper where MLP has four hidden layers with 8, 16, 32, and 64 hidden neurons, respectively.

NPR [14]. This is a neural version of BPR with two GMFs to implement the pairwise structure and the BPR objective function.

Reproducibility. All code, data, and experimental settings can be found at https://github.com/Zziwei/Joint-Collaborative-Autoencoder. We implement the proposed model based on Tensorflow [1] and use Adam [8] optimizer. We set the learning rate 0.003 for all methods.

[2]http://files.grouplens.org/datasets/movielens/ml-1m.zip
[3]https://www.yelp.com/dataset/challenge
[4]http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/ratings_Video_Games.csv

	ML1M			Yelp			Games		
	@1	@5	@10	@1	@5	@10	@1	@5	@10
BPR	.0427	.1306	.1704	.0068	.0185	.0221	.0138	.0189	.0175
MF	.0477	.1427	.1837	.0148	.0307	.0333	.0255	.0294	.0265
NCF	.0515	.1454	.1870	.0147	.0316	.0343	.0246	.0319	.0281
NPR	.0486	.1427	.1806	.0110	.0267	.0295	.0158	.0221	.0198
I-AutoRec	.0472	.1249	.1602	.0139	.0278	.0296	.0198	.0266	.0241
U-AutoRec	.0526	.1541	.1968	.0148	.0311	.0333	.0217	.0290	.0265
UI-AutoRec	.0540	.1562	.1984	.0149	.0319	.0350	**.0270**	**.0340**	.0298
CDAE	**.0552**	**.1578**	**.1999**	**.0154**	**.0321**	**.0353**	.0252	.0331	**.0301**
JCA	**.0602**	**.1634**	**.2080**	**.0164**	**.0346**	**.0376**	**.0297**	**.0364**	**.0322**
%improve	8.9%	3.5%	4.1%	6.5%	7.8%	6.5%	10.0%	7.1%	7.0%

Table 3: F1@k comparison between the proposed JCA and eight state-of-the-art baselines, where the results of proposed method and the best baselines are marked in bold.

For the mini-batch size, we set 256 for single side based methods (AutoRec and CDAE), and 1,500 for other methods (both bs^U and bs^I are set to be 1500 for JCA). For the regularization trade-off parameter, we grid search the best ones over [0.5, 0.1, 0.05, 0.01, 0.005, 0.001] for each model by the validation set. For ML1M dataset: we set $\lambda = 0.001$ for JCA, 0.05 for JCA-MSE, and 0.01 for JCA-BPR; for Yelp dataset: $\lambda = 0.005$ for JCA, 0.05 for JCA-MSE, and 0.001 for JCA-BPR; for Games dataset $\lambda = 0.001$ for JCA, 0.001 for JCA-MSE, and 0.001 for JCA-BPR. Moreover, we set the margin parameter $d = 0.15$ for JCA. We grid search hidden layer dimension over [40, 80, 120, 160, 200, 240] and set 160 hidden neurons for JCA (we set the hidden dimensions the same for both user and item component, fine tuning the dimensions separately may provide better performance). We re-sample negative samples in each epoch and set the negative sampling rate 1 for all models. Other model-specific hyper-parameters of baselines are fine-tuned over validation sets.

5.3 Proposed Model vs. State-of-the-art Models

We begin by investigating the recommendation quality of JCA compared with eight baselines over three datasets. The precision and recall results are presented in Figure 3 and Figure 4. For both metrics, we see that JCA produces the best performance. Besides, we also calculate the F1-score to further evaluate the recommendation quality as shown in Table 3, where the F1 scores for the proposed method and the best baselines are marked in bold, and the improvement rates are calculated for all settings. In terms of average *F1@k* result (across three different values of k), JCA can outperform the best baseline by around 5.5% on the MovieLens dataset, around 6.9% on the Yelp dataset, and around 8% on the Games dataset. We further do a t-test on the small improvement cases (ML1M @5 and @10), where we find the p-value is smaller than 1.5% for both, showing the statistical significance of the performance improvement.

In addition, we observe that: (i) the performance improvement is larger for sparser datasets, which may be because in high sparsity scenarios, user-user correlations or item-item correlations alone are not enough, while joint learning of user-item correlations can extract more useful information; (ii) the performance improvement is larger for smaller k (k defines how many items to be recommended to users), which indicates that JCA is more competitive when limited number of items should be recommended to users.

5.4 Proposed Model and Objective Function

To demonstrate the effectiveness of the proposed JCA structure and the pairwise hinge-based objective function, we implement two variations of JCA using alternative objective functions (MSE and

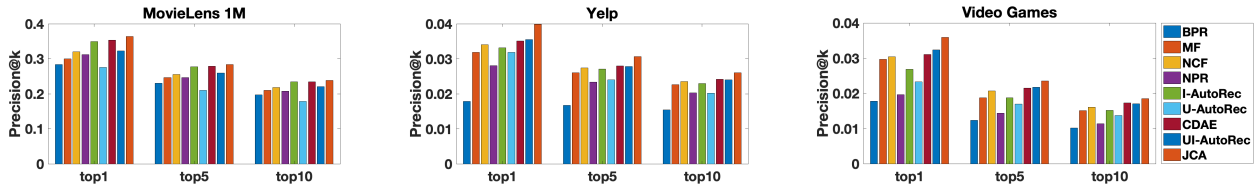

Figure 3: Precision@k results of the proposed method vs. baselines for the three datasets.

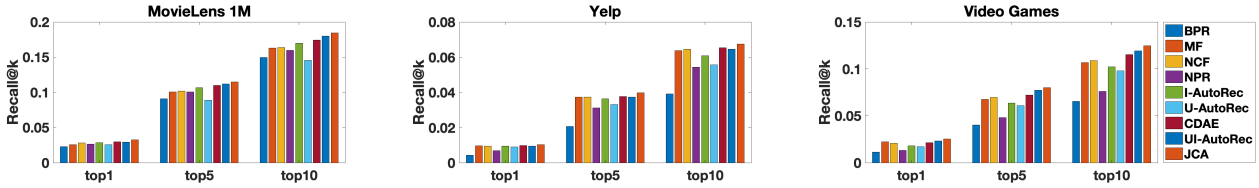

Figure 4: Recall@k results of the proposed method vs. baselines for the three datasets.

	ML1M			Yelp			Games		
	P@10	R@10	F@10	P@10	R@10	F@10	P@10	R@10	F@10
JCA-MSE	**.2346**	**.1768**	**.2017**	**.0256**	**.0665**	**.0370**	**.0178**	**.1205**	**.0311**
NCF	.2182	.1637	.1870	.0235	.0636	.0343	.0161	.1089	.0281
UI-AutoRec	.2209	**.1801**	.1984	.0240	.0646	.0350	.0171	**.1192**	.0298
CDAE	**.2344**	.1743	**.1999**	**.0242**	**.0654**	**.0353**	**.0173**	.1154	**.0301**
JCA-BPR	**.2106**	**.1597**	**.1817**	**.0234**	**.0639**	**.0343**	**.0173**	**.1160**	**.0301**
NPR	**.2078**	**.1594**	**.1806**	**.0203**	**.0543**	**.0295**	**.0114**	**.0762**	**.0198**
BPR	.1975	.1498	.1704	.0154	.0391	.0221	.0101	.0655	.0175

Table 4: Comparison between JCA models and baselines with same objective functions, where the results of the variations of JCA and the best baselines are marked in bold.

	ML1M			Yelp			Games		
	P@10	R@10	F@10	P@10	R@10	F@10	P@10	R@10	F@10
JCA	**.2384**	**.1845**	**.2080**	**.0261**	**.0674**	**.0376**	**.0185**	**.1247**	**.0322**
JCA-MSE	.2346	.1768	.2017	.0256	.0665	.0370	.0178	.1205	.0311
JCA-BPR	.2106	.1597	.1817	.0234	.0639	.0343	.0173	.1160	.0301

Table 5: Comparison between JCA models with different objective functions, where the best results are marked in bold.

BPR), and compare the performance with other models using the same objective functions. We've argued in Section 4.2 that these objective functions are not aligned with top-K recommendation, so we seek to untangle their impact. We denote the model combine JCA structure with MSE objective function as JCA-MSE, and the one with BPR loss function as JCA-BPR.

Here we report the comparison between the JCA variations (JCA-MSE and JCA-BPR) and the best three models using the MSE objective function – CDAE, UI-AutoRec and NCF – and the models using the BPR objective function – NPR and BPR. We follow the same parameter tuning process as introduced in 5.2 for the two JCA variations. We report the Precision, Recall and F1 at top 10, but note that the results for top 1 and 5 have similar patterns.

First, let's focus on the comparison between JCA variations and other models with the same objective functions to evaluate the effectiveness of the JCA structure. Table 4 shows that JCA-MSE produces better performance than CDAE, UI-AutoRec and NCF for all three datasets in terms of F1@10 and Precision@10. From the perspective of recall, there are only one exceptional case: on ML1M dataset UI-AutoRec is 1.8% better than JCA-MSE for Recall@10. For the BPR objective function, JCA-BPR outperforms NPR and BPR significantly. Based on these comparisons, we can conclude that the proposed JCA model is effective for implicit top-K recommenders, regardless of the choice of objective function.

	ML1M			Yelp			Games		
	P@10	R@10	F@10	P@10	R@10	F@10	P@10	R@10	F@10
JCA	**.2384**	**.1845**	**.2080**	**.0261**	**.0674**	**.0376**	**.0185**	**.1247**	**.0322**
JCA-NF	.2303	.1777	.2006	.0257	.0669	.0371	.0179	.1214	.0313

Table 6: Comparison between JCA models with/without the learnable item normalization factor, where the best results are marked in bold.

Next, we turn to investigate whether the pairwise hinge-based objective function is effective or not. Comparing the results of the three variations of JCA in Table 5, we can observe the recommendation performance trend: JCA > JCA-MSE > JCA-BPR for all three metrics and all three datasets. This provides empirical evidence that the proposed hinge objective function is more effective and suitable for implicit top-K recommenders.

Further, to validate the effectiveness of the learnable item normalization factor in the proposed JCA structure, we also implement a variation of JCA without the item normalization factor named JCA-NF in comparison with JCA. The result is presented in Table 6, which shows that JCA model with the item normalization factor produces better recommendation performance for all three datasets, and the performance improvement rate is around 5%.

6 CONCLUSION AND FUTURE WORK

In this paper, we show that user-based AutoRec and item-based AutoRec models explore different information from the data and produce complementary recommendations. Based on this observation, we propose the Joint Collaborative Autoencoder framework that learns both user-user and item-item correlations simultaneously, leading to improved implicit top-K recommendation quality. Besides, the proposed JCA adopts a pairwise hinge-based objective function to optimize the top-K precision and recall. We also present a mini-batch optimization algorithm so that JCA can be trained on large datasets without the (impractical) need to load the entire rating matrix as in naive batch learning approach. By extensive experiments on three public datasets, we show that the proposed framework outperforms state-of-the-art baselines. In our continuing work, we are exploring how to incorporate auxiliary information, such as textual and visual information, into the framework to further improve the recommendation quality.

Acknowledgments. This work is supported in part by NSF IIS-1841138.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning.. In *OSDI*, Vol. 16. 265–283.

[2] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 191–198.

[3] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. 2015. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 278–288.

[4] Kostadin Georgiev and Preslav Nakov. 2013. A non-iid framework for collaborative filtering with restricted boltzmann machines. In *International Conference on Machine Learning*. 1148–1156.

[5] 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*. International World Wide Web Conferences Steering Committee, 173–182.

[6] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. Ieee, 263–272.

[7] Santosh Kabbur, Xia Ning, and George Karypis. 2013. Fism: factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 659–667.

[8] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[9] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 426–434.

[10] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

[11] Chenyi Lei, Dong Liu, Weiping Li, Zheng-Jun Zha, and Houqiang Li. 2016. Comparative deep learning of hybrid representations for image recommendations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2545–2553.

[12] Gai Li and Weihua Ou. 2016. Pairwise probabilistic matrix factorization for implicit feedback collaborative filtering. *Neurocomputing* 204 (2016), 17–25.

[13] Daryl Lim, Julian McAuley, and Gert Lanckriet. 2015. Top-n recommendation with missing implicit feedback. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 309–312.

[14] Wei Niu, James Caverlee, and Haokai Lu. 2018. Neural Personalized Ranking for Image Recommendation. In *Proceedings of 11th ACM International Conference on Web Search and Data Mining*.

[15] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 502–511.

[16] Aghny Arisya Putra, Rahmad Mahendra, Indra Budi, and Qorib Munajat. 2017. Two-steps graph-based collaborative filtering using user and item similarities: Case study of E-commerce recommender systems. In *Data and Software Engineering (ICoDSE), 2017 International Conference on*. IEEE, 1–6.

[17] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[18] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*. ACM, 791–798.

[19] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. ACM, 285–295.

[20] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 111–112.

[21] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1235–1244.

[22] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. 2006. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 501–508.

[23] Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*, Vol. 11. 2764–2770.

[24] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 153–162.

[25] Pratibha Yadav and Shweta Tyagi. 2017. Hybrid fuzzy collaborative filtering: an integration of item-based and user-based clustering techniques. *International Journal of Computational Science and Engineering* 15, 3-4 (2017), 295–310.

[26] Akihiro Yamashita, Hidenori Kawamura, and Keiji Suzuki. 2011. Adaptive fusion method for user-based and item-based collaborative filtering. *Advances in Complex Systems* 14, 02 (2011), 133–149.

[27] Shuang-Hong Yang, Bo Long, Alexander J Smola, Hongyuan Zha, and Zhaohui Zheng. 2011. Collaborative competitive filtering: learning recommender using context of user choice. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 295–304.