The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Collaborative Filtering with User-Item Co-Autoregressive Models

Chao Du,[†] Chongxuan Li,[†] Yin Zheng,[‡] Jun Zhu,[*†] Bo Zhang[†]
[†]Dept. of Comp. Sci. & Tech., State Key Lab of Intell. Tech. & Sys., TNList Lab,
[†]Center for Bio-Inspired Computing Research, Tsinghua University, Beijing, 100084, China
[‡]Tencent AI Lab, Shenzhen, Guangdong, China

Abstract

Deep neural networks have shown promise in collaborative filtering (CF). However, existing neural approaches are either user-based or item-based, which cannot leverage all the underlying information explicitly. We propose CF-UIcA, a neural co-autoregressive model for CF tasks, which exploits the structural correlation in the domains of both users and items. The co-autoregression allows extra desired properties to be incorporated for different tasks. Furthermore, we develop an efficient stochastic learning algorithm to handle large scale datasets. We evaluate CF-UIcA on two popular benchmarks: MovieLens 1M and Netflix, and achieve state-of-the-art performance in both rating prediction and top-N recommendation tasks, which demonstrates the effectiveness of CF-UIcA.

1 Introduction

With the fast development of electronic commerce, social networks and music/movie content providers, recommendation systems have attracted extensive research attention (Burke 2002; Schafer et al. 2007). As one of the most popular methods, collaborative filtering (CF) (Schafer et al. 2007; Billsus and Pazzani 1998; Resnick et al. 1994; Salakhutdinov, Mnih, and Hinton 2007) predicts users' preferences for items based on their previous behaviors (rating/clicking/purchasing etc.) in a recommendation system. CF enjoys the benefit of content-independence of the items being recommended. Thus, it does not need expert knowledge about the items when compared with content-based methods (Van den Oord, Dieleman, and Schrauwen 2013; Gopalan, Charlin, and Blei 2014) and could possibly provide cross-domain recommendations.

The basic assumption behind CF is that there exist correlations between the observed user behaviors, and these correlations can be generalized to their future behaviors. Basically, the correlations can be categorized as *User-User Correlations* (UUCs)—the correlations between different users' behaviors on a same item, and *Item-Item Correlations* (IICs)—the correlations between a user's behaviors on different items. These two types of underlying correlations usually exist crisscrossing in the partially observed user behaviors, making CF a difficult task.

[*]corresponding author.

Extensive work has studied how to effectively exploit the underlying correlations to make accurate predictions. Early approaches (Resnick et al. 1994; Sarwar et al. 2001) consider UUCs or IICs by computing the similarities between users or items. As one of the most popular classes of CF methods, matrix factorization (MF) (Billsus and Pazzani 1998; Koren, Bell, and Volinsky 2009; Salakhutdinov and Mnih 2007) assumes that the partially observed matrix (of ratings) is low-rank and embeds both users and items into a shared latent space. MF methods consider both UUCs and IICs implicitly as a prediction is simply the inner product of the latent vectors of the corresponding user and item. Recently, deep learning methods have achieved promising results in various tasks (Bahdanau, Cho, and Bengio 2014; Mnih et al. 2015; Silver et al. 2016) due to their ability to learn a rich set of abstract representations. Inspired by these advances, neural networks based CF methods (Salakhutdinov, Mnih, and Hinton 2007; Sedhain et al. 2015; Wu et al. 2016; Zheng et al. 2016b), which employ highly flexible transformations to model a user's behavior profile (all behaviors) with a compact representation, are widely studied as alternatives to MF. These methods essentially consider all UUCs explicitly as they take inputs of users' all observed behaviors. (See more details in Sec. 2.)

Though previous neural networks based methods are promising, one common drawback of these methods lies in that they cannot exploit both UUCs and IICs together, making them further improvable. To this end, we propose a novel neural autoregressive model for CF, named User-Item co-Autoregressive model (CF-UIcA), which considers the autoregression in the domains of both users and items, so as to model both UUCs and IICs together. The introduced co-autoregression naturally provides a principled way to select the UUCs and IICs to be generalized that are helpful for prediction. This allows us to incorporate extra desired properties into the model for different tasks, which is not studied in previous work. We further develop a stochastic learning algorithm for CF-UIcA to make it applicable for large datasets. We demonstrate our method on two real benchmarks: MovieLens 1M and Netflix, achieving state-of-the-art results in both rating prediction and top-N recommendation tasks, which is rarely accomplished in previous work. In addition, the visualization demonstrates that CF-UIcA learns semantic patterns without extra supervision.

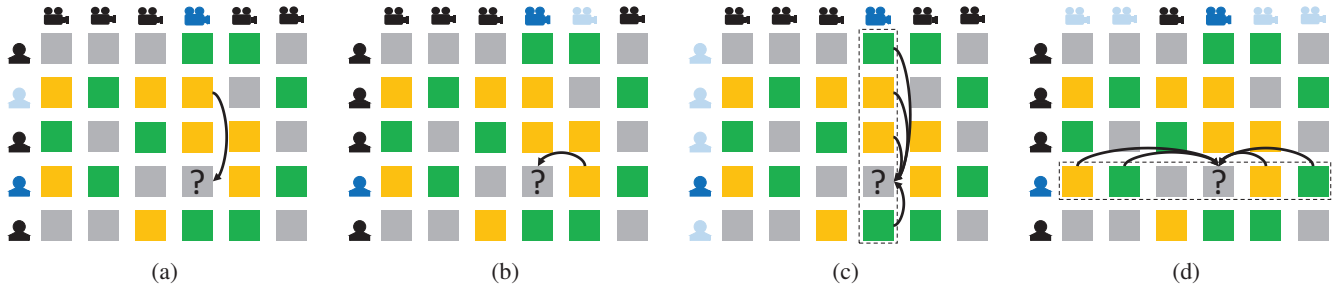


Figure 1: Illustration of predictions in a toy recommendation system with 5 users and 6 items (best viewed in color). Each square in green/yellow/gray corresponds to a positive/negative/unobserved behavior, respectively. The behavior (whose associating user and item are colored in deep blue) being predicted is marked with a question mark. (a) Predict with a single User-User Correlation: the behavior is predicted according to the behavior of another user (labeled as light blue); (b) Predict with a single Item-Item Correlation; (c) Predict with multiple User-User Correlations, and (d) Predict with multiple Item-Item Correlations.

2 Related Work

Collaborative filtering methods make predictions based on user behaviors, which could reveal certain patterns for generalization. These phenomena involve two types of information: *User-User Correlations* (UUCs) and *Item-Item Correlations* (IICs). As shown in Fig. 1a, UUC depicts that a user's behavior is usually related to the one of some other users on the same item, especially when they have similar habits/tastes. Similarly, IIC depicts that a user's behavior on an item is related to his/her behaviors on other items, especially when these items are similar in nature, as shown in Fig. 1b. Predictions are then possible to be made by integrating these correlations. Fig. 1c and Fig. 1d show all the UUCs and IICs of the unknown preference marked by the question mark. Intuitively, integrating multiple UUCs and IICs can potentially lead to a more precise prediction.

Existing CF methods either implicitly or explicitly exploit these correlations. Early methods model the correlations via some similarity functions on the raw preferences, such as k-NN collaborative filtering (kNN-CF) (Resnick et al. 1994; Sarwar et al. 2001). These methods make predictions with the top k UUCs or IICs explicitly. Matrix factorization (MF) methods characterize both users and items by vectors in a low-dimensional latent space. The predictions are modeled with the inner products of the latent vectors of the corresponding users and items. Representative works include SVD-based methods (Billsus and Pazzani 1998; Sarwar et al. 2000) and the probabilistic MF (PMF) (Salakhutdinov and Mnih 2007; 2008). Recent approaches improve MF by loosing the constrains of linearity and low-rank assumption. Bias MF (Koren, Bell, and Volinsky 2009) introduces bias terms associated with users and items. Lee et al. (2013) propose Local Low-Rank Matrix Approximation (LLORMA) by assuming the observed rating matrix is a weighted sum of low-rank matrices. NNMF (Dziugaite and Roy 2015) and NeuMF (He et al. 2017) replace the inner product operations in MF with neural networks. Since MF methods make predictions with the learned latent vectors of the users and the items, the UUCs and IICs are not modeled explicitly.

With the success in many tasks (Krizhevsky, Sutskever, and Hinton 2012; Graves, Mohamed, and Hinton 2013; Bahdanau, Cho, and Bengio 2014; Mnih et al. 2015; Silver et al. 2016), deep learning has been integrated into CF methods with great success. Salakhutdinov, Mnih, and Hinton (2007) propose RBM-CF, a CF methods based on Restricted Boltzmann Machines, which has shown its power in Netflix prize challenge (Bennett and Lanning 2007). Sedhain et al. (2015) propose AutoRec, a discriminative model based on autoencoders. A similar model known as CDAE is concurrently proposed by Wu et al. (2016). Recently, Zheng et al. (2016b) propose CF-NADE, a tractable model based on Neural Autoregressive Distribution Estimators (NADE) (Larochelle and Murray 2011), and achieve the state-of-the-art results on several CF benchmarks. These methods share two aspects: 1) different models are built for different users by sharing parameters; and 2) predictions are made for a user according to his/her behavior profile. Note that as the role of users and items are exchangeable, these methods usually have a user-based and an item-based variants. As a result, these methods make predictions with either the UUCs or the IICs explicitly.

Our CF-UIcA differs from existing CF methods in that it can capture both UUCs and IICs explicitly and simultaneously. Similar as in CF-NADE, we adopt neural autoregressive architectures to model the probabilities of the behaviors. The crucial difference is that CF-NADE models the rating vectors of each user, making the users independent from each other, while CF-UIcA models the behaviors across all users and items in order to consider UUCs and IICs jointly. Moreover, we analyze the significance of the co-autoregression in a novel perspective and demonstrate its effectiveness, which is another step beyond CF-NADE.

Hybrid recommendation (Burke 2002) is a class of methods focusing on combining different techniques at a high level, e.g., combining CF-based methods and content-based methods together. Different with the existing hybrid recommendation methods, our model focuses on utilizing both user-based and item-based information. Wang, De Vries, and Reinders (2006) share similar motivation with ours. However, their method is memory-based and unifies user-based and item-based models by similarity. While our method is model-based and combines user-based and item-based information by autoregressive neural networks.

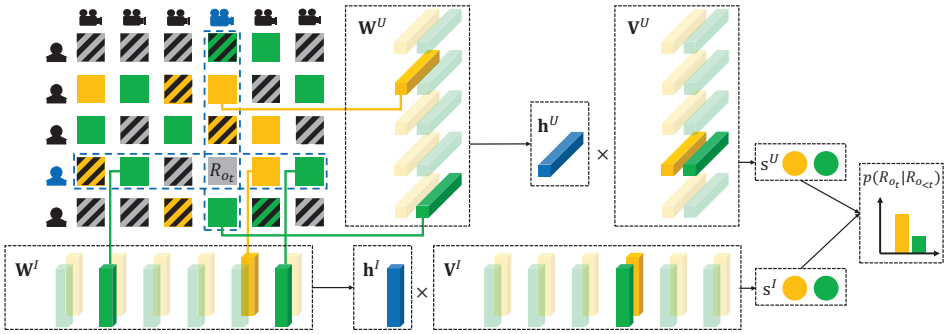


Figure 2: An illustration of the conditional model. The yellow, green and gray entries are interpreted same as in Fig. 1. Suppose R_{o_t} is the current behavior being modeled. The black striped lines mark the entries of $R_{o_{>t}}$. The blue dashed boxes line out the UUCs and IICs for R_{o_t}. The cuboids represent the columns of $\mathbf{W}^U, \mathbf{W}^I, \mathbf{V}^U, \mathbf{V}^I$, with the color corresponding to the behaviors. The hidden representations $\mathbf{h}^U$ and $\mathbf{h}^I$ are computed by summing over the corresponding columns (with the uncorresponding columns marked in lighter colors) of UUCs and IICs. The activations s^U and s^I are computed by multiplying the hidden representations and the corresponding columns of $\mathbf{V}^U$ and $\mathbf{V}^I$. We omit the bias terms for clarity.

3 Method

We now present CF-UIcA which models both UUCs and IICs with co-autoregressvie architectures.

Let N and M denote the number of users and items, respectively. We define the behavior matrix $\mathbf{R} \in \mathbb{R}^{N \times M}$ from user behaviors by assigning entries of $\mathbf{R}$ with different labels for different behaviors: For explicit feedback, e.g. K-star scale ratings, we define $R_{i,j} = k$ for the behavior "user i rates item j with k stars"; For implicit feedback, e.g. clicks, we define $R_{i,j} = 1$ for the behavior "user i clicks item j". And we define $R_{i,j} = 0$ for unobserved entries. Let $\mathcal{D} = \{R_{i_d,j_d}\}_{d=1}^{D}$ be all the observed behaviors, which form the training set. The goal of CF is then to predict an unknown behavior $R_{i^*,j^*} \notin \mathcal{D}$ based on the observed behaviors $\mathcal{D}$.

3.1 The Model

Autoregressive models (Frey 1998; Larochelle and Murray 2011; Lauly et al. 2017) offer a natural way to introduce interdependencies, which is desired for CF tasks, as analyzed in Sec. 2. We start with a very generic autoregressive assumption to model the probability of the behavior matrix:

$$p(\mathbf{R}) = \prod_{t=1}^{N \times M} p(R_{o_t}|R_{o_{<t}}), \tag{1}$$

where o is a permutation of all $\langle$user, item$\rangle$ pairs that serves as an ordering of all the entries in the behavior matrix $\mathbf{R}$, and $R_{o_{<t}}$ denotes the first $t-1$ entries of $\mathbf{R}$ indexed by o. For example, $o_t = (i', j')$ indicates that the behavior $R_{i',j'}$ is at the t-th position in o, i.e., $R_{o_t} = R_{i',j'}$. Let $o_t^i = i'$ and $o_t^j = j'$ denote the first and second dimension of o_t, which index the user and the item, respectively.

Basically, there are $(N \times M)!$ possible orderings of all the entries of $\mathbf{R}$. For now we assume that o is fixed. (We will discuss the orderings latter.) If we consider o as the ordering of the timestamps of the behaviors observed by the system, then the conditional in Eqn. (1) means that the model predicts behavior R_{o_t} at time t depends on all the observed behaviors before t.

The Conditional Model According to Sec. 2, both UUCs and IICs are informative for prediction. We therefore define a conditional model that exploits both UUCs and IICs:

$$p(R_{o_t}|R_{o_{<t}}) = p(R_{o_t}|R_{o_t}^{UUC}, R_{o_t}^{IIC}), \tag{2}$$

where we have defined $R_{o_t}^{UUC} = \{R_{o_{t'}} : t' < t, o_{t'}^j = o_t^j\}$ as the behaviors on item o_t^j in $R_{o_{<t}}$, which form all the UUCs of R_{o_t} (by time t). $R_{o_t}^{IIC}$ is defined symmetrically.

Inspired by NADE (Larochelle and Murray 2011) and CF-NADE (Zheng et al. 2016b), we model the conditional in Eqn. (2) with neural networks due to the rich expressive ability. Specifically, CF-UIcA models the UUCs and IICs of R_{o_t} with hidden representations respectively:

$$\mathbf{h}^U(R_{o_t}^{UUC}) = f\Big(\sum\nolimits_{t'<t:o_{t'}^j=o_t^j} \mathbf{W}^U_{:,o_{t'}^i,R_{o_{t'}}} + \mathbf{c}^U\Big), \tag{3}$$

$$\mathbf{h}^I(R_{o_t}^{IIC}) = f\Big(\sum\nolimits_{t'<t:o_{t'}^i=o_t^i} \mathbf{W}^I_{:,o_{t'}^j,R_{o_{t'}}} + \mathbf{c}^I\Big), \tag{4}$$

where $f(\cdot)$ is a nonlinear function, such as $\tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$, $\mathbf{W}^U \in \mathbb{R}^{H_U \times N \times K}$ and $\mathbf{W}^I \in \mathbb{R}^{H_I \times M \times K}$ are 3-order tensors, $\mathbf{c}^U \in \mathbb{R}^{H_U}$ and $\mathbf{c}^I \in \mathbb{R}^{H_I}$ are the bias terms. H_U and H_I are the dimensions of the hidden representations for UUCs and IICs, respectively. The column $\mathbf{W}^I_{:,j,k}$ denotes how much "behaving k on item j" contributes to the hidden representations of the IICs while the column $\mathbf{W}^U_{:,i,k}$ denotes the contribution of "user i behaves k" to the hidden representation of the UUCs.

CF-UIcA explains the hidden representations of the UUCs and the IICs by computing the activations:

$$s^U_{o_t^i,k}(\mathbf{h}^U(R_{o_t}^{UUC})) = \mathbf{V}^U_{:,o_t^i,k}{}^\top \mathbf{h}^U(R_{o_t}^{UUC}) + b^U_{o_t^i,k}, \tag{5}$$

$$s^I_{o_t^j,k}(\mathbf{h}^I(R_{o_t}^{IIC})) = \mathbf{V}^I_{:,o_t^j,k}{}^\top \mathbf{h}^I(R_{o_t}^{IIC}) + b^I_{o_t^j,k}, \tag{6}$$

where $\mathbf{V}^U \in \mathbb{R}^{H_U \times N \times K}$ and $\mathbf{V}^I \in \mathbb{R}^{H_I \times M \times K}$ are 3-order tensors, $\mathbf{b}^U \in \mathbb{R}^{N \times K}$ and $\mathbf{b}^I \in \mathbb{R}^{M \times K}$ are the bias terms. The column $\mathbf{V}^I_{:,j,k}$ is the coefficients that determine how the hidden representation of the IICs affects the activation $s^I_{j,k}$ for "behaving k on item j". Higher activation $s^I_{j,k}$ indicates

that the considered IICs suggest higher probability that the user will carry out a behavior k on item j. The activation $s^U_{i,k}$ is interpreted similarly.

Finally, to combine the activations of UUCs and IICs of R_{o_t} and produce a probability distribution, we define the final conditional model as a softmax function of the summation of the two activations:

$$p(R_{o_t} = k|R_{o_{<t}}) = \frac{\exp\left(s^U_{o^i_t,k} + s^I_{o^j_t,k}\right)}{\sum_{k'=1}^{K} \exp\left(s^U_{o^i_t,k'} + s^I_{o^j_t,k'}\right)}. \quad (7)$$

Fig. 2 illustrates the conditional model.

Orderings in Different Tasks We now discuss the effect of different orderings on the model and show what kinds of orderings are considered for two major CF tasks detailedly.

In fact, the ordering o decides the conditional model for each observed behavior $R_{i',j'}$. Specifically, according to our model (See Eqns. (2) to (4)), the contributions of UUCs and IICs to a given behavior $R_{i',j'}$, i.e. $R^{UUC}_{i',j'}$ and $R^{IIC}_{i',j'}$, depend on where the ordering o places $R_{i',j'}$ and what o places before $R_{i',j'}$. In general, different orderings result in different conditional models or dependency relations (see Fig. 3 for an illustration) and any possible conditional models can be induced by some specific orderings. Such a property leaves us freedom to control what kind of dependencies we would like the model to exploit in different tasks, as shown below.

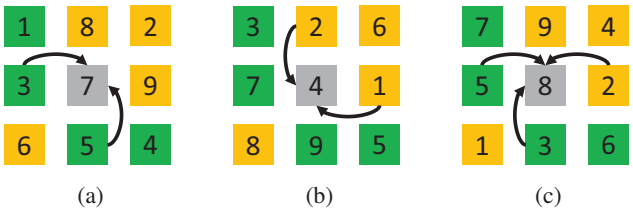
CF methods are usually evaluated on rating prediction tasks (Zheng et al. 2016b; Sedhain et al. 2015), or more generally, matrix completion tasks, by predicting randomly missing ratings/values. For matrix completion tasks, taking all UUCs and IICs into consideration leads the model to exploit the underlying correlations to a maximum extent. Therefore, we should consider all possible conditional models for each behavior, i.e., all orderings, in such tasks. The objective could then be defined as the expected (over all orderings) negative log-likelihood (NLL) of the training set:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \mathop{\mathbb{E}}_{o\in\mathfrak{S}_D} -\log p(\mathcal{D}|\boldsymbol{\theta}, o) \\ &= -\mathop{\mathbb{E}}_{o\in\mathfrak{S}_D} \sum\nolimits_{d=1}^{D} \log p(R_{o_d}|R_{o_{<d}}, \boldsymbol{\theta}, o), \end{aligned} \quad (8)$$

where $\boldsymbol{\theta}$ denotes all the model parameters and $\mathfrak{S}_D$ is the set of all the permutations of $\mathcal{D}$[1]. Note that taking the expectation over all orderings is equivalent to integrating them out. Thus the training procedure does not depend on any particular ordering and no manually chosen ordering is needed.

Recent works (Rendle et al. 2009; He et al. 2017) also evaluate CF on top-N recommendation tasks, aiming to suggest a short list of future preferences for each user, which is closer to the goal of real-world recommendation systems. In these tasks, not all IICs are useful. For example, people who have just watched *Harry Potter 1* (*HP1*) are very likely to be interested in *Harry Potter 2* (*HP2*), however those who have just watched *HP2* are less likely to have interest in *HP1*, as he/she may have known some spoiler about *HP1*. To this end, we should expect the model to capture the chronological IICs, which only include the dependencies from later behaviors to previous behaviors of each user, and all UUCs in the meanwhile. Then, an appropriate objective should be the expected NLL of the training set over all orderings that do not break the chronological order of each user's behaviors. Note that this can be implemented equivalently by re-defining $R^{IIC}_{i',j'} = \{R_{i',j''} : T(R_{i',j''}) < T(R_{i',j'})\}$, where $T(\cdot)$ is the timestamp when the system observes the behavior, and using Eqn. (8) as the objective[2]. Hence we still need not to choose any ordering manually.

The above examples show that extra desired properties can be incorporated into CF-UIcA for different tasks by considering different orderings in the objective, which indeed benefits from the co-autoregressive assumption.



(a) (b) (c)

Figure 3: An illustration of how the orderings decide the conditional models. Colors are interpreted same as in Fig. 2. (a) - (c) show 3 conditional models for the central (gray) behavior in an example behavior matrix under 3 different orderings. The numbers 1 - 9 indicate the indices of the corresponding behaviors in the orderings. The arrows indicate the dependencies involved in the conditional models.

3.2 Learning

The remaining challenge is to optimize the objective in Eqn. (8). A common strategy to deal with complicate integrations or large-scale datasets is to adopt stochastic optimization approaches, e.g. stochastic gradient descent (SGD) (Bottou 2010; Kingma and Ba 2014), which require an unbiased estimator of the objective or its gradient. SGD and its variants have been widely adopted in various areas due to its efficiency, including many CF methods (Sedhain et al. 2015; Zheng et al. 2016b). However, unlike in the most existing neural networks based methods (Wu et al. 2016; Zheng et al. 2016b), the users are not modeled independently in CF-UIcA, resulting the objective cannot be estimated stochastically by simply sub-sampling the users. To tackle this challenge, we derive an unbiased estimator of Eqn. (8) below, which completes the proposed method.

By exchanging the order of the expectation and the summation in Eqn. (8) and doing some simple math, we get:

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{(i',j')\in\mathcal{D}} \mathop{\mathbb{E}}_{d} \mathop{\mathbb{E}}_{o\in\mathfrak{S}_D|o_d=(i',j')} \log p(R_{i',j'}|R_{o_{<d}}, \boldsymbol{\theta}, o). \quad (9)$$

According to the definition of the conditional model from Eqns. (3) to (7), the log-probability of $R_{i',j'}$ in Eqn. (9) depends on at most $\mathbf{R}_{i',\neg j'} = \mathbf{R}_{i',:} \backslash \{R_{i',j'}\}$ (behaviors of user

[1] Given the training set $\mathcal{D}$, the first D elements of o will be automatically restricted to $\mathcal{D}$. As we only evaluate the likelihood of the training set $\mathcal{D}$, the number of equivalence orderings are $D!$.

[2] In this case $R^{IIC}_{i',j'}$ is deterministic and only $R^{UUC}_{i',j'}$ depends on the ordering o.

i' except $R_{i',j'}$) and $\mathbf{R}_{\neg i',j'} = \mathbf{R}_{:,j'} \backslash \{R_{i',j'}\}$ (behaviors on item j' except $R_{i',j'}$). Specifically, given $o_d = (i', j')$, the log-probability of $R_{i',j'}$ depends on exactly the set $R^{IIC}_{i',j'} = \mathbf{R}_{i',\neg j'} \cap R_{o_{<d}}$ and the set $R^{UUC}_{i',j'} = \mathbf{R}_{\neg i',j'} \cap R_{o_{<d}}$. As we treat the ordering o as a random variable uniformly distributed over $\mathfrak{S}_D$, $R^{IIC}_{i',j'}$ and $R^{UUC}_{i',j'}$ are also random. Moreover, since $R^{IIC}_{i',j'} \cap R^{UUC}_{i',j'} = \varnothing$, they are independent given their sizes $m = |R^{IIC}_{i',j'}|$ and $n = |R^{UUC}_{i',j'}|$, i.e., $R^{IIC}_{i',j'}|m \perp R^{UUC}_{i',j'}|n$. By expanding the second expectation in Eqn. (9) based on the above analysis, we have:

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{i'=1}^{N}\sum_{j'=1}^{M} \mathop{\mathbb{E}}_{d} \mathop{\mathbb{E}}_{m,n|d} \mathop{\mathbb{E}}_{R^{IIC}_{i',j'}|m} \mathop{\mathbb{E}}_{R^{UUC}_{i',j'}|n} \log p(R_{i',j'}|R^{UUC}_{i',j'}, R^{IIC}_{i',j'}, \boldsymbol{\theta}) \mathbb{I}_{[(i',j')\in\mathcal{D}]}, \tag{10}$$

where m, n, $R^{IIC}_{i',j'}|m$ and $R^{UUC}_{i',j'}|n$ are all random and are decided by the random ordering o. Note the summation over $\mathcal{D}$ is replaced by an equivalent representation using an indicator function $\mathbb{I}_{[(i',j')\in\mathcal{D}]}$. Given $R^{UUC}_{i',j'}$ and $R^{IIC}_{i',j'}$, the log-probability term and the indicator term can be computed easily. From now we omit these two terms for simplicity.

According to symmetry, it is easy to know that $R^{IIC}_{i',j'}|m$ and $R^{UUC}_{i',j'}|n$ are uniformly distributed over all subsets of size m of $\mathbf{R}_{\neg i',j'} \cap \mathcal{D}$ and subsets of size n of $\mathbf{R}_{i',\neg j'} \cap \mathcal{D}$, respectively. However, these distributions have different supports since the numbers of the observed behaviors for users (items) are different, which makes the sampling unparallelizable. Note that the process of drawing o from $\mathfrak{S}_D$ can be equivalently simulated by first randomly drawing σ from $\mathfrak{S}_{N\times M}$, which can be viewed as an ordering of all the entries of $\mathbf{R}$, and then dropping the unobserved entries $\mathbf{R}\backslash\mathcal{D}$. The resulted ordering on $\mathcal{D}$ is still uniformly distributed over $\mathfrak{S}_D$. Then Eqn. (10) can be written equivalently as:

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{i'=1}^{N}\sum_{j'=1}^{M} \mathop{\mathbb{E}}_{r} \mathop{\mathbb{E}}_{y,z|r} \mathop{\mathbb{E}}_{\mathcal{M}\subseteq[M]\backslash\{j'\}|y} \mathop{\mathbb{E}}_{\mathcal{N}\subseteq[N]\backslash\{i'\}|z}, \tag{11}$$

where r is the index of $R_{i',j'}$ in σ, y and z are the number of entries in $\mathbf{R}_{i',\neg j'} \cap R_{\sigma_{<r}}$ and $\mathbf{R}_{\neg i',j'} \cap R_{\sigma_{<r}}$, respectively. $\mathcal{M}$ is a subset of size y of $[M]\backslash\{j'\}$ and $\mathcal{N}$ is a subset of size z of $[N]\backslash\{i'\}$, where $[N]$ denotes $\{1, \cdots, N\}$. $R^{UUC}_{i',j'}$ and $R^{IIC}_{i',j'}$ are therefore $\mathbf{R}_{\mathcal{N},j'} \cap \mathcal{D}$, $\mathbf{R}_{i',\mathcal{M}} \cap \mathcal{D}$.

Finally, with some simple math we obtain:

$$\mathcal{L}(\boldsymbol{\theta}) = -NM \mathop{\mathbb{E}}_{r} \mathop{\mathbb{E}}_{y,z|r} \mathop{\mathbb{E}}_{\mathcal{M}\subseteq[M]|y} \mathop{\mathbb{E}}_{\mathcal{N}\subseteq[N]|z} \mathop{\mathbb{E}}_{i'\in[N]\backslash\mathcal{N}} \mathop{\mathbb{E}}_{j'\in[M]\backslash\mathcal{M}}. \tag{12}$$

In Eqn. (12), y and z can be computed after sampling r and σ. $\mathcal{M}$ and $\mathcal{N}$ are sampled by uniformly choosing y and z elements in $[M]$ and $[N]$ without replacement, respectively. The last two expectations can be estimated unbiasedly by sampling B_U elements from $[N]\backslash\mathcal{N}$ and B_I elements from $[M]\backslash\mathcal{M}$, respectively, where B_U and B_I can be viewed as the minibatch sizes of users and items. Finally, we get an unbiased estimation of the objective $\mathcal{L}(\boldsymbol{\theta})$, which can be then adopted in SGD algorithms.

Note that though the training objective involves expectations over multiple orderings (which help exploit the desired UUCs and IICs during training), the prediction procedure is simple and deterministic. For an unknown behavior R_{i^*,j^*}, the prediction is evaluated with $\hat{R}_{i^*,j^*} = \mathbb{E}_{p(R_{i^*,j^*}=k|\mathcal{D})}[k]$ with $R^{UUC}_{i^*,j^*} = \mathbf{R}_{\neg i^*,j^*} \cap \mathcal{D}$ and $R^{IIC}_{i^*,j^*} = \mathbf{R}_{i^*,\neg j^*} \cap \mathcal{D}$, where we have assumed $R_{i^*,j^*} = R_{o_{D+1}}$ and $R_{o_{<D+1}} = \mathcal{D}$.

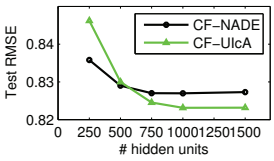


Figure 4: The performance on MovieLens 1M of CF-UIcA and CF-NADE w.r.t. the number of hidden units.

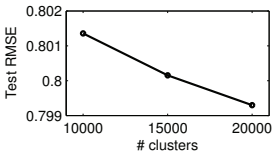


Figure 5: The performance on Netflix of CF-UIcA w.r.t. the number clusters of users.

4 Experiments

We now present a series of experimental results of the proposed CF-UIcA to demonstrate its effectiveness. We compare CF-UIcA with other popular CF methods on two major kinds of CF tasks: rating prediction and top-N recommendation. The experiments are conducted on two representative datasets: MovieLens 1M (Harper and Konstan 2016) and Netflix (Bennett and Lanning 2007). MovieLens 1M consists of $1,000,209$ ratings of $3,952$ movies (items) rated by $6,040$ users. Netflix consists of $100,480,507$ ratings of $17,770$ movies rated by $480,189$ users. The ratings in both datasets are 1-5 stars scale, i.e., $K = 5$. For all experiments, we use Adam (Kingma and Ba 2014) to optimize the objectives with an initial learning rate 0.001. During training, we anneal the learning rate by factor 0.25 until no significant improvement can be observed on validation set. Note that in Eqn. (12) the sizes of $[N]\backslash\mathcal{N}$ and $[M]\backslash\mathcal{M}$, i.e. $N - z$ and $M - y$, vary from 1 to $N-1$ and to $M-1$, respectively. As a consequence, the minibatch sizes of users/items should be set dynamically. Nevertheless, we choose fixed minibatch sizes of users/items B_U/B_I, which only take effect when $M - y > B_I$ or $N-z > B_U$. We adopt weight decay on model parameters to prevent the model from overfitting. Other hyper parameters and detailed experimental settings will be specified latter for each task. The codes and more detailed settings can be found at https://github.com/thu-ml/CF-UIcA.

4.1 Rating Prediction

We use the same experimental settings with LLORMA (Lee et al. 2013), AutoRec (Sedhain et al. 2015) and CF-NADE (Zheng et al. 2016b). We randomly select 90% of the ratings in each of the datasets as the training set, leaving the remaining 10% of the ratings as the test set. Among the

Table 1: Test RMSE on MovieLens 1M and Netflix. All the baseline results are taken from Zheng et al. (2016b).

Method	MovieLens 1M	Netflix
PMF	0.883	-
U-RBM	0.881	0.845
U-AutoRec	0.874	-
LLORMA-Global	0.865	0.874
I-RBM	0.854	-
BiasMF	0.845	0.844
U-CF-NADE-S (2 layers)	0.845	0.803
NNMF	0.843	-
LLORMA-Local	0.833	0.834
I-AutoRec	0.831	0.823
I-CF-NADE-S (2 layers)	0.829	-
CF-UIcA (H^U=H^I=500)	**0.823**	**0.799**

ratings in the training set, 5% are hold out for validation. We compare the predictive performance with other state-of-the-art methods in terms of the common used *Root Mean Squared Error* (RMSE) $= (\sum_{(i,j)\in\mathcal{D}_{\text{test}}}(\hat{R}_{i,j} - R_{i,j})^2/D_{\text{test}})^{1/2}$, where $\mathcal{D}_{\text{test}}$ is the test set of D_{test} unknown ratings, $R_{i,j}$ is the true rating and $\hat{R}_{i,j}$ is the prediction. The reported results are averaged over 10 random splits, with standard deviations less than 0.0002.

MovieLens 1M For experiments on MovieLens 1M, we set B_U/B_I to $1,000/1,000$ and the weight decay to 0.0001.

Since CF-UIcA has a connection with CF-NADE (Zheng et al. 2016b) as mentioned in Sec. 2, we first present a comparison between CF-UIcA and CF-NADE in Fig. 4. CF-NADE models each user with a latent representation, similar with our hidden representation of UUCs or IICs. For fairness, we compare the two methods with the same number of hidden units, where in our CF-UIcA the number of hidden units is H^U+H^I. Note that in CF-UIcA H^U is not necessarily equal to H^I, we nevertheless choose H^U=H^I for simplicity. We report the item-based CF-NADE results under best setting as described in (Zheng et al. 2016b).

From Fig. 4 we observe that for small number of hidden units, e.g. 250, our method gives a worse result than CF-NADE. This is attributed to that the hidden dimensions allocated to the hidden representation of UUCs and IICs are too small (H^U=H^I=125) to capture the underlying information. As the number of hidden units increases, we observe CF-UIcA outperforms CF-NADE since CF-UIcA can capture both UUCs and IICs while the item-based CF-NADE can only capture UUCs. One thing worth mentioning is that the total number of parameters in CF-UIcA is only around 83% of the number of parameters in CF-NADE for MovieLens 1M when the number of hidden units are same, which implies that CF-UIcA can capture more information than CF-NADE with fewer parameters.

Table 1 (middle column) compares the performance of CF-UIcA with state-of-the-art methods on MovieLens 1M. The hidden dimensions of CF-UIcA are $H^U = H^I = 500$. Our method achieves an RMSE of 0.823, outperforming all the existing strong baselines. Note that as RMSE scores have been highly optimized in previous work, our 0.006 RMSE improvement w.r.t. CF-NADE is quite significant, compared to the 0.002 by which CF-NADE improves over AutoRec and 0.002 by which AutoRec improves over LLORMA.

Table 2: Test HR@10 and NDCG@10 of various methods on MovieLens 1M. The results of baseline methods (except CDAE) are kindly provided by He et al. (2017).

Method	HR@10	NDCG@10
ItemPop	0.454	0.254
ItemKNN	0.623	0.359
BPR (Rendle et al. 2009)	0.690	0.419
eALS (He et al. 2016)	0.704	0.433
NeuMF	0.730	0.447
CDAE	0.726	0.448
CF-UIcA (Uniform)	0.692	0.406
CF-UIcA (Inverse)	0.616	0.353
CF-UIcA	**0.736**	**0.492**

Netflix The Netflix dataset is much bigger than MovieLens 1M, especially the number of users. We opt to cluster all the $480,189$ users into $10K$, $15K$ and $20K$ groups, respectively, and make the users in same clusters sharing their corresponding columns in $\mathbf{W}^U$ and $\mathbf{V}^U$. To cluster the users, we first run matrix factorization (Juan et al. 2016) with rank 100 on the training set. (Predicting the test set with the learned vectors by MF gives an RMSE of 0.865.) Then the clustering process is simply done by running a K-means clustering algorithm on the latent vectors of users learned by MF. For CF-UIcA, the weight decay is set to 5×10^{-6} as the dataset is sufficiently large. The minibacth sizes of users and items are set to $B_U = 4,000$ and $B_I = 1,000$. The hidden dimensions are $H^U = H^I = 500$.

Fig. 5 shows the performance of CF-UIcA with different number of clusters. We observe that the performance improves as the number of clusters increases, which can be attributed to that using more clusters empowers the model to capture more variety among users. Another observation is that the performance can be potentially improved by further increasing the number of clusters. We do not increase the number of clusters due to the limitation of GPU memory.

Table 1 (right column) summarizes our best result and other state-of-the-art results. Symbol "-" indicates that the authors didn't report the result, probably due to the lack of scalability[3]. Our method with $20,000$ clusters of users achieves a state-of-the-art RMSE of 0.799, which, together with the results on MovieLens 1M, proves that our CF-UIcA has the ability to predict users' ratings precisely.

4.2 Top-N Recommendation

In most real scenarios, the goal of recommendation systems is to suggest a top-N ranked list of items that are supposed to be appealing for each user. Moreover, implicit feedback (Zheng et al. 2016a) has attracted increasing interests because it is

[3]We confirmed with the authors of (Zheng et al. 2016b) that I-CF-NADE is not scalable to Netflix. For AutoRec, the authors reported that I-AutoRec is their best model.

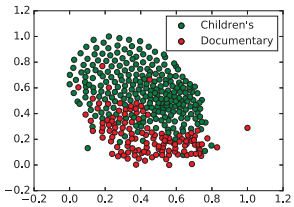


Figure 6: t-SNE embedding of the learned vectors for MovieLens 1M.

usually collected automatically and is thus much more easier to obtain. We follow the experimental settings of NeuMF (He et al. 2017) to test the recommendation quality of CF-UIcA with implicit feedback. We transform MovieLens 1M into implicit data by marking all ratings as 1 indicating that the user has rated the item. We adopt the *leave-one-out* (Rendle et al. 2009; He et al. 2017) evaluation: The latest rated item of each user is held out as the test set; The second latest rated item of each user is choosen as the validation set and the remaining data are used as the training set. At test time, we adopt the common strategy (Koren 2008; Elkahky, Song, and He 2015) that randomly samples 100 items that are not rated by the user, and ask the algorithm to rank the test item among the 100 sampled items. We evaluate the quality of the ranked list for the user by computing the *Hit Ratio* (HR) and the *Normalized Discounted Cumulative Gain* (NDCG) (He et al. 2015). Specifically,

$$\text{HR} = \frac{\text{\#hits}}{\text{\#users}}, \quad \text{NDCG} = \frac{1}{\text{\#users}} \sum_{i=1}^{\text{\#hits}} \frac{1}{\log_2(p_i+1)}, \tag{13}$$

where #hits is the number of users whose test item appears in the recommended list and p_i is the position of the test item in the list for the i-th hit. For both metrics, the ranked list is truncated at 10.

Since the model is always asked to make predictions of latest behaviors based on former behaviors, we train the model under the expectation over orderings that maintain the chronological order of each user's behaviors, as anylized in Sec. 3.1. An important difficulty of CF with implicit feedback is that only positive signals are observed. To handle the absence of negative signals, we follow the common strategy (Pan et al. 2008; He et al. 2017) that randomly samples negative instances from unobserved entries dynamically during the training procedure.

The minibatch sizes of users and items are set to 200. The hidden dimensions are $H^U = H^I = 256$ and the weight decay is 1×10^{-5}. the results are averaged over 5 runs with different random seeds, with standard deviations less than 0.0005. Table 2 compares the results in HR@10 and NDCG@10 with state-of-the-art methods for top-N recommendation with implicit feedback on MovieLens 1M. The baseline results are provided by He et al. (2017), except the result of CDAE (Wu et al. 2016), which is evaluated with our implementation. We can see that CF-UIcA achieves the best performance under both measures. Importantly, our method gives an NDCG@10 0.492, which outperforms the state-of-the-art method NeuMF by a large margin 0.045 (relative improvement 10.1%). To demonstrate the significance of the co-autoregression, we train another two CF-UIcA models under the expectation over: (Uniform Case) all possible orderings, which cover all UUCs and IICs; and (Inverse Case) all orderings that reverse the chronological order of each user's behaviors. The results are shown in Table 2. We can observe that the orderings significantly affect the performance. Compared to the result (NDCG@10 0.406) of Ignore case where all UUCs and IICs are captured, our best result brings a 0.09 improvment, demonstrating the effectiveness of the orderings and the power of the co-autoregression.

Table 3: Average running time for each minibatch of CF-UIcA on different datasets.

Dataset	B_U/B_I	H^U/H^I	Time
ML 1M	$1,000/1,000$	$500/500$	0.77s
Netflix	$4,000/1,000$	$500/500$	3.4s

Table 4: Average test time of different methods and tasks on MovieLens 1M.

Method	Task	Test Time
CF-NADE	Rating Prediction	0.68s
CF-UIcA	Rating Prediction	0.80s
CF-UIcA	Top-N Recommendation	0.73s

4.3 Visualization

In MovieLens 1M, each movie is marked with one or more genres. There are totally 18 different genres including *Action*, *Children's*, *Drama*, etc. We visualize the learned weight matrices $\mathbf{V}^I$ in Sec. 4.1. Specifically, once the model is learned, $\mathbf{V}^I_{:,j,:}$ can be viewed as $H^I \times K$ dimensional vectors associated with item j. We apply t-SNE (Maaten and Hinton 2008) to embed these vectors into a 2-dimensional plane. Fig. 6 shows the t-SNE embedding of two most exclusive genres: *Children's* and *Documentary*. We can observe the learned vectors are distributed semantically in the gross.

4.4 Running Time and Memory

We analyze the running time and memory cost of the proposed method. All the experiments are conducted on a single Nvidia TITAN X GPU with Theano (Theano Development Team 2016) codes. As explained in Sec. 4, the minibatch sizes of users and items are not deterministic during training and thus there is no standard way to train CF-UIcA epoch by epoch. We report the average training time for each minibatch in Table 3. As for testing time, we compare CF-UIcA with other state-of-the-art methods in Table 4.

The running memory cost of CF-UIcA is mainly for saving the 3-dimensional weight tensors. Specifically, the memory complexity of CF-UIcA is $O((NH^U + MH^I)K)$. In our experiments we always let $H^U = H^I = H$, resulting the memory cost is proportional to $(N + M)HK$.

5 Conclusion

We propose CF-UIcA, a neural co-autoregressive model for collaborative filtering, with a scalable stochastic learning algorithm. CF-UIcA performs autoregression in both users and

items domains, making it able to capture both correlations between users and items explicitly and simultaneously. Experiments show that our method achieves state-of-the-art results, and is able to learn semantic information by visualization, verifying that the autoregression provides a principled way to incorporate the correlations.

Acknowledgments

This work is supported by the National NSF of China (Nos. 61620106010, 61621136008, 61332007), the MIIT Grant of Int. Man. Comp. Stan (No. 2016ZXFB00001) and the NVIDIA NVAIL Program.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bennett, J., and Lanning, S. 2007. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, 35.

Billsus, D., and Pazzani, M. J. 1998. Learning collaborative information filters. In *ICML*, volume 98, 46–54.

Bottou, L. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*. Springer. 177–186.

Burke, R. 2002. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction* 12(4):331–370.

Dziugaite, G. K., and Roy, D. M. 2015. Neural network matrix factorization. *arXiv preprint arXiv:1511.06443*.

Elkahky, A. M.; Song, Y.; and He, X. 2015. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *WWW*, 278–288.

Frey, B. J. 1998. *Graphical models for machine learning and digital communication*.

Gopalan, P. K.; Charlin, L.; and Blei, D. 2014. Content-based recommendations with poisson factorization. In *NIPS*.

Graves, A.; Mohamed, A.-r.; and Hinton, G. 2013. Speech recognition with deep recurrent neural networks. In *ICASSP*.

Harper, F. M., and Konstan, J. A. 2016. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5(4):19.

He, X.; Chen, T.; Kan, M.-Y.; and Chen, X. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *CIKM*.

He, X.; Zhang, H.; Kan, M.-Y.; and Chua, T.-S. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*, 549–558.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural collaborative filtering. In *WWW*.

Juan, Y.-C.; Chin, W.-S.; Zhuang, Y.; Yuan, B.-W.; Yang, M.-Y.; and Lin, C.-J. 2016. Libmf: A matrix-factorization library for recommender systems. https://www.csie.ntu.edu.tw/~cjlin/libmf/.

Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer* (8):30–37.

Koren, Y. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *SIGKDD*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

Larochelle, H., and Murray, I. 2011. The neural autoregressive distribution estimator. In *AISTATS*.

Lauly, S.; Zheng, Y.; Allauzen, A.; and Larochelle, H. 2017. Document neural autoregressive distribution estimation. *JMLR* 18(113):1–24.

Lee, J.; Kim, S.; Lebanon, G.; and Singer, Y. 2013. Local low-rank matrix approximation. In *ICML*, 82–90.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *JMLR* 9:2579–2605.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Pan, R.; Zhou, Y.; Cao, B.; Liu, N. N.; Lukose, R.; Scholz, M.; and Yang, Q. 2008. One-class collaborative filtering. In *ICDM*.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, 452–461.

Resnick, P.; Iacovou, N.; Suchak, M.; Bergstrom, P.; and Riedl, J. 1994. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, 175–186. ACM.

Salakhutdinov, R., and Mnih, A. 2007. Probabilistic matrix factorization. In *NIPS*.

Salakhutdinov, R., and Mnih, A. 2008. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML*.

Salakhutdinov, R.; Mnih, A.; and Hinton, G. 2007. Restricted boltzmann machines for collaborative filtering. In *ICML*.

Sarwar, B.; Karypis, G.; Konstan, J.; and Riedl, J. 2000. Application of dimensionality reduction in recommender system – a case study. In *ACM WEBKDD WORKSHOP*.

Sarwar, B.; Karypis, G.; Konstan, J.; and Riedl, J. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*.

Schafer, J. B.; Frankowski, D.; Herlocker, J.; and Sen, S. 2007. Collaborative filtering recommender systems. In *The adaptive web*. Springer. 291–324.

Sedhain, S.; Menon, A. K.; Sanner, S.; and Xie, L. 2015. Autorec: Autoencoders meet collaborative filtering. In *WWW*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*.

Van den Oord, A.; Dieleman, S.; and Schrauwen, B. 2013. Deep content-based music recommendation. In *NIPS*.

Wang, J.; De Vries, A. P.; and Reinders, M. J. 2006. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *SIGIR*, 501–508.

Wu, Y.; DuBois, C.; Zheng, A. X.; and Ester, M. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *WSDM*, 153–162.

Zheng, Y.; Liu, C.; Tang, B.; and Zhou, H. 2016a. Neural autoregressive collaborative filtering for implicit feedback. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, 2–6. ACM.

Zheng, Y.; Tang, B.; Ding, W.; and Zhou, H. 2016b. A neural autoregressive approach to collaborative filtering. In *ICML*.