Learning to Recommend from Sparse Data via Generative User Feedback

Wenlin Wang
Department of Electrical and Computer Engineering, Duke University
wlwang616@gmail.com

Abstract

Traditional collaborative filtering (CF) based recommender systems tend to perform poorly when the user-item interactions/ratings are highly scarce. To address this, we propose a learning framework that improves collaborative filtering with a synthetic feedback loop (CF-SFL) to simulate the user feedback. The proposed framework consists of a "recommender" and a "virtual user". The "recommender" is formulated as a CF model, recommending items according to observed user preference. The "virtual user" estimates rewards from the recommended items and generates a *feedback* in addition to the observed user preference. The "recommender" connected with the "virtual user" constructs a closed loop, that recommends users with items and imitates the *unobserved* feedback of the users to the recommended items. The synthetic feedback is used to augment the observed user preference and improve recommendation results. Theoretically, such model design can be interpreted as inverse reinforcement learning, which can be learned effectively via rollout (simulation). Experimental results show that the proposed framework is able to enrich the learning of user preference and boost the performance of existing collaborative filtering methods on multiple datasets.

Introduction

Recommender systems are important modules for abundant online applications, helping users explore items of potential interest. As one of the most effective approaches, collaborative filtering (Sarwar et al. 2001; Koren and Bell 2015; He et al. 2017) and its deep neural networks based variants (He et al. 2017; Wu et al. 2016; Liang et al. 2018; Li and She 2017; Yang et al. 2017; Wang et al. 2018) have been widely studied. These methods leverage patterns across similar users and items, predicting user preferences and have demonstrated encouraging results in recommendation tasks (Bennett, Lanning et al. 2007; Hu, Koren, and Volinsky 2008; Schedl 2016). Among these works, beside "user-item" pair data (e.g., ratings or interaction/purchase history), side information, *e.g.*, user reviews and scores on items, has also been leveraged (Menon et al. 2011; Fang and Si 2011). Such side information is a kind of user feedback to the recommended items, which is often useful for improving the recommendation systems.

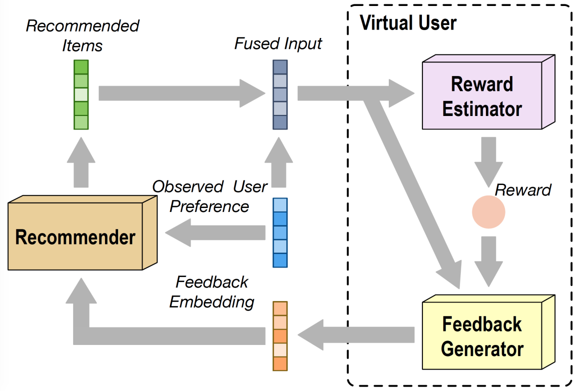

Figure 1: Illustration of our proposed CF-SFL framework for collaborative filtering.

Unfortunately, the user-item pairs and user feedback are extremely sparse as compared to the search space of items. Moreover, when the recommendation systems are trained on static observations, the user feedback is unavailable until it is deployed in real-world applications; in both training and validation phases, the target systems have no access to any feedback because no one has observed the recommended items. Therefore, the recommendation systems may suffer from overfitting, and their performance may degrade accordingly, especially in the initial phase of deployment. Although real-world recommendation systems are usually updated in an online manner with the help of increasing observed user preference (Rendle and Schmidt-Thieme 2008; Agarwal, Chen, and Elango 2010; He et al. 2016), introducing a feedback learning mechanism *during* their training phases can potentially improve the system that is eventually deployed. However, this aspect has largely been neglected by the existing recommender system frameworks.

Motivated by these observations, we propose a novel framework that achieves collaborative filtering with a synthetic feedback loop (CF-SFL). As shown in Figure 1, the proposed framework consists of a "recommender" and a "virtual user." The recommender is a collaborative filtering (CF) model, which predicts items based on observed user preference. The observed user preference vector reflects intrinsic preferences of the user, while the recommended items vector represents the model's estimated preferences of the user on the items. Taking a combination of the observed user preference and the recommended items (the estimated user preference) as inputs, which we refer to as *input* (cf., Figure 1), the virtual user, which is the key aspect of our model, imitates a real-world setting and provides a *synthesized* user feedback (in form of an embedding) to the CF module. In particular, the virtual user contains a *reward estimator* and a *feedback generator*. The *reward estimator* estimates rewards based on the fused inputs (the combined representation of the user observation and its recommended items), learned with a generative adversarial regularizer. The *feedback generator*, conditioned on the estimated rewards as well as the fused inputs, generates the feedback embedding to augment the original user preference vector. Such a framework constructs a closed loop between the target CF model and the virtual user, synthesizing user feedback as side information to improve recommendation results.

The proposed CF-SFL framework can be interpreted as an inverse reinforcement learning (IRL) set-up, in which the recommender learns to recommend the user items (policy) with the estimated guidance (feedback) from the proposed virtual user. The proposed feedback loop can be understood as an effective rollout procedure for recommendation, jointly updating the recommender (policy) and the virtual user (consisting of the reward estimator and the feedback generator). Essentially, even if side information ($i.e.$, real-world user feedback) is *unavailable*, our model is still applicable to *synthesize* feedback during both training as well as inference phases. The proposed framework is general and the recommender module can use any of the various existing CF methods, making our framework significantly modular. A comprehensive set of experimental results show that the performance of existing CF models can be remarkably improved within the proposed framework.

Proposed Framework

In this section, we first describe the problem set-up and provide a detailed description of each module that is part of the proposed framework.

Problem Statement

Suppose we have N users with M items in total. We denote the observed user-item matrix as $\boldsymbol{X} = [\boldsymbol{x}_i] \in \{0,1\}^{N\times M}$, where each vector $\boldsymbol{x}_i = [x_{ij}] \in \mathcal{R}^M$, $i = 1, ..., N$, represents the *observed* user preferences for the i-th user. Here $x_{ij} = 1$ indicates the the j-th item is bought or clicked by the i-th user; otherwise the j-th item is either irrelevant to the i-th user or we do not have knowledge about their relationship. A recommendation system outputs the estimated user preferences, denoted as $\boldsymbol{a}_i = [a_{ij}] \in \mathcal{R}^M$, whose element a_{ij} indicates the estimated preference of the i-th user to the j-th item. The system can then recommend each user the top few items (e.g., given a specified consumption budget) based on the value of the estimated preferences a_{ij}'s.

In practice, for each user, the vector $\boldsymbol{x}_i$ just contains partial information about the user's preferences on items and an ideal recommendation system works dynamically with a closed loop — users often generate feedback on the recommended items while the system considers these feedback to revise recommended items in the future. Formally, this *feedback-driven* recommendation process can be written formally as

$$\boldsymbol{a}_i^t = \pi(\boldsymbol{x}_i, \boldsymbol{v}_i^t), \;\; \boldsymbol{v}_i^{t+1} = f(\boldsymbol{x}_i, \boldsymbol{a}_i^t), \;\; \text{for } i = 1, ..., N, \quad (1)$$

where $\pi(\cdot)$ represents the target recommender while $f(\cdot)$ represents the coupled feedback mechanism of the system. Here, $\boldsymbol{v}_i \in \mathcal{R}^d$ denotes the embedding of the aggregated user feedback on the previously recommended items. At time-step t, the recommender predicts preferred items according to the observed user preference $\boldsymbol{x}_i$ and previous feedback $\boldsymbol{v}_i^t$, subsequently, the user generates an updated feedback $\boldsymbol{v}_i^{t+1}$ to the recommender. Note that Eq. (1) is different from existing sequential recommendation models (Mishra, Kumar, and Bhasker 2015; Wang et al. 2016) because these methods do not have the feedback loop, and they just update the recommender module π according to observed sequential observations, *i.e.*, $\boldsymbol{x}_i^t$ for different time-steps t's.[1]

Unfortunately, the feedback information is often unavailable during training and inference. Accordingly, most existing collaborative filtering-based recommendation methods do not have a feedback loop in the system, and learn the recommendation system purely from the statically observed user-item matrix $\boldsymbol{X}$ (Liang et al. 2018; Li and She 2017). Although, in some settings, side information like user reviews is associated with the observation matrix, the methods using such information often treat it as a source of static knowledge rather than a dynamic feedback. They mainly focus on fitting the ground-truth recommended items with the recommender $\pi(\cdot)$ given fixed $\boldsymbol{x}_i$'s and fixed $\boldsymbol{v}_i$'s, while ignoring the whole recommendation-feedback loop in Eq. (1). Without a feedback mechanism, $f(\cdot)$, $\pi(\cdot)$ may overfit the (possibly scarce) user observations and the static side information, especially in dynamic settings.

To overcome the aforementioned problems, we propose a collaborative filtering framework with a synthetic feedback loop (CF-SFL). As shown in Figure 1, besides the traditional recommendation module, the proposed framework further introduces a *virtual user*, which imitates the recommendation-feedback loop, even if the real user feedback is unavailable.

The Recommender

In our framework, the recommender implements the function $\pi(\cdot)$ in Eq. (1), which takes the observed user preference $\boldsymbol{x}_i$ and the user's previous feedback embedding $\boldsymbol{v}_i^t$ as inputs and recommends items accordingly. In principle, the recommender $\pi(\cdot)$ can be defined with high flexibility, which can be based on any existing shallow/deep collaborative filtering method that predicts items from user representations, such

[1] When the static observation $\boldsymbol{x}_i$ in Eq. (1) is replaced with sequential observation $\boldsymbol{x}_i^t$, Eq. (1) is naturally extended to a sequential recommendation system with a feedback loop. In this work, we focus on the case with static observations and train a recommender system accordingly.

as WMF (Hu, Koren, and Volinsky 2008), CDAE (Wu et al. 2016), VAE (Liang et al. 2018), etc.

Our goal is to mimic and integrate dynamic user feedback and therefore, in this work, we develop an inverse reinforcement learning (IRL) based framework, adapted for CF. To the best of our knowledge, none of the existing CF approaches incorporate such a dynamic feedback loop, and our work is the first such attempt in this direction.

In particular, the recommendation-feedback loop generates a sequence of interactions between each user and the recommender, $i.e.$, $(\boldsymbol{s}_i^t, \boldsymbol{a}_i^t)_{t=1}^T$ for $i = 1, ..., N$. Here, $\boldsymbol{s}_i^t = [\boldsymbol{x}_i; \boldsymbol{v}_i^t]$ is the *representation* of user i at time t, which is a sample in the state space $\mathcal{S}$ describing user preferences; $\boldsymbol{a}_i^t$ is a vector of the estimated user preferences for user i, which is a sample in the action space $\mathcal{A}$ of the recommender. Accordingly, we can model the recommendation-feedback loop as a Markov Decision Process (MDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R \rangle$, where $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ is the transition probability of user preferences and $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function used to evaluate recommended items. We further assume that the recommender $\pi(\cdot)$ works as a policy parametrized by $\boldsymbol{\theta}$, $i.e.$, $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})$, which corresponds to the distribution of the estimated item preferences $\boldsymbol{a}$, conditioned on the user representation $\boldsymbol{s}$. The target recommender should be an optimal policy that maximizes the expected reward: $J(\pi_\theta) = \sum_{t=1}^T \mathbb{E}_{\pi_{\boldsymbol{\theta}}} [R_{\boldsymbol{\phi}}(\boldsymbol{s}^t, \boldsymbol{a}^t)]$, where $R_{\boldsymbol{\phi}}(\boldsymbol{s}^t, \boldsymbol{a}^t)$ means the reward for the state-action pair $(\boldsymbol{s}^t, \boldsymbol{a}^t)$. For the i-th user, given $\boldsymbol{s}_i^t$, the recommender selects potentially-preferred items by finding the optimal item-preference vector as follows

$$\boldsymbol{a}_i^t = \arg\ \max\nolimits_{\boldsymbol{a} \in \mathcal{A}} \pi_\theta(\boldsymbol{a}|\boldsymbol{s}_i^t). \tag{2}$$

and then recommending the top few items with largest values in the vector $\boldsymbol{a}_i^t$. Note that, different from traditional reinforcement learning tasks, in which both $\mathcal{S}$ and $\mathcal{A}$ are available while P and R are with limited accessibility, our recommender receives only *partial* information of the state — it does not observe users' feedback embedding $\boldsymbol{v}_i$. In other words, to optimize the recommender, we need to build a reward model and a feedback generator *jointly*, which motivates us to introduce a virtual user into the framework.

The Virtual User

The virtual user module aims to implement the feedback function $f(\cdot)$ in Eq. (1), which not only models the reward of the items provided by the recommender but also generates feedback $\boldsymbol{v}_i^t$ to complete the representations of the state $\boldsymbol{s}_i^t = [\boldsymbol{x}_i; \boldsymbol{v}_i^t]$. Accordingly, the virtual user contains the following two modules:

Reward Estimator: it parametrizes the function of reward, which takes the current estimation $\boldsymbol{a}_i^t$ and user preference $\boldsymbol{s}_i^t$ as input and evaluate their compatibility. In this work, we assume that the reward estimator is parametrized by parameters $\boldsymbol{\phi}$, and is defined as

$$R_{\boldsymbol{\phi}}(\boldsymbol{s}_i^t, \boldsymbol{a}_i^t) = \text{sigmoid}(g(h(\boldsymbol{x}_i, \boldsymbol{a}_i^t))). \tag{3}$$

For the reward estimator, we use the static part of the state $\boldsymbol{s}_i^t$, $i.e.$, the observed user preference $\boldsymbol{x}_i$ as input. In Eq. (3), $h(\cdot, \cdot)$ denotes the fusion function which merges $\boldsymbol{x}_i$ and $\boldsymbol{a}_i^t$ into a real value vector, and $g(\cdot)$ is the single value regression function that translates the fused input into a single reward value. The sigmoid function squashes the predicted reward value between 0 and 1.

Feedback Generator: it connects the reward estimator with the recommender module via generating a feedback embedding, $i.e.$,

$$\boldsymbol{v}_{t+1}^i = F_{\boldsymbol{\psi}}(h(\boldsymbol{x}_i, \boldsymbol{a}_i^t), R_\phi(\boldsymbol{s}_i^t, \boldsymbol{a}_i^t)), \tag{4}$$

where $\boldsymbol{\psi}$ represents the parameters of the generator. Specifically, the parametric function $F_{\boldsymbol{\psi}}(\cdot, \cdot)$ considers the fused input and the estimated reward and returns a feedback embedding $\boldsymbol{v}_t^i \in R^d$ to the recommender. In this work, we use a multilayer perceptron to model the function $F_{\boldsymbol{\psi}}$. In Eq. (4), $R_\phi(\boldsymbol{s}_i^t, \boldsymbol{a}_t^i)$ is as the scalar reward (as defined in Eq. (3)) denoting the compatibility between the recommended items and user preferences, and $h(\boldsymbol{x}_i, \boldsymbol{a}_t^i)$, which is a vector rather than a scalar like reward, further enriches the information of the reward to generate feedback embeddings. Consequently, the recommender receives the informative feedback as a complementary component of the static observation $\boldsymbol{x}_i$ to make an improved recommendation via Eq. (2).

The Learning Algorithm

Learning Task

Based on the proposed framework, we need to jointly learn the policy corresponding to the recommender $\pi_{\boldsymbol{\theta}}$, the reward estimator $R_{\boldsymbol{\phi}}$, and the feedback generator $F_{\boldsymbol{\psi}}$. Assuming we have a set of user observations $\mathcal{D} = \{\boldsymbol{x}_i\}$, where $\boldsymbol{x}_i \in \mathbb{R}^M$ is the vector of observed user preferences for user i. We formulate the learning task as the following min-max optimization problem

$$\min\nolimits_{\pi_{\boldsymbol{\theta}}, F_{\boldsymbol{\psi}}}\ \max\nolimits_{R_{\boldsymbol{\phi}}} \mathcal{L}(\pi_{\boldsymbol{\theta}}, R_{\boldsymbol{\phi}}, F_{\boldsymbol{\psi}}), \tag{5}$$

where

$$\mathcal{L}(\pi_{\boldsymbol{\theta}}, R_{\boldsymbol{\phi}}, F_{\boldsymbol{\psi}}) = \underbrace{\sum\nolimits_i \mathcal{L}_{\text{rec}}(\boldsymbol{a}_i, \boldsymbol{x}_i; \pi_{\boldsymbol{\theta}}, F_{\boldsymbol{\psi}})}_{\text{Reconstruction loss}}$$
$$- \underbrace{\mathbb{E}_{\boldsymbol{a}\sim\pi_{\boldsymbol{\theta}}}[\log(R_{\boldsymbol{\phi}}(\boldsymbol{s}, \boldsymbol{a}))] - \mathbb{E}_{\boldsymbol{a}\sim\mathcal{D}}[1 - \log(R_{\boldsymbol{\phi}}(\boldsymbol{s}, \boldsymbol{a}))]}_{\text{Collaboration with adversarial regularizer}}$$

In particular, the first term $\mathcal{L}_{\text{rec}}$ in Eq. (6) can be any reconstruction loss based on user preferences $\mathcal{D}$, $e.g.$, the evidence lower bound (ELBO) proposed in VAEs (Liang et al. 2018) (and used in our work). This term ensures the recommender to fit the observed user preference. The second term considers the following types of interactions among the various modules:

- The *collaboration* between the recommender policy $\pi_{\boldsymbol{\theta}}$ and the feedback generator $F_{\boldsymbol{\psi}}$ towards a better predictive recommender.
- The *adversarial game* between the recommender policy $\pi_{\boldsymbol{\theta}}$ and the reward estimator $R_{\boldsymbol{\phi}}$.

Accordingly, given the current reward model, we update the recommender policy $\pi_{\boldsymbol{\theta}}$ and the feedback generator $F_{\boldsymbol{\psi}}$ to

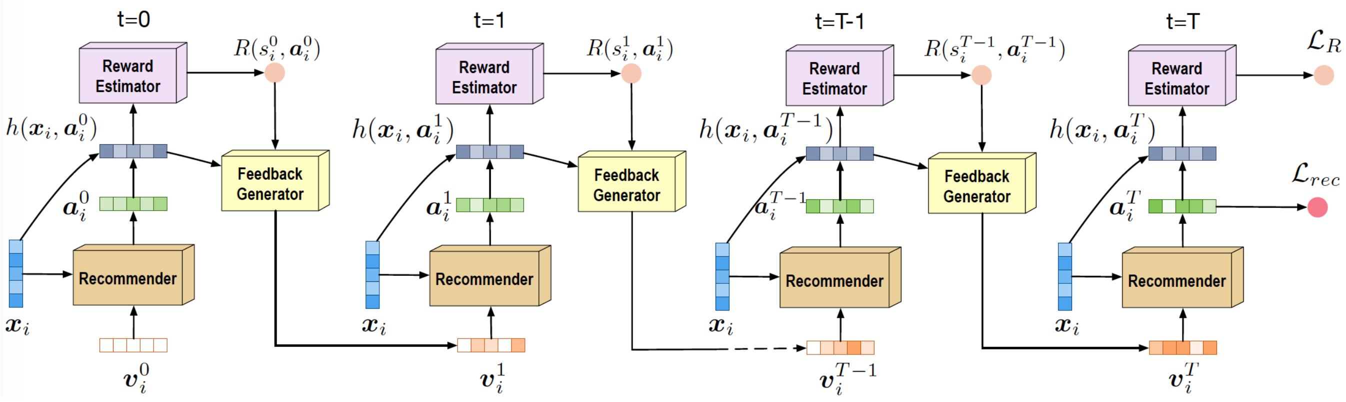

Figure 2: Unrolling the recurrent CF-SFL framework into an iterative learning process with T time steps.

maximize the expected reward derived from the generated user representation $\boldsymbol{s}$ and the estimated user preference $\boldsymbol{a}$. Likewise, given the recommended policy and the feedback generator, we improve the reward estimator $R_{\boldsymbol{\phi}}$ by sharpening its criterion — the updated reward estimator maximizes the expected reward derived from the generated user representation and the observed user preference, while minimizing the expected reward based on the generated user representation and the estimated user preference.

Therefore, we solve Eq. (5) via alternating optimization, updating of $\pi_{\boldsymbol{\theta}}$ and $F_{\boldsymbol{\psi}}$ by minimizing

$$\begin{aligned}\mathcal{L}_C(\pi_{\boldsymbol{\theta}}, F_{\boldsymbol{\psi}}) = &\sum\nolimits_i \mathcal{L}_{\text{rec}}(\boldsymbol{a}_i, \boldsymbol{x}_i; \pi_{\boldsymbol{\theta}}, F_{\boldsymbol{\psi}}) \\ &- \mathbb{E}_{\boldsymbol{a}\sim\pi_\theta}[\log(R_{\boldsymbol{\phi}}(\boldsymbol{s}, \boldsymbol{a}))].\end{aligned} \tag{6}$$

We update $R_{\boldsymbol{\phi}}$ is achieved by maximizing

$$\begin{aligned}\mathcal{L}_A(R_{\boldsymbol{\phi}}) = &- \mathbb{E}_{\boldsymbol{a}\sim\pi_\theta}[\log(R_{\boldsymbol{\phi}}(\boldsymbol{s}, \boldsymbol{a}))] \\ &- \mathbb{E}_{\boldsymbol{a}\sim\mathcal{D}}[1 - \log(R_{\boldsymbol{\phi}}(\boldsymbol{s}, \boldsymbol{a}))].\end{aligned} \tag{7}$$

Both these update steps can be solved efficiently via stochastic gradient descent.

Unrolling for Learning and Inference

Because the proposed framework contains a closed loop among learnable modules, during training we unroll the loop and let the recommender interact with the virtual user in T steps. Specifically, at the initial stage, the recommender takes the observed user preference vector $\boldsymbol{x}_i$ and an all-zero initial feedback embedding vector $\boldsymbol{v}_i^0$, to make recommendations. At each step t, the recommender outputs the estimated user preferences $\boldsymbol{a}_i^t$ given $\boldsymbol{x}_i$ and $\boldsymbol{v}_i^t$ to the virtual user, and receives the feedback embedding $\boldsymbol{v}_i^{t+1}$. The loss is defined according to the output of the last step, *i.e.*, $\boldsymbol{a}^T$ and $\boldsymbol{v}^T$, and the modules are updated accordingly. After the model is learned, in the testing phase we need to infer the recommended item in the same manner, unrolling the feedback loop and deriving $\boldsymbol{a}^T$ as the final estimated user preferences. The details of unrolling process are illustrated in Figure 2

CF-SFL as Inverse Reinforcement Learning

Our CF-SFL framework automatically discovers informative user feedback as side information and gradually improve the training for the recommender. Theoretically, it is closely related to Inverse Reinforcement Learning (IRL). Specifically, we jointly learn the reward estimator R_ϕ and the policy (recommender) $\pi_{\boldsymbol{\theta}}$ from the *expert trajectories* $\mathcal{D}$ (*i.e.,* the observed labeled data), which typically consists of state-action pairs generated from some expert policy π_E with the corresponding environment dynamics ρ_E. The IRL is to recover the optimal reward function R^* as well as the optimal recommender π^*. Formally, the IRL is defined as:

$$\begin{aligned}\{R^*, \pi^*\} \triangleq \text{IRL}(\pi_E) = \arg\max_{\boldsymbol{\phi}} &\sum_{\boldsymbol{s},\boldsymbol{a}} \rho_E(\boldsymbol{s}, \boldsymbol{a}) R_{\boldsymbol{\phi}}(\boldsymbol{s}, \boldsymbol{a}) \\ - [\max_{\boldsymbol{\theta}} H(\pi) + &\sum\nolimits_{\boldsymbol{s},\boldsymbol{a}} \rho(\boldsymbol{s}, \boldsymbol{a}) R_{\boldsymbol{\phi}}(\boldsymbol{s}, \boldsymbol{a})],\end{aligned} \tag{8}$$

which can be rewritten as

$$\max_{\boldsymbol{\phi}} \min_{\boldsymbol{\theta}} \underbrace{\sum\nolimits_{\boldsymbol{s},\boldsymbol{a}} (\rho_E(\boldsymbol{s}, \boldsymbol{a}) - \rho(\boldsymbol{s}, \boldsymbol{a})) R_{\boldsymbol{\phi}}(\boldsymbol{s}, \boldsymbol{a}) - H(\pi)}_{\mathcal{L}(\pi, R)}$$

Intuitively, the objective enforces the expert policy π_E to induce higher rewards (the $\max$ part), than all other policies. This objective is sub-optimal if the expert trajectories are noisy, *i.e.*, the expert is not perfect and its trajectories are not optimal. This will make the learned policy always perform worse than the expert one. Besides, the illed-defined IRL objective often induces multiple solutions due to the flexible solution space, *i.e.*, one can assign an arbitrary reward to trajectories not from expert, as long as these trajectories yield lower rewards than the expert trajectories. To alleviate these issues, some constraints can be incorporated into the objective functions, *e.g.*, a convex reward functional, $\psi : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \mathbb{R}$, which usually works as a regularizer.

$$\{R^*, \pi^*\} = \arg\max_{\boldsymbol{\phi}} \min_{\boldsymbol{\theta}} \mathcal{L}(\pi_{\boldsymbol{\theta}}, R_{\boldsymbol{\phi}}) - \psi(R_{\boldsymbol{\phi}}). \tag{9}$$

To imitate the expert policy and provide better generalization, we adopt the adversarial regularizer (Ho and Ermon 2016), which defines ψ with the following form:

$$\psi(R_{\boldsymbol{\phi}}) \triangleq \begin{cases} \mathbb{E}_{\pi_E}[q(R_{\boldsymbol{\phi}}(\boldsymbol{s}, \boldsymbol{a}))] & \text{if } R_{\boldsymbol{\phi}}(\boldsymbol{s}, \boldsymbol{a}) \geq 0 \\ +\infty & \text{otherwise} \end{cases},$$

	ML-20M	**Netflix**	**MSD**
# of users	136,677	463,435	571,355
# of items	20,108	17,769	41,140
# of interactions	10.0M	56.9M	33.6M
# of held-out-users	10.0K	40.0K	50.0K
% of sparsity	0.36%	0.69%	0.14%

Table 1: Dataset statistics

where $q(x) = x - \log(1 - e^{-x})$. This regularizer places low penalty on reward functions R that assign an amount of positive value to expert state-action pairs; however, if R assigns low value (close to zero, which is the lower bound) to the expert, then the regularizer will heavily penalize $R_{\boldsymbol{\phi}}$. With the adversarial regularizer, we obtain a new imitation learning algorithm for the recommender:

$$\min_\theta \psi^*(\rho_\pi - \rho_{\pi_E}) - \lambda H(\pi_\theta) \tag{10}$$

Intuitively, we want to find a saddle point (R_ϕ, π_θ) of the expression:

$$\mathbb{E}_{\pi_\theta}[\log(R_{\boldsymbol{\phi}}(\boldsymbol{s}, \boldsymbol{a}))] + \mathbb{E}_{\pi_E}[1 - \log(R_{\boldsymbol{\phi}}(\boldsymbol{s}, \boldsymbol{a}))] - \lambda H(\pi_\theta),$$

where $R_{\boldsymbol{\phi}}(\boldsymbol{s}, \boldsymbol{a}) \in (0, 1)$. Note that Eq. (9) is derived from the objective of traditional IRL. However, distinct from the traditional approach, we propose a feedback generator to provide feedback to the recommender. In terms of the reward estimator, it tends to assign lower rewards to the predicted results by the recommender π_θ and higher rewards for the expert policy π_E, which aims to discriminate π_θ from π_E, similar to Eq. (7):

$$\mathcal{L}_R = \mathbb{E}_{\pi_\theta}[\log(R_{\boldsymbol{\phi}}(\boldsymbol{s}, \boldsymbol{a}))] + \mathbb{E}_{\pi_E}[1 - \log(R_{\boldsymbol{\phi}}(\boldsymbol{s}, \boldsymbol{a}))]. \tag{11}$$

Similar to standard IRL, we update the generator to maximize the expected reward with respect to $\log R_{\boldsymbol{\phi}}(\boldsymbol{s}, \boldsymbol{a})$, moving towards expert-like regions of user-item space. In practice, we incorporate feedback embedding to update the user preferences, and the objective of the recommender is:

$$\mathcal{L}_F = \mathbb{E}_{\pi_\theta}[-\log(R([\boldsymbol{x}_i, \boldsymbol{v}_i^t], \boldsymbol{a}))] - \lambda H(\pi_\theta) \tag{12}$$

where $\boldsymbol{v}_i^t = F_\psi(h(\boldsymbol{x}_i, \boldsymbol{a}_i^t), R_{\boldsymbol{\phi}}(\boldsymbol{s}_i^t, \boldsymbol{a}_i^t))$. It is obvious that $\mathcal{L}_F$ recovers the second term in Eq. (6).

Related Work

Collaborative Filtering (CF). Existing CF approaches primary operate in one of the following two settings: CF with implicit feedback (Bayer et al. 2017; Hu, Koren, and Volinsky 2008) and CF with explicit feedback (Koren 2008; Liu et al. 2010). In implicit CF, user-item interactions are binary in nature (*i.e.*, 1 if clicked and 0 otherwise) as opposed to explicit CF where they represent item ratings (e.g., 1-5 stars). The implicit CF setting is more common/natural in many applications and has been widely studied, examples including factorization of user-item interactions (He et al. 2016; Koren 2008; Liu et al. 2016; Rendle 2010; Rennie and Srebro 2005) and ranking based approach (Rendle et al. 2012). Our CF-SFL framework is also designed for the implicit CF.

Recommender	**Reward Est.**	**Feedback Gen.**
Input $\mathcal{R}^M$	Input $\mathcal{R}^{64}$	Input $\mathcal{R}^{65}$
$M \times 600$, tanh	64×128, ReLU	64×128, ReLU
600×200 (x2)	128×128, ReLU	
Sample $\mathcal{R}^{200}$		128×128, ReLU
200×600, tanh	128×128, ReLU	
$600 \times M$ softmax	128×1, sigmoid	128×128

Table 2: Architecture of our CF-SFL framework.

Currently, neural network based models have achieved state-of-the-art performance for various recommender systems (Cheng et al. 2016; He et al. 2018, 2017; Zhang et al. 2018b; Liang et al. 2018). Among these methods, NCF (He et al. 2017) casts the matrix factorization algorithm into an entire neural framework, combing the shallow inner-product based learner with a series of stacked nonlinear transformations. This method outperforms various traditional baselines and has motivated many follow-up works such as NFM (He et al. 2017), Deep FM (Guo et al. 2017) and Wide and Deep (Cheng et al. 2016). Recently, deep generative has also achieved remarkable success. In particular, VAE based approach to CF uses variational inference to scale up the algorithm for large-scale dataset and has shown significant success in recommender systems using multinomial (Liang et al. 2018) or negative binomial (Zhao et al. 2020) likelihoods. CF-SFL is a general framework which can be integrated with such models seamlessly.

Reinforcement Learning in CF. For RL-based methods, contextual multi-armed bandits have been utilized to model the interactive nature of recommender systems. Thompson Sampling (TS) (Chapelle and Li 2011; Kveton et al. 2015; Zhang et al. 2018a) and Upper Confident Bound (UCB) (Li et al. 2010) are used to balance the trade-off between exploration and exploitation. Matrix factorization is combined with a bandit set-up in (Zhao, Zhang, and Wang 2013) to include latent vectors of items and users for better exploration. The MDP-Based CF model can be viewed as a partial observable MDP (POMDP) with partial observation of user preferences (Sunehag et al. 2015). Value function approximation and policy based optimization can be employed to solve the MDP. Modeling web page recommendation as a Q-Learning problem was proposed in (Zheng et al. 2018) and (Taghipour and Kardan 2008) and to make recommendations from web usage data. In this paper, we consider the recommending procedure as sequential interactions between virtual users and recommender; and leverage feedback from virtual users to improve the recommendation.

Experiments

Datasets. We investigate the effectiveness of the proposed CF-SFL framework on three benchmark datasets of recommendation systems. (*i*) MovieLens-20M (ML-20M), taken from a movie recommendation service containing tens of millions user-movie ratings; (*ii*) Netflix-Prize (Netflix), another user-movie ratings dataset collected by the Netflix Prize (Bennett, Lanning et al. 2007); (*iii*) Million Song Dataset (MSD), a user-song rating dataset, which is released

Methods	ML-20M			Netflix			MSD		
	R@20	R@50	NDCG@100	R@20	R@50	NDCG@100	R@20	R@50	NDCG@100
SLIM	0.370	0.495	0.401	0.347	0.428	0.379	-	-	-
WMF	0.360	0.498	0.386	0.316	0.404	0.351	0.211	0.312	0.257
CDAE	0.391	0.523	0.418	0.343	0.428	0.376	0.188	0.283	0.237
aWAE	0.391	0.532	0.424	0.354	0.441	0.381	-	-	-
VAE	0.395	0.537	0.426	0.351	0.444	0.386	0.266	0.364	0.316
VAE*	0.395	0.535	0.425	0.350	0.444	0.386	0.260	0.356	0.311
VAE†	0.396	0.536	0.426	0.352	0.445	0.387	0.263	0.360	0.314
CF-SFL	**0.404**	**0.542**	**0.435**	**0.355**	**0.449**	**0.394**	**0.273**	**0.369**	**0.323**

Table 3: Performance comparison between our CF-SFL framework and various baselines. VAE* is the results based on our own runs and VAE† is the VAE model with our reward estimator.

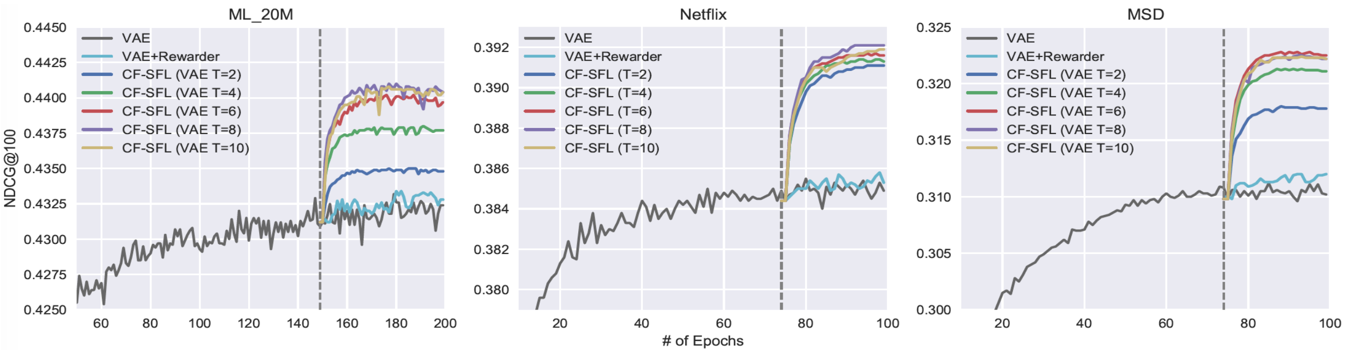

Figure 3: Performance (NDCG@100) boost on the validation sets.

as part of the Million Song Dataset (Bertin-Mahieux et al. 2011). To directly compare with existing work, we employed the same pre-processing procedure as (Liang et al. 2018). A summary statistics of these datasets are provided in Table 1.

Evaluation Metrics. We employ Recall@r[2] together with NDCG@r[3] as the evaluation metric for recommendation, which measures the similarity between the recommended items and the ground truth. Recall@r considers top-r recommended items equally, while NDCG@r ranks the top-r items and emphasizes the importance of the items that are with high ranks.

Set-up. For our CF-SFL framework, the architectures of its recommender, reward estimator and feedback generator are shown in Table 2. To represent the user preference, we normalize $\boldsymbol{x}_i$ and $\boldsymbol{v}_i^t$ $(t > 0)$ independently and concatenate the two into a single vector. To learn the model, we pre-train the recommender (150 epochs for ML-20M and 75 epochs for Netflix and MSD) and optimize the entire framework (50 epochs for ML-20M and 25 epochs for the other two). ℓ_2 regularization with a penalty term 0.01 is applied to the recommender, and Adam optimizer (Kingma and Ba 2014) with batch in size of 500 is employed.

Baselines. To demonstrate the efficacy of our framework, we consider multiple state-of-the-art approaches as baselines, which can be categorized into two types: (*i*) Linear models: SLIM (Ning and Karypis 2011) and WMF (Hu, Koren, and Volinsky 2008); and (*ii*) Deep neural network based models: CDAE (Wu et al. 2016), VAE (Liang et al. 2018), and aWAE (Zhang, Zhong, and Liu 2020). It should be noted that our CF-SFL is a generalized framework, which is compatible with all these approaches. In particular, as shown in Table 2, we implement our recommender as the VAE-based model (Liang et al. 2018) for a fair comparison. In the following experiments, we will show that besides such a setting the recommender can be implemented by other existing models as well.

All the evaluation metrics are averaged across all the test sets.

(i) Quantitative Results: we test various methods and report their results in Table 3. With the proposed CF-SFL framework, we observe improvements over the baselines on all the evaluation metrics. These experimental results demonstrate the power of the proposed CF-SFL framework, which provides informative feedback as side information. Particularly, we observed that the performance of the base model (VAE*) is similar to that of its variation with the reward estimator (VAE†). It implies that simply learning a feedback from the reward estimator via back-propagation is not much helpful. Compared with such a naïve strategy, the proposed CF-SFL provides more informative feedback to the recommender, and is able to improve recommendation results more effectively.

(ii) Learning Comparison: In Figure 3, we show the train-

[2] https://en.wikipedia.org/wiki/Precision_and_recall

[3] https://en.wikipedia.org/wiki/Discounted_cumulative_gain

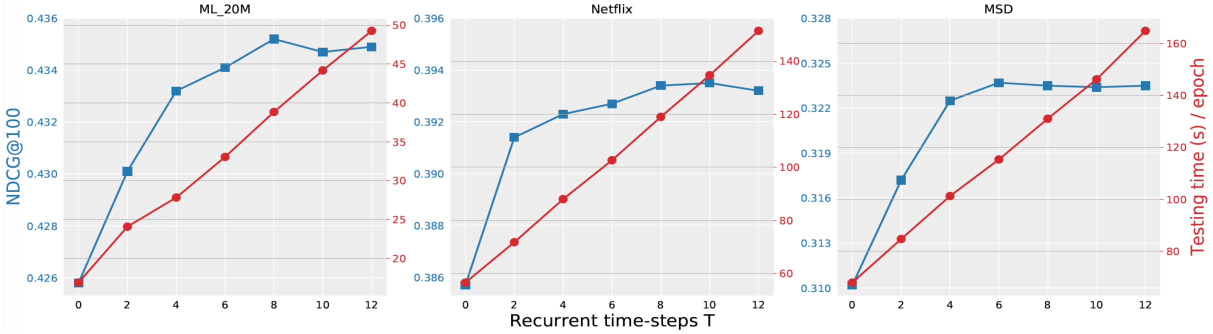

Figure 4: The blue curve summarizes NDCG@100 and red curves report the computational cost for model inference in each epoch. In each sub-figure, we vary the time steps from 0 to 12 ($T = 0$ is the base recommender).

ing trajectory of the baselines (VAE, VAE+reward estimator) and the CF-SFL with multiple time steps. There are several interesting findings: (a) The performance of the base VAE doesn't improve after the pre-training steps, *e.g.,* 75 epochs for Netflix. In comparison, the proposed CF-SFL framework can further improve the performance once the whole model is triggered; (b) The CF-SFL yields fast convergence once the whole framework is activated; (c) Consistent with the results in Table 3, the trajectory of VAE† in Figure 3 is similar to that of the base VAEs (VAE*). In contrast, the trajectories of our CF-SFL methods are more smooth and are to converge to a much better local minimum. This phenomenon further verifies that our CF-SFL learns informative user feedback with better stability; (d) With an increase in the number of time steps T in a particular range ($T \leq 8$ for ML-20M), CF-SFL achieves faster and better performance; One possible explanation is the learning with our unrolled structure — parameters are shared across different time-steps, and a more accurate gradient is found towards the local minimum; and (e) We find ML-20M and MSD are more sensitive to the choice of T when compared with Netflix. Therefore, the choice of T should be adjusted for different datasets.

(iii) CF-SFL with Dynamic Time Steps: As shown in Figure 2, learning of CF-SFL involves a recurrent structure with T times steps. We investigate the choice of T and report its influence on the performance of our method. Specifically, the NDCG@100 with different T values is shown in Figure 4. Within 6 time steps, CF-SFL consistently boots the performance on all the three datasets. Even with a larger time step, the results remain stable. Additionally, the inference time of CF-SFL is linear in T. To achieve a trade-off between performance and efficiency, in our experiments we set T to 8 for ML-20M and Netflix and 6 for MSD.

Relative Improvements due to Generative Feedback. As mentioned earlier, our CF-SFL is a generalized framework which is compatible with many existing collaborative filtering approaches. We study the usefulness of our CF-SFL on various recommendation systems and present the results in Table 4. Specifically, two types of recommenders are being considered: linear approaches like WARP (Weston, Bengio, and Usunier 2011) and MF (Hu, Koren, and Volinsky 2008), and deep learning methods, *e.g.*, DAE (Liang et al. 2018) and the variation of VAE in (Liang et al. 2018). We find that our CF-SFL is capable of generalizing most such collaborative filtering approaches and boosts their performance accordingly. The gains achieved by our CF-SFL may vary depending on the choice of recommender.

Recommender	**w/o CF-SFL**	**w CF-SFL**	**Gain** (10^{-3})
WARP	0.31228	0.33987	+27.59
MF	0.41587	0.41902	+3.15
DAE	0.42056	0.42307	+2.51
VAE	0.42546	0.43472	+9.26
VAE-(Gaussian)	0.42019	0.42751	+7.32
VAE-($\beta = 0$)	0.42027	0.42539	+5.02
VAE-Linear	0.41563	0.41597	+0.34

Table 4: Comparisons for various recommenders.

Conclusion

We presented CF-SLF, a novel framework for making recommendation from sparse data by simulating user feedback. It constructs a virtual user to provide informative side information as user feedback. We formulate the framework as an IRL problem and learn the optimal policy by feeding back the action and reward. Specifically, a recurrent architecture was built to unrolled the framework for efficient learning. Empirically we improve the performance of state-of-the-art collaborative filtering methods with a non-trivial margin. Our framework serves as a practical solution making IRL feasible over large-scale collaborative filtering. It will be interesting to investigate the framework in other applications, such as sequential recommender systems.

Acknowledgments

The Duke University component of this work was supported in part by DARPA, DOE, NIH, ONR and NSF, and a portion of the work performed by the first two authors was performed when they were affiliated with Duke. Additionally, Hongteng Xu was supported in part by Beijing Outstanding Young Scientist Program (NO. BJJWZYJH012019100020098) and National Natural Science Foundation of China (No. 61832017).

References

Agarwal, D.; Chen, B.-C.; and Elango, P. 2010. Fast online learning through offline initialization for time-sensitive recommendation. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 703–712.

Bayer, I.; He, X.; Kanagal, B.; and Rendle, S. 2017. A generic coordinate descent framework for learning from implicit feedback. In *Proceedings of the 26th International Conference on World Wide Web*, 1341–1350.

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

Bertin-Mahieux, T.; Ellis, D. P.; Whitman, B.; and Lamere, P. 2011. The million song dataset .

Chapelle, O.; and Li, L. 2011. An empirical evaluation of thompson sampling. *Advances in neural information processing systems* 24: 2249–2257.

Cheng, H.-T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, 7–10.

Fang, Y.; and Si, L. 2011. Matrix co-factorization for recommendation with rich side information and implicit feedback. In *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, 65–69.

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* .

He, X.; Du, X.; Wang, X.; Tian, F.; Tang, J.; and Chua, T.-S. 2018. Outer product-based neural collaborative filtering. *arXiv preprint arXiv:1808.03912* .

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, 173–182.

He, X.; Zhang, H.; Kan, M.-Y.; and Chua, T.-S. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 549–558.

Ho, J.; and Ermon, S. 2016. Generative adversarial imitation learning. *arXiv preprint arXiv:1606.03476* .

Hu, Y.; Koren, Y.; and Volinsky, C. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, 263–272. Ieee.

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

Koren, Y. 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*, 426–434.

Koren, Y.; and Bell, R. 2015. Advances in collaborative filtering. *Recommender systems handbook* 77–118.

Kveton, B.; Szepesvari, C.; Wen, Z.; and Ashkan, A. 2015. Cascading bandits: Learning to rank in the cascade model. In *International Conference on Machine Learning*, 767–776. PMLR.

Li, L.; Chu, W.; Langford, J.; and Schapire, R. E. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, 661–670.

Li, X.; and She, J. 2017. Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 305–314.

Liang, D.; Krishnan, R. G.; Hoffman, M. D.; and Jebara, T. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 world wide web conference*, 689–698.

Liu, N. N.; Xiang, E. W.; Zhao, M.; and Yang, Q. 2010. Unifying explicit and implicit feedback for collaborative filtering. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, 1445–1448.

Liu, Y.; Zhao, P.; Liu, X.; Wu, M.; and Li, X.-L. 2016. Learning optimal social dependency for recommendation. *arXiv preprint arXiv:1603.04522* .

Menon, A. K.; Chitrapura, K.-P.; Garg, S.; Agarwal, D.; and Kota, N. 2011. Response prediction using collaborative filtering with hierarchies and side-information. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 141–149.

Mishra, R.; Kumar, P.; and Bhasker, B. 2015. A web recommendation system considering sequential information. *Decision Support Systems* 75: 1–10.

Ning, X.; and Karypis, G. 2011. Slim: Sparse linear methods for top-n recommender systems. In *2011 IEEE 11th International Conference on Data Mining*, 497–506. IEEE.

Rendle, S. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*, 995–1000. IEEE.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* .

Rendle, S.; and Schmidt-Thieme, L. 2008. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*, 251–258.

Rennie, J. D.; and Srebro, N. 2005. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, 713–719.

Sarwar, B.; Karypis, G.; Konstan, J.; and Riedl, J. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, 285–295.

Schedl, M. 2016. The lfm-1b dataset for music retrieval and recommendation. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, 103–110.

Sunehag, P.; Evans, R.; Dulac-Arnold, G.; Zwols, Y.; Visentin, D.; and Coppin, B. 2015. Deep reinforcement learning with attention for slate markov decision processes with high-dimensional states and actions. *arXiv preprint arXiv:1512.01124* .

Taghipour, N.; and Kardan, A. 2008. A hybrid web recommender system based on q-learning. In *Proceedings of the 2008 ACM symposium on Applied computing*, 1164–1168.

Wang, Q.; Yin, H.; Hu, Z.; Lian, D.; Wang, H.; and Huang, Z. 2018. Neural memory streaming recommender networks with adversarial training. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2467–2475.

Wang, W.; Yin, H.; Sadiq, S.; Chen, L.; Xie, M.; and Zhou, X. 2016. SPORE: A sequential personalized spatial item recommender system. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, 954–965. IEEE.

Weston, J.; Bengio, S.; and Usunier, N. 2011. Wsabie: Scaling up to large vocabulary image annotation .

Wu, Y.; DuBois, C.; Zheng, A. X.; and Ester, M. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the ninth ACM international conference on web search and data mining*, 153–162.

Yang, C.; Bai, L.; Zhang, C.; Yuan, Q.; and Han, J. 2017. Bridging collaborative filtering and semi-supervised learning: a neural approach for poi recommendation. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1245–1254.

Zhang, R.; Li, C.; Chen, C.; and Carin, L. 2018a. Learning structural weight uncertainty for sequential decision-making. In *International Conference on Artificial Intelligence and Statistics*, 1137–1146. PMLR.

Zhang, S.; Yao, L.; Sun, A.; Wang, S.; Long, G.; and Dong, M. 2018b. Neurec: On nonlinear transformation for personalized ranking. *arXiv preprint arXiv:1805.03002* .

Zhang, X.; Zhong, J.; and Liu, K. 2020. Wasserstein autoencoders for collaborative filtering. *Neural Computing and Applications* 1–10.

Zhao, H.; Rai, P.; Du, L.; Buntine, W.; Phung, D.; and Zhou, M. 2020. Variational autoencoders for sparse and overdispersed discrete data. In *International Conference on Artificial Intelligence and Statistics*, 1684–1694. PMLR.

Zhao, X.; Zhang, W.; and Wang, J. 2013. Interactive collaborative filtering. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 1411–1420.

Zheng, G.; Zhang, F.; Zheng, Z.; Xiang, Y.; Yuan, N. J.; Xie, X.; and Li, Z. 2018. DRN: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 World Wide Web Conference*, 167–176.