PLASTIC: Prioritize Long and Short-term Information in Top-n Recommendation using Adversarial Training

Wei Zhao[1,2], **Benyou Wang**[2], **Jianbo Ye**[3], **Yongqiang Gao**[2], **Min Yang**[1*], **Xiaojun Chen**[4]
[1] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[2] Tencent Inc.
[3] Pennsylvania State University
[4] Shenzhen University

Abstract

Recommender systems provide users with ranked lists of items based on individual's preferences and constraints. Two types of models are commonly used to generate ranking results: long-term models and session-based models. While long-term models represent the interactions between users and items that are supposed to change slowly across time, session-based models encode the information of users' interests and changing dynamics of items' attributes in short terms. In this paper, we propose a PLASTIC model, **P**rioritizing **L**ong **A**nd **S**hort-**T**erm **I**nformation in top-n re**C**ommendation using adversarial training. In the adversarial process, we train a generator as an agent of reinforcement learning which recommends the next item to a user sequentially. We also train a discriminator which attempts to distinguish the generated list of items from the real list recorded. Extensive experiments show that our model exhibits significantly better performances on two widely used datasets.[1]

1 Introduction

With the sheer volume of online information, much attention has been given to data-driven recommender systems. Numerous recommendation techniques have been developed. Three main categories of them are collaborative filtering methods, content-based methods and hybrid methods [Bobadilla *et al.*, 2013; Lu *et al.*, 2015]. In this paper, we aim to develop a method producing a ranked list of n items to a user by collaborative filtering that exploits historical user-item interactions.

Matrix factorization (MF) [Koren *et al.*, 2009] is one of the most successful techniques in the practice of recommendation due to its simplicity, attractive accuracy and scalability. It models the user preference matrix approximately as a product of two lower-rank latent feature matrices representing user profiles and item profiles respectively. Matrix factorization is robust to the data sparsity and cold-start issues in the recommender system.

Despite the appeal of matrix factorization, this technique does not explicitly consider the temporal variability of data [Wu *et al.*, 2017]. Firstly, the popularity of an item may change over time. For example, movie popularity booms or fades, which can be triggered by external events such as the appearance of an actor in a new movie. Secondly, users may change their interests and baseline ratings over time. For instance, a user who tended to rate an average movie as ”4 stars”, may now rate such a movie as ”3 stars”. Recently, recurrent neural network (RNN) [Hochreiter and Schmidhuber, 1997] has gained significant attention by considering such temporal dynamics for both users and items, and achieved high recommendation quality [Wu *et al.*, 2016b; 2017].

More recent work [Wu *et al.*, 2017] reveals that matrix factorization based and RNN based recommendation approaches are complementary to each other. Specifically, the matrix factorization recommendation approaches make item predictions based on users' long-term interests which change very slowly with respect to time. On the contrary, the RNN recommendation approaches predict which item will the user consume next, respecting the dynamics of users' behaviors and items' attributes in the short term. It therefore motivates us to devise a joint approach that takes advantage of both matrix factorization and RNN, exploiting both long-term and short-term associations among users and items.

In this paper, we propose a novel PLASTIC model, which **P**rioritizes **L**ong **A**nd **S**hort-**T**erm **I**nformation in top-n re**C**ommendation using adversarial training. The PLASTIC model employs an adversarial framework to combine the MF and RNN based models for the top-n recommendation, taking the best of each to improve the final recommending performance. In the adversarial process, we simultaneously train two models: a generative model G and a discriminative model D. In particular, the generator G takes the user u_i and time t as input, and predicts the recommendation list for user i at time t based on the historical user-item interactions. We implement the discriminator D via a Siamese Network that incorporates long-term and session-based ranking model in a pair-wise scenario. The two point-wise networks of Siamese Network share the same set of parameters. The generator G and the discriminator D are optimized with a minimax two-

*Min Yang is corresponding author (min.yang@siat.ac.cn)

[1]Codes are publicly available at `https://goo.gl/WUVK9a`.

player game. The discriminator D tries to distinguish the real high-rated items in the training data from the recommendation list generated by the generator G, while the training procedure of generator G is to maximize the probability of D making a mistake. Thus, this adversarial process can eventually adjust G to generate plausible and high-quality recommendation list.

We summarize our main contributions as follows:

- To the best of our knowledge, we are the first to use GAN framework to leverage the MF and RNN approaches for top-n recommendation. This joint model adaptively adjusts how the contributions of the long-term and short-term information of users and items are mixed together.
- We propose four hard and soft mixture mechanisms to integrate MF and RNN. We use the hard mechanism to calculate the mixing score straightforwardly and explore several soft mechanisms to learn the temporal dynamics with the help of the long-term profiles.
- Our model uses reinforcement learning to optimize the generator G for generating highly rewarded recommendation list. Thus, it effectively bypasses the non-differentiable task metric issue by directly performing policy gradient update.
- The experimental results show that our model consistently outperforms the state-of-the-art methods.

2 Related Work

Recommender system is an active research field. The authors of [Bobadilla *et al.*, 2013; Lu *et al.*, 2015] describe most of the existing techniques for recommender systems.

Matrix factorization for recommendation Modeling the long-term interests of users, the matrix factorization method and its variants have grown to become dominant in the literature [Rennie and Srebro, 2005; Koren *et al.*, 2009; Hernando *et al.*, 2016; He *et al.*, 2016; Tu *et al.*, 2015]. In the standard matrix factorization, the recommendation task can be formulated as inferring missing values of a partially observed user-item matrix [Koren *et al.*, 2009]. Srebro *et al.* [2005] suggested the Maximum Margin Matrix Factorization (MMMF), which used low-norm instead of low-rank factorizations. Mnih and Salakhutdinov [2008] presented the Probabilistic Matrix Factorization (PMF) model that characterized the user preference matrix as a product of two lower-rank user and item matrices. The PMF model was especially effective at making better predictions for users with few ratings. He *et al.* [2016] proposed a new MF method which considers the implicit feedback for on-line recommendation.

Recurrent neural network for recommendation These traditional MF methods for recommendation systems are based on the assumption that the user interests and item attributes are near static, which is however not consistent with reality. Koren [2010] discussed the effect of temporal dynamics in recommender systems and proposed a temporal extension of the SVD++ (called TimeSVD++) to explicitly model the temporal bias in data. However, the features used in TimeSVD++ were hand-crafted and computationally expensive to obtain. Recently, there have been increasing interests in employing recurrent neural network to model temporal dynamic in recommendation systems. For example, Hidasi *et al.* [2015] applied recurrent neural network (GRU) to session-based recommender systems. Wu *et al.* [2016a] proposed a recurrent neural network to perform the time heterogeneous feedback recommendation. Wu *et al.* [2017] used LSTM autoregressive model for the user and item dynamics and employed matrix factorization to model the stationary components that encode fixed properties. Different from their work, we use GAN framework to leverage the MF and RNN approaches for top-n recommendation, aiming to generate plausible and high-quality recommendation lists.

Generative adversarial network for recommendation In parallel, previous work has demonstrated the effectiveness of generative adversarial network (GAN) [Goodfellow *et al.*, 2014] in various tasks such as image generation [Reed *et al.*, 2016], image captioning [Chen *et al.*, 2017], and sequence generation[Yu *et al.*, 2017; Liu *et al.*, 2018]. The most related work to ours is [Wang *et al.*, 2017], which proposed a novel IRGAN mechanism to iteratively optimize a generative retrieval component and a discriminative retrieval component. Our approach differs from theirs in several aspects. First, we combine the MF approach and the RNN approach with GAN, exploiting the performance contributions of both approaches. Second, IRGAN does not attempt to estimate the future behavior. In fact, they use future trajectories to infer the historical records, which seems not useful in real-life applications.

3 Our Model

Suppose there is a sparse user-item rating matrix R that consists of U users and M items. Each entry $r_{ij,t}$ denotes the rating of user i on item j at time step t. The rating is represented by numerical values from 1 to 5, where the higher value indicates the stronger preference. Instead of predicting the rating of a specific user-item pair as is done in [McNee *et al.*, 2006], our model aims to provide users with ranked lists of items (top-n recommendation) [Liu and Yang, 2008].

The PLASTIC model employs an adversarial framework to combine the MF and RNN based models for the top-n item recommendation. In the adversarial process, we simultaneously train two models: a generative model G and a discriminative model D.

3.1 Matrix Factorization (MF)

The MF framework [Mnih and Salakhutdinov, 2008] models the long-term states (global information) for both users ($\mathbf{e}^u$) and items ($\mathbf{e}^m$). In its standard setting, the recommendation task can be formulated as inferring missing values of a partially observed user-item rating matrix R. The formulation of MF is given by:

$$\operatorname*{argmin}_{\mathbf{e}^u,\mathbf{e}^m} \sum_{i,j} I_{ij} \left(r_{ij} - \rho \left((\mathbf{e}_i^u)^T \mathbf{e}_j^m \right) \right)^2 + \lambda^u \|\mathbf{e}^u\|_F^2 + \lambda^m \|\mathbf{e}^m\|_F^2 \quad (1)$$

where $\mathbf{e}_i^u$ and $\mathbf{e}_j^m$ represent the user and item latent factors in the shared d-dimension space respectively. r_{ij} denotes the user i's rating on item j. I_{ij} is an indicator function and

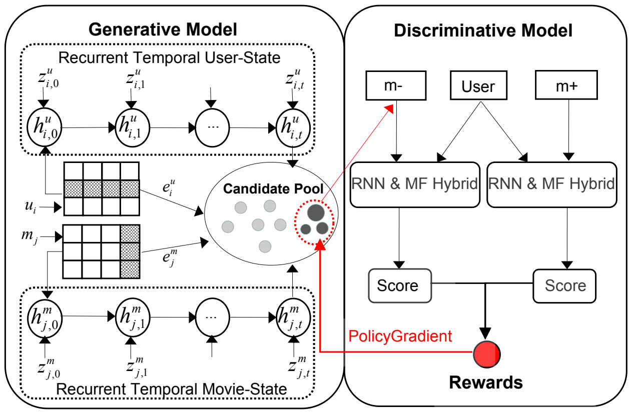

Figure 1: The Architecture of Long-term and Session-based Ranking Model with Adversarial Network

equals 1 if $r_{ij} > 0$, and 0 otherwise. λ^u and λ^m are regularization coefficients. The $\rho(\cdot)$ is a logistic scoring function that bounds the range of outputs.

Because minimizing the objective function – the squared errors – does not perfectly align with the goal to optimize the ranking order. In this paper, we apply MF for ranking prediction (top-n recommendation) directly, similar to [Wang *et al.*, 2017].

3.2 Recurrent Neural Network (RNN)

The RNN based recommender system focuses on modeling session-based trajectories instead of global (long-term) information [Wu *et al.*, 2017]. It predicts future behaviors and provides users with a ranking list given the users' past history. The main purpose of using RNN is to capture time-varying state for both users and items. Particularly, we use LSTM cell as the basic RNN unit. Each LSTM unit at time t consists of a memory cell c_t, an input gate i_t, a forget gate f_t, and an output gate o_t. These gates are computed from previous hidden state $\mathbf{h}_{t-1}$ and the current input $\mathbf{x}_t$:

$$[f_t, i_t, o_t] = \text{sigmoid}(W[\mathbf{h}_{t-1}, \mathbf{x}_t]) \tag{2}$$

The memory cell c_t is updated by partially forgetting the existing memory and adding a new memory content $\mathbf{l}_t$:

$$\mathbf{l}_t = \tanh(V[\mathbf{h}_{t-1}, \mathbf{x}_t]) \tag{3}$$

$$\mathbf{c}_t = f_t \odot \mathbf{c}_{t-1} + i_t \odot \mathbf{l}_t \tag{4}$$

Once the memory content of the LSTM unit is updated, the hidden state at time step t is given by:

$$\mathbf{h}_t = o_t \odot \tanh(\mathbf{c}_t) \tag{5}$$

For simplicity of notation, the update of the hidden states of LSTM at time step t is denoted as $\mathbf{h}_t = \text{LSTM}(\mathbf{h}_{t-1}, \mathbf{x}_t)$.

Here, we use $\mathbf{z}^u_{i,t} \in \mathbb{R}^M$ and $\mathbf{z}^m_{j,t} \in \mathbb{R}^U$ to represent the rating vector of user i and item j given time t respectively. Both $\mathbf{z}^u_{i,t}$ and $\mathbf{z}^m_{j,t}$ serve as the input to the LSTM layer at time t to infer the new states of the user and the item:

$$\mathbf{h}^u_{i,t} = \text{LSTM}(\mathbf{h}^u_{i,t-1}, \mathbf{z}^u_{i,t}) \tag{6}$$

$$\mathbf{h}^m_{j,t} = \text{LSTM}(\mathbf{h}^m_{j,t-1}, \mathbf{z}^m_{j,t}) \tag{7}$$

Here, $\mathbf{h}^u_{i,t}$ and $\mathbf{h}^m_{j,t}$ denote hidden states for user i and item j at time step t respectively.

3.3 RNN and MF Hybrid

The session-based model deals with temporal dynamics of the user and item states, we further incorporate the long-term preference of users and the fixed properties of items. To exploit their advantages together, similar to [Wu *et al.*, 2017], we define the rating prediction function as:

$$r_{ij,t} = g(\mathbf{e}^u_i, \mathbf{e}^m_j, \mathbf{h}^u_{i,t}, \mathbf{h}^m_{j,t}) \tag{8}$$

where $g(.)$ is a score function, $\mathbf{e}^u_i$ and $\mathbf{e}^m_j$ denote the global latent factors of user i and item j learned by Eq. (1); $\mathbf{h}^u_{i,t}$ and $\mathbf{h}^m_{j,t}$ denote the hidden states at time step t of two RNNs learned by Eq. (6) and Eq. (7) respectively. In this work, we study four strategies to calculate the score function g, integrating MF and RNN. The details are described below.

PLASTIC-V1 This is a hard mechanism, using a simple way to calculate the mixing score from MF and RNN with the following formulation:

$$r_{ij,t} = g(\mathbf{e}^u_i, \mathbf{e}^m_j, \mathbf{h}^u_{i,t}, \mathbf{h}^m_{j,t}) = \frac{1}{1 + \exp(-\mathbf{s})} \tag{9}$$

$$\mathbf{s} = \mathbf{e}^u_i \cdot \mathbf{e}^m_j + \mathbf{h}^u_{i,t} \cdot \mathbf{h}^m_{j,t} + b^u_i + b^m_j \tag{10}$$

where b^u_i and b^m_j are the biases of user i and item j; $\mathbf{h}^u_{i,t}$ and $\mathbf{h}^m_{j,t}$ are computed by Eq. (6) and Eq. (7).

In fact, PLASTIC-V1 does not exploit the global factors in learning the temporal dynamics. In this paper, we also design a soft mixture mechanism and provide three strategies to account for the global factors $\mathbf{e}^u_i$ and $\mathbf{e}^m_j$ in learning $\mathbf{h}^u_{i,t}$ and $\mathbf{h}^m_{j,t}$, as described below (i.e., PLASTIC-V2, PLASTIC-V3 and PLASTIC-V4).

PLASTIC-V2 We use the latent factors of user i ($\mathbf{e}^u_i$) and item j ($\mathbf{e}^m_j$) pre-trained by MF model to initialize the hidden states of the LSTM cells $\mathbf{h}^u_{i,0}$ and $\mathbf{h}^m_{j,0}$ respectively.

PLASTIC-V3 We extend PLASTIC-V2 by treating $\mathbf{e}^u_i$ (for user i) and $\mathbf{e}^m_j$ (for item j) as the static context vectors, and feed them as an extra input into the computation of the temporal hidden states of users and items by LSTM. At each time step, the context information assists the inference of the hidden states of LSTM model.

PLASTIC-V4 This method uses an attention mechanism to compute a weight for each hidden state by exploiting the global factors. The mixing scores at time t can be reformulated by:

$$r_{ij,t} = g(\mathbf{e}^u_i, \mathbf{e}^m_j, \mathbf{h}^u_{i,t-1}, \mathbf{h}^m_{j,t-1}, \mathbf{c}^u_{i,t}, \mathbf{c}^m_{j,t}) = \frac{1}{1 + \exp(-\mathbf{s})} \tag{11}$$

$$\mathbf{s} = \mathbf{e}^u_i \cdot \mathbf{e}^m_j + \mathbf{h}^u_{i,t} \cdot \mathbf{h}^m_{j,t} + b_i + b_j \tag{12}$$

where $\mathbf{c}^u_{i,t}$ and $\mathbf{c}^m_{j,t}$ are the context vectors at time step t for user i and item j; $\mathbf{h}^u_{i,t}$ and $\mathbf{h}^m_{j,t}$ are the hidden states of LSTMs at time step t, computed by

$$\mathbf{h}^u_{i,t} = \text{LSTM}(\mathbf{h}^u_{i,t-1}, \mathbf{z}^u_{i,t}, \mathbf{c}^u_{i,t}) \tag{13}$$

$$\mathbf{h}^m_{j,t} = \text{LSTM}(\mathbf{h}^m_{j,t-1}, \mathbf{z}^m_{j,t}, \mathbf{c}^m_{j,t}) \tag{14}$$

The context vectors $\mathbf{c}^u_{i,t}$ and $\mathbf{c}^m_{j,t}$ act as extra input in the computation of the hidden states in LSTMs to make sure that every time step of the LSTMs can get full information of the context (long-term information). The context vectors $\mathbf{c}^u_{i,t}$ and $\mathbf{c}^m_{j,t}$ are the dynamic representations of the relevant long-term information for user i and item j at time t, calculated by

$$\mathbf{c}^u_{i,t} = \sum_{k=1}^{U} \alpha^i_{k,t}\mathbf{e}^u_k; \quad \mathbf{c}^m_{j,t} = \sum_{p=1}^{M} \beta^j_{p,t}\mathbf{e}^m_p \tag{15}$$

where U and M are the number of users and items. The attention weights $\alpha^i_{k,t}$ and $\beta^j_{p,t}$ for user i and item j at time step t are computed by

$$\alpha^i_{k,t} = \frac{\exp(\sigma(\mathbf{h}^u_{i,t-1}, \mathbf{e}^u_k))}{\sum_{k'=1}^{U}\exp(\sigma(\mathbf{h}^u_{i,t-1}, \mathbf{e}^u_{k'}))} \tag{16}$$

$$\beta^j_{p,t} = \frac{\exp(\sigma(\mathbf{h}^m_{j,t-1}, \mathbf{e}^m_p))}{\sum_{p'=1}^{M}\exp(\sigma(\mathbf{h}^m_{j,t-1}, \mathbf{e}^m_{p'}))} \tag{17}$$

where σ is a feed-forward neural network to produce a real-valued score. The attention weights α^i_t and β^j_t together determine which user and item factors should be selected to generate $r_{ij,t}$.

3.4 Generative Adversarial Network (GAN) for Recommendation

Generative adversarial network (GAN) [Goodfellow *et al.*, 2014] consists of a generator G and a discriminator D that compete in a minimax game with two players: The discriminator tries to distinguish real high-rated items on training data from ranking or recommendation list predicted by G, and the generator tries to fool the discriminator to generate(predict) well-ranked recommendation list. Concretely, D and G play the following game on V(D,G):

$$\min_G \max_D v(D,G) = \mathbb{E}_{x\sim P_{true}(x)}[\log D(x)] + \mathbb{E}_{z\sim P_z(z)}[\log(1 - D(G(z)))] \tag{18}$$

Here, x is the input data from training set, z is the noise variable sampled from normal distribution.

We propose an adversarial framework to iteratively optimize two models: the generative model G predicting recommendation list given historical user-item interactions, and the discriminative model D predicting the relevance of the generated list. Like the standard generative adversarial networks (GANs) [Goodfellow *et al.*, 2014], our model also optimizes the two models with a minimax two-player game. D tries to distinguish the real high-rated items in the training data from the recommendation list generated by G, while G maximizes the probability of D making a mistake. Hopefully, this adversarial process can eventually adjust G to generate plausible and high-quality recommendation list. We further elaborate the generator and discriminator below.

Discriminative Model

As depicted in Figure 1 (right side), we implement the discriminator D via a Siamese Network that incorporates long and session-based ranking models in a pair-wise scenario. The discriminator D has two symmetrical point-wise networks that share parameters and are updated by minimizing a pair-wise loss.

The objective of discriminator D is to maximize the probability of correctly distinguishing the ground truth items from generated recommendation items. For G fixed, we can obtain the optimal parameters for the discriminator D with the following formulation.

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{i\in\mathcal{U}} \Big(\mathbb{E}_{m_+,m_-\sim p_{true}}[\log D_\theta(u_i, m_-, m_+|t)] + \mathbb{E}_{m_+\sim p_{true}, m_{g,t}\sim G_\phi(m_{g,t}|u_i,t)}[\log(1 - D_\theta(u_i, m_{g,t}, m_+|t)] \Big) \tag{19}$$

where $\mathcal{U}$ denotes the user set, u_i denotes user i, m_+ is a positive (high-rating) item, m_- is a negative item randomly chosen from the entire negative (low-rating) item space, θ and ϕ are parameters of D and G, and $m_{g,t}$ is the generated item by G given time t. Here, we adopt hinge loss as our training objective since it performs better than other training objectives. Hinge loss is widely adopted in various learning to rank scenario, which aims to penalize the examples that violate the margin constraint:

$$D(u_i, m_-, m_+|t) = \max\Big\{0, \epsilon - g(\mathbf{e}^u_i, \mathbf{e}^m_{m_+}, \mathbf{h}^u_{i,t}, \mathbf{h}^m_{m_+,t}) + g(\mathbf{e}^u_i, \mathbf{e}^m_{m_-}, \mathbf{h}^u_{i,t}, \mathbf{h}^m_{m_-,t})\Big\} \tag{20}$$

where ϵ is the hyper-parameter determining the margin of hinge loss, and we compress the outputs to the range of $(0, 1)$.

Generative Model

Similar to conditional GANs proposed in [Mirza and Osindero, 2014], our generator G takes in the auxiliary information (user u_i and time t) as input, and generates the ranking list for user i. Specifically, when D is optimized and fixed after computing Eq. 19, the generator G can be optimized by minimizing the following formulation:

$$\phi^* = \underset{\phi}{\operatorname{argmin}} \sum_{m\in\mathcal{M}} \left(\mathbb{E}_{m_{g,t}\sim G_\phi(m_{g,t}|u_i,t)}[\log(1 - D(u_i, m_{g,t}, m_+|t)]\right) \tag{21}$$

Here, $\mathcal{M}$ denotes the item set. As in [Goodfellow *et al.*, 2014], instead of minimizing $\log(1-D(u_i, m_{g,t}, m_+|t))$, we train G to maximize $\log(D(u_i, m_{g,t}, m_+|t))$.

Policy Gradient

Since the sampling of recommendation list by generator G is discrete, it cannot be directly optimized by gradient descent as in the standard GAN formulation. Therefore, we use policy gradient based reinforcement learning algorithm [Sutton *et al.*, 2000] to optimize the generator G so as to generate highly rewarded recommendation list. Concretely, we have the following derivations:

$$\begin{aligned}
\nabla_\phi J^G(u_i) &= \nabla_\phi \mathbb{E}_{m_{g,t}\sim G_\phi(m_{g,t}|u_i,t)}[\log D(u_i, m_{g,t}, m_+|t)] \\
&= \sum_{m\in\mathcal{M}} \nabla_\phi G_\phi(m|u_i,t)\log D(u_i, m, m_+|t) \\
&= \sum_{m\in\mathcal{M}} G_\phi(m|u_i,t)\nabla_\phi \log G_\phi(m|u_i,t)\log D(u_i, m, m_+|t) \\
&\approx \frac{1}{K}\sum_{k=1}^{K} \nabla_\phi \log G_\phi(m_k|u_i,t)\log D(u_i, m_k, m_+|t)
\end{aligned} \tag{22}$$

where K is number of items sampled by the current version of generator and m_k is the k-th sampled item. With reinforcement learning terminology, we treat the term $\log D(u_i, m_k, m_+|t)$ as the reward at time step t, and take an action m_k at each time step. To accelerate the convergence,

Algorithm 1: Long and Session-based Ranking Model with Adversarial Network

```
1  Input: generator $G_\phi$, discriminator $D_\theta$, training data $S$.
2  Initialize models $G_\phi$ and $D_\theta$ with random weights, and
     pre-train them on training data $S$.
3  repeat
4  for g-steps do
5      Generate recommendation list for user $i$ at time $t$
         using the generator $G_\phi$.
6      Sample $K$ candidates from recommendation list.
7      for $k \in \{1, ..., K\}$ do
8          Sample a positive item $m_+$ from S.
9          Compute the reward $\log D(u_i, m_k, m_+|t)$ with
             Eq.(20)
10     Update generator $G_\phi$ via policy gradient Eq.(22).
11 for d-steps do
12     Use current $G_\phi$ to generate a negative item and
         combined with a positive item sampled from $S$.
13     Update discriminator $D_\theta$ with Eq.(19).
14 until convergence
```

the rewards within a batch are normalized with a Gaussian distribution to make the significant differences.

The overall procedure is summarized in Algorithm 1. During the training stage, the discriminator and the generator are trained alternatively in an adversarial manner via Eq.(19) and Eq.(22), respectively.

4 Experimental Setup

4.1 Datasets

In order to evaluate the effectiveness of our model, we conduct experiments on two widely-used real-life datasets: Movielens100K and Netflix (called "Netflix-Full"). For each dataset, we split the whole data into several training and testing intervals based on time, as done in [Wu *et al.*, 2017], to simulate the actual situation of predicting future behaviors of users given the data that occurred strictly before current time step. Then, each testing interval is randomly divided into a validation set and a testing set. We removed the users and items that do not appear in training set from the validation and test sets. The detailed statistics are presented in Table **??**[2]. The users and items that are not shown in the training data are removed from the test data. Following [Wang *et al.*, 2017], we treat "5-star" in Netflix, "4-start" and "5-start" in Movielens100K as positive feedback and all others as unknown (negative) feedback.

4.2 Implementation Details

Matrix Factorization. We use matrix factorization with 5 and 16 factor numbers for Movielens and Netflix respectively [Wang *et al.*, 2017]. The parameters are randomly initialized by a uniform distribution ranged in [-0.05, 0.05]. We take gradient-clipping to suppress the gradient to the range of [0.2,0.2]. L2 regularization (with $\lambda = 0.05$) is used to the weights and biases of user and item factors.

[2]"Density" shows the average amount of 5-ratings for the user per day. "Sparsity" shows the filling-rate of user-item rating matrix as used in [Wu *et al.*, 2017]

Dataset	Movielens-100K	Netflix-Full
Users	943	480,189
Items	1,6831	17,770
Ratings	100,000	100,480,507
Train Data	09/97-03/98	12/99-11/05
Test Data	03/98-04/98	12/05
Train Ratings	77,714	98,074,901
Test Ratings	21,875	2,405,578
Sparsity [Wu *et al.*, 2017]	0.063	0.012

Table 1: Characteristics of the datasets.

Recurrent Neural Network. We use a single-layer LSTM with 10 hidden neurons, 15-dimensional input embeddings, and 4-dimensional dynamic states where each state contains 7-days users/items behavioral trajectories. That is, we take one month as the length of a session. The parameters are initialized in the same way as in MF. L2 regularization (with $\lambda = 0.05$) is used to the weights and biases of the LSTM layer to avoid over-fitting.

Generative Adversarial Nets. We pre-train G and D on the training data with a pair-wise scenario, and use SGD algorithm with learning rate 1×10^{-4} to optimize its parameters. In addition, we use matrix factorization model to generate 25 candidate items, and then re-rank these items with LSTM. In all experiments, we conduct mini-batch training with batch size 128.

4.3 Evaluation Metrics

To quantitatively evaluate our method, we adopt the rank-based evaluation metrics to measure the performance of top-n recommendation [Liu and Yang, 2008], including Precision@N, Normalised Discounted Cumulative Gain (NDCG@N), Mean Average Precision (MAP) and Mean Reciprocal Ranking (MRR).

4.4 Comparison to Baselines

In the experiments, we compare our approach with several strong baseline methods including Bayesian Personalised Ranking (BPR) [Rendle *et al.*, 2009], Pairwise Ranking Factorization Machine (PRFM) [Qiang *et al.*, 2013], LambdaFM [Yuan *et al.*, 2016], Recurrent Recommender Networks (RRN) [Wu *et al.*, 2017], and IRGAN [Wang *et al.*, 2017]. For all the baseline methods, we adopt the default settings for all hyperparameters.

5 Experimental Results

5.1 Quantitative Results

We first evaluate the performance of top-n recommendation. The experimental results are summarized in Tables **??** and 3. Our model substantially and consistently outperforms the baseline methods by a noticeable margin on all the experimental datasets. In particular, we have explored several

	P@3	**P@5**	**P@10**	**MRR**
BPR	0.2795	0.2664	0.2301	0.4324
PRFM	0.2884	0.2699	0.2481	0.4484
LambdaFM	0.3108	0.2953	0.2612	0.4611
RRN	0.2893	0.2740	0.2480	0.4320
IRGAN	0.3022	0.2885	0.2582	0.4515
PLASTIC-V1	0.2926	0.2743	0.2491	0.4293
PLASTIC-V2	0.3024	0.2883	0.2518	0.4438
PLASTIC-V3	0.3118	0.2993	0.2571	0.4652
PLASTIC-V4	**0.3311**	**0.3115**	**0.2817**	**0.4815**
Impv	6.53%	5.49%	7.85%	4.42%
	NDCG@3	**NDCG@5**	**NDCG@10**	**MAP**
BPR	0.2910	0.2761	0.2550	0.3549
PRFM	0.2937	0.2894	0.2676	0.3885
LambdaFM	0.3302	0.3117	0.2795	0.4014
RRN	0.2951	0.2814	0.2513	0.3631
IRGAN	0.3285	0.3032	0.2678	0.3744
PLASTIC-V1	0.2955	0.2831	0.2624	0.3699
PLASTIC-V2	0.3089	0.2885	0.2814	0.3682
PLASTIC-V3	0.3230	0.3006	0.2912	0.3936
PLASTIC-V4	**0.3497**	**0.3312**	**0.2939**	**0.4223**
Impv	5.91%	6.26%	5.15%	5.21%

Table 2: Movie recommendation results (MovieLens).

	P@3	**P@5**	**P@10**	**MRR**
BPR	0.3011	0.2817	0.2587	0.3840
PRFM	0.2959	0.2837	0.2624	0.4060
LambdaFM	0.3446	0.3301	0.3226	0.4356
RRN	0.3135	0.2954	0.2699	0.3953
IRGAN	0.3320	0.3229	0.3056	0.4248
PLASTIC-V1	0.3092	0.3092	0.2848	0.4070
PLASTIC-V2	0.3367	0.3367	0.3192	0.4408
PLASTIC-V3	0.3407	0.3480	0.3236	0.4412
PLASTIC-V4	**0.3595**	**0.3510**	**0.3498**	**0.4527**
Impv	4.32%	6.33%	8.43%	6.57%
	NDCG@3	**NDCG@5**	**NDCG@10**	**MAP**
BPR	0.2998	0.2870	0.2693	0.3660
PRFM	0.2831	0.2887	0.2789	0.3916
LambdaFM	0.3450	0.3398	0.3255	0.4067
RRN	0.3123	0.3004	0.2810	0.3768
IRGAN	0.3319	0.3260	0.3131	0.4052
PLASTIC-V1	0.3227	0.3128	0.2948	0.3898
PLASTIC-V2	0.3439	0.3391	0.3264	0.4219
PLASTIC-V3	0.3483	0.3431	0.3308	0.4212
PLASTIC-V4	**0.3586**	**0.3502**	**0.3402**	**0.4316**
Impv	3.94%	3.06%	4.51%	6.12%

Table 3: Movie recommendation results (Netflix-Full).

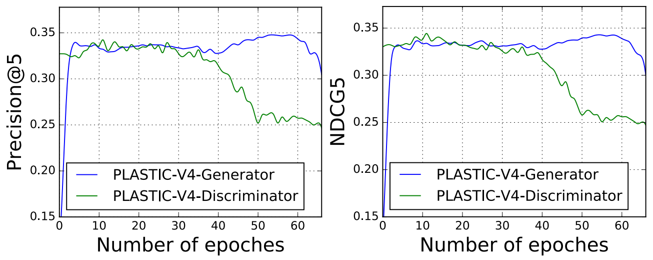

Figure 2: Learning curves of GAN on Netflix-Full

versions of our model with different mixture mechanisms. As one anticipates, PLASTIC-V4 achieves the best results across all evaluation metrics and all datasets. For example, on MovieLens dataset, PLASTIC-V4 improves 7.45% on percision@5 and 6.87% on NDCG@5 over the baseline methods.

Groundtruth	**IRGAN** [Wang *et al.*, 2017]	**RRN** [Wu *et al.*, 2017]	**PLASTIC-SoftV4**
9 Souls The Princess Bride Stuart Saves Family The Last Valley Wax Mask After Hours Session 9 Valentin	**[1]** The Beatles **[2]** Wax Maska ✓ **[3]** Stuart Saves Family ✓ **[4]** After Hours ✓ **[5]** Top Secret! **[6]** Damn Yankees **[7]** Dragon Tales **[8]** Play Misty for Me **[9]** The Last Round **[10]** La Vie de Chateau	**[1]** Falling Down **[2]** 9 Souls ✓ **[3]** Wax Mask ✓ **[4]** After Hours ✓ **[5]** Stuart Saves Family ✓ **[6]** Crocodile Dundee 2 **[7]** The Princess Bride ✓ **[8]** Dragon Tales **[9]** They Were Expendable **[10]** Damn Yankees	**[1]**9 Souls ✓ **[2]**The Princess Bride ✓ **[3]**Stuart Saves Family ✓ **[4]** The Last Valley ✓ **[5]** Wax Mask ✓ **[6]** Session 9 ✓ **[7]** Dragon Tales **[8]** Damn Yankees **[9]** After Hours ✓ **[10]** Valentin ✓
9 Souls Princess Bride	**[1]** Cheech Chong's Up **[2]** Wax Mask **[3]** Damn Yankees **[4]** Dragon Tales **[5]** Top Secret! **[6]** Agent Cody Banks 2 **[7]** After Hours **[8]** Stuart Saves Family **[9]** 9 Souls ✓ **[10]** The Beatles	**[1]** Crocodile Dundee 2 **[2]** 9 Souls ✓ **[3]** Falling Down **[4]** Wax Mask **[5]** After Hours **[6]** Stuart Saves Family **[7]** Session 9 **[8]** The Princess Bride ✓ **[9]** Dragon Tales **[10]** Scream 2	**[1]** 9 Souls ✓ **[2]**The Princess Bride ✓ **[3]**The Last Valley **[4]**Stuart Saves Family **[5]**Wax Mask **[6]**Dragon Tales **[7]**Session 9 **[8]**Crocodile Dundee 2 **[9]**Damn Yankees **[10]**Cheech Chong's Up

Table 4: The recalled items from top-10 candidates.

The main strength of our model comes from its capability of prioritizing both long-term and short-term information in top-n recommendation. In addition, our mixture mechanisms (hard and soft) also seem quite effective to integrate MF and RNN.

To better understand the adversarial training process, we visualize the learning curves of PLASTIC-V4 as shown in Figure 2. Due to the limited space, we only report the Pecision@5 and NDCG@5 scores as in [Wang *et al.*, 2017]. The other metrics exhibit a similar trend. As shown in Figure 2, after about 50 epochs, both Precision@5 and NDCG@5 converge and the winning player is the generator which is used to generate recommendation list for our final top-n item recommendation. The performance of generator G becomes better with the effective feedback (reward) from discriminator D. On the other hand, once we have a set of high-quality recommendation movies, the performance of D deteriorates gradually in the training procedure and makes mistakes for predictions. In our experiments, we use the generator G with the best performance to predict test data.

5.2 Case study

In this section, we will further show the advantages of our models through some quintessential examples. In Table 4, we provide the recommendation lists generated by two strong competitors (i.e., IRGAN and RNN) as well as the proposed PLASTIC-V4 model for two users who are randomly selected from the Netflix-Full dataset. Our model can rank the positive movies in higher positions than compared methods. For example, some emerging movies such as "*Session 9*" and "*The Last Valley*" that are truly attractive to the first user have been recommended by our models, whereas they are ignored by the compared methods. In fact, we can include all positive movies in the top-10 list for the first user and in the top-3 list for the second user.

6 Conclusion

In this paper, we proposed a novel adversarial process for top-n recommendation. Our models incorporate both matrix factorization and recurrent neural network to exploit the benefits of the long-term and short-term knowledge. Experiments on two real-life datasets showed the performance superiority of our models for top-n recommendation.

References

[Bobadilla *et al.*, 2013] Jesus Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutierrez. Recommender systems survey. *Knowledge-based systems*, 46:109–132, 2013.

[Chen *et al.*, 2017] Tseng-Hung Chen, Yuan-Hong Liao, Ching-Yao Chuang, Wan-Ting Hsu, Jianlong Fu, and Min Sun. Show, adapt and tell: Adversarial training of cross-domain image captioner. *arXiv preprint arXiv:1705.00930*, 2017.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[He *et al.*, 2016] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*, pages 549–558. ACM, 2016.

[Hernando *et al.*, 2016] Antonio Hernando, Jesús Bobadilla, and Fernando Ortega. A non negative matrix factorization for collaborative filtering recommender systems based on a bayesian probabilistic model. *Knowledge-Based Systems*, 97:188–202, 2016.

[Hidasi *et al.*, 2015] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *ICLR*, 2015.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

[Koren, 2010] Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[Liu and Yang, 2008] Nathan N Liu and Qiang Yang. Eigenrank: a ranking-oriented approach to collaborative filtering. In *SIGIR*, pages 83–90. ACM, 2008.

[Liu *et al.*, 2018] Linqing Liu, Yao Lu, Min Yang, Qiang Qu, Jia Zhu, and Hongyan Li. Generative adversarial network for abstractive text summarization. In *AAAI*, 2018.

[Lu *et al.*, 2015] Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang, and Guangquan Zhang. Recommender system application developments: a survey. *Decision Support Systems*, 74:12–32, 2015.

[McNee *et al.*, 2006] Sean M McNee, John Riedl, and Joseph A Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI*, pages 1097–1101. ACM, 2006.

[Mirza and Osindero, 2014] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[Mnih and Salakhutdinov, 2008] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264, 2008.

[Qiang *et al.*, 2013] Runwei Qiang, Feng Liang, and Jianwu Yang. Exploiting ranking factorization machines for microblog retrieval. In *CIKM*, pages 1783–1788. ACM, 2013.

[Reed *et al.*, 2016] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *ICML*, pages 1060–1069. JMLR.org, 2016.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[Rennie and Srebro, 2005] Jasson DM Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *ICML*, pages 713–719. ACM, 2005.

[Srebro *et al.*, 2005] Nathan Srebro, Jason Rennie, and Tommi S Jaakkola. Maximum-margin matrix factorization. In *NIPS*, pages 1329–1336, 2005.

[Sutton *et al.*, 2000] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, pages 1057–1063, 2000.

[Tu *et al.*, 2015] Wenting Tu, David W Cheung, Nikos Mamoulis, Min Yang, and Ziyu Lu. Activity-partner recommendation. In *PAKDD*, pages 591–604, 2015.

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

[Wu *et al.*, 2016a] Caihua Wu, Junwei Wang, Juntao Liu, and Wenyu Liu. Recurrent neural network based recommendation for time heterogeneous feedback. *Knowledge-Based Systems*, 109:90–103, 2016.

[Wu *et al.*, 2016b] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, and Alexander J Smola. Joint training of ratings and reviews with recurrent recommender networks. *ICLR*, 2016.

[Wu *et al.*, 2017] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. Recurrent recommender networks. In *WSDM*, pages 495–503. ACM, 2017.

[Yu *et al.*, 2017] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858, 2017.

[Yuan *et al.*, 2016] Fajie Yuan, Guibing Guo, Joemon M Jose, Long Chen, Haitao Yu, and Weinan Zhang. Lambdafm: learning optimal ranking with factorization machines using lambda surrogates. In *CIKM*, pages 227–236. ACM, 2016.