Adversarial Binary Collaborative Filtering for Implicit Feedback

Haoyu Wang,[1] **Nan Shao,**[1] **Defu Lian**[*2,3]
[1]School of Computer Science and Engeering, University of Electronic Science and Technology of China
[2]School of Computer Science and Technology, University of Science and Technology of China
[3]School of Data Science, University of Science and Technology of China
{haoyu.uestc,shaonan.uestc,dove.ustc}@gmail.com

Abstract

Fast item recommendation based on implicit feedback is vital in practical scenarios due to data-abundance, but challenging because of the lack of negative samples and the large number of recommended items. Recent adversarial methods unifying generative and discriminative models are promising, since the generative model, as a negative sampler, gradually improves as iteration continues. However, binary-valued generative model is still unexplored within the min-max framework, but important for accelerating item recommendation. Optimizing binary-valued models is difficult due to non-smooth and non-differentiable. To this end, we propose two novel methods to relax the binarization based on the error function and Gumbel trick so that the generative model can be optimized by many popular solvers, such as SGD and ADMM. The binary-valued generative model is then evaluated within the min-max framework on four real-world datasets and shown its superiority to competing hashing-based recommendation algorithms. In addition, our proposed framework can approximate discrete variables precisely and be applied to solve other discrete optimization problems.

Introduction

Nowadays the recommendation system has become more and more important for a lot of companies(*e.g.*,Amazon, Facebook, Netflix) to help their customers find their desirable products to purchase. Traditional recommender systems aim to predict ratings and recommend items based on explicit ratings.

However, explicit ratings are not always available in many cases. Implicit feedback is more common and abundant, *e.g.*, purchase history, mouse activities and users' video viewing (Bennett, Lanning, and others 2007). So how to utilize implicit feedback is a problem needing to be solved. However, compared with explicit feedback, implicit feedback is more difficult to utilize because lack of negative feedback (Pan et al. 2008). Some work such as WR-MF (Hu, Koren, and Volinsky 2008), BPR (Rendle et al. 2009), LambdaFM (Yuan et al. 2016) etc based on matrix factorization (Koren and Bell 2015) achieved great performance in solving this problem. Recently, some work used GAN (Generative Adversarial Networks) (Goodfellow et al. 2014) to get high-quality negative samples. IRGAN (Wang et al. 2017) which is based on a minimax game and matrix factorization model is one of their representative works. By generating high-quality negative samples, IRGAN is the state-of-art algorithm for implicit feedback tasks.

Because the preferences of customers may change constantly, recommender systems need update in time, which means the efficiency of online recommendation is important. Unfortunately, latent factor models, as generative models in IRGAN, has a critical efficiency bottleneck in top-K task that is almost the most common and important recommendation task today. If there are M users and N items, and the dimension of latent space is k, the time complexity of recommendation is $\mathcal{O}(MNk + MN\log K)$ to extract top-K desirable items for every user because it needs to compute users' preference for all items and rank the preference. To solve this bottleneck, hash technique is applied in recommendation. Hash technique, encoding real-valued vectors compact binary codes(*e.g.*,{0,1},{1,-1}), can provide an efficient way to compute preference because inner product can be computed efficiently by bit operation. It can also be used to find approximate top-K items in sublinear or logarithmic time (Wang, Kumar, and Chang 2012; Muja and Lowe 2009).

To make adversarial-based recommendation for implicit feedback effectively and efficiently, we propose an adversarial binary collaborative filtering framework (ABinCF) following IRGAN. However, binary constraints make the learning generally NP-hard (Håstad 2001). To solve this discrete optimization problem, some work used two-stage methods like BCCF (Zhou and Zha 2012), PPH (Zhang et al. 2014) and CH (Liu et al. 2014)) and some directly learnt binary codes like Discrete Collaborative Filtering (DCF) (Zhang et al. 2016). However, the two-stage methods lead to a large quantization loss (Zhang et al. 2016), and DCF optimizes binary codes bit by bit, which is a local search method (Hromkovič 2013). It searches neighborhoods with the distance of one. So DCF is easy to fall into local optima because our objective function is non-convex and nonlinear. And a recent work (Courbariaux et al. 2016) did optimization by estimating discrete gradient, which use sign function as feed-forward pass and a hard tanh as back-propagation. It may be hard to converge asthe feed-forward

*Corresponding author

pass and back-propagation pass using different activation functions (Cao et al. 2017). To address their problem, inspired by recent continuous methods (Allgower and Georg 2012; Cao et al. 2017; Song 2017), we design **ABinCF-erf** using error function to approximate sign function which can result in a sequence of optimization problems converging to the original problem. At the same time, some work (Maddison, Mnih, and Teh 2016; Jang, Gu, and Poole 2016; Li et al. 2018) approximated Bernoulli distribution by Gumbel-softmax trick. (Maddison, Mnih, and Teh 2016; Jang, Gu, and Poole 2016) add noise $\ln(-\ln(U))$ and (Li et al. 2018) adds noise $\ln(U/(1-U))$, where U follows $\mathrm{Uniform}(0,1)$. However, most values of their noise are too large and they even play a more important role in training processing than user and item latent vectors, which is shown in Figure 1. Therefore, we propose ABinCF-Gn which provides a general Gumbel-softmax method to control the value of noise and approximate Bernoulli distribution with any degree of accuracy.

Our contributions are summarized as follows:

- We propose an adversarial binary collaborative filtering framework for implicit feedback to do accurate and fast recommendation.
- We develop two effective discrete optimization algorithms by approximating sign function and Bernoulli distribution with high accuracy.
- Through extensive experiments performed on four real-world datasets, we show the superiority of the proposed algorithm to the state-of-the-arts.

Related Work

In this section we review some work related to our task including adversarial collaborative filtering and recent hashing-based collaborative filtering methods.

Adversarial Collaborative Filtering

How to generate negative samples effectively is crucial for implicit feedback. Some work used adversarial collaborative filtering to solve it such as IRGAN, APR (He et al. 2018), ACAE (Yuan, Yao, and Benatallah 2018), MNRN-GAN (Wang et al. 2018), etc. (1)IRGAN proposed a minimax game to optimize both models iteratively. Their discriminative model mined signals from labelled and unlabelled data to guide generative model and the generative model generated different examples to fool discriminative model in an adversarial way to minimise its discriminative objective. (2)APR enhanced the pairwise ranking method BPR by performing adversarial training. It played a minimax game where the minimization of BPR objective function defended an adversary. The adversary added adversarial noise to maximize BPR objective function at the same time. (3)ACAE proposed a general adversarial training framework for neural network-based recommendation models to improve model robustness and performance and made a trade-off between them. (4)MNRN-GAN designed a streaming recommender model based on neural memory networks and an adaptive negative sampling framework based on GAN to optimize the streaming recommender model.

Discrete Hashing for Collaborative Filtering

An early work was based on Locality-Sensitive Hashing to generate binary codes for Google News readers by their click history (Das et al. 2007). Later, (Karatzoglou, Smola, and Weimer 2010) mapped user/item latent representation learned from MF into Hamming space to get hash codes. Following this, some two-stage methods were proposed which relaxed binary constraints firstly and then did binary quantization (Zhou and Zha 2012; Zhang et al. 2014). However, according to (Zhang et al. 2016),these two-stage methods incur large quantization loss. Hence DCF proposed a method to optimize binary codes directly. Unfortunately, these algorithms were designed for rating prediction which had less wide application range than implicit feedback.

Adversarial Binary Collaborative Filtering

Preliminaries

Adversarial Collaborative Filtering Objective Function The original IRGAN combined objective function (Wang et al. 2017) is

$$J^{G^*,D^*} = \min_{\theta}\max_{\phi}\sum_{n=1}^{N}(\mathbb{E}_{d\sim p_{true}(d|q_n,r)}[\ln D(d|q_n)] + \mathbb{E}_{d\sim p_\theta(d|q_n,r)}[\ln(1-D(d|q_n))])$$

where d means documents and q means queries. When optimizing generative retrieval G^*, it is difficult to optimize it directly by gradient descent because the sample of d is discrete. It uses policy gradient to compute gradient (Williams 1992):

$$\begin{aligned}&\nabla_\theta J^G(q_n)\\ =&\sum_{i=1}^{M}\nabla_\theta p_\theta(d_i|q_n,r)\ln(1+\exp(f_\phi(d_i,q_n)))\\ =&\mathbb{E}_{d\sim p_\theta(d|q_n,r)}[\nabla_\theta \ln p_\theta(d|q_n,r)\ln(1+\exp(f_\phi(d_i,q_n)))]\end{aligned}$$

When it is used to item recommendation, the d means items and q means users.The model of G is $p_\theta(j|i) = \mathrm{softmax}(g_\theta(i,j)/\tau)$ and the model of D is $D(j|i) = \sigma(g_\phi(i,j))$, where $g_\theta(i,j)$, $g_\phi(i,j)$ are scoring functions and i, j denote the ith user and the jth item. A widely adopted approach for recommendation is matrix factorization, so the scoring function is set as $s(i,j) = b_j + \mathbf{u}_i^T\mathbf{v}_j$, where b_j is the bias term for item j and $\mathbf{u}_i$, $\mathbf{v}_j \in \mathbb{R}^k$ are the latent vectors of user i and item j. So, the combined objective function for item recommendation is

$$\begin{aligned}D^* &= \max_{\phi}\sum_{i=1}^{M}(\mathbb{E}_{k\sim p_{true}(j|i)}[\ln\sigma(s_\phi(i,j))] + \mathbb{E}_{k\sim p_\theta(j|i)}[\ln(1-\sigma(s_\phi(i,j)))])\\ G^* &= \min_{\theta}\sum_{i=1}^{M}\sum_{k\sim p_\theta(j|i)}\ln p_\theta(j|i)\ln(1-\sigma(s_\phi(i,j)))\end{aligned}$$

Problem Formulation

After having the latent representation of users and items, generating top-K preferred items for each user is considered

as a similarity-based retrieval problem. However, if the similarity is computed by inner product and the top-K items are extracted through the max-heap structure, the scheme costs $\mathcal{O}(Nk + NlogK)$. When N is large, it will lead to crucial low-efficiency issues.

If latent vectors are represented as binary codes, the similarity-based search can be accelerated by computing the inner product much more efficiently via the Hamming distance. Denoting $\mathbf{u}_i \in \{1, -1\}^k$ and $\mathbf{v}_j \in \{1, -1\}^k$ as the binary codes of user u and item i, the inner product is represented as $\mathbf{u}_i^T \mathbf{v}_j = k - 2H(\mathbf{u}_i, \mathbf{v}_j)$, where $H(\cdot)$ denotes the Hamming distance between binary codes. Particularly, if denoting $\mathbf{u}_i \in \{0, 1\}^k$ and $\mathbf{v}_j \in \{0, 1\}^k$, the the inner product is represented as $\mathbf{u}_i^T \mathbf{v}_j = \text{popcount}(k - 2\text{xor}(\mathbf{u}_i, \mathbf{v}_j))$. Hamming distance can be computed extremely efficiently by fast bit operations.

Following IRGAN which uses generative model to do recommendation, we try to make the parameters of generative model be binary codes while the parameters of discriminative model are still continuous to keep accuracy. However, the bias term is not binary in generative model because the bias term is not involved in inner product and if it is continuous, it can help reduce accuracy loss of the generative model. The combined objective function is

$$
\begin{aligned}
D^* = \max_{\phi} \sum_{i=1}^{M} (&\mathbb{E}_{k \sim p_{true}(j|i)}[\ln \sigma(s_\phi(i,j))] \\
&+ \mathbb{E}_{k \sim p_\theta(j|i)}[\ln(1 - \sigma(s_\phi(i,j)))]) + \lambda \|\phi\|_{\mathcal{L}_2} \\
G^* = \min_{\theta} \sum_{i=1}^{M} &\sum_{k \sim p_\theta(j|i)} \ln p_\theta(j|i) \ln(1 - \sigma(s_\phi(i,j))) \\
&s.t. \theta = H(\theta^*)
\end{aligned}
$$

where θ^* is real-valued and $H(\cdot)$ is a hash function which make θ^*(except bias) be binary codes, $\|\phi\|_{\mathcal{L}_2}$ is the $\mathcal{L}_2$ regularization of ϕ and λ is a coefficient of the regularization.

Train ABinCF

To get adversarial binary collaborative filtering recommender system, we propose **ABinCF** algorithm based on two continuation methods: one uses $\text{erf}(\cdot)$ to relax binary constraint and the other one uses a general gumbel-softmax method to approximate. The whole training processing is shown in Algorithm 1. Then we explain the two algorithms respectively.

Error Function Relaxation We try to optimize the objective function while $\boldsymbol{v}_i$ is binary, since it is difficult to use optimization method based on gradient to solve the problem directly. To address the discrete optimization problem, we relax the discrete objective function into a continuous function which is easy to optimize. When we adjust the value of the parameter gradually which controls the degree of relaxation, the original optimization problem is replaced with a series of continuation optimization problems which can converge to the original problem (Cao et al. 2017). The relaxation method is based on the following equation:

$$\lim_{\beta \to 0^+} \text{erf}(x/\beta) = \text{sign}(x)$$

Proof. Consider the density function of normal distribution $f(x) = \frac{1}{\sqrt{2\pi}\beta} e^{-x^2/(2\beta^2)}$, the following equation holds:

$$\lim_{\beta \to 0^+} \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}\beta} e^{-x^2/(2\beta^2)} dx = \lim_{\beta \to 0^+} \frac{1}{2}(\text{erf}(\frac{z}{\beta}) + 1)$$

According to Dominated convergence theorem (Royden and Fitzpatrick 1988), we get

$$\lim_{\beta \to 0^+} \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}\beta} e^{-\frac{x^2}{2\beta^2}} dx = \int_{-\infty}^{z} \lim_{\beta \to 0^+} \frac{1}{\sqrt{2\pi}\beta} e^{-\frac{x^2}{2\beta^2}} dx$$

When $\beta \to 0^+$, if $x = 0$, then $f(x) = +\infty$. And $f(x) = 0$ if $x \neq 0$. Therefore the equality holds.

When we reduce the value of β, $\text{erf}(x/\beta)$ can approximate $\text{sign}(x)$by any precision. In other words, we can decrease the value of β gradually to approximate binary codes. Based on the above conclusion, we design the following algorithm for optimizing generative models. In the beginning, we use $\text{erf}(x)$ to train the generative model because it is easiest to train by this relaxation function. Then the value of β will decrease afterwards and train generative model until convergence. Particularly, we name this algorithm as **ABinCF-erf** for short in this paper.

General Gumbel-softmax Method Gumbel-softmax trick (Maddison, Mnih, and Teh 2016; Jang, Gu, and Poole 2016) is an efficient method to approximate discrete distribute and it is widely used in stochastic computational graphs. However, they are not robust to solve this problem. Considering that the noise added in these two methods, if the value of U is close to 0 or 1, the noise will tend to ∞. If we want to control the range of noise, let $|\text{noise}| < \varepsilon$ and the range of U is extremely narrow. To solve this problem, we propose a general Gumbel-softmax method firstly, and then choose Gaussian distribution as the noise to avoid the value of noise being too large. We list three methods approximating Bernoulli distribution based on Gumbel-softmax trick in Table 1, where $\alpha \in \mathbb{R}$, $\tau > 0$, $x \sim N(0, \sigma^2)$ and $U \sim \text{Uniform}(0, 1)$.

Table 1: Two Gumbel-based approximation methods

Method	Approximation function	Noise
Gumbel-softmax	$G(\alpha, \tau) = \sigma(\frac{\alpha - \ln(-\ln(U))}{\tau})$	$\ln(-\ln(U)$
G^2-LSTM	$G(\alpha, \tau) = \sigma(\frac{\alpha + \ln(\frac{U}{1-U})}{\tau})$	$\ln(\frac{U}{1-U})$
ABinCF-Gn	$G(\alpha, \tau) = \sigma((\alpha - x)/\tau)$	x

Theorem 1 Assume $\sigma(\cdot)$ is the sigmoid function. Given the $\tau > 0$ and $\alpha \in \mathbb{R}$, we define a random variable $D \sim B(F(\alpha))$ where $F(\cdot)$ is the CDF of a particular distribution and $G(\alpha, \tau) = \sigma(\frac{\alpha - x}{\tau})$ where x is a random variable and its CDF is $F(\cdot)$. If $F(\cdot)$ is $\rho-$ Lipschitz continuous, then for any $\varepsilon \in (0, 1/2)$ the following two inequalities hold:

$$
\begin{aligned}
P(D = 1) - \rho\tau \ln(1/\varepsilon) &\leq P(G(\alpha, \tau) \geq 1 - \varepsilon) \\
&\leq P(D = 1) \\
P(D = 0) - \rho\tau \ln(1/\varepsilon) &\leq P(G(\alpha, \tau) \leq \varepsilon) \\
&\leq P(D = 0)
\end{aligned}
$$

Proof. Because the proof of the two inequalities is almost the same, we just prove the first one.

Firstly, computing the following probability and we have

$$\begin{aligned} P(G(\alpha,\tau) \geq 1-\varepsilon) =& P(x \leq \alpha - \tau \ln(1/\varepsilon - 1)) \\ =& F(\alpha - \tau \ln(1/\varepsilon - 1)) \end{aligned}$$

Considering that $F(\cdot)$ is $\rho-$ Lipschitz continuous and it is monotonically increasing, we get

$$\begin{aligned} & P(D=1) - P(G(\alpha,\tau) \geq 1-\varepsilon) \\ =& F(\alpha) - F(\alpha - \tau \ln(1/\varepsilon - 1)) \\ \leq& \rho\tau \ln(1/\varepsilon - 1) \\ \leq& \rho\tau \ln(1/\varepsilon) \end{aligned} \tag{1}$$

and $P(D=1) - P(G(\alpha,\tau) \geq 1-\varepsilon) \geq 0$. So, the inequality holds.

Corollary 1 Given the $\tau > 0$ and $\alpha \in \mathbb{R}$, we define a random variable $D \sim B(F(\alpha))$ where $F(\cdot) \sim N(0,\sigma^2)$ and $G(\alpha,\tau) = \sigma(\frac{\alpha - x}{\tau})$ where $x \sim N(0,\sigma^2)$. For any $\varepsilon \in (0, 1/2)$ the following two inequalities hold:

$$\begin{aligned} P(D=1) - \rho\tau ln(1/\varepsilon) \leq& P(G(\alpha,\tau) \geq 1-\varepsilon) \\ \leq& P(D=1) \\ P(D=0) - \rho\tau ln(1/\varepsilon) \leq& P(G(\alpha,\tau) \leq \varepsilon) \\ \leq& P(D=0) \end{aligned}$$

where $\rho = 1/(\sqrt{2\pi}\sigma)$.
So, when $\tau \to 0^+$, we have

$$\begin{aligned} P(\lim_{\tau \to 0^+} G(\alpha,\tau) = 1) = P(D=1) \\ P(\lim_{\tau \to 0^+} G(\alpha,\tau) = 0) = P(D=0) \end{aligned}$$

This means that $G(\alpha,\tau)$ is an approximation of $\mathrm{Bernoulli}(F(\alpha))$ and the rate of convergence is showed by Eqn. (1).

In the sequel, we apply this Normal distribution Gumbel-softmax method to solve problems with binary constraint. We will explain the reason why choose normal distribution rather than any other distributions and how to use this method. First of all, normal distribution satisfies "three-sigma rule of thumb", which means the value of most samples are in the range $(-3\sigma, 3\sigma)$, so we can control the value of noise by controlling the value of the variance. Then we replace $\alpha \in \{0,1\}$ with $\sigma((\alpha - x)/\tau)$ where $\alpha \in \mathbb{R}$ to relax binary variables. Specifically, in matrix factorization model, the $\mathbf{u}_i$ is a k dimension vector, so we use the following way to solve it: Given $\boldsymbol{\alpha} \in \mathbb{R}^k$ and $\tau \geq 0$, $G(\alpha,\tau) = \sigma((\boldsymbol{\alpha} - \boldsymbol{x})/\tau)$ where $\boldsymbol{x}$ is a random vector which every element $\boldsymbol{x}_i$ is sampled independently from $N(0,\sigma^2)$, where $i = 1, 2, 3, ..., k$. If we set τ as a small value or we decrease the value of τ gradually, we can approximate binary vector well by any optimization methods based on gradient like SGD, Adam and so on. In the following section, we choose normal distribution and we call this method **ABinCF-Gn** for short.

Experiments

In this section, we evaluate our proposed hashing framework with the aim of answering the following research questions.

Algorithm 1: ABinCF Algorithm

Input: A sequence of temperature;generator $p_\theta(i_k|u_n)$;discriminator $f_\phi(u_n, i_k)$;training dataset $\mathcal{S}$
Output: Generative model with $H(x)$ as hash function

```
Initialize weights θ, φ randomly.
Pre-train p_θ(i_k|u_n), f_φ(u_n, i_k) by S
repeat
    Train Generative model with R(·) as relaxation function;
    Set converged Generative model as next Generative model initialization;
    Decrease temperature;
    Train Discriminator model;
until Convergence;
```

1. Does the recommendation performance of the proposed ABinCF outperform the state-of-the-art hashing-based recommender systems?
2. Whether our proposed Normal Gumbel-softmax method can approximate Bernoulli distribution well? And whether our method is more effective than Gumbel and G^2-LSTM method?
3. How the temperature setting influences the results?
4. How about the advantage of hashing for online recommendation over real-based frameworks?

We introduce the experimental settings firstly and then answer the above questions in following sections.

Experiment Settings

In this section, we introduce datasets used in our experiments in the beginning. Then we introduce five important baselines, followed by the introduction of evaluation metric.

Datasets We use the four public available datasets from various real-word online websites to evaluate the proposed algorithm.

MovieLens datasets are collected by the GroupLens Research Project at the University of Minnesota, and MovieLens100k and MovieLens10M are two of them. There are originally 10,000,054 ratings from 0.5 to 5 with 0.5 interval from 71,567 users on 10,681 items in MovieLens10M and there are 100,000 ratings from 1 to 5 from 943 users on 1,683 items in MovieLens100k.

Amazon contains user rating and reviews on Amazon of 24 product categories and we evaluate our method on one of the largest product categories, Book dataset. It includes 1,732,060 ratings from 35,151 users on 33,195 items.

Yelp is the latest Yelp challenge dataset. The scores are integers from 1 to 5. And Yelp dataset includes 2,685,066 ratings from 409,117 users and 85,539 items.

Due to the extreme sparsity of MovieLens10M, Amazon and Yelp original datasets, we remove users who have less than 20 ratings and remove items which are rated by less than 20 users. Because MovieLens100k is not very sparse, we don't filter it. Since the proposed algorithm is suitable

for implicit feedback, we follow BPR (Rendle et al. 2009) to convert rating data into implicit feedback. Particularly, we set ratings greater than 3.5 as positive feedback. Table 2 summarizes the filtered datasets. For each user, we randomly sampled 50% ratings as training as the rest as testing. We repeated for five random splits and reported the averaged results.

Table 2: Statistics of datasets

Dataset	#User	#Item	#Rating	Density
MovieLens100k	943	1,683	100,000	6.30%
MovieLens10M	69,838	8,939	9,983,758	1.60%
Book	35,151	33,195	1,732,060	0.15%
Yelp	13,679	12,922	640,143	0.36%

Comparison Methods For hashing-based recommender system, we compare ABinCF with 3 very popular and state-of-art methods:DCF, PPH, BCCF. Note that DCF tackles a discrete optimization problem directly which is subject to de-correlated and balanced constraints for seeking compact and informative binary codes for users and items. PPH and BCCF are both two-stage method to learn hash code. BCCF is a Binary Code learning method for Collaborative Filtering and PPH is a two-stage Preference Preserving Hashing. However, they haven't been designed for implicit feedback.

For real-based recommender system, we only compare the latest and state-of-art method: IRGAN. It uses generative and discriminative information retrieval models to do recommendation and it is especially designed for implicit feedback problems, which outperforms almost other continuous models including BPR (Rendle et al. 2009), LambdaFM (Yuan et al. 2016).

Evaluation Metric We evaluate the recommendation system performance by a widely used ranking based metric: NDCG(Normalized Discounted Cumulative Gain) and Precision. NDCG is the normalization of DCG(Discounted Cumulative Gain) which can measure the ranking quality. Precision is the ratio of the number of relevant results to the total number of results. In our experiment, we predicted the top-K preference items for each user from testing datasets.

Parameter Settings In our experiments, we set $\beta = \max(\exp(-\text{epoch}), 0.01)$ for ABinCF-erf on all datasets and $\tau = \exp(-0.1 * \text{epoch})$ on MovieLens100K, $\tau = 0.9$ on other datasets for ABinCF-Gn. For all datasets, we set σ of ABinCF-Gn as 0.01, set the dimension of latent factor k as 16 and set λ as $0.1/\text{batchsize}$. We set the sampling temperature to 0.2, and the number of generated relevant items to 5 for policy gradient and the number of negative items to the number of positive ones for discriminative learning for IRGAN and ABinCF on all datasets following (Wang et al. 2017).

Besides, for other baselines, we held-out evaluation method on randomly splits of training data to tune the optimal hyper-parameters for them by grid search. The settings of them are listed in Table 3.

Table 3: Parameter Settings

	DCF		BCCF	PPH
	α	β	λ	λ
MovieLens100k	1	10	0.09	16
MovieLens10M	0.001	10	0.09	16
Book	10	10	0.01	16
Yelp	0.001	10	0.01	8

Comparison with Baselines

Although hashing recommendation has significant advantages of time and storage over real-valued recommendation, it often suffers from low recommendation accuracy because binary codes lose a lot of information compared with real-valued codes due to the discrete constraints. ABinCF is proposed to improve the recommendation accuracy.

In this part, we will answer the first question at the beginning of the experiment section. The recommendation accuracy comparisons including Precision@10 and NDCG@10 are shown in Table 4, Table 5, Table 6 and Table 7.

Compared with other hash-based recommendation, the performance of ABinCF-Gn far surpasses all other hash algorithms including DCF, BCCF and PPH. And ABinCF-erf has huge advantages over other hash-based algorithms on Amazon, Yelp and MovieLens10M dataset while the value of Precision@100 is a little smaller than PPH on MovieLens100k. Because Amazon, Yelp and MovieLens10M are sparser than MovieLens100k, so our model based on adversarial collaborative filtering can show more advantages on generating high quality negative samples which are significant in implicit feedback tasks. So ABinCF-erf achieves the best performance about Precision@100 and NDCG@100 on Amazon. ABinCF-Gn has better performance than others including ABinCF-erf on MovieLens and Yelp because Normal Gumbel-softmax method makes it approximate Bernoulli distribution well. In addition, DCF, BCCF and PPH are designed to make binary codes for explicit recommendation, and BCCF is a two-stage method and PPH is based on quantization method while ABinCF relax sign function directly, so ABinCF has much better results.

Compared with the state-of-art real-valued recommendation algorithm IRGAN, precision loss of ABinCF-erf is less than 30% in almost datasets while the gap between ABinCF-erf and IRGAN is larger than that between ABinCF-Gn and IRGAN. Because IRGAN is real-based method, it obtains more information from real-valued codes. And ABinCF approximates both in sampling of policy gradient and making hash codes, so IRGAN performs better than ours.

The Effectiveness of Normal Gumbel-softmax Method

We next show the effectiveness of Normal Gumbel-softmax method as follows:

(1)Normal Gumbel-softmax method can approximate Bernoulli distribution greatly. We set $\tau = 0.001$, $\alpha = 0$, which means 0 and 1 have same probability according to $F(0) = 0.5$. Then we sample $x \sim N(0, 1)$, $x \sim N(0, 0.1)$

Table 4: Item Recommendation Results(MovieLens100k)

	Precision@3	**Precision@10**	**Precision@100**	**NDCG@3**	**NDCG@10**	**NDCG@100**
IRGAN	0.4396	0.3627	0.1578	0.4542	0.4074	0.4807
DCF	0.0964	0.0962	0.0752	0.1135	0.1058	0.1609
BCCF	0.1336	0.1223	0.0974	0.1331	0.1266	0.2071
PPH	**0.1863**	**0.1779**	**0.1065**	**0.1952**	**0.1893**	**0.2853**
ABinCF-erf	0.2327	0.2005	0.1034	0.2387	0.2204	0.2970
ABinCF-Gn	**0.2742**	**0.2192**	**0.1129**	**0.2814**	**0.2466**	**0.3244**

Table 5: Item Recommendation Results(MovieLens10M)

	Precision@3	**Precision@10**	**Precision@100**	**NDCG@3**	**NDCG@10**	**NDCG@100**
IRGAN	0.2576	0.2104	0.0972	0.2646	0.2336	0.2683
DCF	**0.0589**	0.0356	0.0201	**0.0748**	**0.0502**	0.0639
BCCF	0.0328	**0.0358**	**0.0330**	0.0325	0.0348	0.0521
PPH	0.0418	0.0348	0.0244	0.0492	0.0423	**0.0783**
ABinCF-erf	01156	0.0759	0.0273	0.1372	0.1025	0.0840
ABinCF-Gn	**0.2582**	**0.2066**	**0.0590**	**0.2648**	**0.2293**	**0.2043**

Table 6: Item Recommendation Results(Amazon)

	Precision@3	**Precision@10**	**Precision@100**	**NDCG@3**	**NDCG@10**	**NDCG@100**
IRGAN	0.0458	0.0377	0.0194	0.0474	0.0416	0.0629
DCF	0.0113	0.0115	0.0097	0.0109	0.0114	0.0266
BCCF	**0.0142**	**0.0141**	**0.0111**	**0.0140**	**0.0142**	**0.0280**
PPH	0.0065	0.0038	0.0030	0.0065	0.0091	0.0090
ABinCF-erf	0.0334	0.0263	**0.0156**	0.0351	0.0295	**0.0479**
ABinCF-Gn	**0.0352**	**0.0274**	0.0143	**0.0364**	**0.0307**	0.0457

Table 7: Item Recommendation Results(Yelp)

	Precision@3	**Precision@10**	**Precision@100**	**NDCG@3**	**NDCG@10**	**NDCG@100**
IRGAN	0.0873	0.0705	0.0356	0.0896	0.0800	0.1475
DCF	**0.0102**	**0.0104**	**0.0095**	0.0098	**0.0107**	**0.0325**
BCCF	0.0101	0.0094	0.0093	**0.0100**	0.0100	0.0307
PPH	0.0077	0.0078	0.0068	0.0079	0.0083	0.0240
ABinCF-erf	0.0474	0.0340	0.0218	0.0434	0.0384	0.0831
ABinCF-Gn	**0.0510**	**0.0422**	**0.0223**	**0.0520**	**0.0472**	**0.0893**

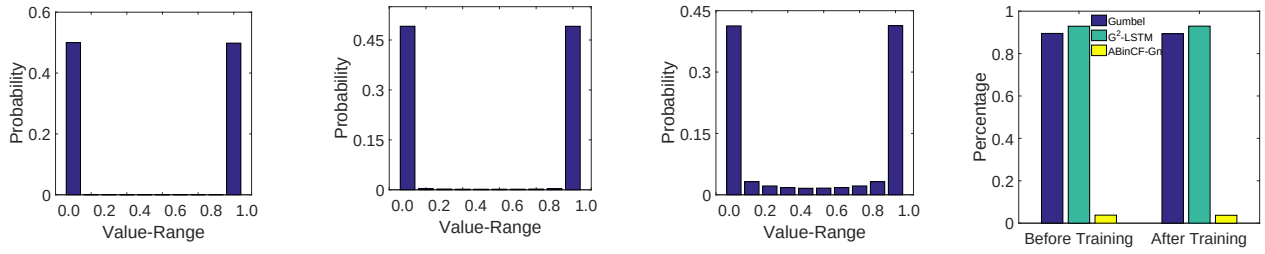
and $x \sim N(0, 0.01)$ 1M times respectively and compute $G = \sigma(\frac{\alpha - x}{\tau})$. The value interval of G is shown in Figure 1. In case of $\sigma = 1$, $\sigma = 0.1$ or $\sigma = 0.01$, it is clear to find that our method can approximate well.

(2)Normal Gumbel-softmax method is more effective than Gumbel noise and G^2-LSTM noise. We count the number of elements in $|\mathbf{u}_i|$ less than absolute value of Gumbel-based noise and ABinCF-Gn noise at the beginning and the end of the first training on MovieLens100k. The result is shown in Figure 1 where the left part is the beginning of first training and the right part is the end of the first training. If noise is larger than parameters needed updating, it will have a greater impact on the gradient. From Figure 1, noise of Gumbel-based method is too large and ours can update parameters much more effectively. Therefore, this experiment verifies the analysis in the part of General Gumbel-softmax method.

The Setting of Temperature

In this section, we answer the third question. We show the value of Precision@10 and NDCG@10 via iterations with different ways to set temperature in ABinCF-erf and AbinCF-Gn. In this experiment, we set the number of epoch as 15 and the number of iteration within each epoch as 20. We test temperature=$\exp(-0.01 * \text{epoch})$, temperature=$\exp(-0.1 * \text{epoch})$, temperature=$\max(\exp(-\text{epoch}), 0.01)$ and temperature=0.1. We did this experiment in MovieLens100k.

The experiment results are shown in Figure 2. The two



(a) (b)

Figure 1: (a):The statistics of interval of random variables using the Normal Gumbel-softmax method with different values of σ. Left.$\sigma = 1$.Middle.$\sigma = 0.1$.Right.$\sigma = 0.01$. (b):The percentage of the absolute value of user latent vectors smaller than the absolute value of stochastic noise by three methods before and after the first training

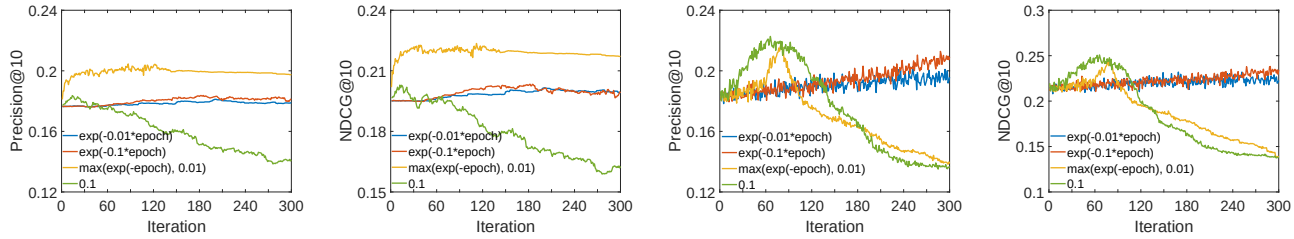


Figure 2: Left 1.Precision@10 vs Temperature(ABinCF-erf);Left 2.NDCG@10 vs Temperature(ABinCF-erf);Right 1.Precision@10 vs Temperature(ABinCF-Gn);Right 2.NDCG@10 vs Temperature(ABinCF-Gn)

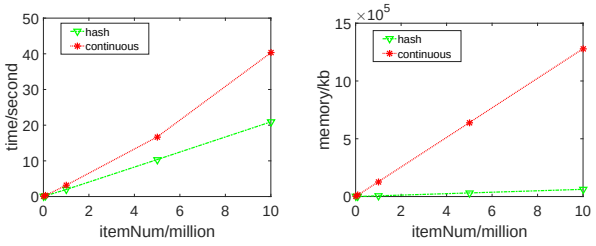


Figure 3: Time and Storage Cost vs Item Size

left figures of Figure 2 shows Precision@10 and NDCG@10 of ABinCF-erf. It is clear to observe that when temperature is set as temperature=$\max(\exp(-\text{epoch}), 0.01)$, it has much better performance than setting temperature 0.1 because such low temperature at the beginning is hard to train, so our method updating temperature is valid. We also find the green line in the two right ones of Figure 2 representing temperature=0.1 achieving the highest results in the first 120 iterations but it drops after. This is because the temperature is too low to update parameters. Besides, the red and blue curves rise during all the training time. Thus we can set temperature as a constant a bit larger than 0.1 or set temperature changes with the increase of epoch.

Efficiency Comparison

According to the analysis before, the significant advantage of hashing method of recommendation over real-based recommendation is the efficiency of online recommendation. We follow PPH to study the efficiency of hashing-based recommendation on the Amazon dataset. However, the number of candidate items is only 33,195, so it is impossible to test efficiency of recommendation in the case of large size of candidate items. To this end, we assume real-valued item latent factors from IRGAN's generative model following multivariable Gaussian distribution, and then sample latent factors of 10K, 100K, 1M, 5M, and 10M items based on the estimated mean and covariance. Some users' latent factors from IRGAN's generative model are used as a query code. Given this synthetic dataset, we compared the efficiency of recommendation and storage cost between real-valued and binary-valued models. The real-valued model directly uses latent factors of the user and items while the binary-valued model exploits the binarzied representation of these latent factors. The effectiveness of hashing-based recommendation compared with real-based recommendation is shown in Figure 3.

Time Complexity The time cost variation over different item sizes is shown in Figure 3. We can conclude that the hashing-based recommendation spends much less time than real-valued recommendation. When the number of users and items becomes much larger which is above millions, the time consuming is extremely huge. Therefore, hashing-based recommendation is much more effective in terms of time than real-based recommendation.

Storage Complexity The storage cost variation over different item sizes is shown in Figure 3. When items become 10M, the storage space is only 1/12 of real-valued features.

The storage superiority of hash codes will show more completely when items become larger.

Conclusions

In this paper, we propose an adversarial-based CF hashing framework ABinCF for implicit feedback. By our proposed learning strategy and approximation methods, ABinCF-erf and ABinCF-Gn both outperform the state-of-the-art hashing-based collaborative filtering algorithms and have small accuracy loss compared to real-based algorithm IR-GAN for implict feedback on four public datasets. Our experiments also show the proposed ABinCF has great advantages in speed of online recommendation and storage. Therefore, it reconciles both effectiveness and efficiency.

Acknowledgements

This research is supported by the National Natural Science Foundation of China (Grant No. 61502077 and 61631005).

References

Allgower, E. L., and Georg, K. 2012. *Numerical continuation methods: an introduction*, volume 13. Springer Science & Business Media.

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

Cao, Z.; Long, M.; Wang, J.; and Philip, S. Y. 2017. Hashnet: Deep learning to hash by continuation. In *ICCV*, 5609–5618.

Courbariaux, M.; Hubara, I.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*.

Das, A. S.; Datar, M.; Garg, A.; and Rajaram, S. 2007. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, 271–280. ACM.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680.

Håstad, J. 2001. Some optimal inapproximability results. *Journal of the ACM (JACM)* 48(4):798–859.

He, X.; He, Z.; Du, X.; and Chua, T.-S. 2018. Adversarial personalized ranking for recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 355–364. ACM.

Hromkovič, J. 2013. *Algorithmics for hard problems: introduction to combinatorial optimization, randomization, approximation, and heuristics*. Springer Science & Business Media.

Hu, Y.; Koren, Y.; and Volinsky, C. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, 263–272. Ieee.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Karatzoglou, A.; Smola, A.; and Weimer, M. 2010. Collaborative filtering on a budget. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 389–396.

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

Li, Z.; He, D.; Tian, F.; Chen, W.; Qin, T.; Wang, L.; and Liu, T.-Y. 2018. Towards binary-valued gates for robust lstm training. *arXiv preprint arXiv:1806.02988*.

Liu, X.; He, J.; Deng, C.; and Lang, B. 2014. Collaborative hashing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2139–2146.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Muja, M., and Lowe, D. G. 2009. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)* 2(331-340):2.

Pan, R.; Zhou, Y.; Cao, B.; Liu, N. N.; Lukose, R.; Scholz, M.; and Yang, Q. 2008. One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, 502–511. IEEE.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, 452–461. AUAI Press.

Royden, H., and Fitzpatrick, P. 1988. Real analysis, 3'rd edition.

Song, J. 2017. Binary generative adversarial networks for image retrieval. *arXiv preprint arXiv:1708.04150*.

Wang, J.; Yu, L.; Zhang, W.; Gong, Y.; Xu, Y.; Wang, B.; Zhang, P.; and Zhang, D. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, 515–524. ACM.

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. ACM.

Wang, J.; Kumar, S.; and Chang, S.-F. 2012. Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis & Machine Intelligence* (12):2393–2406.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Yuan, F.; Guo, G.; Jose, J. M.; Chen, L.; Yu, H.; and Zhang, W. 2016. Lambdafm: learning optimal ranking with factorization machines using lambda surrogates. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 227–236. ACM.

Yuan, F.; Yao, L.; and Benatallah, B. 2018. Adversarial collaborative auto-encoder for top-n recommendation. *arXiv preprint arXiv:1808.05361*.

Zhang, Z.; Wang, Q.; Ruan, L.; and Si, L. 2014. Preference preserving hashing for efficient recommendation. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, 183–192. ACM.

Zhang, H.; Shen, F.; Liu, W.; He, X.; Luan, H.; and Chua, T.-S. 2016. Discrete collaborative filtering. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 325–334. ACM.

Zhou, K., and Zha, H. 2012. Learning binary codes for collaborative filtering. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 498–506. ACM.