
There may be differences between this version and the published version. You are advised to consult the publisher’s version if you wish to cite from it.


https://eprints.gla.ac.uk/265372/

Deposited on: 15 February 2022

Enlighten – Research publications by members of the University of Glasgow https://eprints.gla.ac.uk
Learning Robust Recommenders through Cross-Model Agreement

Yu Wang
University of Science and Technology of China
wy2001@mail.ustc.edu.cn

Xin Xin
Shandong University
ximxin@sdu.edu.cn

Zaiqiao Meng
Cambridge University
zaiqiao.meng@gmail.com

Joemon M Jose
University of Glasgow
Joemon.Jose@glasgow.ac.uk

Fuli Feng
University of Science of Technology of China
fulifeng93@gmail.com

Xiangnan He
University of Science and Technology of China
xiangnanhe@gmail.com

ABSTRACT

Learning from implicit feedback is one of the most common cases in the application of recommender systems. Generally speaking, interacted examples are considered as positive while negative examples are sampled from uninteracted ones. However, noisy examples are prevalent in real-world implicit feedback. A noisy positive example could be interacted but it actually leads to negative user preference. A noisy negative example which is uninteracted because of user unawareness could also denote positive potential user preference. Conventional training methods overlook these noisy examples, leading to sub-optimal recommendations.

In this work, we propose a general framework to learn robust recommenders from implicit feedback. Through an empirical study, we find that different models make relatively similar predictions on noisy examples. These predictions on noisy examples vary much more across different models. Motivated by this observation, we propose denoising with cross-model agreement (DeCA) which minimizes the KL-divergence between the real user preference distributions parameterized by two recommendation models while maximizing the likelihood of data observation. We instantiate DeCA on four representative recommendation models, empirically demonstrating its superiority over normal training and existing denoising methods. Codes are available at https://github.com/wangyu-ustc/DeCA.

KEYWORDS

Recommender System, Denoised Recommendation, Robust Learning, Implicit Feedback.

1 INTRODUCTION

Recommender systems are widely used for information filtering, especially for various online services such as E-commerce [21], multimedia platforms [3, 32] and social media [1]. Among existing systems, training using implicit feedback (e.g., view and click behaviors) is more prevalent than that using explicit ratings, since implicit feedback is much easier to be collected [14, 27]. Specifically, the interacted user-item samples are assumed to be positive examples while negative examples are sampled from missing interactions.

Under this assumption, the interacted examples are regarded to denote positive user preference (i.e., like) and thus are trained towards higher prediction scores. The sampled missing interactions are considered as negative user preferences (i.e., dislike) and are pushed down in the training procedure. Figure 1(a) illustrates this normal training assumption.

However, this assumption seldom holds for real-world applications since noisy examples are prevalent, as shown in Figure 1(b). For example, an interacted user-item pair may only be attributed to the popularity bias [2], which could actually lead to negative user preference. These interactions can be considered as noisy-positive examples [15, 23, 39]. Besides, a missing interaction could also be attributed to user unawareness because the item is not exposed to this user. Such noisy-negative examples could have signaled potential positive user preferences. The normal training procedure overlooks these noisy examples, leading to the misunderstanding of the real user preference and sub-optimal recommendation [2, 39].

To denoise implicit feedback, some recent efforts have been done by using re-sampling methods [4, 6, 8, 34, 43] or re-weighting methods [35]. Re-sampling methods focus on designing more effective samplers. For example, [8] considers that the missing interactions of popular items are highly likely to be real negative examples.
However, the performance of re-sampling methods depends heavily on the sampling distribution and suffers from high variance [45]. [35] proposed a re-weighting method that assigns lower weights or zeroes weights to high-loss samples since the noisy examples would have higher losses. However, [31] shows that hard yet clean examples also tend to have high losses. As a result, [35] could encounter difficulties distinguishing between hard clean and noisy examples. Besides, some research focuses on utilizing auxiliary information to denoise implicit feedback [18, 22, 24, 42] but these kinds of methods need additional input.

In this work, we propose a general training framework to learn robust recommenders from implicit feedback without introducing external knowledge sources. The signal for denoising comes from an insightful observation: different models tend to make relatively similar predictions for clean examples which represent the real user preference, while the predictions for noisy examples would vary much more among different models. (See figure 2 for more details.) To this end, we propose denoising with cross-model agreement (DeCA) which minimizes the Kullback–Leibler (KL) divergence between the real user preference distributions parameterized by two recommendation models, and meanwhile, maximizes the likelihood of the data observation given the real user preference. DeCA can be considered as a framework that incorporates the weakly supervised signals from different model predictions to denoise the target recommender. The proposed DeCA acts as a robust learning framework and can be naturally incorporated with any recommendation model.

To summarize, the main contributions of this work are as follows:

- We find that the different models tend to make more similar predictions for clean examples than for noisy ones. This observation provides new hints to devise denoising methods for implicit feedback recommendation.

- We propose the robust learning frameworks DeCA which utilizes the agreement across different model predictions as the denoising signal for implicit feedback and then infers the real user preference from corrupted binary data.

- We instantiate DeCA with four classical recommendation models and conduct extensive experiments on four datasets. The results demonstrate the effectiveness of our proposal.

2 MOTIVATION

According to the theory of robust learning, [10, 17], noisy examples tend to have relatively high loss values. However, the hard yet
We calculate all prediction differences on clean examples and noisy examples respectively and then report the average.

Figure 2(a) shows the mean prediction differences between two different models on clean positive examples and noisy positive examples. It’s obvious that the model prediction differences on noisy examples are significantly larger than the differences on clean examples. In other words, different models tend to make agreement predictions for clean examples compared with noisy examples. That is to say, different models tend to fit different parts of the corrupted data but clean examples are the robust agreement component that every model attempts to fit. This observation also conforms with the nature of robust learning. Besides, we also find that even one model trained on the same dataset with different random seeds tends to make more consistent prediction agreement on clean examples compared with noisy examples, as shown in Figure 2(b). One intuitive solution to instantiate this observation is the ensemble methods. However, the data encountered during the inference stage is clean. Then ensemble methods will make consistent while corrupted predictions on the test data, since every model is corrupted. Nevertheless, for our method, we will use the signals from different models to refine one model so that this single target model could potentially yield better predictions.

3 METHODOLOGY

In this section, we propose the learning framework DeCA which uses the observations described in section 2 as the denoising signals to learn robust implicit recommenders. Before a detailed description of the methods, some notations and problem formulation are given.

3.1 Notations and Problem Formulation

We use $u \in U$ and $i \in I$ to denote the user and item with $U$ and $I$ being the user and item sets respectively. $R \in \mathbb{R}^{|U| \times |I|}$ is the binary matrix for corrupted data. $r_{ui}$ represents the $(u, i)$-th entry of $R$ and is defined as $r_{ui} = 1$ if there is an interaction between user $u$ and item $i$, otherwise $r_{ui} = 0$. Due to the existence of noisy examples, the real preference matrix is different from $R$. We use $\tilde{R} \in \mathbb{R}^{|U| \times |I|}$ to denote the real preference matrix and $\tilde{r}_{ui}$ is the real preference of user $u$ over item $i$. What we have is the corrupted binary $R$ while the clean $\tilde{R}$ is not accessible.

Formally, we assume $r_{ui}$ is drawn from a Bernoulli distribution:

$$r_{ui} \sim \text{Bernoulli}(\eta_{ui}) = \text{Bernoulli}(f_\theta(u, i)), \quad (1)$$

where $\eta_{ui}$ describes the probability of positive preference (i.e., $r_{ui} = 1$) and is approximated by $f_\theta(u, i)$. $\theta$ denotes the parameters of $f$. We use $f_\theta$ as our target recommender that generates the final recommendation. Specifically, the output value after sigmoid of the logits from the model is denoted as $f_\theta(u, i)$, which is also the probability that $\eta_{ui}$ equals to 1. Generally speaking, $f_\theta$ is expected to have high model expressiveness. Since the denoising signal comes from the agreement across different model predictions, we introduce another auxiliary Bernoulli distribution parameterized by $g_\phi(u, i)$. In this paper, we use a relatively simple model matrix factorization (MF) [19] as the auxiliary model $g_\phi$. Recent research has demonstrated that MF is still one of the most effective models to capture user preference for recommendation [28].

Considering both the noisy positive and noisy negative examples, we assume that given the real preference $r_{ui}$, the corrupted binary $\tilde{r}_{ui}$ is also drawn from Bernoulli distributions as

$$\tilde{r}_{ui}|r_{ui} = 0 \sim \text{Bernoulli}(h_g(u, i))$$
$$\tilde{r}_{ui}|r_{ui} = 1 \sim \text{Bernoulli}(h'_g(u, i)),$$ \quad (2)

where $h_g(u, i)$ and $h'_g(u, i)$, parameterized by $\phi$ and $\psi$ respectively, are two models describing the consistency between the real user preference and data observation. We use two MF models as $h$ and $h'$ without special mention.

The task of this paper is, given the corrupted binary data $\tilde{R}$, to infer the real user preference $R$ and its underlying model $f_\theta$, then to use $f_\theta$ to generate recommendation.

3.2 Denoising with Cross-Model Agreement

As discussed in section 2, for clean examples which denote the real user preference, different models tend to make more consistent predictions compared with noisy examples. For simplicity, in this subsection we use $P(R)$ to denote the real user preference $\text{Bernoulli}(\eta)$. $P_f(R)$ and $P_g(R)$ are used to represent the approximated $\text{Bernoulli}(f_\theta)$ and $\text{Bernoulli}(g_\phi)$, correspondingly.

Due to the fact that $P_f(R)$ and $P_g(R)$ both approximate the real user preference $R$, they should remain a relatively small KLD-divergence according to section 2, which is formulated as

$$D[P_g(R)||P_f(R)] = E_{R \sim P_g}[\log P_g(R) - \log P_f(R)].$$ \quad (3)

However, naively optimizing Eq.(3) is meaningless since we do not have the supervision signal of $R$. As a result, we need to introduce supervision signals from the corrupted data observation $\tilde{R}$. Using the Bayes theorem, $P_f(R)$ can be approximated as

$$P_f(R) \approx p(R) = \frac{P(R)P(\tilde{R}|R)}{P(\tilde{R})}. \quad (4)$$

Combining Eq.(4) and Eq.(3), we can obtain

$$D[P_g(R)||P_f(R)] = E_{R \sim P_g} [\log P_g(R) - \log P_f(R)]$$
$$\approx E_{R \sim P_g} [\log P_g(R) - \log \frac{P(R)P(\tilde{R}|R)}{P(\tilde{R})}]$$
$$= E_{R \sim P_g} [\log P_g(R) - \log P(R) - \log P(\tilde{R}) + \log P(\tilde{R})]$$
$$= D[P_g(R)||P(R)] - \log P(\tilde{R}) + E_{R \sim P_g} [\log P(R)]. \quad (5)$$

We then rearrange the terms in Eq.(5) and obtain

$$E_{R \sim P_g} [\log P(R)] - D[P_g(R)||P_f(R)] = \log P(\tilde{R}) - D[P_g(R)||P(\tilde{R})].$$ \quad (6)

We can see the meaning of maximizing the left side of Eq.(6) is maximizing the likelihood of data observation given real user preference (i.e., $\log P(\tilde{R})$) and meanwhile minimizing the KL-divergence between two models which both approximate the real user preference (i.e., $D[P_g(R)||P_f(R)]$). Since the KL-divergence $D[P_g(R)||P(R)]$ is larger than zero, the left side of Eq.(6) can also be seen as the lower bound of $\log P(\tilde{R})$. The bound is satisfied only if $P_g(R)$ perfectly recovers $P(R|R)$, in other words, $P_g(R)$ perfectly approximates the real user preference distribution given the corrupted data. A naive solution is to directly maximize the left side of Eq.(6) with an end-to-end fashion. However, it would not yield satisfactory
performance. The reason is that the left side of Eq.(6) is based on the expectation over \( P_y \). The learning process is equivalent to training \( g_y \) with the corrupted data \( \mathbf{R} \) and then uses \( D[P_y(\mathbf{R})|P_y(\mathbf{R})] \) to transmit the information from \( g_y \) to \( f_0 \). However, the information is corrupted and it will affect the performance of our target model \( f_0 \). To fix the problem, we notice that when the training process is converged, two distributions \( P_y(\mathbf{R}) \) and \( P_y(\mathbf{R}) \) would be close to each other. We can then modify the left side of Eq.(6) as

\[
E_{R-P_f} \left[ \log P(\mathbf{R}|\mathbf{R}) - D[P_y(\mathbf{R})|P_y(\mathbf{R})] \right] \\
\approx \log P(\mathbf{R}) - D[P_y(\mathbf{R})|P(\mathbf{R}|\mathbf{R})].
\]

Then optimizing the left side of Eq.(7) is actually training \( f_0 \) with the corrupted \( \mathbf{R} \) and then transmit information to \( g_y \). However, since \( g_y \) is relatively simple than \( f_0 \), \( g_y \) could only fit the robust data component (i.e., clean examples). Thus \( g_y \) would not affect the learning of \( f_0 \) on clean examples, while in the meantime, pull back \( f_0 \) on noisy samples, or in other words, downgrade the noisy signal. To this end, the denoising objective function can be formulated as

\[
\mathcal{L} = -E_{R-P_f} \left[ \log P(\mathbf{R}|\mathbf{R}) + D[P_y(\mathbf{R})|P_f(\mathbf{R})] \right] \\
+ \alpha D[P_y(\mathbf{R})|P_f(\mathbf{R})],
\]

where \( \alpha \in [0, 1] \) is a hyper-parameter.

In fact, the two models \( f_0 \) and \( g_y \) in Eq.(9) can play a symmetric role, but we still need a model (here we use \( f_0 \)) to do the inference. Thus, the other model (here we use \( g_y \)) serves as an auxiliary model. For Co-trained DeCA, we chose \( g_y \) to be MF to (1) achieve more efficient computation and more stable learning because of the simplicity of MF; (b) yield good performances since MF is shown to be still effective (as stated in Section 3.1); (c) make the proposed DeCA more applicable without tuning on the selection of \( g_y \) as MF is one of the most general models. More precisely, we can give the detailed formulation of the terms of 9 in Supplement A.2.

### 3.3 DeCA with Fixed Pre-training

As described in subsection 3.2, DeCA utilizes an auxiliary model \( g_y \) as an information filter which helps to downgraude the effect of noisy signal. The auxiliary model \( g_y \) is co-trained jointly with the target model \( f_0 \). As discussed before, the left side of Eq.(6) is the lower bound of \( \log P(\mathbf{R}) \), which is satisfied when \( P_y(\mathbf{R}) \approx P(\mathbf{R}|\mathbf{R}) \). Then as the training process converges, we have \( P_y(\mathbf{R}) \approx P_f(\mathbf{R}) \). Thus we could expect \( P(\mathbf{R}|\mathbf{R}) \approx P_f(\mathbf{R}) \).

To this end, we modify the assumption of Eq.(1) as

\[
r_{ui} \mid \tilde{r}_{ui} \sim \text{Bernoulli}(\eta_{ui}) \approx \text{Bernoulli}(f_0(u, i)).
\]

The underlying intuition is that whether the data observation is known or not should not affect the robust real user preference, which is reasonable and keeps inline with the nature of robust learning. As a result, we should minimize the following KL-divergence

\[
\mathcal{L}_{\text{DeCA}(p)} = \bar{E}_{R-P_f} \left[ \log P(\mathbf{R}|\mathbf{R}) + \alpha D[P_y(\mathbf{R})|P_f(\mathbf{R})] \right] \\
+ (1 - \alpha) D[P_y(\mathbf{R})|P_f(\mathbf{R})].
\]

Through Bayesian transformation, the following equation can be deduced:

\[
\mathcal{L}_{\text{DeCA}(p)} = \bar{E}_{R-P_f} \left[ \log P(\mathbf{R}|\mathbf{R}) - D[P_y(\mathbf{R}|\mathbf{R})|P(\mathbf{R})] \right] \\
= \log P(\mathbf{R}) - D[P_y(\mathbf{R}|\mathbf{R})|P(\mathbf{R})].
\]

Similarly with Eq.(9), we propose another variant loss function as DeCA with pre-training (DeCA(p)):

\[
\mathcal{L}_{\text{DeCA}(p)} = \left( \mathcal{L}_{\text{DeCA}(p)} \right)^2.
\]

We use a pre-trained and fixed model \( f_0 \) which has the same structure as our target model \( f_0 \) but is trained with different random seeds to model \( P(\mathbf{R}) \). This setting is motivated by the observation that one model trained with different random seeds tends to make high variance predictions on noisy examples but more consistent agreement predictions on clean examples, as shown in Figure 2(b).

The major difference between DeCA and DeCA(p) can be summarized as follows:

- The auxiliary model \( g_y \) of DeCA is a simple MF while the involved \( f_0 \) in DeCA(p) has the same structure as the target model.
- DeCA co-trains the auxiliary model \( g_y \) with \( f_0 \) but DeCA(p) uses a pre-trained and fixed \( f_0 \) to describe the prior distribution.

### 3.4 Training Routine

We can see that the objective functions of both DeCA and DeCA(p) contain the term of \( E[\log P(\mathbf{R}|\mathbf{R})] \). A naive solution to optimize \( E[\log P(\mathbf{R}|\mathbf{R})] \) is direct performing calculation as shown in Eq.(17). However, we find that it won’t yield satisfactory results. The reason is that computing \( E[\log P(\mathbf{R}|\mathbf{R})] \) involves updating three models \( h_\phi \), \( h_\psi \), and \( f_0 \) simultaneously. Then these models will interfere with each other. The corrupted outputs from \( h_\phi \) and \( h_\psi \) will prevent \( f_0 \) from learning well, and vice versa. To handle this issue, we split the denoising problem into two dual tasks: Denoising Positive (DP) which aims to denoise the noisy positive examples from interacted samples as shown in Figure 1(c); and Denoising Negative (DN) which aims to denoise noisy negative examples from sampled missing interactions as illustrated in Figure 1(d). Then we optimize these two sub-tasks iteratively. Each time we only consider either \( h_\phi \) or \( h_\psi \), and fix the other one to be the function of an approximate large value denoted as \( C_1 \) and \( C_2 \) as illustrated in the following part.

We can then obtain information from the data to update \( f_0 \), and also refine one of the \( h \)-models according to the outputs of \( f_0 \).

#### 3.4.1 Denoising Positive

In this situation, we only focus on denoising noisy positive examples, which means we will regard every negative sample as a clean example, as shown in Figure 1(c). That is to say, given \( r_{ui} = 1 \) we will always have \( \tilde{r}_{ui} = 1 \) (i.e., \( h_\psi(u, i) = 1 \)). Thus in this sub-task, only the model \( h_\phi \) will be trained. Then the term \( E[\log P(\mathbf{R}|\mathbf{R})] \) becomes

\[
E[\log P(\mathbf{R}|\mathbf{R})] = \sum_{(u, i)} \log h_\phi(u, i) \cdot (1 - f_0(u, i)) - \sum_{(u, i)} C_1 \cdot f_0(u, i) + \log(1 - h_\psi(u, i)) \\
+ (1 - f_0(u, i)).
\]
where $C_1$ is a large positive hyperparameter to substitute $-\log(1-h^\psi (u, i))$.

3.4.2 Denoising Negative. In this sub-task, we only focus on denoising noisy negative examples in the sampled missing interactions, which means that we will regard every positive example as a clean example, as shown in Figure 1(d). That is to say, given $r_{ui} = 0$ we will always have $h^\psi (u, i) = 0$. Thus in this sub-task, only model $h^\psi$ will be trained. The term $E[\log P(R|R)]$ is expanded as

$$E[\log P(R|R)] = \sum_{(u,i)|r_{ui}=0} \log(1-h^\psi (u, i)) \cdot f_0(u, i)$$

$$+ \sum_{(u,i)|r_{ui}=1} \log h^\psi (u, i) \cdot f_0(u, i) - C_2 \cdot (1 - f_0(u, i)),$$

3.4.2 Denoising Negative. In this sub-task, we only focus on denoising noisy negative examples in the sampled missing interactions, which means that we will regard every positive example as a clean example, as shown in Figure 1(d). That is to say, given $r_{ui} = 0$ we will always have $h^\psi (u, i) = 0$. Thus in this sub-task, only model $h^\psi$ will be trained. The term $E[\log P(R|R)]$ is expanded as

$$E[\log P(R|R)] = \sum_{(u,i)|r_{ui}=0} \log(1-h^\psi (u, i)) \cdot f_0(u, i)$$

$$+ \sum_{(u,i)|r_{ui}=1} \log h^\psi (u, i) \cdot f_0(u, i) - C_2 \cdot (1 - f_0(u, i)),$$

where $C_2$ is a large positive hyperparameter to replace $-\log(h^\psi(u, i))$.

In the training procedure of DeCA and DeCA(p), we will expand $E[\log P(R|R)]$ as Eq.(14) and Eq.(15) iteratively to denoise both noisy positive examples and noisy negative examples. The detailed training pseudo-code of DeCA and DeCA(p) can be found in the supplement.

4 EXPERIMENTS

In this section, we instantiate the proposed DeCA and DeCA(p) with four state-of-the-art recommendation models as the target model $f$ and conduct experiments on four real-world datasets to verify the effectiveness of our methods. We aim to answer the following research questions:

- **RQ1**: How do the proposed methods perform compared to normal training and other denoising methods? Can the proposed methods help to downgrade the effect of noisy examples?
- **RQ2**: How does the design of proposed methods affect the recommendation performance, including the iterative training, hyperparameter study, and model selection?
- **RQ3**: Can the proposed methods generate reasonable preference distribution given the corrupted binary data observation?

4.1 Experimental Settings

4.1.1 Datasets. We conduct experiments with four public accessible datasets: MovieLens, Modcloth, Adressa and Electronics. More statistics about these three datasets are listed in Table 1. For each dataset, we construct the clean test set with only clean examples which denote the real user preference.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Users</th>
<th># Items</th>
<th># Interactions</th>
<th>Sparsity</th>
</tr>
</thead>
<tbody>
<tr>
<td>MovieLens</td>
<td>943</td>
<td>1,682</td>
<td>100,000</td>
<td>0.93695</td>
</tr>
<tr>
<td>Modcloth</td>
<td>44,783</td>
<td>1,020</td>
<td>99,893</td>
<td>0.99781</td>
</tr>
<tr>
<td>Adressa</td>
<td>212,231</td>
<td>6,596</td>
<td>419,491</td>
<td>0.99970</td>
</tr>
<tr>
<td>Electronics</td>
<td>1,157,633</td>
<td>9,560</td>
<td>1,292,954</td>
<td>0.99988</td>
</tr>
</tbody>
</table>

where $C_1$ is a large positive hyperparameter to substitute $-\log(1-h^\psi (u, i))$.

4.1.2 Evaluation protocols. We adopt cross-validation to evaluate the performance. For Adressa and MovieLens, we split the user-item interactions into the training set, validation set, and test set according to the ratio of 8:1:1 in chronological order [35]. As for the other two datasets, we randomly split the historical interactions according to the ratio of 8:1:1. After that, the clean test set is constructed for each dataset as described in section 4.1.1. The performance is measured by two widely used top-K recommendation metrics [12, 41]: recall@K and ndcg@K. By default, we set K = 5, 20 for Modcloth and Adressa, and K = 10, 50 for Electronics since the number of items in Electronics is larger. All experiments are run 3 times. The average and standard deviation are reported.

4.1.3 Baselines. We select four state-of-the-art recommendation models as the target model $f$ of DeCA and DeCA(p):

- **GMF** [12]: This is a generalized version of MF by changing the inner product to the element-wise product and a dense layer.
- **NeuMF** [12]: The method is a state-of-the-art neural CF model which combines GMF with a Multi-Layer Perceptron (MLP).
- **CDAE** [40]: CDAE corrupts the observed interactions with random noises, and then employs several linear layers to reconstruct the original datasets, which will increase its anti-noise abilities.
- **LightGCN** [13]: LightGCN is a newly proposed graph-based recommendation model which learns user and item embeddings by linearly propagating them on the interaction graph.

Each model is trained with the following approaches:

- **Normal**: Train the model with the assumption shown in Figure 1(a) and simple binary-cross-entropy (BCE) loss.
- **WBPR** [9]: This is a re-sampling based denoising method, which considers the popular but uninteracted items are highly likely to be real negative ones.
- **T-CE** [35]: This is a re-weighting based denoising method, which uses the Truncated BCE to assign zero weights to large-loss examples with a dynamic threshold in each iteration.
- **Ensemble**: This is an ensemble-based approach, which aggregates the results from two models with different random seeds.
- **DeCA** and **DeCA(p)**: Our proposed methods.

Besides, [43] proposed a noisy robust learning method. We do not compare with this method because it has been shown to be only applicable to the MF model [43].

MovieLens [11]. This is one of the most popular datasets in the task of recommendation. For evaluation, the clean test set is constructed based on user-item pairs with ratings equal to 5.

Modcloth. It is from an e-commerce website that sells women’s clothing and accessories. For evaluation, the clean test set is built on user-item pairs whose rating scores are equal to 5.

Adressa. This is a real-world news reading dataset from Adressa-sen. It contains the interaction records of anonymous users and the news. According to [18], we use interactions with dwell time longer than 10 seconds to construct the clean test set.

Electronics. Electronics is collected from the Electronics category on Amazon [26, 33]. The clean test set is built with user-item pairs with rating scores equal to 5.

Note that all the ratings and dwell time are only used to construct the clean test set. The models are trained with only the corrupted binary implicit feedback.
### Table 2: Overall performance comparison. The highest scores are in Boldface. R is short for Recall and N is short for NDCG.
The results with improvements over the best baseline larger than 5% are marked with `*`.

<table>
<thead>
<tr>
<th>MovieLens</th>
<th>R@5</th>
<th>R@20</th>
<th>N@5</th>
<th>N@20</th>
<th>R@5</th>
<th>R@20</th>
<th>N@5</th>
<th>N@20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>0.021±0.001</td>
<td>0.095±0.001</td>
<td>0.035±0.003</td>
<td>0.059±0.001</td>
<td>0.025±0.002</td>
<td>0.104±0.006</td>
<td>0.041±0.002</td>
<td>0.064±0.003</td>
</tr>
<tr>
<td>WBPR</td>
<td>0.023±0.002</td>
<td>0.082±0.003</td>
<td>0.045±0.002</td>
<td>0.056±0.000</td>
<td>0.022±0.003</td>
<td>0.086±0.001</td>
<td>0.038±0.005</td>
<td>0.054±0.002</td>
</tr>
<tr>
<td>T-CE</td>
<td>0.017±0.002</td>
<td>0.098±0.001</td>
<td>0.026±0.003</td>
<td>0.054±0.001</td>
<td>0.026±0.004</td>
<td>0.106±0.002</td>
<td>0.047±0.003</td>
<td>0.068±0.002</td>
</tr>
<tr>
<td>Ensemble</td>
<td>0.021±0.000</td>
<td>0.095±0.001</td>
<td>0.035±0.001</td>
<td>0.059±0.000</td>
<td>0.030±0.001</td>
<td>0.108±0.003</td>
<td>0.048±0.002</td>
<td>0.070±0.002</td>
</tr>
<tr>
<td>DeCA</td>
<td>0.022±0.001</td>
<td>0.109±0.007*</td>
<td>0.038±0.001</td>
<td>0.064±0.002</td>
<td>0.022±0.001</td>
<td>0.099±0.006</td>
<td>0.032±0.003</td>
<td>0.059±0.003</td>
</tr>
<tr>
<td>DeCA(p)</td>
<td>0.027±0.002*</td>
<td>0.099±0.001</td>
<td>0.051±0.003*</td>
<td>0.066±0.001*</td>
<td>0.035±0.000*</td>
<td>0.120±0.003*</td>
<td>0.064±0.000*</td>
<td>0.081±0.002*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Modcloth</th>
<th>R@5</th>
<th>R@20</th>
<th>N@5</th>
<th>N@20</th>
<th>R@5</th>
<th>R@20</th>
<th>N@5</th>
<th>N@20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>0.063±0.006</td>
<td>0.225±0.008</td>
<td>0.043±0.005</td>
<td>0.084±0.005</td>
<td>0.082±0.003</td>
<td>0.242±0.004</td>
<td>0.055±0.002</td>
<td>0.101±0.002</td>
</tr>
<tr>
<td>WBPR</td>
<td>0.067±0.002</td>
<td>0.224±0.002</td>
<td>0.046±0.001</td>
<td>0.090±0.001</td>
<td>0.092±0.001</td>
<td>0.247±0.020</td>
<td>0.064±0.001</td>
<td>0.108±0.006</td>
</tr>
<tr>
<td>T-CE</td>
<td>0.067±0.003</td>
<td>0.235±0.002</td>
<td>0.045±0.003</td>
<td>0.093±0.001</td>
<td>0.065±0.010</td>
<td>0.228±0.008</td>
<td>0.044±0.010</td>
<td>0.091±0.008</td>
</tr>
<tr>
<td>Ensemble</td>
<td>0.064±0.004</td>
<td>0.228±0.007</td>
<td>0.043±0.002</td>
<td>0.089±0.002</td>
<td>0.093±0.003</td>
<td>0.269±0.004</td>
<td>0.061±0.001</td>
<td>0.109±0.002</td>
</tr>
<tr>
<td>DeCA</td>
<td>0.072±0.001</td>
<td>0.245±0.001</td>
<td>0.050±0.001</td>
<td>0.099±0.001</td>
<td>0.099±0.005</td>
<td>0.268±0.006</td>
<td>0.065±0.001</td>
<td>0.113±0.001</td>
</tr>
<tr>
<td>DeCA(p)</td>
<td>0.074±0.001*</td>
<td>0.247±0.001*</td>
<td>0.052±0.001*</td>
<td>0.100±0.000*</td>
<td>0.087±0.005</td>
<td>0.265±0.006</td>
<td>0.059±0.004</td>
<td>0.110±0.004</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Adressa</th>
<th>R@5</th>
<th>R@20</th>
<th>N@5</th>
<th>N@20</th>
<th>R@5</th>
<th>R@20</th>
<th>N@5</th>
<th>N@20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>0.116±0.003</td>
<td>0.209±0.003</td>
<td>0.080±0.002</td>
<td>0.112±0.001</td>
<td>0.149±0.004</td>
<td>0.312±0.004</td>
<td>0.131±0.002</td>
<td>0.180±0.003</td>
</tr>
<tr>
<td>WBPR</td>
<td>0.115±0.007</td>
<td>0.210±0.007</td>
<td>0.084±0.003</td>
<td>0.116±0.003</td>
<td>0.172±0.002</td>
<td>0.311±0.003</td>
<td>0.132±0.001</td>
<td>0.181±0.001</td>
</tr>
<tr>
<td>T-CE</td>
<td>0.109±0.000</td>
<td>0.209±0.001</td>
<td>0.070±0.000</td>
<td>0.104±0.000</td>
<td>0.172±0.003</td>
<td>0.312±0.003</td>
<td>0.134±0.001</td>
<td>0.183±0.002</td>
</tr>
<tr>
<td>Ensemble</td>
<td>0.110±0.002</td>
<td>0.191±0.006</td>
<td>0.078±0.002</td>
<td>0.105±0.003</td>
<td>0.180±0.001</td>
<td>0.311±0.001</td>
<td>0.134±0.001</td>
<td>0.180±0.000</td>
</tr>
<tr>
<td>DeCA</td>
<td>0.125±0.002*</td>
<td>0.220±0.001*</td>
<td>0.091±0.002</td>
<td>0.126±0.002</td>
<td>0.170±0.006</td>
<td>0.318±0.002</td>
<td>0.130±0.001</td>
<td>0.181±0.000</td>
</tr>
<tr>
<td>DeCA(p)</td>
<td>0.123±0.005</td>
<td>0.220±0.000</td>
<td>0.093±0.004*</td>
<td>0.127±0.003*</td>
<td>0.183±0.009</td>
<td>0.316±0.004</td>
<td>0.137±0.002</td>
<td>0.181±0.005</td>
</tr>
</tbody>
</table>

For 4.1.4 Parameter settings. We optimize all models using Adam optimizer. The batch size is set as 2048. The learning rate is tuned as 0.001 on four datasets. For each training instance, we sample one interacted sample and one randomly sampled missing interaction to feed the model. We use the recommended network settings of all models. Besides, the embedding size of users and items in GMF and NeuMF, the hidden size of CDAE, are all set to 32. For LightGCN, the embedding size is set to 64 without dropout [13]. The $L_2$ regularization coefficient is tuned in $[0.01, 0.1, 1, 10, 100, 1000]$ divided by the number of users in each dataset. For DeCA and DeCA(p), there are three hyper-parameters: $C_1$, $C_2$ and $\alpha$. We apply a grid search for hyperparameters: $C_1$ and $C_2$ are searched among $[1, 10, 100, 1000]$, $\alpha$ is tuned in $[0.05, 1]$. Note that the hyperparameters of recommendation models keep exactly the same across all training approaches for a fair comparison. It also indicates that the proposed methods can be easily integrated with downstream recommendation models without exhaustive hyperparameter refinement.

### 4.2 Performance Comparison (RQ1)

Table 2 shows the performance comparison on Movielens, Modcloth and Adressa. The results on Electronics are provided in the supplement due to space limitations. From the table, we can have the following observations:

- The proposed DeCA and DeCA(p) can effectively improve the recommendation performance of all the four recommendation models over all datasets. Either DeCA or DeCA(p) achieves the best performance compared with normal training and other denoising methods, except for few cases. Even if the CDAE model...
4.3.2 Hyperparameter Study. In this part, we conduct experiments to show the effect of \( C_1 \) and \( C_2 \), which are two hyper-parameters used in the iterative training routine. More specifically, a larger \( C_1 \) means we are more confident that the negative samples are truly negative. A larger \( C_2 \) denotes that an interacted sample is more likely to be real positive. Figure 5 shows the results on Adressa when using GMF as the target recommendation model. Results on other datasets are provided in the supplement. We find that the optimal setting of \( C_1 \) (i.e., \( C_1 = 1000 \)) is larger than the optimal setting of \( C_2 \) (i.e., \( C_2 = 10 \) or 100). This observation indicates that the probability that a missing interaction is real negative is larger than the probability that an interacted sample is real positive.

4.3.3 Effect of Model Selection. We use MF as \( h \) and \( h' \) to model the probability \( P(R|R) \) as our default settings. Since the model capability of MF might be limited, in this part we conduct experiments to see how the performance would be if we use more complicated models for \( h \) and \( h' \). Table 4 shows the result on Modcloth. Results on other datasets are provided in the supplement. The target recommendation model is GMF. We can see that replacing MF with more complicated models will not boost the performance significantly. The reason might be that modelling the probability \( P(R|R) \) is not a very complex task. Accomplishing this task with MF already works well. Besides, as we can see that no matter what model we use, the performance of the proposed DeCA and DeCA(p) are significantly better than normal training.

4.4 Method Interpretation (RQ3)

In this part, we conduct experiments to see whether DeCA and DeCA(p) generate the reasonable user preference distribution given the corrupted data, which can be used to show the interpretability of our methods. For DeCA, we have the following equation:

\[
P(r_{ui} = 1|\tilde{r}_{ui} = 1) = \frac{P(\tilde{r}_{ui} = 1|r_{ui} = 1)P(r_{ui} = 1)}{P(\tilde{r}_{ui} = 1)}
\]

\[
= \frac{h'_i(u, i) + h_i(u, i)(1 - f_0(u, i))}{h'_i(u, i) + h_i(u, i) + h_0(u, i)}
\]

For DeCA(p), \( P(r_{ui} = 1|\tilde{r}_{ui} = 1) \) can be directly computed as \( P(r_{ui} = 1|\tilde{r}_{ui} = 1) = f_0(u, i) \) with condition \( \tilde{r}_{ui} = 1 \). \( P(r_{ui} = 1|\tilde{r}_{ui} = 1) \) describes the probability of an interacted sample to be real positive. Figure 4 shows the relationship between ratings and this mean real positive probability on the datasets MovieLens, Modcloth and Electronics. We can see that the probability gets larger as ratings go higher, which is consistent with the impression that examples with higher ratings should be more likely to be real positive. Besides, we can see that the probability trend is consistent with different \( h \) and \( h' \) models, which further demonstrates the generalization ability of our methods.

5 RELATED WORK

Modern recommender systems are usually trained using implicit feedback. The training of implicit recommenders usually needs to sample negative examples from missing interactions and then feed the interacted items and sampled negative instances for pairwise ranking [27] or BCE loss function. Besides, there are also attempts to investigate non-sampling approaches [1, 14, 45] for implicit feedback. Regarding the recommendation models, MF [19]...
Table 3: Performance comparison when considering one sub-task. DeCA-DP and DeCA-DN denote training DeCA with only either DP or DN. DeCA(p)-DP and DeCA(p)-DN denote training DeCA(p) with only either DP or DN.

<table>
<thead>
<tr>
<th>Method</th>
<th>GMF R@5</th>
<th>GMF R@20</th>
<th>GMF N@5</th>
<th>GMF N@20</th>
<th>NeuMF R@5</th>
<th>NeuMF R@20</th>
<th>NeuMF N@5</th>
<th>NeuMF N@20</th>
<th>CDAE R@5</th>
<th>CDAE R@20</th>
<th>CDAE N@5</th>
<th>CDAE N@20</th>
<th>LightGCN R@5</th>
<th>LightGCN R@20</th>
<th>LightGCN N@5</th>
<th>LightGCN N@20</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeCA-DP</td>
<td>0.072</td>
<td>0.245</td>
<td>0.050</td>
<td>0.098</td>
<td>0.097</td>
<td>0.236</td>
<td>0.061</td>
<td>0.109</td>
<td>0.077</td>
<td>0.236</td>
<td>0.050</td>
<td>0.095</td>
<td>0.066</td>
<td>0.223</td>
<td>0.042</td>
<td>0.086</td>
</tr>
<tr>
<td>DeCA-DN</td>
<td>0.051</td>
<td>0.187</td>
<td>0.032</td>
<td>0.071</td>
<td>0.045</td>
<td>0.198</td>
<td>0.027</td>
<td>0.070</td>
<td>0.066</td>
<td>0.236</td>
<td>0.044</td>
<td>0.093</td>
<td>0.068</td>
<td>0.240</td>
<td>0.047</td>
<td>0.096</td>
</tr>
<tr>
<td>DeCA(p)</td>
<td>0.072</td>
<td>0.245</td>
<td>0.050</td>
<td>0.099</td>
<td>0.099</td>
<td>0.268</td>
<td>0.065</td>
<td>0.113</td>
<td>0.086</td>
<td>0.250</td>
<td>0.056</td>
<td>0.102</td>
<td>0.064</td>
<td>0.221</td>
<td>0.041</td>
<td>0.085</td>
</tr>
<tr>
<td>DeCA(p)-DP</td>
<td>0.075</td>
<td>0.243</td>
<td>0.052</td>
<td>0.099</td>
<td>0.094</td>
<td>0.268</td>
<td>0.064</td>
<td>0.113</td>
<td>0.091</td>
<td>0.245</td>
<td>0.059</td>
<td>0.102</td>
<td>0.065</td>
<td>0.220</td>
<td>0.042</td>
<td>0.086</td>
</tr>
<tr>
<td>DeCA(p)-DN</td>
<td>0.060</td>
<td>0.221</td>
<td>0.039</td>
<td>0.084</td>
<td>0.092</td>
<td>0.255</td>
<td>0.062</td>
<td>0.108</td>
<td>0.073</td>
<td>0.238</td>
<td>0.050</td>
<td>0.097</td>
<td>0.072</td>
<td>0.240</td>
<td>0.049</td>
<td>0.096</td>
</tr>
<tr>
<td>DeCA(p)</td>
<td>0.074</td>
<td>0.247</td>
<td>0.052</td>
<td>0.100</td>
<td>0.087</td>
<td>0.265</td>
<td>0.059</td>
<td>0.110</td>
<td>0.089</td>
<td>0.251</td>
<td>0.057</td>
<td>0.103</td>
<td>0.073</td>
<td>0.235</td>
<td>0.051</td>
<td>0.096</td>
</tr>
</tbody>
</table>

Figure 4: Mean real positive probability of different ratings on three datasets.

Figure 5: Hyperparameter study on Adressa.

Table 4: Effect of model selection on Modcloth.

<table>
<thead>
<tr>
<th>Method</th>
<th>$h$ and $h'$</th>
<th>R@5</th>
<th>R@20</th>
<th>N@5</th>
<th>N@20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>0.0629</td>
<td>0.2246</td>
<td>0.0430</td>
<td>0.0884</td>
<td></td>
</tr>
<tr>
<td>DeCA</td>
<td>MF</td>
<td>0.0717</td>
<td>0.2452</td>
<td>0.0500</td>
<td>0.0985</td>
</tr>
<tr>
<td></td>
<td>GMF</td>
<td>0.0740</td>
<td>0.2453</td>
<td>0.0513</td>
<td>0.0989</td>
</tr>
<tr>
<td></td>
<td>NeuMF</td>
<td>0.0747</td>
<td>0.2448</td>
<td>0.0519</td>
<td>0.0991</td>
</tr>
<tr>
<td>DeCA(p)</td>
<td>MF</td>
<td>0.0743</td>
<td>0.2465</td>
<td>0.0515</td>
<td>0.0996</td>
</tr>
<tr>
<td></td>
<td>GMF</td>
<td>0.0740</td>
<td>0.2464</td>
<td>0.0508</td>
<td>0.0989</td>
</tr>
<tr>
<td></td>
<td>NeuMF</td>
<td>0.0751</td>
<td>0.2464</td>
<td>0.0514</td>
<td>0.0992</td>
</tr>
</tbody>
</table>

is one of the most notable and effective models. Recently, plenty of work has proposed deep learning-based recommendation models, such as GMF, NeuMF [12], CDAE [40] and Wide&Deep [12]. The key idea is to use deep learning to increase model expressiveness. Besides, graph neural networks also demonstrated their capability in recommendation. Plenty of models have emerged, such as HOPRec [41], KGAT [36], NGCF [37] and LightGCN [13].

Recently, there has been some researches [16, 23] pointing out that the observed implicit feedback could be easily corrupted by different factors, such as popularity bias, exposure bias and so on [2]. As a result, there have been some efforts aiming to address the noisy problem, which can be categorized into re-sampling methods [4–6, 8, 43], re-weighting methods [25, 29, 35] and methods utilizing additional knowledge input [18, 24, 46]. Re-sampling methods aim to design more effective samplers. For example, [8] proposed to sample popular but not interacted items as negative examples while [4] proposed that the viewed but not purchased items are highly likely to be real negative. [5, 38] proposed to use reinforcement learning for negative sampling. The performance of re-sampling depends heavily on the sampling strategy [45], which is usually developed heuristically [43]. Re-weighting methods usually identify the noisy examples as samples with high loss values and then assign lower weights to them. Besides, additional knowledge such as dwell time [18], gaze pattern [46] and auxiliary item features [24] can also be used to denoise implicit feedback.

6 CONCLUSION

In this work, we propose model-agnostic training frameworks to learn robust recommenders from implicit feedback. We find that different models tend to make more consistent agreement predictions for clean examples compared with noisy ones. To this end, we propose denoising with cross-model agreement (DeCA), which utilizes predictions from different models as the denoising signal. We employ the proposed methods on four recommendation models and conduct extensive experiments on four datasets. The results demonstrate the effectiveness of our methods. To the best of our knowledge, this is the first paper to consider the agreement across different models as the denoising signal, and it’s also the first work to denoise both noisy positive examples and noisy negative examples. Future work includes using DeCA for other domain applications and generalizing DeCA for multi-class scenarios.

ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China (2020AAA0106000) and the National Natural Science Foundation of China (U19A2079, 62121002).
REFERENCES


[23] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2019. Reinforcement Learning Robust Recommenders through Cross-Model Agreement WWW ’22, April 25–29, 2022, Virtual Event, Lyon, France
A SUPPLEMENT

A.1 Pseudo-code for DeCA and DeCA(p)

Algorithm 1: Learning algorithm for DeCA

Input: Corrupted data \( R \), learning rate \( \beta \), epochs \( T \), hyper-parameter \( \alpha, C_1, C_2 \), regularization weight \( \lambda \), target recommender \( f \), auxiliary model \( g, h, h' \).
Output: Parameters \( \theta, \mu, \phi, \psi \) for \( f, g, h, h' \), correspondingly.

1. Initialize all parameters;
2. \( count \leftarrow 0; \)
3. while Not early stop and epoch < \( T \) do
   4. Draw a minibatch of \((u, i)\) from \( \{(u, i)|r_{ui} = 1\} \);
   5. Draw \((u, i)\) from \( \{(u, i)|r_{ui} = 0\} \) for each \((u, i)\);
   6. if \( count \% 2 = 0 \) then
      7. Compute \( \mathcal{L}_{\text{DEC}(\alpha)} \) according to Eq.(9) and Eq.(14);
   else
      8. Compute \( \mathcal{L}_{\text{DEC}(\alpha)} \) according to Eq.(9) and Eq.(15);
   9. Add regularization term: \( \mathcal{L}_{\text{DEC}(\alpha)} = \mathcal{L}_{\text{DEC}(\alpha)} + \lambda ||\theta||^2 \);
   10. for each parameter \( \Theta \) in \( \{\theta, \mu, \phi, \psi\} \) do
       11. Compute \( \frac{\partial \mathcal{L}_{\text{DEC}(\alpha)}}{\partial \theta} \) by back-propagation;
       12. \( \Theta \leftarrow \Theta - \beta \frac{\partial \mathcal{L}_{\text{DEC}(\alpha)}}{\partial \Theta} \);
   end
   13. \( count \leftarrow count + 1; \)
end

Algorithm 2: Learning algorithm for DeCA(p)

Input: Corrupted data \( R \), learning rate \( \beta \), epochs \( T \), hyper-parameter \( \alpha, C_1, C_2 \), regularization weight \( \lambda \), target recommender \( f \), auxiliary model \( g, h, h' \).
Output: Parameters \( \theta, \mu, \phi, \psi \) for \( f, g, h, h' \), correspondingly.

1. Set random seed to \( s_1 \), initialize another copy of \( \Theta \) as \( \Theta' \).
2. while Not early stop and epoch < \( T \) do
   3. Draw a minibatch of \((u, i)\) from \( \{(u, i)|\tilde{r}_{ui} = 1\} \);
   4. Draw \((u, i)\) from \( \{(u, i)|\tilde{r}_{ui} = 0\} \) for each \((u, i)\);
   5. Compute binary cross-entropy \( \mathcal{L}_{\text{BCE}} \) on \((u, i)\) and \((u, i)\) with \( f_\theta \);
   6. Add regularization: \( \mathcal{L}_{\text{BCE}} = \mathcal{L}_{\text{BCE}} + \lambda ||\theta||^2 \);
   7. Compute \( \frac{\partial \mathcal{L}_{\text{BCE}}}{\partial \Theta'} \) by back-propagation;
   8. \( \Theta' \leftarrow \Theta' - \beta \frac{\partial \mathcal{L}_{\text{BCE}}}{\partial \Theta'} \);
   9. end
10. Freeze \( \Theta' \), set random seed to \( s_2 \) and initialize \( \Theta, \phi, \psi \);
11. \( count \leftarrow 0; \)
12. while Not early stop and epoch < \( T \) do
13. Draw \((u, l_i)\) and \((u, l_i)\) similarly with l=3-4;
14. if \( count \% 2 = 0 \) then
   15. Compute \( \mathcal{L}_{\text{DEC}(\alpha)} \) according to Eq.(13) and Eq.(14);
16. else
   17. Compute \( \mathcal{L}_{\text{DEC}(\alpha)} \) according to Eq.(13) and Eq.(15);
   18. Add regularization term: \( \mathcal{L}_{\text{DEC}(\alpha)} = \mathcal{L}_{\text{DEC}(\alpha)} + \lambda ||\Theta||^2 \);
   19. for each parameter \( \Theta, \phi, \psi \) do
       20. Compute \( \frac{\partial \mathcal{L}_{\text{DEC}(\alpha)}}{\partial \Theta} \) by back-propagation;
       21. \( \Theta \leftarrow \Theta - \beta \frac{\partial \mathcal{L}_{\text{DEC}(\alpha)}}{\partial \Theta} \);
   end
   22. \( count \leftarrow count + 1; \)
end

A.2 Mathematical Formulations

The detailed formulation of the term \( E_{R \cdot P_f} [\log P(\hat{R}|R)] \) in \( \mathcal{L}_{\text{DEC}} \) is shown as:

\[
E_{R \cdot P_f} [\log P(\hat{R}|R)] = \sum_{(u,i)} E_{r_{ui}} [P(\hat{r}_{ui}|r_{ui})]
\]

Thus maximizing Eq.(18) will optimize \( f_\theta(u, i) \) to fit the observational data. The loss for the dataset is similar to cross-entropy loss, and it will converge.

A.3.3 Analysis of the Run Time Complexity. For DeCA, the main additional complexity comes from the forward and backward of
Table 5: Overall performance comparison. The highest scores are in Boldface. R is short for Recall and N is short for NDCG. The results with improvements over the best baseline larger than 5% are marked with ∗.

<table>
<thead>
<tr>
<th>Electronics</th>
<th>R@10</th>
<th>R@50</th>
<th>N@10</th>
<th>N@50</th>
<th>R@10</th>
<th>R@50</th>
<th>N@10</th>
<th>N@50</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GMF</td>
<td></td>
<td></td>
<td></td>
<td>NeuMF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>0.023±0.000</td>
<td>0.063±0.001</td>
<td>0.013±0.000</td>
<td>0.021±0.000</td>
<td>0.075±0.000</td>
<td>0.191±0.002</td>
<td>0.048±0.000</td>
<td>0.072±0.001</td>
</tr>
<tr>
<td>WBPR</td>
<td>0.026±0.001</td>
<td>0.072±0.001</td>
<td>0.014±0.000</td>
<td>0.024±0.000</td>
<td>0.077±0.000</td>
<td>0.201±0.001</td>
<td>0.048±0.000</td>
<td>0.074±0.000</td>
</tr>
<tr>
<td>T-CE</td>
<td>0.024±0.000</td>
<td>0.063±0.001</td>
<td>0.013±0.000</td>
<td>0.022±0.000</td>
<td>0.071±0.004</td>
<td>0.189±0.008</td>
<td>0.045±0.002</td>
<td>0.071±0.003</td>
</tr>
<tr>
<td>Ensemble</td>
<td>0.014±0.001</td>
<td>0.034±0.001</td>
<td>0.010±0.000</td>
<td>0.016±0.000</td>
<td>0.051±0.000</td>
<td>0.108±0.001</td>
<td>0.040±0.000</td>
<td>0.055±0.000</td>
</tr>
<tr>
<td>DeCA</td>
<td>0.045±0.001</td>
<td>0.104±0.000</td>
<td>0.023±0.000</td>
<td>0.037±0.000</td>
<td>0.071±0.001</td>
<td>0.191±0.001</td>
<td>0.044±0.002</td>
<td>0.066±0.001</td>
</tr>
<tr>
<td>DeCA(p)</td>
<td>0.042±0.001</td>
<td>0.110±0.000</td>
<td>0.024±0.000</td>
<td>0.038±0.000</td>
<td>0.074±0.002</td>
<td>0.200±0.001</td>
<td>0.047±0.000</td>
<td>0.074±0.000</td>
</tr>
</tbody>
</table>

Table 6: Effect of model selection on MovieLens, Adressa and Electronics.

<table>
<thead>
<tr>
<th>Method</th>
<th>MovieLens</th>
<th>Addressa</th>
<th>Electronics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R@5</td>
<td>R@20</td>
<td>R@5</td>
</tr>
<tr>
<td>Normal</td>
<td>0.0214</td>
<td>0.0953</td>
<td>0.0352</td>
</tr>
<tr>
<td>DeCA</td>
<td>0.0217</td>
<td>0.1089</td>
<td>0.0382</td>
</tr>
<tr>
<td></td>
<td>0.0186</td>
<td>0.1054</td>
<td>0.0351</td>
</tr>
<tr>
<td></td>
<td>0.0225</td>
<td>0.1077</td>
<td>0.0349</td>
</tr>
<tr>
<td>DeCA(p)</td>
<td>0.0269</td>
<td>0.0987</td>
<td>0.0538</td>
</tr>
<tr>
<td></td>
<td>0.0262</td>
<td>0.0980</td>
<td>0.0536</td>
</tr>
<tr>
<td></td>
<td>0.0255</td>
<td>0.0963</td>
<td>0.0539</td>
</tr>
</tbody>
</table>

Figure 6: Hyperparameter study on Movielens, Modcloth and Electronics.

A.4.1 Performance Comparison on Electronics. The performance comparison between models and baselines on the Electronics dataset is shown in Table 5.

A.4.2 Supplementary Results for Hyperparameter Study. Figure 6 shows the supplementary results for hyperparameter study on MovieLens, Modcloth and Electronics.

A.4.3 Supplementary Results for Model Selection. Table 6 shows the supplementary results for the selection of h and h' on MovieLens, Adressa and Electronics.

A.4 Supplementary Experimental Results

A.4.1 Performance Comparison on Electronics. The performance comparison between models and baselines on the Electronics dataset is shown in Table 5.