Wang, Y., Xin, X., Meng, Z., Jose, J. M., Feng, F. and He, X. (2022) Learning Robust Recommenders Through Cross-Model Agreement. In: ACM Web Conference 2022, Lyon, France, 25-29 April 2022, pp. 2015-2025. ISBN 9781450390965.

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.

© Association for Computing Machinery 2022. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in the Proceedings of the ACM Web Conference 2022, Lyon, France, 25-29 April 2022, pp. 2015-2025. ISBN 9781450390965. https://doi.org/10.1145/3485447.3512202.

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

Deposited on: 15 February 2022
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  
xinxin@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 potential positive 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 clean examples which denote the real user preference, while the 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.

CCS CONCEPTS
- Mathematics of computing → Probability and statistics;  
- Information systems → Recommender systems.

†Xin Xin is the corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France  
© 2022 Association for Computing Machinery.  
ACM ISBN 978-1-4503-9096-5/22/04 ... $15.00  
https://doi.org/10.1145/3485447.3512202

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 clean examples which fall near the classification boundary also have high losses [31]. These hard clean examples play an important role to boost the recommendation performance of implicit feedback [5, 38, 44]. Denoising methods based on loss values could encounter difficulties distinguishing between noisy examples and hard clean examples, leading to sub-optimal solutions. However, we find that another signal which can help to identify noisy examples can be obtained from the agreement across different model predictions.

To illustrate such signal, we conduct an empirical study on the MovieLens-100k dataset. Similar observations also exist on other datasets. We train four notable recommendation models (i.e. MF [19], GMF and NeuMF [12] and LightGCN [13]) with the binary user-item implicit feedback with the normal training assumption (i.e., all interacted items are considered as positive examples) and uninteracted ones are considered as negative examples, as shown in (a). However, in real-world applications, there are both noisy-positive examples in interacted items and noisy-negative examples in uninteracted items, as shown in (b). (c) denotes only denoising positive examples [24, 35], while (d) only denoises negative examples [4, 43].

Figure 1: Different assumptions in recommendation training on implicit feedback. Gray denotes positive user preference (i.e., ‘like’) while white denotes negative preference (i.e., ‘dislike’). In normal training, interacted instances are considered as positive examples and uninteracted ones are considered as negative examples, as shown in (a). However, in real-world applications, there are both noisy-positive examples in interacted items and noisy-negative examples in uninteracted items, as shown in (b). (c) denotes only denoising positive examples [24, 35], while (d) only denoises negative examples [4, 43].

Figure 2: Mean prediction differences on clean and noisy examples in Modcloth from (a) different models or (b) one model trained with two different random seeds. NMF and LGN is short for NeuMF and LightGCN respectively.
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 $u \in \mathcal{U}$ and $i \in \mathcal{I}$ to denote the user and item with $\mathcal{U}$ and $\mathcal{I}$ being the user and item sets respectively. $\mathbf{R} \in \mathbb{R}^{\mid \mathcal{U}\mid \times \mid \mathcal{I}\mid}$ is the binary matrix for corrupted data. $\tilde{r}_{ui}$ represents the $(u, i)$-th entry of $\mathbf{R}$ and is defined as $\tilde{r}_{ui} = 1$ if there is an interaction between user $u$ and item $i$, otherwise $\tilde{r}_{ui} = 0$. Due to the existence of noisy examples, the real preference matrix is different from $\mathbf{R}$. We use $\mathbf{R} \in \mathbb{R}^{\mid \mathcal{U}\mid \times \mid \mathcal{I}\mid}$ to denote the real preference matrix and $r_{ui}$ is the real preference of user $u$ over item $i$. What we have is the corrupted binary $\mathbf{R}$ while the clean $\mathbf{R}$ is not accessible.

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

$$r_{ui} \sim \text{Bernoulli}(\eta_{ui}) \equiv \text{Bernoulli}(f_0(u, i)), \quad (1)$$

where $\eta_{ui}$ describes the probability of positive preference (i.e., $r_{ui} = 1$) and is approximated by $f_0(u, i)$. $\theta$ denotes the parameters of $f$. We use $f_0$ as our target recommender that generates the final recommendation. Specifically, the output value after sigmoid of the logits from the model is denoted as $f_0(u, i)$, which is also the probability that $\eta_{ui}$ equals to 1. Generally speaking, $f_0$ 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

$$\tilde{r}_{ui}|r_{ui} = 0 \sim \text{Bernoulli}(h_\phi(u, i))$$

$$\tilde{r}_{ui}|r_{ui} = 1 \sim \text{Bernoulli}(h_\phi(u, i)),$$  \quad (2)

where $h_\phi(u, i)$ and $h_\phi'(u, i)$, 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 $\tilde{\mathbf{R}}$, to infer the real user preference $\mathbf{R}$ and its underlying model $f_0$, then to use $f_0$ 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 $P(\mathbf{R})$ to denote the real user preference $Bernoulli(\eta)$. $P_f(\mathbf{R})$ and $P_g(\mathbf{R})$ are used to represent the approximated $Bernoulli(f_0)$ and $Bernoulli(g_\phi)$, correspondingly.

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

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

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

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

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

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

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

$$E_{R \sim \tilde{R}}[\log P(\tilde{\mathbf{R}}) - D[P_g(\mathbf{R})||P_f(\mathbf{R})] = \log P(\tilde{\mathbf{R}}) - D[P_g(\mathbf{R})||P_f(\mathbf{R})]].$$ \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., $\log P(\tilde{\mathbf{R}})$) and meanwhile minimizing the KL-divergence between two models which both approximate the real user preference (i.e., $D[P_g(\mathbf{R})||P_f(\mathbf{R})]$). Since the KL-divergence $D[P_g(\mathbf{R})||P_f(\mathbf{R})]$ is larger than zero, the left side of Eq.(6) can also be seen as the lower bound of $\log P(\tilde{\mathbf{R}})$. The bound is satisfied only if $P_g(\mathbf{R})$ perfectly recovers $P(\mathbf{R})$, in other words, $P_g(\mathbf{R})$ 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
Then as the training process converge, we have

Thus, the other model (here we use \( f_b \)) will be more effective (as stated in Section 3.1); (c) make the proposed DeCA more suitable when tuning on the selection of \( g_p \), as MF is one of the most general models. More precisely, we can give the detailed formulation of the terms of Eq.(9) in Supplement A.2.

Considering that the gradient of \( D(P_g || P_f) \) and \( D(P_f || P_g) \) to \( \theta \) is different, which could slightly affect the performance, we then formulate the final denoising objective function as

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

In fact, the two models \( f_b \) and \( g_p \) in Eq.(9) can play a symmetric role, but we still need a model (here we use \( f_b \)) to do the inference. Thus, the other model (here we use \( g_p \)) serves as an auxiliary model. For Co-trained DeCA, we chose \( g_p \) 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 \( g_p \), 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 \( g_p \) as an information filter which helps to downgrade the effect of noisy signal. The auxiliary model \( g_p \) is co-trained jointly with the target model \( f_b \). As discussed before, the left side of Eq.(6) is the lower bound of \( \log P(\hat{R}) \), which is satisfied when \( P_g(\hat{R}) \approx P(\hat{R}) \). Then as the training process converge, we have \( P_g(\hat{R}) \approx P_f(\hat{R}) \).

Thus we could expect \( P(\hat{R}) \approx P_f(\hat{R}) \).

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

The underlying intuition is 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

Through Bayesian transformation, the following equation can be deduced:

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

We use a pre-trained and fixed model \( f_b \) which has the same structure as our target model \( f_b \) but is trained with different random seeds to model \( P(\hat{R}) \). 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 \( g_p \) of DeCA is a simple MF while the involved \( f_b \) in DeCA(p) has the same structure as the target model.
- DeCA co-trains the auxiliary model \( g_p \) with \( f_b \) but DeCA(p) uses a pre-trained and fixed \( f_b \) 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 \( E[\log P(\hat{R})] \). A naive solution to optimize \( E[\log P(\hat{R})] \) is direct performing calculation as shown in Eq.(17). However, we find that it won’t yield satisfactory results. The reason is that computing \( E[\log P(\hat{R})] \) involves updating three models \( h_\phi, h_\psi, f_b \) simultaneously. Then these models will interfere with each other. The corrupted outputs from \( h_\phi, h_\psi \) will prevent \( f_b \) 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). We then 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. Then we can obtain information from the data to update \( f_b \), and also refine one of the \( h \)-models according to the outputs of \( f_b \).

#### 3.4.1 Denoising Positive

In this situation, we only focus on denoising noise positive examples, which means we will regard every noisy positive example 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 \( E[\log P(\hat{R})] \) becomes

where \( \alpha \) is a hyper-parameter.
where $C_1$ is a large positive hyperparameter to substitute $-\log(1 - h'_y(u,i))$.

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 $h_y(u,i) = 0$. Thus in this sub-task, only model $h'_y$ will be trained. The term $E[\log P(R|R)]$ is expanded as

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

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

(15)

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

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 conduct the clean test set with only clean examples which denote the real user preference.

#### 4.1.2 Evaluation protocols.

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.

### 4.1.3 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].
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 *.

| MovieLens | R@3 | R@20 | N@3 | N@20 | R@3 | R@20 | N@3 | N@20 |
|-----------|-----|-----|-----|-----|-----|-----|-----|-----|
|           | GMF |     |     |     | NeuMF |     |     |     |     |
| Normal    | 0.021±0.001 0.095±0.001 0.035±0.003 0.059±0.001 | 0.025±0.002 0.104±0.006 0.041±0.002 0.064±0.003 | 0.025±0.000 0.106±0.001 0.048±0.002 0.071±0.003 |
| WBPR      | 0.032±0.002 0.082±0.003 0.045±0.002 0.056±0.000 | 0.022±0.003 0.086±0.001 0.038±0.005 0.054±0.002 | 0.026±0.004 0.106±0.002 0.047±0.003 0.068±0.002 |
| T-CE      | 0.017±0.002 0.098±0.001 0.026±0.003 0.054±0.001 | 0.026±0.004 0.106±0.002 0.047±0.003 0.068±0.002 |
| Ensemble  | 0.021±0.000 0.095±0.001 0.035±0.001 0.059±0.000 | 0.030±0.001 0.108±0.003 0.048±0.002 0.070±0.002 |
| DeCA      | 0.022±0.001 0.109±0.007* 0.038±0.001 0.064±0.002 | 0.022±0.001 0.099±0.006 0.032±0.003 0.059±0.003 |
| DeCA(p)   | 0.027±0.002* 0.099±0.001 0.054±0.003 0.066±0.001 | 0.035±0.000* 0.120±0.003* 0.064±0.000* 0.081±0.002* |

| Modcloth  | R@3 | R@20 | N@3 | N@20 | R@3 | R@20 | N@3 | N@20 |
|-----------|-----|-----|-----|-----|-----|-----|-----|-----|
|           | GMF |     |     |     | NeuMF |     |     |     |     |
| Normal    | 0.063±0.006 0.225±0.008 0.043±0.005 0.088±0.005 | 0.082±0.003 0.242±0.004 0.055±0.002 0.101±0.002 | 0.076±0.002 0.252±0.004 0.070±0.003 0.108±0.006 |
| WBPR      | 0.067±0.002 0.224±0.002 0.046±0.001 0.090±0.001 | 0.092±0.001 0.247±0.002 0.064±0.001 0.108±0.006 |
| T-CE      | 0.067±0.004 0.235±0.002 0.045±0.003 0.093±0.001 | 0.065±0.010 0.228±0.008 0.044±0.010 0.091±0.008 |
| Ensemble  | 0.064±0.004 0.228±0.002 0.045±0.002 0.089±0.002 | 0.093±0.003 0.269±0.004 0.061±0.001 0.109±0.002 |
| DeCA      | 0.072±0.001 0.245±0.001 0.050±0.001 0.099±0.001 | 0.099±0.005* 0.268±0.006 0.065±0.001 0.113±0.001 |
| DeCA(p)   | 0.074±0.001* 0.247±0.001* 0.052±0.001* 0.100±0.000* | 0.087±0.005 0.265±0.006 0.059±0.004 0.110±0.004 |

| Adressa   | R@3 | R@20 | N@3 | N@20 | R@3 | R@20 | N@3 | N@20 |
|-----------|-----|-----|-----|-----|-----|-----|-----|-----|
|           | GMF |     |     |     | NeuMF |     |     |     |     |
| Normal    | 0.116±0.003 0.209±0.003 0.080±0.002 0.112±0.001 | 0.149±0.004 0.312±0.004 0.131±0.002 0.180±0.003 | 0.149±0.004 0.312±0.004 0.131±0.002 0.180±0.003 |
| WBPR      | 0.115±0.007 0.210±0.007 0.084±0.003 0.116±0.003 | 0.172±0.002 0.311±0.003 0.132±0.001 0.181±0.001 |
| T-CE      | 0.109±0.000 0.209±0.001 0.070±0.000 0.104±0.000 | 0.172±0.003 0.312±0.003 0.134±0.001 0.183±0.002 |
| Ensemble  | 0.110±0.002 0.191±0.006 0.078±0.002 0.105±0.003 | 0.180±0.001 0.311±0.001 0.134±0.001 0.180±0.000 |
| DeCA      | 0.125±0.002* 0.220±0.001* 0.091±0.002 0.126±0.002 | 0.170±0.006 0.318±0.002 0.130±0.001 0.181±0.000 |
| DeCA(p)   | 0.123±0.005 0.220±0.001 0.093±0.004* 0.127±0.003* | 0.183±0.009 0.316±0.004 0.137±0.002 0.181±0.005 |

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 set to 32. For LightGCN, the embedding size is set to 64 without dropout [13]. The $L_2$ regularization coefficient is tuned in $[0.0, 0.1, 0.1, 10, 100, 1000]$ 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 $[0, 0.5, 1]$. 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)
Tabel 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...
Recall@20
•
•
•
•
•
•

Figure 3: Recall along the training process

itself is based on the denoising auto-encoder which is more robust to corrupted data, there still exists improvement when training with our proposed methods, especially in the MovieLens dataset.

• Simple ensemble of the results from different models sometimes cannot lead to better performance, for example, in the Electronic dataset. The reason could be that the Electronic data is super sparse. The results from different models vary much more. Simple aggregating two very different results could lead to poor performance in such cases.

• The performance of LightGCN is not so good on Modcloth, Adressa and Electronics. This could be attributed to the bias of datasets. We can see from Table 1 that users in three datasets would only be connected to a small number of items while items are connected to a large amount of users. The imbalance of interaction graphs could affect the performance of LightGCN.

Besides, we also analyse how DeCA and DeCA(p) affect the memorization of noisy samples. Figure 3 shows the learning curve of normal training and the proposed DeCA and DeCA(p) on Modcloth, Adressa and Electronics when using GMF as the target recommender. We can see that in normal training, performance on the clean test set decreases in the late training stage. The reason is as the training goes on, the model tends to memorize all samples in the dataset, both noisy and clean ones. However, we can see that both DeCA and DeCA(p) remain more stable and better performance along the whole training stage. The results demonstrate that the proposed DeCA and DeCA(p) successfully prevent the model from being affected by noisy samples.

4.3 Model Investigation (RQ2)

4.3.1 Ablation Study. DeCA and DeCA(p) utilize an iterative training routine that contains two sub-tasks DP and DN. In this part, we discuss how these two sub-tasks contribute to the whole framework and how they perform separately. The results on Modcloth dataset are shown in Table 3. We can have following observations:

• In most cases, DeCA and DeCA(p) are better than only considering one sub-task of denoising. Some abnormal cases could be attributed to the instability of DN. During each training epoch, the interacted samples are fixed while the sampled negative instances could change frequently. Since DN aims to denoise noisy negative examples from the sampled missing interactions, the performance of DN could be unstable.

• DP is better than DN in most cases, except for the LightGCN model. This observation to some extent indicates that finding noisy examples from interacted samples is much easier than finding potential positive preference from the massive uninteracted missing samples.

4.3.2 Hyperparameter Study. In this part, we conduct experiments to show the effect of C1 and C2, which are two hyper-parameters used in the iterative training routine. More specifically, a larger C1 means we are more confident that the negative samples are truly negative. A larger C2 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 C1 (i.e., C1 = 1000) is larger than the optimal setting of C2 (i.e., C2 = 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:

\[
P(r_{ui} = 1|\tilde{r}_{ui} = 1) = \frac{P(\tilde{r}_{ui} = 1|r_{ui} = 1)P(r_{ui} = 1)}{P(\tilde{r}_{ui} = 1)} = \frac{h'_x(u, i)f_0(u, i) + h_y(u, i)(1 - f_0(u, i))}{\sum_{u, i} h'_x(u, i)f_0(u, i) + h_y(u, i)(1 - f_0(u, i))}
\]

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 demonstrate 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.

| Model         | R@5   | R@20  | N@5   | N@20  |
|---------------|-------|-------|-------|-------|
| DeCA-DP       | 0.072 | 0.245 | 0.050 | 0.098 |
| DeCA-DN       | 0.051 | 0.187 | 0.032 | 0.071 |
| DeCA          | 0.072 | 0.245 | 0.050 | 0.099 |
| DeCA(p)-DP    | 0.075 | 0.243 | 0.052 | 0.099 |
| DeCA(p)-DN    | 0.060 | 0.221 | 0.039 | 0.084 |
| DeCA(p)       | 0.074 | 0.247 | 0.052 | 0.100 |

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).
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 $a, C_1, C_2$, regularization weight $\lambda$, target recommender $f$, auxiliary model $g, h'$

Output: Parameters $\theta, \mu, \phi, \psi$ for $f, g, h'$, correspondingly

1. Initialize all parameters;
2. $\text{count} \leftarrow 0$
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 $\text{count} \mod 2 = 0$ then
      7. Compute $L_{DeCA}$ according to Eq.(9) and Eq.(14);
   else
      8. Compute $L_{DeCA}$ according to Eq.(9) and Eq.(15);
   9. Add regularization term: $L_{DeCA} \leftarrow L_{DeCA} + \lambda ||\theta||^2$;
   10. for each parameter $\Theta$ in $\{\Theta, \mu, \phi, \psi\}$ do
       11. Compute $L_{DeCA}/\Theta$ by back-propagation;
       12. $\Theta \leftarrow \Theta - \beta L_{DeCA}/\Theta$
   13. end
   14. $\text{count} \leftarrow \text{count} + 1$
end

Algorithm 2: Learning algorithm for DeCA(p)

Input: Corrupted data $\hat{R}$, learning rate $\beta$, epochs $T$, hyper-parameter $a, C_1, C_2$, regularization weight $\lambda$, target recommender $f$, auxiliary model $g, h'$

Output: Parameters $\theta, \mu, \phi, \psi$ for $f, g, h'$, 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 $(u, i)|\hat{r}_{ui} = 1$;
   4. Draw $(u, i)$ from $(u, i)|\hat{r}_{ui} = 0$ for each $(u, i)$;
   5. Compute binary cross-entropy $L_{BCE}$ on $(u, i)$ and $(u, i)$ with $f$;
   6. Add regularization term: $L_{BCE} \leftarrow L_{BCE} + \lambda ||\theta||^2$;
   7. Compute $L_{BCE}/\Theta$ by back-propagation;
   8. $\theta' \leftarrow \theta' - \beta L_{BCE}/\Theta$
end

A.2 Mathematical Formulations

The detailed formulation of the term $\mathbb{E}_{\hat{R}-P_f}[\log P(\hat{R}|R)]$ in $L_{DeCA}$ is shown as:

$$E_{\hat{R}-P_f}[\log P(\hat{R}|R)] = \sum_{(u,i)} E_{r_{ui}-P_f}[\log P(\hat{r}_{ui}=1)]$$

The KL divergence term $D[P_g||P_f]$ can be calculated as

$$D[P_g||P_f] = g(u,i) \cdot \log g(u,i)/f_0(u,i) + (1-g(u,i)) \cdot \log (1-g(u,i))/f_0(u,i)$$

A.3 Additional Discussion

In this subsection, we provide a brief discussion to illustrate the relationship between our methods and other related methods.

A.3.1 Relationship with Variational Auto-Encoder. We can see that Eq.(12) is exactly the objective function of a variational auto-encoder (VAE) [7]. Specifically, the real user preference $P_f(R|R)$ which is parameterized by our target model $f_0$ maps the corrupted data $\hat{R}$ to the latent variables $R$ and thus can be seen as the encoder. The likelihood $P(\hat{R}|R)$ describes the distribution of corrupted data $\hat{R}$ given the real user preference $R$ and acts as the decoder. Finally, $P(R)$ is the fixed prior distribution. However, we claim that the proposed DeCA(p) is substantially different from VAE. For example, DeCA(p) does not utilize the re-parameterization of VAE. VAE is just an interpretation of DeCA(p). Besides, although there are some methods utilizing VAE for recommendation [20, 30], few of them are designed for recommendation denoising. Finally, existing VAE-based recommendation methods are actually recommendation models while the proposed DeCA(p) acts a model-agnostic training framework which is then instantiated with all kinds of downstream models. They serve on different levels.

A.3.2 Analysis of the Convergence. During training, if the parameters $C_1$ and $C_2$ are large enough, the process will act similarly to the normal training, since (1) the gradients from the fixed part will dominate the update of the model when $C_1$ and $C_2$ are large. (2) With large $C_1$ and $C_2$, we will demonstrate that the target model $P_f$ has the same target as cross-entropy loss as follows:

If we simply replace $-\log(1-h(y, u, i))$ and $-\log(1-h(y, u, i))$ with $C_1$ and $C_2$ in the term $E[\log P(\hat{R}|R)]$, we could obtain the following expansion:

$$E[\log P(\hat{R}|R)] = -\sum_{(u,i)|r_{ui}=0} C_1 f_0(u,i) - \sum_{(u,i)|r_{ui}=1} C_2 (1-f_0(u,i))$$

Thus maximizing Eq.(18) will optimize $f_0(u,i)$ 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 *. 

| Electronics | R@10  | R@50  | N@10  | N@50  | \(\text{GMP}^\star\) | R@10  | R@50  | N@10  | N@50  | \(\text{NeuMF}^\star\) |
|------------|------|------|------|------|----------------|------|------|------|------|----------------|
| Normal     | 0.023±0.000 | 0.063±0.001 | 0.013±0.000 | 0.021±0.000 | 0.075±0.000 | 0.191±0.002 | 0.048±0.000 | 0.072±0.001 |
| WBPR       | 0.026±0.001 | 0.073±0.001 | 0.014±0.000 | 0.024±0.000 | 0.077±0.000 | 0.201±0.001 | 0.048±0.000 | 0.074±0.000 |
| T-CE       | 0.024±0.000 | 0.067±0.001 | 0.013±0.000 | 0.022±0.000 | 0.071±0.004 | 0.189±0.008 | 0.045±0.002 | 0.071±0.003 |
| Ensemble   | 0.014±0.001 | 0.034±0.001 | 0.010±0.000 | 0.016±0.000 | 0.051±0.000 | 0.108±0.001 | 0.040±0.000 | 0.055±0.000 |
| DeCA       | 0.045±0.001* | 0.107±0.000 | 0.023±0.000 | 0.057±0.000 | 0.071±0.001 | 0.191±0.001 | 0.044±0.002 | 0.069±0.001 |
| DeCA(p)    | 0.042±0.001 | 0.110±0.000* | 0.024±0.000* | 0.038±0.000* | 0.074±0.002 | 0.206±0.001 | 0.047±0.000 | 0.074±0.000 |

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

| Method | \(h\) and \(h'\) | MovieLens | Addressa | Electronics |
|--------|-----------------|-----------|-----------|-------------|
|        | R@5  | R@20 | N@5  | N@20 | R@5  | R@20 | N@5  | N@20 | R@10 | R@50 | N@10 | N@50 |
| Normal |      |      |      |      |      |      |      |      |      |      |      |      |      |
| MF     | 0.021±0.000 | 0.116±0.000 | 0.080±0.001 | 0.112±0.000 | 0.022±0.000 | 0.103±0.000 | 0.036±0.000 | 0.072±0.000 |
| GMF    | 0.018±0.000 | 0.109±0.000 | 0.077±0.000 | 0.072±0.000 | 0.022±0.000 | 0.076±0.000 | 0.028±0.000 | 0.072±0.000 |
| NeuMF  | 0.022±0.000 | 0.110±0.000 | 0.036±0.000 | 0.056±0.000 | 0.018±0.000 | 0.049±0.000 | 0.023±0.000 | 0.072±0.000 |
| DeCA   |      |      |      |      |      |      |      |      |      |      |      |      |      |
| MF     | 0.026±0.000 | 0.123±0.000 | 0.093±0.000 | 0.127±0.000 | 0.041±0.000 | 0.110±0.000 | 0.032±0.000 | 0.072±0.000 |
| GMF    | 0.026±0.000 | 0.122±0.000 | 0.091±0.000 | 0.124±0.000 | 0.042±0.000 | 0.112±0.000 | 0.035±0.000 | 0.086±0.000 |
| NeuMF  | 0.025±0.000 | 0.121±0.000 | 0.090±0.000 | 0.124±0.000 | 0.049±0.000 | 0.115±0.000 | 0.037±0.000 | 0.086±0.000 |

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

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.