

Learning Robust Recommenders through Cross-Model Agreement

Yu Wang

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

Jeoman Jose

University of Glasgow
Joeman.Jose@glasgow.ac.uk

Xin Xin*

University of Glasgow
x.xin.1@research.gla.ac.uk

Zaiqiao Meng

Cambridge University
zaiqiao.meng@gmail.com

Fuli Feng

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

Xiangnan He

University of Science and Technology
of China
hexn@ustc.edu.cn

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>

KEYWORDS

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

ACM Reference Format:

Yu Wang, Xin Xin, Zaiqiao Meng, Jeoman Jose, Fuli Feng, and Xiangnan He. 2022. Learning Robust Recommenders through Cross-Model Agreement. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France*. ACM, New York, NY, USA, 11 pages. <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.

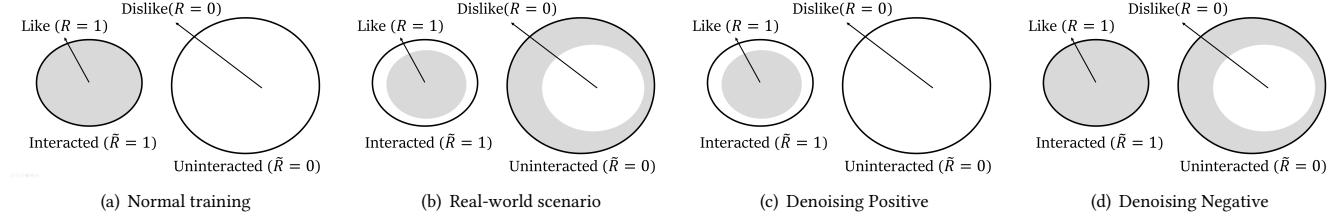


Figure 1: Different assumptions in recommendation training on implicit feedback. Gray denotes positive user preference (i.e., ‘like’) while white denotes negative preference (i.e., ‘dislike’). In normal training, interacted instances are considered as positive examples and uninteracted ones are considered 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].

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.

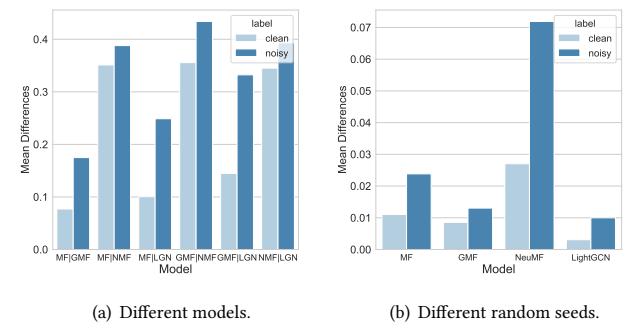


Figure 2: Mean prediction differences on clean and noisy examples in Modcloth from (a) different models or (b) one model trained with two different random seeds. NMF and LGN is short for NeuMF and LightGCN respectively.

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 dataset¹. 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 negative examples are sampled from missing interactions, as shown in Figure 1(a)).

More precisely, we consider interactions whose ratings are 4 and 5 as clean positive examples while interactions whose ratings are 1 and 2 as noisy positive examples. The prediction difference

¹<https://grouplens.org/datasets/movielens/100k/>

between two models is defined as $|I(\delta(y_{ui})) - I(\delta(y'_{ui}))|$, where y_{ui} and y'_{ui} are the predicted scores from two models regarding user u and item i , δ is the sigmoid function. $I(x)$ is an indicator function and is defined as $I(x) = 1$ when $x \geq 0.5$ and $I(x) = 0$ otherwise. 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. $\tilde{\mathbf{R}} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$ is the binary matrix for corrupted data. \tilde{r}_{ui} represents the (u, i) -th entry of $\tilde{\mathbf{R}}$ 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 $\tilde{\mathbf{R}}$. We use $\mathbf{R} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$ 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 $\tilde{\mathbf{R}}$ while the clean \mathbf{R} is not accessible.

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

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

where η_{ui} describes the probability of positive preference (i.e., $r_{ui} = 1$) and is approximated by $f_\theta(u, i)$. θ denotes the parameters of f . We use f_θ 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 η_{ui} equals to 1. Generally speaking, f_θ 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 $g_\mu(u, i)$. In this paper, we use a relatively simple model

matrix factorization (MF) [19] as the auxiliary model g_μ . 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

$$\begin{aligned} \tilde{r}_{ui}|r_{ui} = 0 &\sim \text{Bernoulli}(h_\phi(u, i)) \\ \tilde{r}_{ui}|r_{ui} = 1 &\sim \text{Bernoulli}(h'_\psi(u, i)), \end{aligned} \quad (2)$$

where $h_\phi(u, i)$ and $h'_\psi(u, i)$, parameterized by ϕ and ψ 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_θ , then to use f_θ 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 $\text{Bernoulli}(\eta)$. $P_f(\mathbf{R})$ and $P_g(\mathbf{R})$ are used to represent the approximated $\text{Bernoulli}(f_\theta)$ and $\text{Bernoulli}(g_\mu)$, 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

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

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

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

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

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

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

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

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., $\log P(\tilde{\mathbf{R}}|\mathbf{R})$) 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 $D[P_g(\mathbf{R})||P(\mathbf{R}|\tilde{\mathbf{R}})]$ 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 $P(\mathbf{R}|\tilde{\mathbf{R}})$, 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 performance. The reason is that the left side of Eq.(6) is based on the expectation over P_g . The learning process is equivalent to training g_μ with the corrupted data $\tilde{\mathbf{R}}$ and then uses $D[P_g(\mathbf{R})||P_f(\mathbf{R})]$ to transmit the information from g_μ to f_θ . However, the information is corrupted and it will affect the performance of our target model f_θ . To fix the problem, we notice that when the training process is converged, two distributions $P_f(\mathbf{R})$ and $P_g(\mathbf{R})$ would be close to each other. We can then modify the left side of Eq.(6) as

$$\begin{aligned} E_{\mathbf{R} \sim P_f} [\log P(\tilde{\mathbf{R}}|\mathbf{R})] - D[P_g(\mathbf{R})||P_f(\mathbf{R})] \\ \approx \log P(\tilde{\mathbf{R}}) - D[P_g(\mathbf{R})||P(\mathbf{R}|\tilde{\mathbf{R}})]. \end{aligned} \quad (7)$$

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

$$\mathcal{L} = -E_{\mathbf{R} \sim P_f} [\log P(\tilde{\mathbf{R}}|\mathbf{R})] + D[P_g(\mathbf{R})||P_f(\mathbf{R})]. \quad (8)$$

Considering that the gradient of $D[P_g||P_f]$ and $D[P_f||P_g]$ to θ is different, which could slightly affect the performance, we then formulate the final denoising objective function as

$$\mathcal{L}_{DeCA} = -E_{\mathbf{R} \sim P_f} [\log P(\tilde{\mathbf{R}}|\mathbf{R})] + \alpha D[P_g||P_f] + (1 - \alpha) D[P_f||P_g], \quad (9)$$

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

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

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

$$r_{ui}|\tilde{r}_{ui} \sim Bernoulli(\eta_{ui}) \sim Bernoulli(f_\theta(u, i)). \quad (10)$$

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

$$D[P_f(\mathbf{R}|\tilde{\mathbf{R}})||P(\mathbf{R}|\tilde{\mathbf{R}})]. \quad (11)$$

Through Bayesian transformation, the following equation can be deduced:

$$\begin{aligned} E_{P_f} [\log P(\tilde{\mathbf{R}}|\mathbf{R})] - D[P_f(\mathbf{R}|\tilde{\mathbf{R}})||P(\mathbf{R}|\tilde{\mathbf{R}})] \\ = \log P(\tilde{\mathbf{R}}) - D[P_f(\mathbf{R}|\tilde{\mathbf{R}})||P(\mathbf{R}|\tilde{\mathbf{R}})]. \end{aligned} \quad (12)$$

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

$$\begin{aligned} \mathcal{L}_{DeCA(p)} = -E_{P_f} [\log P(\tilde{\mathbf{R}}|\mathbf{R})] + \alpha D[P_f(\mathbf{R}|\tilde{\mathbf{R}})||P(\mathbf{R})] \\ + (1 - \alpha) D[P(\mathbf{R})||P_f(\mathbf{R}|\tilde{\mathbf{R}})]. \end{aligned} \quad (13)$$

We use a pre-trained and fixed model $f_{\theta'}$ which has the same structure as our target model f_θ 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_μ of DeCA is a simple MF while the involved $f_{\theta'}$ in DeCA(p) has the same structure as the target model.
- DeCA co-trains the auxiliary model g_μ with f_θ but DeCA(p) uses a pre-trained and fixed $f_{\theta'}$ 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 $E[\log P(\tilde{\mathbf{R}}|\mathbf{R})]$. A naive solution to optimize $E[\log P(\tilde{\mathbf{R}}|\mathbf{R})]$ is direct performing calculation as shown in Eq.(17). However, we find that it won't yield satisfactory results. The reason is that computing $E[\log P(\tilde{\mathbf{R}}|\mathbf{R})]$ involves updating three models h_ϕ , h'_ψ and f_θ simultaneously. Then these models will interfere with each other. The corrupted outputs from h_ϕ and h'_ψ will prevent f_θ 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_ϕ or h'_ψ , 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_θ , and also refine one of the h -models according to the outputs of f_θ .

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_ϕ will be trained. Then the term $E[\log P(\tilde{\mathbf{R}}|\mathbf{R})]$ becomes

$$E[\log P(\tilde{\mathbf{R}}|\mathbf{R})] = \sum_{(u,i)|\tilde{r}_{ui}=1} \log h_\phi(u, i) \cdot (1 - f_\theta(u, i))$$

Table 1: Statistics of the datasets

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

$$-\sum_{(u,i)|\tilde{r}_{ui}=0} C_1 \cdot f_\theta(u, i) + \log(1 - h'_\psi(u, i)) \cdot (1 - f_\theta(u, i)), \quad (14)$$

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 examples as a clean example, as shown in Figure 1(d). That is to say, given $r_{ui} = 0$ we will always have $\tilde{r}_{ui} = 0$ (i.e., $h'_\psi(u, i) = 0$). Thus in this sub-task, only model h'_ψ will be trained. The term $E[\log P(\tilde{\mathbf{R}}|\mathbf{R})]$ is expanded as

$$\begin{aligned} E[\log P(\tilde{\mathbf{R}}|\mathbf{R})] &= \sum_{(u,i)|\tilde{r}_{ui}=0} \log(1 - h'_\psi(u, i)) \cdot f_\theta(u, i) \\ &\quad + \sum_{(u,i)|\tilde{r}_{ui}=1} \log h'_\psi(u, i) \cdot f_\theta(u, i) - C_2 \cdot (1 - f_\theta(u, i)), \end{aligned} \quad (15)$$

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 $E[\log P(\tilde{\mathbf{R}}|\mathbf{R})]$ 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, Modcloth², Adressa³ and Electronics⁴. 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.

²<https://github.com/MengtingWan/marketBias>

³<https://github.com/WenjieWWJ/DenoisingRec>

⁴<https://github.com/MengtingWan/marketBias>

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 Adressavisen. 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.

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].

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	0.103±0.006	0.041±0.002	0.064±0.003
	WBPR	0.023±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
	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
	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
	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
	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*
CDAE								
Normal	0.017±0.000	0.095±0.002	0.028±0.004	0.052±0.001	0.025±0.000	0.106±0.001	0.048±0.000	0.065±0.000
	WBPR	0.017±0.003	0.087±0.004	0.028±0.003	0.050±0.002	0.031±0.000	0.093±0.001	0.054±0.000
	T-CE	0.014±0.002	0.095±0.007	0.024±0.004	0.050±0.001	0.011±0.001	0.080±0.007	0.020±0.002
	Ensemble	0.015±0.003	0.099±0.000	0.026±0.004	0.052±0.001	0.026±0.001	0.105±0.002	0.049±0.001
	DeCA	0.029±0.001*	0.109±0.000	0.048±0.001*	0.066±0.000	0.027±0.000	0.104±0.001	0.044±0.002
	DeCA(p)	0.028±0.002	0.110±0.002*	0.047±0.002	0.067±0.001*	0.029±0.001	0.118±0.001*	0.055±0.001
LightGCN								
Modcloth	R@5	R@20	N@5	N@20	R@5	R@20	N@5	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
	WBPR	0.067±0.002	0.224±0.002	0.046±0.001	0.090±0.001	0.092±0.001	0.247±0.020	0.064±0.001
	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
	Ensemble	0.064±0.004	0.228±0.007	0.043±0.002	0.089±0.002	0.090±0.003	0.260±0.004	0.061±0.001
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
CDAE								
Normal	0.082±0.004	0.242±0.003	0.052±0.002	0.098±0.001	0.065±0.001	0.220±0.002	0.043±0.001	
WBPR	0.079±0.000	0.238±0.004	0.050±0.000	0.095±0.001	0.072±0.001	0.222±0.002	0.046±0.001	
T-CE	0.075±0.003	0.243±0.008	0.048±0.002	0.096±0.002	0.071±0.000	0.231±0.001	0.049±0.000	
Ensemble	0.084±0.002	0.250±0.002	0.054±0.001	0.100±0.001	0.068±0.001	0.227±0.001	0.046±0.000	0.090±0.000
	DeCA	0.086±0.004	0.250±0.003	0.056±0.001	0.102±0.000	0.064±0.001	0.221±0.002	0.041±0.000
	DeCA(p)	0.089±0.004*	0.251±0.005	0.057±0.003*	0.103±0.003*	0.073±0.001	0.235±0.001	0.051±0.001
LightGCN								
Adressa	R@5	R@20	N@5	N@20	R@5	R@20	N@5	N@20
	GMF				NeuMF			
	Normal	0.116±0.003	0.209±0.005	0.080±0.002	0.112±0.001	0.169±0.004	0.312±0.004	0.131±0.002
	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
	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
	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
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
CDAE								
Normal	0.162±0.000	0.317±0.001	0.123±0.000	0.178±0.000	0.085±0.004	0.215±0.005	0.064±0.003	
WBPR	0.161±0.001	0.315±0.005	0.121±0.002	0.173±0.004	0.118±0.003	0.211±0.006	0.089±0.002	
T-CE	0.161±0.001	0.317±0.003	0.122±0.002	0.176±0.004	0.119±0.001	0.206±0.003	0.091±0.001	
Ensemble	0.162±0.000	0.317±0.001	0.122±0.000	0.176±0.000	0.104±0.001	0.217±0.001	0.078±0.001	0.118±0.001
	DeCA	0.162±0.000	0.319±0.000	0.123±0.000	0.178±0.000	0.112±0.001	0.221±0.001	0.077±0.005
	DeCA(p)	0.163±0.000	0.320±0.002	0.123±0.000	0.178±0.001	0.121±0.001	0.222±0.001	0.089±0.001
LightGCN								

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 α . We apply a grid search for hyperparameters: C_1 and C_2 are searched among {1, 10, 100, 1000}, α 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

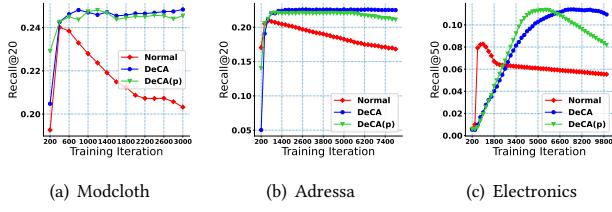


Figure 3: Recall along the training process

best performance compared with normal training and other denoising methods, except for few cases. Even if the CDAE model 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 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 $P(\tilde{\mathbf{R}}|\mathbf{R})$ 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 $P(\tilde{\mathbf{R}}|\mathbf{R})$ 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:

$$\begin{aligned} P(r_{ui} = 1 | \tilde{r}_{ui} = 1) &= \frac{P(\tilde{r}_{ui} = 1 | r_{ui} = 1)P(r_{ui} = 1)}{P(\tilde{r}_{ui} = 1)} \\ &= h'_\psi(u, i)f_\theta(u, i) \\ &= h'_\psi(u, i)f_\theta(u, i) + h_\phi(u, i)(1 - f_\theta(u, i)) \end{aligned} \quad (16)$$

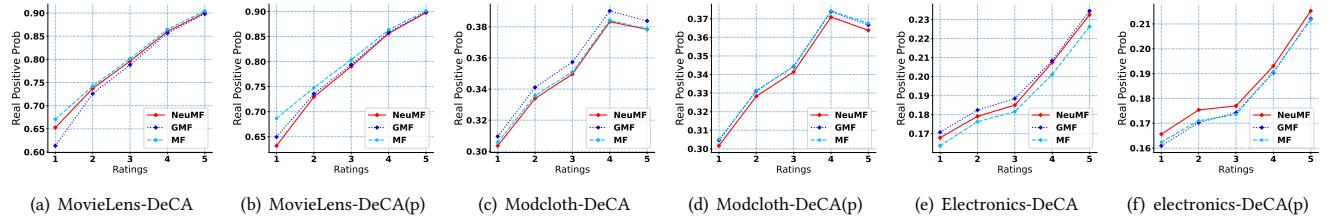
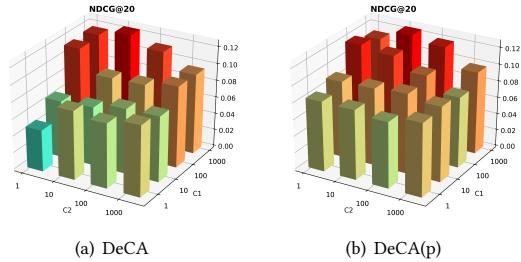
For DeCA(p), $P(r_{ui} = 1 | \tilde{r}_{ui} = 1)$ can be directly computed as $P(r_{ui} = 1 | \tilde{r}_{ui} = 1) = f_\theta(u, i)$ with condition $\tilde{r}_{ui} = 1$. $P(r_{ui} = 1 | \tilde{r}_{ui} = 1)$ describes the probability of an interacted sample to be real positive. Figure 4 shows the relationship 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

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.

	GMF				NeuMF				CDAE				LightGCN			
	R@5	R@20	N@5	N@20												
DeCA-DP	0.072	0.245	0.050	0.098	0.089	0.256	0.061	0.109	0.077	0.236	0.050	0.095	0.066	0.223	0.042	0.086
DeCA-DN	0.051	0.187	0.032	0.071	0.045	0.198	0.027	0.070	0.066	0.236	0.044	0.093	0.068	0.240	0.047	0.096
DeCA	0.072	0.245	0.050	0.099	0.099	0.268	0.065	0.113	0.086	0.250	0.056	0.102	0.064	0.221	0.041	0.085
DeCA(p)-DP	0.075	0.243	0.052	0.099	0.094	0.268	0.064	0.113	0.091	0.245	0.059	0.102	0.065	0.220	0.042	0.086
DeCA(p)-DN	0.060	0.221	0.039	0.084	0.092	0.255	0.062	0.108	0.073	0.238	0.050	0.097	0.072	0.240	0.049	0.096
DeCA(p)	0.074	0.247	0.052	0.100	0.087	0.265	0.059	0.110	0.089	0.251	0.057	0.103	0.073	0.235	0.051	0.096

**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.**

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

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] 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 HOP-Rec [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

- [1] Chong Chen, Min Zhang, Chenyang Wang, Weizhi Ma, Minming Li, Yiqun Liu, and Shaoping Ma. 2019. An efficient adaptive transfer neural network for social-aware recommendation. In *SIGIR*. 225–234.
- [2] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2020. Bias and Debias in Recommender System: A Survey and Future Directions. *arXiv preprint arXiv:2010.03240* (2020).
- [3] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, et al. 2010. The YouTube video recommendation system. In *Recsys*. 293–296.
- [4] Jingtao Ding, Fuli Feng, Xiangnan He, Guanghui Yu, Yong Li, and Depeng Jin. 2018. An improved sampler for bayesian personalized ranking by leveraging view data. In *WWW*. 13–14.
- [5] Jingtao Ding, Yuhua Quan, Xiangnan He, Yong Li, and Depeng Jin. 2019. Reinforced Negative Sampling for Recommendation with Exposure Data. In *IJCAI*. 2230–2236.
- [6] Jingtao Ding, Guanghui Yu, Xiangnan He, Fuli Feng, Yong Li, and Depeng Jin. 2019. Sampler design for bayesian personalized ranking by leveraging view data. *IEEE Trans. Knowl. Data. Eng.* (2019).
- [7] Carl Doersch. 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908* (2016).
- [8] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2012. Personalized ranking for non-uniformly sampled items. In *KDD Cup*. 231–247.
- [9] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2012. Personalized ranking for non-uniformly sampled items. In *KDD Cup 2011*. 231–247.
- [10] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *NeurIPS* 31 (2018).
- [11] F. Maxwell Harper and Joseph A. Konstan. 2016. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4 (2016), 19:1–19:19.
- [12] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *SIGIR*. 355–364.
- [13] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgc: Simplifying and powering graph convolution network for recommendation. In *SIGIR*. 639–648.
- [14] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*. 549–558.
- [15] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM*. 263–272.
- [16] Rolf Jagerman, Harrie Oosterhuis, and Maarten de Rijke. 2019. To model or to intervene: A comparison of counterfactual and online learning to rank from user interactions. In *SIGIR*. 15–24.
- [17] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. Mentor-net: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*. 2304–2313.
- [18] Youngho Kim, Ahmed Hassan, Ryen W White, and Imed Zitouni. 2014. Modeling dwell time to predict click-level satisfaction. In *WSDM*. 193–202.
- [19] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
- [20] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *WWW*. 689–698.
- [21] Tzu-Heng Lin, Chen Gao, and Yong Li. 2019. Cross: Cross-platform recommendation for social e-commerce. In *SIGIR*. 515–524.
- [22] Chao Liu, Ryen W White, and Susan Dumais. 2010. Understanding web browsing behaviors through Weibull analysis of dwell time. In *SIGIR*. 379–386.
- [23] Hongyu Lu, Min Zhang, and Shaoping Ma. 2018. Between clicks and satisfaction: Study on multi-phase user preferences and satisfaction for online news reading. In *SIGIR*. 435–444.
- [24] Hongyu Lu, Min Zhang, Weizhi Ma, Ce Wang, Feng Xia, Yiqun Liu, Leyu Lin, and Shaoping Ma. 2019. Effects of User Negative Experience in Mobile News Streaming. In *SIGIR*. 705–714.
- [25] Duc Tam Nguyen, Chaithanya Kumar Mummadi, Thi Phuong Nhun Ngo, Thi Hoai Phuong Nguyen, Laura Beggel, and Thomas Brox. 2019. Self: Learning to filter noisy labels with self-ensembling. *arXiv preprint arXiv:1910.01842* (2019).
- [26] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *EMNLP-IJCNLP*. 188–197.
- [27] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars BPR Schmidt-Thieme. 2014. Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.
- [28] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural collaborative filtering vs. matrix factorization revisited. In *Recsys*. 240–248.
- [29] Yanyao Shen and Sujay Sanghavi. 2019. Learning with bad training data via iterative trimmed loss minimization. In *ICML*. 5739–5748.
- [30] Ilya Shenbin, Anton Alekseev, Elena Tutubalina, Valentin Malykh, and Sergey I. Nikolenko. 2020. RecVAE: A New Variational Autoencoder for Top-N Recommendations with Implicit Feedback. In *WSDM*. 528–536.
- [31] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. 2019. Meta-weight-net: Learning an explicit mapping for sample weighting. In *NeurIPS*, Vol. 32.
- [32] Aäron Van Den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *NeurIPS*, Vol. 26.
- [33] Mengting Wan, Jianmo Ni, Rishabh Misra, and Julian McAuley. 2020. Addressing Marketing Bias in Product Recommendations. In *WSDM*. 618–626.
- [34] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models. In *SIGIR*. 515–524.
- [35] Wenjie Wang, Fuli Feng, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2020. Denoising Implicit Feedback for Recommendation. In *WSDM*. 373–381.
- [36] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *SIGKDD*. 950–958.
- [37] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*. 165–174.
- [38] Xiang Wang, Yaokun Xu, Xiangnan He, Yixin Cao, Meng Wang, and Tat-Seng Chua. 2020. Reinforced negative sampling over knowledge graph for recommendation. In *WWW*. 99–109.
- [39] Hongyi Wen, Longqi Yang, and Deborah Estrin. 2019. Leveraging post-click feedback for content recommendations. In *Recsys*. 278–286.
- [40] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *WSDM*. 153–162.
- [41] Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, and Ming-Feng Tsai. 2018. HOP-rec: high-order proximity for implicit recommendation. In *Recsys*. 140–144.
- [42] Xing Yi, Liangjie Hong, Erheng Zhong, Nanthan Nan Liu, and Suju Rajan. 2014. Beyond clicks: dwell time for personalization. In *Recsys*. 113–120.
- [43] Wenhui Yu and Zheng Qin. 2020. Sampler Design for Implicit Feedback Data by Noisy-label Robust Learning. In *SIGIR*. 861–870.
- [44] Fajie Yuan, Guibing Guo, Joemon M Jose, Long Chen, Haitao Yu, and Weinan Zhang. 2016. Lambdafm learning optimal ranking with factorization machines using lambda surrogates. In *CIKM*. 227–236.
- [45] Fajie Yuan, Xin Xin, Xiangnan He, Guibing Guo, Weinan Zhang, Chua Tat-Seng, and Joemon M Jose. 2018. fbgd: Learning embeddings from positive unlabeled data with bgd. In *UAI*. 198–207.
- [46] Qian Zhao, Shuo Chang, F Maxwell Harper, and Joseph A Konstan. 2016. Gaze prediction for recommender systems. In *Recsys*. 131–138.

A SUPPLEMENT

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

Algorithm 1: Learning algorithm for DeCA

Input: Corrupted data $\tilde{\mathbf{R}}$, learning rate β , epochs T , hyper-parameter α, C_1, C_2 , regularization weight λ , target recommender f , auxiliary model g, h, h'

Output: Parameters θ, μ, ϕ, ψ 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 $\{(u, i) | \tilde{r}_{ui} = 1\}$;
- 5 Draw (u, i_-) from $\{(u, i) | \tilde{r}_{ui} = 0\}$ for each (u, i_+) ;
- 6 **if** $count \% 2 == 0$ **then**
- 7 | Compute \mathcal{L}_{DeCA} according to Eq.(9) and Eq.(14);
- 8 **else**
- 9 | Compute \mathcal{L}_{DeCA} according to Eq.(9) and Eq.(15);
- 10 Add regularization term: $\mathcal{L}_{DeCA} \leftarrow \mathcal{L}_{DeCA} + \lambda ||\theta||^2$;
- 11 **for** each parameter Θ in $\{\theta, \mu, \phi, \psi\}$ **do**
- 12 | Compute $\partial \mathcal{L}_{DeCA} / \partial \Theta$ by back-propagation;
- 13 | $\Theta \leftarrow \Theta - \beta \partial \mathcal{L}_{DeCA} / \partial \Theta$
- 14 **end**
- 15 $count \leftarrow count + 1$;
- 16 **end**

Algorithm 2: Learning algorithm for DeCA(p)

Input: Corrupted data $\tilde{\mathbf{R}}$, learning rate β , epochs T , hyper-parameter α, C_1, C_2 , regularization weight λ , target recommender f , auxiliary model h, h'

Output: Parameters θ, ϕ, ψ for f, h, h' , correspondingly

- 1 Set random seed to s_1 , initialize another copy of θ as θ' ;
- 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}_{BCE} on (u, i_+) and (u, i_-) with $f_{\theta'}$;
- 6 Add regularization: $\mathcal{L}_{BCE} \leftarrow \mathcal{L}_{BCE} + \lambda ||\theta'||^2$;
- 7 Compute $\partial \mathcal{L}_{BCE} / \partial \theta'$ by back-propagation;
- 8 $\theta' \leftarrow \theta' - \beta \partial \mathcal{L}_{BCE} / \partial \theta'$;
- 9 **end**
- 10 Freeze θ' , set random seed to s_2 and initialize θ, ϕ, ψ ;
- 11 $count \leftarrow 0$;
- 12 **while** Not early stop and epoch $< T$ **do**
- 13 Draw (u, i_+) and (u, i_-) similarly with line3-4;
- 14 **if** $count \% 2 == 0$ **then**
- 15 | Compute $\mathcal{L}_{DeCA(p)}$ according to Eq.(13) and Eq.(14);
- 16 **else**
- 17 | Compute $\mathcal{L}_{DeCA(p)}$ according to Eq.(13) and Eq.(15);
- 18 Add regularization term: $\mathcal{L}_{DeCA(p)} \leftarrow \mathcal{L}_{DeCA(p)} + \lambda ||\theta||^2$;
- 19 **for** each parameter Θ in $\{\theta, \phi, \psi\}$ **do**
- 20 | Compute $\partial \mathcal{L}_{DeCA(p)} / \partial \Theta$ by back-propagation;
- 21 | $\Theta \leftarrow \Theta - \beta \partial \mathcal{L}_{DeCA(p)} / \partial \Theta$;
- 22 **end**
- 23 $count \leftarrow count + 1$;
- 24 **end**

A.2 Mathematical Formulations

The detailed formulation of the term $E_{\mathbf{R} \sim P_f} [\log P(\tilde{\mathbf{R}}|\mathbf{R})]$ in \mathcal{L}_{DeCA} is shown as:

$$E_{\mathbf{R} \sim P_f} [\log P(\tilde{\mathbf{R}}|\mathbf{R})] = \sum_{(u,i)} E_{r_{ui} \sim P_f} [\log P(\tilde{r}_{ui}|r_{ui})]$$

$$\begin{aligned}
 &= \sum_{(u,i)|\tilde{r}_{ui}=1} \left\{ \begin{array}{l} \log P(\tilde{r}_{ui} = 1|r_{ui} = 1) \cdot P_f(r_{ui} = 1) \\ + \log P(\tilde{r}_{ui} = 1|r_{ui} = 0) \cdot P_f(r_{ui} = 0) \end{array} \right. \\
 &+ \sum_{(u,i)|\tilde{r}_{ui}=0} \left\{ \begin{array}{l} \log P(\tilde{r}_{ui} = 0|r_{ui} = 1) \cdot P_f(r_{ui} = 1) \\ + \log P(\tilde{r}_{ui} = 0|r_{ui} = 0) \cdot P_f(r_{ui} = 0) \end{array} \right. \\
 &= \sum_{(u,i)|\tilde{r}_{ui}=1} \log h'_\psi(u, i) \cdot f_\theta(u, i) + \log h_\phi(u, i) \cdot (1 - f_\theta(u, i)) \\
 &+ \sum_{(u,i)|\tilde{r}_{ui}=0} \log(1 - h'_\psi(u, i)) \cdot f_\theta(u, i) + \log(1 - h_\phi(u, i)) \cdot (1 - f_\theta(u, i)). \tag{17}
 \end{aligned}$$

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

$$D[P_g||P_f] = g_\mu(u, i) \cdot \log \frac{g_\mu(u, i)}{f_\theta(u, i)} + (1 - g_\mu(u, i)) \cdot \log \frac{1 - g_\mu(u, i)}{1 - f_\theta(u, i)}.$$

$D[P_f||P_g]$ is computed similarly with $D[P_g||P_f]$.

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 \mathbf{R} is the latent variable. $P_f(\mathbf{R}|\tilde{\mathbf{R}})$ which is parameterized by our target model f_θ maps the corrupted data $\tilde{\mathbf{R}}$ to the latent variables \mathbf{R} and thus can be seen as the encoder. The likelihood $P(\tilde{\mathbf{R}}|\mathbf{R})$ describes the distribution of corrupted data $\tilde{\mathbf{R}}$ given the real user preference \mathbf{R} and acts as the decoder. Finally, $P(\mathbf{R})$ 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 reparameterization 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 objective has the same target as cross-entropy loss as follows.

If we simply replace $-\log(1 - h'_\psi(u, i))$ and $-\log(1 - h_\phi(u, i))$ with C_1 and C_2 in the term $E[\log P(\tilde{\mathbf{R}}|\mathbf{R})]$, we could obtain the following expansion:

$$E[\log P(\tilde{\mathbf{R}}|\mathbf{R})] = - \sum_{(u,i)|\tilde{r}_{ui}=0} C_1 \cdot f_\theta(u, i) - \sum_{(u,i)|\tilde{r}_{ui}=1} C_2 \cdot (1 - f_\theta(u, i)) \tag{18}$$

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 *.

Electronics	R@10	R@50	N@10	N@50	R@10	R@50	N@10	N@50
	GMF				NeuMF			
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	0.072±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	0.063±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*	0.104±0.000	0.023±0.000	0.037±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	0.200±0.001	0.047±0.000	0.074±0.000
CDAE								
Normal	0.074±0.001	0.197±0.001	0.047±0.000	0.073±0.000	0.024±0.000	0.081±0.001	0.016±0.000	0.030±0.000
WBPR	0.077±0.000	0.190±0.000	0.048±0.000	0.072±0.000	0.022±0.000	0.076±0.001	0.015±0.000	0.028±0.000
T-CE	0.067±0.002	0.187±0.010	0.044±0.001	0.067±0.003	0.024±0.001	0.086±0.001	0.017±0.000	0.031±0.000
Ensemble	0.043±0.000	0.110±0.001	0.036±0.000	0.056±0.000	0.018±0.000	0.049±0.001	0.014±0.000	0.023±0.000
DeCA	0.077±0.000	0.201±0.001	0.048±0.000	0.075±0.000	0.025±0.000	0.103±0.001	0.018±0.000	0.036±0.000
DeCA(p)	0.077±0.000	0.202±0.001	0.048±0.000	0.075±0.000	0.026±0.001	0.104±0.001*	0.019±0.000	0.037±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		0.0214	0.0953	0.0352	0.0593	0.1163	0.2089	0.0801	0.1125	0.0227	0.0633	0.0128	0.0214
DeCA	MF	0.0217	0.1089	0.0382	0.0643	0.1255	0.2253	0.0907	0.1250	0.0451	0.1040	0.0234	0.0365
	GMF	0.0186	0.1054	0.0351	0.0634	0.1239	0.2259	0.0908	0.1260	0.0447	0.1047	0.0235	0.0368
	NeuMF	0.0225	0.1077	0.0349	0.0621	0.1268	0.2270	0.0915	0.1260	0.0447	0.1042	0.0233	0.0365
DeCA(p)	MF	0.0269	0.0987	0.0538	0.0658	0.1233	0.2203	0.0932	0.1270	0.0411	0.1102	0.0222	0.0372
	GMF	0.0262	0.0980	0.0536	0.0646	0.1221	0.2182	0.0912	0.1245	0.0428	0.1125	0.0235	0.0386
	NeuMF	0.0255	0.0963	0.0539	0.0653	0.1216	0.2178	0.0906	0.1240	0.0429	0.1115	0.0237	0.0386

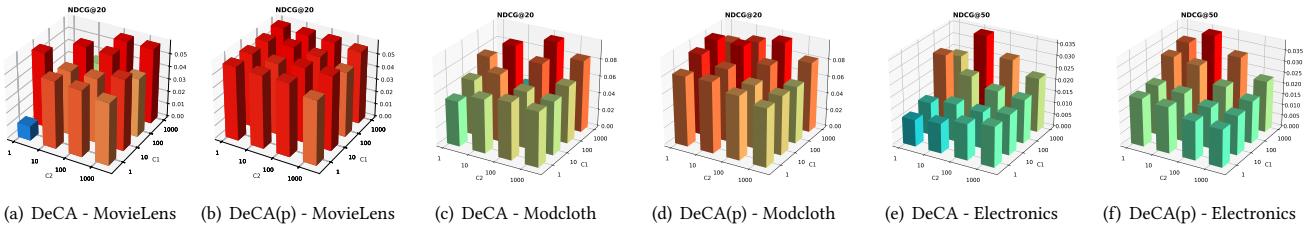


Figure 6: Hyperparameter study on MovieLens, Modcloth and Electronics.

the MF model g_μ . However, since MF is the simplest model, the time complexity will be much smaller than twice the complexity of normal training.

For DeCA(p), the total time complexity will be almost exactly twice of normal training, since the complexity of the second time is almost the same as the first time, i.e., normal training.

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.

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.