Self-Guided Learning to Denoise for Robust Recommendation

Yunjun Gao, Yuntao Du, Yujia Hu,
Lu Chen, Xinjun Zhu, Ziquan Fang
College of Computer Science, Zhejiang University, China
{gaoyj,ytdu,yjhu,luchen,xjzhu,zqfang}@zju.edu.cn

Baihua Zheng
School of Computing and Information Systems,
Singapore Management University, Singapore
bhzheng@smu.edu.sg

ABSTRACT

The ubiquity of implicit feedback makes them the default choice to build modern recommender systems. Generally speaking, observed interactions are considered as positive samples, while unobserved interactions are considered as negative ones. However, implicit feedback is inherently noisy because of the ubiquitous presence of *noisy-positive* and *noisy-negative* interactions. Recently, some studies have noticed the importance of denoising implicit feedback for recommendations, and enhanced the robustness of recommendation models to some extent. Nonetheless, they typically fail to (1) capture the hard yet clean interactions for learning comprehensive user preference, and (2) provide a universal denoising solution that can be applied to various kinds of recommendation models.

In this paper, we thoroughly investigate the memorization effect of recommendation models, and propose a new denoising paradigm, *i.e.,* Self-Guided Denoising Learning (SGDL), which is able to collect memorized interactions at the early stage of the training (*i.e.,* "noise-resistant" period), and leverage those data as denoising signals to guide the following training (*i.e.,* "noise-sensitive" period) of the model in a meta-learning manner. Besides, our method can automatically switch its learning phase at the memorization point from memorization to self-guided learning, and select clean and informative memorized data via a novel adaptive denoising scheduler to improve the robustness. We incorporate SGDL with four representative recommendation models (*i.e.,* NeuMF, CDAE, NGCF and Light-GCN) and different loss functions (*i.e.,* binary cross-entropy and BPR loss). The experimental results on three benchmark datasets demonstrate the effectiveness of SGDL over the state-of-the-art denoising methods like T-CE, IR, DeCA, and even state-of-the-art robust graph-based methods like SGCN and SGL.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → *Learning from implicit feedback.*

KEYWORDS

Recommender System; Denoising Recommendation; Implicit Feedback; Robust Learning

1 INTRODUCTION

Recommender systems have been widely deployed to perform personalized information filtering, especially for various online services such as E-commerce [17, 30], social media [33], and news portals [38]. Most of existing recommender systems use implicit feedback (*e.g.,* view and click) to develop machine learning models, due to its large volume and availability [13, 14, 21, 34]. Specifically, the observed interactions between users and items are viewed as positive instances, while unobserved interactions are viewed as negative instances. However, implicit feedback is inherently noisy because of the ubiquitous presence of *noisy-positive* and *noisy-negative* interactions [3, 24, 28, 42]. Take the E-commerce scenario as an example. A large portion of click behaviors are triggered by the curiosity of users, which cannot directly indicate the users' positive views of the products. On the other hand, unobserved interactions may attribute to the unawareness of users because the items are simply not exposed to them. Hence, blindly fitting the implicit feedback to recommender systems without considering the inherent noise would fail to understand users' true preferences, and eventually harm user experiences and degrade recommendation performance [39, 46].

Considering the widespread use of implicit feedback and its large impact on the recommendation model, some recent studies have noticed the importance of denoising implicit feedback for recommendations. Existing efforts on tackling this problem can be roughly divided into two categories: *sample selection methods* [12, 16, 42, 50] and *sample re-weighting methods* [23, 39, 41]. Sample selection methods focus on designing more effective samplers to collect clean samples for learning users' preferences, while their performance suffers from high variance since they heavily depend on the sampling distribution [51]. On the other hand, sample re-weighting methods aim to distinguish noisy interactions from clean data in terms of the loss values, and assign lower weights to noisy interactions with high loss values during training. Their key idea is consistent with *memorization effect* [2]: models tend to initially learn easy and clean patterns (*i.e.,* user preferences) in the early stage of their learning, and eventually memorize all training interactions. Benefiting from this principle, sample re-weighting methods are able to successfully identify noisy interactions.
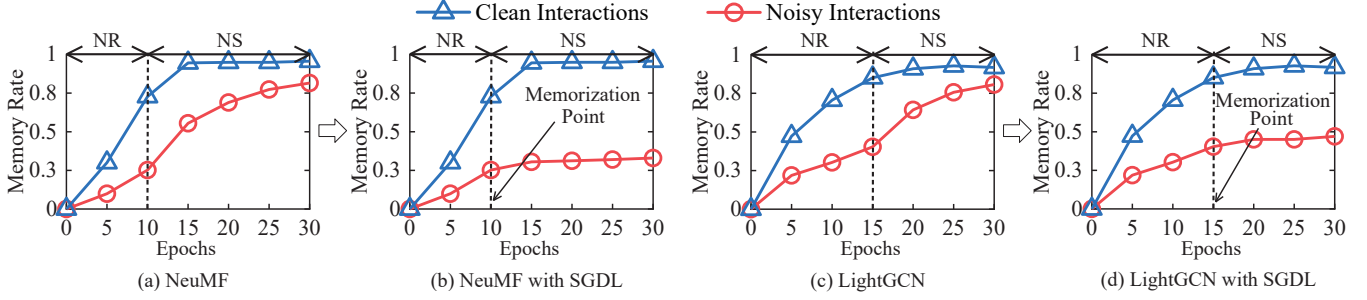
Figure 1: Key idea of SGDL: (a) and (c) show the memory rate when training NeuMF and LightGCN on the MovieLens dataset, respetively; (b) and (d) show the memory rate when training the corresponding model with SGDL. The memory rate is the proportion of memorized data (see Section 3.1.1 for details) in the clean and noisy interactions. NR denotes "noise-resistant" period and NS represents "noise-sensitive" period, respectively.

Although sample re-weighting methods can achieve promising performance for denoising implicit feedback, we want to highlight that they commonly suffer from following *two* problems:

- **Abandon of Hard Clean Interactions.** These methods heavily rely on the loss values. They simply assume that interactions with large loss values are noisy and penalize them with small weights. However, it has been reported that some clean interactions (*aka.* hard yet clean interactions) may also have high loss values at the beginning of the training, and those interactions play an important role in understanding users' behaviors and preferences [11, 23, 41]. Nonetheless, these interactions are simply discarded by existing methods due to high loss values, incurring insufficient understanding of users' true preferences.

- **Lack of Adaptivity and Universality.** Existing sample re-weighting methods are able to achieve good performance *only* when the reweighting configurations (e.g., weights and thresholds) are properly specified. However, obtaining proper configurations normally requires time-consuming procedures (*e.g.,* grid search), and the optimal configurations for one dataset typically are not applicable for other datasets because of the different data distributions. Meanwhile, these methods are only applicable to the recommender models with predefined pointwise loss function (*e.g.,* cross-entropy loss in [39, 41]), which makes it hard to incorporate other popular ranking loss functions (*e.g.,* BPR loss [34]) and limits their applications.

In this regard, we have thoroughly explored the memorization of the representative recommendation models (*e.g.,* NeuMF [20] and LightGCN [19]) on implicit feedback, including the clean interactions and the noisy interactions. By analyzing the learning processes of different models, we have observed the existence of two learning periods. As shown in Figure 1, a typical learning process consists of the "noise-resistant" (NR) period and the "noise-sensitive" (NS) period. The former is the duration when the memorization of noisy interactions is insignificant because the models focus on memorizing easy and clean patterns at their early stage of training; while the latter is the duration when the memorization of noisy interactions rapidly increases since models eventually begin memorizing all the implicit feedback at the late stage of training. The timestamp in the training process that can best differentiate above two periods is defined as *memorization point* of the model training, which are represented by dotted lines in Figure 1.

These observations motivate us to design a new approach that better leverages the memorization nature of recommendation models. Specifically, we propose a new denoising paradigm, namely, S̲elf-G̲uided D̲enoising L̲earning (SGDL), which is able to collect memorized interactions at the early stage of training (*i.e.,* "noise-resistant" period), and leverage those data as denoising signals to guide the subsequent training (*i.e.,* "noise-sensitive" period) of the model in a meta-learning manner. Besides, SGDL can automatically transit from noise-resistant period to noise-sensitive period in order to stop accumulating memorized interactions at the *memorization point*, since the noisy interactions are gradually memorized with the training process. In a nutshell, corresponding to the two observed periods commonly existing in the training process of different models, SGDL contains two key phases, *i.e., memorization* and *self-guided learning*:

- **Memorization.** Owing to the negligible memorization of noisy interactions during "noise-resistant" period, the model is initially trained with all the implicit feedback. To better reveal the underlying memorization nature of the model during training, we design the new memorization-based metrics to define the memorization states of data. As the memorized data are mostly easy and clean interactions at the early stage of training, we collect them as denoising signals to guide the following denoising training process. Moreover, SGDL is able to automatically estimate the best memorization point, from which the learning moves from the "noise-resistant" period to "noise-sensitive" period, to stop accumulating memorized data without *any* supervision.

- **Self-Guided Learning.** To avoid the memorization of noisy interactions in the "noise-sensitive" period, we leverage the memorized data collected from the "noise-resistant" period to represent user preferences. Specifically, a denoising module is proposed to learn a parameterized weighting function for the implicit feedback, which is guided by memorized data and updated with the learning process of the model. Moreover, since some of the memorized data can also be noisy, we further develop a novel adaptive denoising scheduler to prevent the denoising module from being corrupted by noisy yet memorized samples. Technically, the adaptive denoising scheduler characterizes the contribution of each memorized data to the denoising performance, and decides whether to use the sample by predicting its probability being sampled. The scheduler is also simultaneously optimized together with the learning process to enhance the robustness of the model.

Through the above two phases, SGDL can denoise implicit feedback with the help of memorized data, which are naturally collected by exploiting the noise-resistant period of training. Compared with standard training, SGDL can dramatically reduce the memory ratio of noisy samples in the noise-sensitive period, and enhance the robustness of recommendation models, as shown in Figure 1. Moreover, since the model is constantly trained with all implicit feedback data, our method can help the model learn users' true preferences with hard yet clean samples, leading to better recommendation performance. Last but not the least, our method does not need *any* thresholds or predefined weighting functions, and is easy to be applied to *any* learning-based recommendation models. We conduct extensive experiments on three real-world datasets with four representative recommendation models (*i.e.,* NeuMF [20], CDAE [45], NGCF [40], and LightGCN [19]) and two loss functions (*i.e.,* binary cross-entropy loss and BPR loss [34]). Experimental results show that our SGDL significantly outperforms all state-of-the-art denoising methods, and it achieves comparable (in many cases even better) performance to the state-of-the-art robust graph-based methods like SGCN [5] and SGL [43].

In summary, we make three key contributions in this paper, as listed below.

- We develop a new denoising paradigm, *i.e., self-guided denoising learning (SGDL)*, which leverages the self-labeled memorized data as guidance to offer denoising signals for robust recommendation without defining *any* weighting functions or requiring *any* auxiliary information.
- We carefully exploit the memorization effect of recommendation models, and design two training phases that can collect memorized data and utilize them as guidance to denoise implicit feedback, respectively. Besides, a novel adaptive denoising scheduler is introduced to further improve the robustness.
- We incorporate SGDL with four representative recommendation models, and conduct extensive experiments on three public benchmark datasets with various state-of-the-art methods to demonstrate the superiority and universality of SGDL.

2 PROBLEM FORMULATION

We first introduce the common paradigm of user preferences learning from implicit feedback, and then formulate our task.

Preference learning from implicit feedback. In this paper, we focus on learning the user preferences from the implicit feedback [34]. Specifically, the behavior data (*e.g.,* click and review) $\mathcal{D} = \{u, i, y_{ui} | u \in \mathcal{U}, i \in \mathcal{I}\}$ involves a set of users $\mathcal{U} = \{u\}$ and a set of items $\mathcal{I} = \{i\}$, as well as the interaction $y_{ui} = \{0, 1\}$ that indicates whether user u has interacted with item i. Most of the state-of-the-art recommendation methods (*e.g.,* NeuMF [20] and LightGCN [19]) assume that the interaction y_{ui} could represent the users' true preferences, and directly learn the model f with parameters θ by minimizing a ranking loss function over $\mathcal{D}$.

Denoising implicit feedback for recommendations. However, due to the existence of inherent noise in implicit feedback, recommendation models might fail to learn the users' true preferences with typical training process, resulting in suboptimal performance. Thus, the task of the paper is, given the noisy implicit feedback $\mathcal{D}$
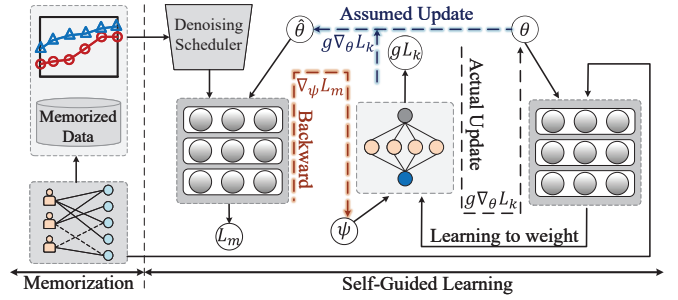
Figure 2: The overall framework of SGDL. During Phase I (memorization), memorized interactions are collected as denoising signals; During Phase II (self-guided learning), the weighting function is simultaneously learned with the recommender model, which is guided by the memorization data in a meta-learning manner (see Section 3.2.1 for details).

that contains both noisy-positive and noisy-negative feedback, to infer users' true preferences with optimal model parameters θ^*.

3 METHODOLOGY

In this section, we detail SGDL that comprises the following two phases: (i) *memorization*, which exploits the memorization effect of models to collect memorized data during the noise-resistant period and estimates the best memorization point to automatically transit to phase II; and (ii) *self-guided learning*, which leverages the memorized samples as denoise signals to guide the model learning during the noise-sensitive period, and discards the potential noisy interactions in memorized data with a novel adaptive denoising scheduler for robustness. We detail the two phases as follows.

3.1 Phase I: Memorization

Initially, Phase I tries to train the recommendation model in a conventional way during the noise-resistant period, where the memorization of noisy interactions is assumed to be suppressed, as we discover in Figure 1. Since most of the memorized interactions until the memorization point are clean, we collect them to form the memorized data, which can be used as the denoising signals to guide the training in Phase II. Therefore, the major challenges in Phase I include 1) how to define the memorization of interactions and 2) how to estimate the memorization point.

3.1.1 **Memorized Interactions.** Previous studies [39, 41] mainly use loss values of training data to demonstrate the memorization effect of recommendation models. However, we argue that loss values are insufficient to reflect the learning process, since they are inconsistent with the optimization target of recommendation models (*i.e.,* personalized ranking), and unable to distinguish hard interactions from noisy ones. Thus, it is necessary to design a new memorization-based metric, which takes the learning process of recommendation models into consideration.

Inspired by the widely used hit ratio metric [9, 34], we define that an interaction (u, i) is *memorized* at epoch t by the model θ if item i is in the ranking list of user u, denoted as $m_t(u, i)$. To ensure the reliability of ranking results, for each user u, we include the top-N of the ranking items into its ranking list, where N is the length of observed interactions. However, simply calculating
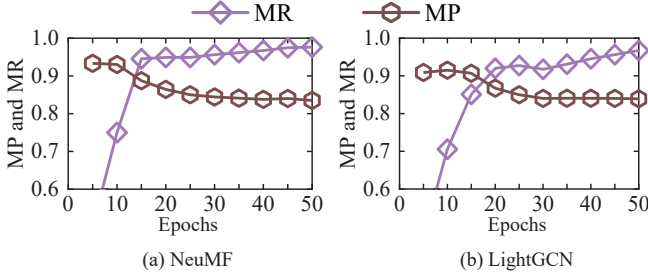
Figure 3: The monotonicity of MP and MR when training NeuMF and LightGCN on the MovieLens dataset.

Figure 4: Loss distribution of MovieLens dataset.

the memorization of interactions of a single epoch could lead to unstable results, since the model is not well trained during the noise-resistant period. Hence, we trace the memorization states of interactions for the most recent h epochs, and define the final memorization of interaction (u, i) as follows:

$$m_t^h(u, i) = \frac{1}{|\mathcal{P}_t^h(u, i)|} \sum_{m_j(u,i) \in \mathcal{P}_t^h(u,i)} m_i(u, i) \qquad (1)$$

where $\mathcal{P}_t^h(u, i) = \{m_{t-h+1}(u, i), \cdots, m_t(u, i)\}$ captures the most recent h memorization histories of interaction (u, i). We define an interaction (u, i) *memorized* by a model θ if the majority of the recent histories $\mathcal{P}_t^h(u, i)$ coincides with the memorization state (*i.e.,* $m_t^h(u, i)$ is larger than 0.5). It is worth mentioning that the definition of memorized interactions dose not need *any* labels for supervision, and is able to effectively indicate the underlying memorization effect of recommendation models.

*3.1.2 **Memorization Point Estimation.*** As shown in Figure 1, the model predominantly learns clean interactions until the noise-sensitive period begins. In others words, during noise-resistant period, the recommendation model (1) not only learns *sufficient* information from the clean interactions, (2) but also accumulates *some* noise from the noisy implicit feedback. Therefore, we aim to design two metrics to reflect the above two memorization characteristics of the model. Formally, we use $\mathcal{M}_t$ to denote a set of memorized interactions at epoch t, and use y_{ui}^* to represent the *true* label of interaction (u, i), which is not available because of the noise in implicit feedback. Inspired by recent advances in robust learning [35, 36] and recommendation evaluation metrics, we propose two memorization-based metrics, namely, *memorization precision* (*MP*) and *memorization recall* (*MR*), to address the memorization effect of recommendation models respectively:

$$MP_t = \frac{|\mathcal{R}_t|}{|\mathcal{M}_t|}, \quad MR_t = \frac{|\mathcal{R}_t|}{|\mathcal{G}|} \qquad (2)$$

where $\mathcal{R}_t = \{(u, i) \in \mathcal{M}_t : y_{ui} = y_{ui}^*\}$ denotes the set of memorized data whose true labels are consistent with predictions, and $\mathcal{G} = \{(u, i) \in \mathcal{D} : y_{ui} = y_{ui}^*\}$ is the set of true labeled data in implicit feedback. According to the definition of *MP* and *MR*, we can conclude that: *MP* monotonically *decreases* since the model tends to memorize clean data first and then gradually memorizes all the noisy interactions as the training progresses; and *MR* monotonically *increases* because the model eventually memorizes all clean interactions as the training progresses, as depicted in Figure 3
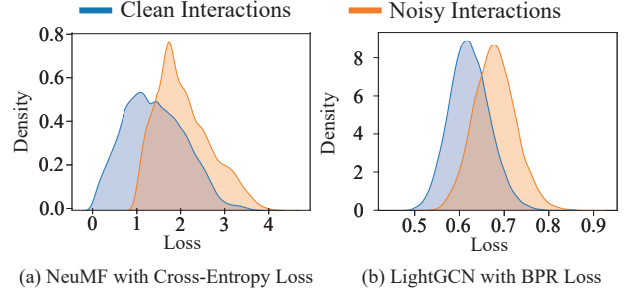
(please refer to Section 3.3 for the theoretical analysis of the monotonicity of the two metrics). Thus, the best memorization point t_m is naturally the best trade-off epoch when *MP* and *MR* share the same value, *i.e.,* $MP_t = MR_t$. By substituting it into Equation (2), the best memorization point t_m can be calculated as:

$$\mathcal{M}_{t_m} = |\{(u, i) \in \mathcal{D}_t : y_{ui} = y_{ui}^*\}| = (1 - \sigma)|\mathcal{D}| \qquad (3)$$

where σ is the noise rate of the implicit feedback. Since σ is typically unknown, we leverage the difference of loss distributions in clean and noisy data to estimate σ, as shown in Figure 4.

Specifically, we first normalize the loss values of all training interactions and then fit them into a two-component Gaussian Mixture Model (GMM) to model the bi-modal distribution [1, 29] of true- and false-labeled samples. The GMM model can be easily trained by using the Expectation-Maximization (EM) algorithm. Hence, we could obtain the probability of an interaction (u, i) being noisy through the posterior probability of loss distributions. Accordingly, the noise rate σ is estimated as:

$$\hat{\sigma} = \mathbb{E}_{(u,i) \in \mathcal{D}} [p(\mu|L_{(u,i)}(\theta)] \qquad (4)$$

where $L_{(u,i)}(\theta)$ is the loss of interaction (u, i) of recommendation model θ, and μ is the Gaussian component with a larger mean, since noisy data have typically larger loss values. Note that there are many approaches [8, 31] available to estimate the noise ratio, we choose GMM because it is easy to apply and has stable performance on different noisy datasets.

Therefore, SGDL transits to phase II when the number of memorized interactions reaches the estimated clean data size (*i.e.,* $|\mathcal{M}_t| \geq (1 - \hat{\sigma})\mathcal{D}$). Note that, SGDL is able to estimate the memorization point t_m, and collect memorized data $\mathcal{M}_{t_m}$ with nearly *zero* computational overhead and *no* additional supervision.

3.2 Phase II: Self-Guided Learning

Phase II aims to robustly learn the recommendation model with the denoising signals of memorized data, which are collected from the training process of Phase I. Previous studies [39, 41] need to pre-specify weighting functions (*e.g.,* exponential function), and require additional hyperparameters (*e.g.,* threshold) to denoise implicit feedback during training. However, we argue that these methods are fairly hard to be generally applied in real-world recommendation systems due to two issues: (1) The proper weighting schemes heavily rely on the training data, which limits their adaptivity. (2) These methods simply abandon the hard yet clean interactions because the fixed weighting functions fail to distinguish the interactions from noisy data, incurring suboptimal recommendation performance.

Thus, we aim to propose a self-guided method that is capable of learning an adaptive weighting function automatically from the collected memorized data to tackle the above two issues.

The above denoising scheme assumes that the memorized data is clean and useful to provide denoising signals for learning user preferences. Nevertheless, as the memorized data is collected by leveraging the memorization effect of recommendation models, it inevitably contains some noise. To prevent the model from being corrupted by such detrimental interactions, we further devise a novel adaptive denoising scheduler to select and use proper memorized data for self-guided learning.

3.2.1 Denoising Learning with Memorized Data.
During the noise-sensitive period, we aim to enhance the robustness of training by imposing a weight on each sample loss. Here, we consider the memorized data that is able to provide denoising signals to learn the weight for each sample, since it is mostly clean thanks to the memorization effect of the model. Specifically, let $L_k(\theta)$ be the k-th sample loss[1] of model θ, and $\mathcal{D}_T$ be the training data. The optimal recommendation model parameter θ is calculated by minimizing the following weighted loss:

$$\theta^*(\psi) = \underset{\theta}{\arg\min} \frac{1}{|\mathcal{D}_T|} \sum_k^{|\mathcal{D}_T|} g(L_k(\theta); \psi) L_k(\theta) \tag{5}$$

where $g(L_k(\theta); \psi)$ is the weight on the k-th sample loss, and ψ represents the current parameters of weighting function $g(\cdot)$. Here, we formulate $g(\cdot)$ as a simple Multi-Layer Perceptron (MLP) with only one hidden layer, since it is known as a universal approximator for almost any continuous function [10, 35]. To adaptively learn the weight for each sample from memorized data, we optimize the weighting parameters ψ given current optimal $\theta^*(\psi)$:[2]

$$\psi^* = \underset{w}{\arg\min} \frac{1}{|\mathcal{M}_{t_m}|} \sum_m^{|\mathcal{M}_{t_m}|} L_m(\theta^*(\psi)) \tag{6}$$

where $L_m(\theta^*(\psi))$ is the m-th memorized sample loss given optimal model parameters $\theta^*(\psi)$. Note that the searching for the optimal θ^* and optimal ψ^* requires two nested loops of optimization (i.e., bi-level optimization) [6, 35], which is costly. Hence, we adopt the idea of meta learning [15, 44], and update θ and ψ alternately in a single loop to guarantee the efficiency of the algorithm. Specifically, as illustrated in Figure 2, we perform the following training procedure in each iteration:

- **Assumed update of θ.** As shown by the blue arrows in Figure 2, we first make an assumed update of θ with current weight ψ:

$$\hat{\theta}(\psi) = \theta - \eta_1 \frac{1}{|\mathcal{D}_T|} \sum_k^{|\mathcal{D}_T|} g(L_k(\theta); \psi) \nabla_\theta L_k(\theta) \tag{7}$$

where we update θ using gradient descent with learning rate η_1.
- **Update of ψ.** As indicated by the brown arrow in Figure 2, the updates of the weighting parameters ψ can be guided by gradient of the memorized data on the updated model:

$$\psi \leftarrow \psi - \eta_2 \frac{1}{|\mathcal{M}_{t_m}|} \sum_m^{|\mathcal{M}_{t_m}|} \nabla_\psi L_m(\hat{\theta}(\psi)) \tag{8}$$

where η_2 is the learning rate of weighting parameters ψ.
- **Actual update of θ.** After receiving the denoising signals (i.e., updated parameters ψ) from memorized data, we use them to update the model:

$$\theta \leftarrow \theta - \eta_1 \frac{1}{|\mathcal{D}_T|} \sum_k^{|\mathcal{D}_T|} g(L_k(\theta); \psi) \nabla_\theta L_k(\theta) \tag{9}$$

We use this alternative strategy to optimize the recommendation model and weighting function during the noise-sensitive period, as illustrated in Figure 2. Although this strategy does not guarantee to find the global optimum, it empirically works well in many bi-level optimization problems [6, 15, 35].

3.2.2 Adaptive Denoising Scheduler.
Above denoising scheme relies on the memorized data to provide denoising signals for training. However, as memorized data inevitably exhibit noise, integrating the noise of them would degrade the denoising performance.

Thus, we propose an adaptive denoising scheduler to select *only* the clean and informative memorized data for denoising learning. Specifically, we define the scheduler as s with parameter ϕ, and choose two representative factors to quantify the contribution of each memorized data for denoising: (1) the loss $L_m(\theta)$ of m-th memorized sample, where θ is the actual updated parameters of the model; and (2) the gradient similarity of m-th memorized sample on assumed updated model parameters $\hat{\theta}$ and actual updated model parameters θ, i.e., $\cos\left(\nabla_{\hat{\theta}} L_m(\hat{\theta}), \nabla_\theta L_m(\theta)\right)$. Here, we use cosine function as the similarity measurement, and other metrics like inner product can also be applied in practice. The two factors are associated with the learning outcome and learning process of the m-th memorized sample, respectively. Specifically, the gradient similarity characterizes the contribution of the memorized sample in the model training. A large loss value may represent a crucial memorized sample if the gradient similarity is also large (i.e., the gradient direction of memorized sample is consistent with the optimization of the model); and a large loss value with small gradient similarity may indicate a noisy memorized sample. Considering the two factors simultaneously, we formulate the sampling probability of m-th memorized data:

$$o_m = s\left(L_m(\theta), \cos(\nabla_{\hat{\theta}} L_m(\hat{\theta}), \nabla_\theta L_m(\theta)); \phi\right) \tag{10}$$

$$\pi_m = \frac{\exp(o_m; \phi)}{\sum_{i \in \mathcal{M}_{t_m}} \exp(o_i; \phi)} \tag{11}$$

where o_m is the output of the scheduler, and π_m is the predicted sampling probability of m-th sample. We choose LSTM network [22] as the scheduler, and feed it with the training factors in each iteration. The intuition behind is that the LSTM could leverage the historical information to capture the prediction variance [4, 48], which shows stable performance in our experiments (different strategies are compared in Section 4.3.3). However, it is intractable to directly optimize the scheduler since the sampling process is not differentiable. To make this procedure differentiable and to jointly optimize the scheduler and the model, we apply the Gumbel-Softmax reparameterization trick [26] to generate differentiable samples:

$$y_m = \frac{\exp(\log(\pi_m) + \epsilon_m)/\tau}{\sum_{i \in \mathcal{M}_{t_m}} \exp(\log(\pi_i) + \epsilon_i)/\tau} \tag{12}$$

---

[1]The loss can be represented as the pointwise or pairwise loss, according to the optimization target of models. We demonstrate the effectiveness of our method on both kinds of loss functions in Section 4.2.

[2]Notice that, here ψ is a variable instead of a quantity, which makes $\theta^*(\psi)$ a function of ψ, and the gradient in Equation (6) can be computed.

where ϵ_m is randomly drawn from uniform distribution between 0 and 1, and τ is the temperature that controls the interpolation between the discrete distribution and continuous categorical densities (we set $\tau = 0.05$ for all experiments). Thus, the scheduler is able to decide which memorized data to use according to its contribution to denoising implicit feedback, and adpatively adjust the sampling probability for more informative guided learning.

3.3 Model Analysis

3.3.1 Analysis on the monotonicity of MP and MR.
We now prove that MP and MR change monotonically over the training time t. Let $\mathcal{N}_t \ (= \mathcal{M}_t \setminus \mathcal{R}_t)$ be the set of memorized data that are falsely predicted by models. Then, $|\mathcal{N}_{t+1}|/|\mathcal{N}_t| \geq |\mathcal{R}_{t+1}|/|\mathcal{R}_t|$ typically holds because the noisy samples are memorized faster than clean samples after the model stabilizes, as depicted in Figure 3. Hence, it is easy to conclude:

$$|\mathcal{N}_{t+1}|/|\mathcal{N}_t| \geq |\mathcal{R}_{t+1}|/|\mathcal{R}_t| \tag{13}$$

$$\implies |\mathcal{N}_{t+1}||\mathcal{R}_t| + |\mathcal{R}_{t+1}||\mathcal{R}_t| \geq |\mathcal{R}_{t+1}||\mathcal{N}_t| + |\mathcal{R}_{t+1}||\mathcal{R}_t| \tag{14}$$

$$\implies (|\mathcal{N}_{t+1}| + |\mathcal{R}_{t+1}|)|\mathcal{R}_t| \geq (|\mathcal{N}_t| + |\mathcal{R}_t|)|\mathcal{R}_{t+1}| \tag{15}$$

Then, $MP_{t+1} \geq MP_t$ can be directly derived from Inequation (15). Besides, considering that the model eventually would memorize all training data [36], we assume that $\mathcal{M}$ would gradually include more observed interactions, including true predicted ones, *i.e.*, $|\mathcal{R}_{t+1}| \geq |\mathcal{R}_t|$. Consequently, MR increases monotonically since $|\mathcal{R}_{t+1}|/|\mathcal{G}| \geq |\mathcal{R}_t|/|\mathcal{G}|$.

3.3.2 Analysis of the Self-Guide Learning Scheme.
We then focus on the guidance strategy to explain how memorized data benefit the denoising training. Formally, we follow [35], and utilize chain rule to derive the update function of ψ:

$$\psi \leftarrow \psi + \frac{\eta_1 \eta_2}{|\mathcal{D}_T|} \sum_{k}^{|\mathcal{D}_T|} \left(\frac{1}{|\mathcal{M}_{t_m}|} \sum_{m}^{|\mathcal{M}_{t_m}|} G_{mk} \right) \nabla_\psi g(L_k(\hat{\theta}); \psi) \tag{16}$$

where $G_{mk} \ (= \nabla_\theta L_m(\hat{\theta})^T \nabla_\theta L_k(\theta))$ measures the gradient similarity between m-th memorized data and k-th training data. Thus, for each k-th training sample, if its gradient is similar to the average gradient of memorized data, it would be considered as a beneficial one for learning, and its weight tends to be increased. Conversely, the weight of the sample inclines to be suppressed. Therefore, memorized data is able to offer a proper weight for each training sample in terms of gradient similarities under the self-guide learning scheme.

3.3.3 Model Size.
The additional parameters of SGDL come from two parts: (1) the parameters $2d_w$ of weighting function, where d_w is the number of the hidden neurons in one layer MLP; and (2) the $4d_l^2 + 12d_l$ parameters of LSTM unit in the adaptive denoising scheduler, where d_l is the dimension of hidden size. Overall, the additional cost of SDGL is negligible, compared with the tremendous parameters of modern recommendation models.

3.3.4 Time Complexity.
Assume that the time complexity of the base model is $O(T)$, the additional complexity of SGDL mainly comes from phase II, which consists of two denoising components: (1) the cost of self-guided learning is also $O(T)$, as the alternative optimization scheme takes no more than three times compared with the normal training; and (2) the computational complexity of

Table 1: Statistics of the datasets used in our experiments.

Dataset	#Users	#Items	#Interactions	Sparsity
Adressa	212,231	6,596	419,491	99.97%
MovieLens	943	1,683	100,000	93.70%
Yelp	45,548	57,396	1,672,520	99.94%

denoising adaptive scheduler is $O(|\mathcal{D}|d_l)$. Therefore, the additional time complexity of SGDL is $O(T + |\mathcal{D}|d_l)$. Under the same experimental settings (*i.e.,* same base model and same embedding size), SGDL achieves better trade-off between efficiency and effectiveness compared with various state-of-the-art models: i) For sample reweighting and sample selection methods, although most of them are more time-saving than SGDL, they suffer from difficult/expensive hyperparameters tuning and unstable performance, as discussed in Section 4.2. ii) For robust graph-based methods, SGDL has a complexity that is comparable with them, since graph-based methods typically leverage on extra graph structure to enhance the robustness of the model.

4 EXPERIMENTS

We provide empirical results to demonstrate the effectiveness of our proposed SGDL. The experiments are designed to answer the following research questions:

- **RQ1:** How does SGDL perform, compared with the state-of-the-art denoising methods as well as the state-of-the-art robust recommender methods?
- **RQ2:** How does each component of SGDL (*i.e.,* memorization point estimation, denoising learning strategy, and adaptive denoising scheduler) affect SGDL?
- **RQ3:** Is SGDL able to distinguish hard yet clean interactions from noisy interactions?

4.1 Experimental Settings

4.1.1 Dataset Description.
We select three real world benchmark datasets to evaluate and compare the performance of SGDL and its competitors. Table 1 lists their statistics.

- **Adressa**[3] is a news reading dataset from Adressavisen, including user click behaviors and the dwell time for each click. Following previous work [39], clicks with dwell time less than 10 seconds are viewed as noisy interactions.
- **Yelp**[4] is an open recommendation dataset released by the Yelp challenge. We use the 2018 version in our experiments. We follow [39] to mark ratings below 3 as noisy interactions.
- **MovieLens** is a widely used dataset for recommendation, which contains 100,000 movie ratings ranging from 1 to 5. Ratings below 3 are regarded as noisy interactions.

4.1.2 Evaluation Metrics.
We adopt cross-validation to verify the performance. Specifically, we follow [39, 41] to split the interactions into the training set, validation set, and clean test set with the ratio of 8:1:1. The performance is measured by two widely used valuation protocols [19, 40]: Recall@K and NDCG@K, where K is set as 5 and 20 by default. We report the average metrics for all users in the test set.

---

[3]https://www.adressa.no/

[4]https://www.yelp.com/dataset/challenge

Table 2: Performance comparison of different denoising methods on robust recommendation. "†" indicates the improvement of the SGDL over the baseline is significant at the level of 0.05. The highest scores are in Bold. R and N refer to Recall and NDCG, respectively.

Database		Adressa				MovieLens				Yelp			
Base Model	Method	R@5	R@20	N@5	N@20	R@5	R@20	N@5	N@20	R@5	R@20	N@5	N@20
NeuMF	Normal	0.1533†	0.3208†	0.1224†	0.1808†	0.1023†	0.2687†	0.2890†	0.2765†	0.0129†	0.0393†	0.0129†	0.0215†
	WBPR	0.1538†	0.3207†	0.1225†	0.1809†	0.1025†	0.2689†	0.2891†	0.2769†	0.0128†	0.0392†	0.0127†	0.0214†
	IR	0.1541†	0.3212†	0.1229†	0.1830†	0.1054†	0.2704†	0.2928†	0.2758†	0.0132†	0.0407†	0.0131†	0.0229†
	T-CE	0.1537†	0.3220†	0.1267†	0.1839†	0.1025†	0.2821†	0.2923†	0.2845†	0.0119†	0.0396†	0.0119†	0.0211†
	DeCA	0.1597	0.3205†	0.1226†	0.1799†	0.1024†	0.2723†	0.2904†	0.2801†	0.0129†	0.0394†	0.0129†	0.0216†
	SGDL	**0.1598**	**0.3291**	**0.1272**	**0.1853**	**0.1135**	**0.2844**	**0.3279**	**0.3032**	**0.0155**	**0.0469**	**0.0158**	**0.0260**
CDAE	Normal	0.1445†	0.3159†	0.0987†	0.1886†	0.0904†	0.2185†	0.2617†	0.2356†	0.0145†	0.0436†	0.0149†	0.0277†
	WBPR	0.1443†	0.3158†	0.0987†	0.1890†	0.0908†	0.2184†	0.2619†	0.2346†	0.0148†	0.0437†	0.0151†	0.0278†
	IR	0.1444	0.3152†	0.0981†	0.1893†	0.0909†	0.2186†	0.2612†	0.2358†	0.0153†	0.0438	0.0152†	0.0278†
	T-CE	0.1415†	0.3106†	0.0991	0.1840†	0.0912†	0.2158†	0.2642	0.2386†	0.0147†	**0.0439**	0.0151†	0.0279†
	DeCA	0.1447†	0.3159†	0.0991	0.1888†	0.0917†	0.2189†	0.2641	0.2378†	0.0158†	0.0438	0.0154†	0.0292†
	SGDL	**0.1450**	**0.3181**	**0.0993**	**0.1956**	**0.0921**	**0.2220**	**0.2643**	**0.2404**	**0.0162**	**0.0439**	**0.0172**	**0.0296**
NGCF	Normal	0.0769†	0.1322†	0.0571†	0.0769†	0.1285†	0.3103†	0.3694†	0.3392†	0.0267†	0.0736†	0.0262†	0.0417†
	WBPR	0.0770†	0.1324†	0.0572†	0.0769†	0.1287†	0.3105†	0.3692†	0.3395†	0.0265†	0.0739†	0.0265†	0.0417†
	IR	0.0772†	0.1337†	0.0570†	0.0768†	0.1280†	0.3104†	0.3701†	0.3395†	0.0269†	0.0737†	0.0261†	0.0412†
	DeCA	0.0760†	0.1326†	0.0571†	0.0766†	0.1304†	0.3113†	0.3729†	0.3401	0.0277	0.0739†	0.0262	0.0418
	SGCN	0.0773†	0.1336†	0.0543†	0.0770	0.1288†	0.3112†	**0.3768**	0.3401	0.0267†	0.0734†	0.0265	**0.0443**
	SGL	0.0775†	0.1345	0.0576	0.0768†	0.1303†	0.3141†	0.3763†	0.3360†	**0.0279†**	**0.0750**	0.0264†	0.0409†
	SGDL	**0.0788**	**0.1347**	**0.0579**	**0.0771**	**0.1309**	**0.3186**	0.3745	**0.3404**	0.0273	0.0746	**0.0267**	0.0420
LightGCN	Normal	0.0951†	0.1817†	0.0713†	0.0994†	0.1258†	0.3173†	0.3678†	0.3358†	0.0334†	0.0912†	0.0332†	0.0515†
	WBPR	0.0958†	0.1845†	0.0733†	0.1006†	0.1262†	0.3189†	0.3701†	0.3510	0.0333†	0.0911†	0.0331†	0.0512†
	IR	0.0953†	0.1822†	0.0726†	0.1003†	0.1285†	0.3194†	0.3681†	0.3361†	0.0305†	0.0909†	0.0326†	0.0510†
	DeCA	0.0974†	0.1855†	0.0758†	0.1162†	0.1293†	0.3076†	0.3575†	0.3270†	0.0337	0.0911†	0.0332†	0.0524
	SGCN	0.0941†	0.1899†	0.0765†	0.1160†	0.1282†	0.3210†	0.3602†	0.3318†	0.0335†	0.0916	**0.0346**	**0.0528**
	SGL	0.0980†	0.1770†	0.0741†	0.0999†	0.1299†	0.3156†	0.3638†	0.3343†	**0.0341**	0.0915	0.0344	0.0526
	SGDL	**0.1134**	**0.2105**	**0.0844**	**0.1178**	**0.1378**	**0.3335**	**0.3844**	**0.3513**	0.0339	**0.0918**	0.0341	0.0525

4.1.3 **Baselines.** We select four state-of-the-art recommendation methods as the base model f of SGDL:

- **NeuMF** [20] is a state-of-the-art model, which generalizes Factorization Machines (FM) with a Multi-Layer Perceptron (MLP).
- **CDAE** [45] is denoising auto-encoder model, which corrupts the interactions with random noises, and then employs a MLP model to reconstruct the original input.
- **NGCF** [40] is a graph model, which applies graph convolution network (GCN) to encode high-order collaborative signals in user-item bipartite graph.
- **LightGCN** [19] is a state-of-the-art graph model, which simplifies the design of GCN by discarding the nonlinear feature transformations for recommendation.

We train the base models with different ranking loss functions to demonstrate the universality of SGDL. Specifically, we train NeuMF and CDAE with binary cross-entropy (BCE) loss, and train NGCF and LightGCN with BPR loss [34]. Each model is trained with the following denoising approaches:

- **Normal** is trained with the original architecture design, without any denoising consideration.
- **WBPR** [16] is a sample selection method, which considers the popular but uninteracted items are likely to be real negative ones.
- **IR** [42] is the state-of-the-art sample selection method, which interactively relabels uncertain samples to mitigate the noise in both observed and unobserved interactions.

- **T-CE** [39] is the state-of-the-art sample re-weighting method, which uses the Truncated BCE to assign zero weights to large-loss examples with a dynamic threshold. Note that, this denoising approach can only be used for BCE loss, and thus, we implement it with NeuMF and CDAE for comparison.
- **DeCA** [41] is a newly proposed sample re-weighting method, which considers the disagreement predictions of noisy samples across different models, and minimizes KL-divergence between the two models' predictions to enhance the robustness of models.

In addition, we also compare SGDL with the state-of-the-art robust graph-based methods to further confirm the effectiveness of our model. Note that the methods can only be applied to graph-based recommenders (*i.e.,* NGCF and LightGCN in our experiments), since they regard the noisy interactions as noisy edges, and devise enhanced graph learning methods for robust recommendation.

- **SGCN** [5] is the state-of-the-art graph structure enhanced method, which attaches the GCN layers with a trainable stochastic binary mask to prune noisy edges in user-item bipartite graph.
- **SGL** [43] is the state-of-the-art self-supervised graph method, which designs different graph views to mine hard negatives and denoise noise in implicit feedback. We choose the Edge Dropout (ED) view as auxiliary supervision signal as it performs the best for most datasets.

4.1.4 **Parameter Settings.** We implement SGDL in Pytorch, and have released our implementations[5] (codes, parameter settings, and training logs) to facilitate reproducibility. We use recommended parameter settings for all models, and optimize them with Adam [27] optimizer. The Xavier initializer [18] is used to initialize the model parameters. We set the batch size as 128 for MovieLens, 1024 for Adressa, and 2048 for Yelp due to the different sizes of each dataset. The learning rate of CDAE, NeuMF, and LightGCN is tuned as 0.001; and for NGCF, the learning rate is set as 0.0001. For NeuMF, the embedding size is 32, and the number of layers is 3. For CDAE, the hidden size is 100, and the dropout ratio is set to 0.5. For LightGCN, we set the embedding size to 64, the number of layers to 3, and train it without dropout. For NGCF, the embedding size and layers are the same as LightGCN; and the node dropout rate is set to 0.1. For SGDL, we set $\eta_1 = \eta_2$ for denoising learning, and tune the learning rate amongst $\{10^{-4}, 10^{-3}, 10^{-2}\}$ for two phases, respectively. The hidden size of MLP and LSTM unit is 64; the length of memorization history h is tuned among $\{2, 5, 10, 20\}$. Note that, the hyperparameters of base models keep exactly the same across all training methods for a fair comparison.

4.2 Performance Comparison (RQ1)

We begin with the performance comparison *w.r.t.* recall@K and NDCG@K, where we test two values (5 and 20) of K. The experimental results are reported in Table 2, and we find that:

- The proposed SGDL can effectively improve the performance of all base models, and outperform all denoising methods over three datasets. Besides, even if the base model is designed to be robust against noisy implicit feedback (*i.e.,* CDAE), our method can still boost its performance by a large margin. We attribute these improvement to the memorization-based denoising schemes of SGDL: (1) By tracing the memorization states of data in the noise-resistant period, SGDL is able to collect memorized interactions during training process to provide valuable denoising signals without any supervision. In contrast, none of the baselines considers explicitly characterizing data from the memorization perspective. (2) Benefiting from our learning to weight strategy and adaptive denoising scheduler, SGDL can adaptively select clean and informative samples from memorized interactions, and use them to guide the learning process of the model. However, other re-weighting baselines (*e.g.,* T-CE and DeCA) are insufficient to provide proper weight for each interaction since they do not have memorized data as guidance.
- Jointly analyzing the performance of SGDL across three datasets, we find that the improvement on MovieLens dataset are less significant than that on other datasets. One possible reason is that the sparsity of MovieLens dataset is denser than the sparsity of Yelp and Adressa. Accordingly, there are sufficient interactions to identify user behavior patterns, which offsets the impact of noisy implicit feedback.
- Jointly analyzing the performance of SGDL across the recommenders, we observe that the relative improvement (*i.e.,* the performance of SGDL over the strongest baselines) on NeuMF and CDAE are more substantial than that on graph-based methods

(*i.e.,* NGCF and LightGCN). This is because our method is model-agnostic, which does not take the graph structure into consideration. In comparison, both SGCN and SGL are designed for graph-based recommendations, and thus, they can leverage the graph structure information to yield better performance. Nonetheless, our method still achieves the best results on most cases, which demonstrate the superior robustness of SGDL.

- All the denoising approaches show better results compared with normal training in most cases, which indicate the necessities of denoising implicit feedback for recommendations. The results are consistent with prior studies [39, 41].
- The performance of sample selection methods (*i.e.,* WBPR and IR) is rather unstable across different recommenders, compared with sample re-weighting methods. This is reasonable since sample selection methods highly depend on the sampling distribution, which makes their performance unstable. For instance, although IR achieves some improvement on Yelp dataset, it performs worse than the base model (*i.e.,* CDAE) on Adressa dataset.
- Robust Graph-based methods (*i.e.,* SGCN and SGL) achieve competitive or even the best performance against other methods. We attribute such improvement to their carefully designed graph structures, which is able to prune noisy and insignificant edges for clean information propagation. However, the methods can only be applied to graph-based models, while SGDL is able to be easily integrated with any learning-based recommender systems, and achieves comparable or even better performance.

4.3 Study of SDGL (RQ2)

As the memorization and self-guided denoising learning are the core of SGDL, we also conduct ablation studies to investigate their effectiveness. Specifically, how the presence of denoising learning and adaptive denoising scheduler, the estimation of memorization point, and the design of scheduler affect our model.

4.3.1 **Impact of Denoising Learning & Scheduler.** We first evaluate the effectiveness of the denoising learning scheme and adaptive denoising scheduler. To this end, two variants of SGDL are constructed by (1) discarding the denoising learning strategy in noise-sensitive period, called SGDL$_{\text{w/o DLS}}$; and (2) removing the scheduler and directly using all memorized data for denoising learning, named SGDL$_{\text{w/o ADS}}$. We summarize the results of NeuMF and LightGCN in Table 3, while skip the results of other models (*i.e.,* CDAE and NGCF) because they demonstrate similar trends, and we have space limitation.

Obviously, compared with SGDL in Table 2, removing the denoising learning scheme (*i.e.,* SGDL$_{\text{w/o DLS}}$) dramatically reduces the predictive accuracy, indicating the necessity of self-guided learning. To be more specific, SGDL$_{\text{w/o DLS}}$ only trains with memorized data in the noise-sensitive period, and thus, it is insufficient to learn user true preference with hard yet clean interactions. Besides, directly leveraging all memorized data as denoising signals would inevitably introduce some noise, and hence, the SGDL$_{\text{w/o ADS}}$ also underperforms the complete model.

4.3.2 **Estimation of Memorization Point.** We then verify the estimation of memorization point, since it plays an important role in our model to transit from noise-resistant period to noise-sensitive

---
[5]https://github.com/ZJU-DAILY/SGDL

Table 3: Impact of denoising Learning and adaptive denoising scheduler.

Database		Adressa				MovieLens				Yelp			
Base Model	Method	R@5	R@20	N@5	N@20	R@5	R@20	N@5	N@20	R@5	R@20	N@5	N@20
NeuMF	w/o DLS	0.1528	0.3107	0.1211	0.1794	0.1055	0.2690	0.2911	0.2774	0.0136	0.0397	0.0131	0.0218
	w/o ADS	0.1576	0.3285	0.1255	0.1801	0.1097	0.2801	0.3210	0.3008	0.0146	0.0438	0.0146	0.0259
LightGCN	w/o DLS	0.0964	0.1810	0.0702	0.0985	0.1244	0.3159	0.3688	0.3349	0.0330	0.0909	0.0331	0.0513
	w/o ADS	0.1013	0.1995	0.0811	0.1007	0.1316	0.3328	0.3824	0.3502	0.0338	0.0914	0.0340	0.0521

Table 4: Impact of estimation of memorization point. R@20 is used to evaluate the performance, and Est. denotes estimated memorization point.

Memorization Point		Early		Est.	Late	
Base Model	Database	+10%	+5%	+0%	-5%	-10%
NeuMF	Adressa	0.3221	0.3275	**0.3291**	0.3256	0.3203
	MovieLens	0.2810	**0.2851**	0.2844	0.2757	0.2704
	Yelp	0.458	0.4420	**0.0469**	0.4430	0.4370
LightGCN	Adressa	0.2006	**0.2114**	0.2105	0.2017	0.1990
	MovieLens	0.3321	0.3325	**0.3335**	0.3262	0.3198
	Yelp	0.0895	0.0912	**0.0918**	0.0904	0.0887

Table 5: Impact of the design of adaptive denoising scheduler.

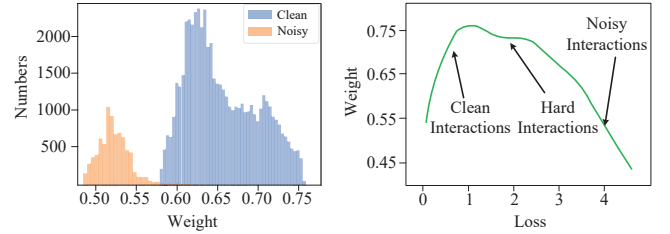| Database		Yelp		MovieLens		Adressa	
Base Model	ADS	R@20	N@20	R@20	N@20	R@20	N@20
NeuMF	top-F	0.3106	0.1766	0.2590	0.2705	0.0386	0.0212
	MLP	0.3257	0.1842	0.2749	0.2841	0.0392	0.0225
	LSTM	0.3291	0.1853	0.2844	0.3032	0.0469	0.0260
LightGCN	top-F	0.1810	0.0985	0.3159	0.3349	0.0909	0.0513
	MLP	0.2066	0.0103	0.3284	0.3427	0.0912	0.0512
	LSTM	0.2105	0.1178	0.3335	0.3513	0.0918	0.0525

Figure 5: (a) Sample weight distribution on MovieLens dataset. (b) Learned weighting function on MovieLens dataset.

period. Specifically, we explore the performance change of SGDL by increasing or decreasing the estimated noisy ratio $\hat{\sigma}$ to force early or late transition. The results of SGDL on NeuMF and LightGCN are presented in Table 4. We observe that:

- Generally speaking, the best performance is achieved at the estimated memorization point. When the memorization point more deviates from the estimated one, the performance tends to be worse. This means the existence of the best memorization point and the effectiveness of our estimation.
- It is worth mentioning that, when the memorization point is slightly earlier than the estimated value, it also shows good performance. We attribute this to the clean memorized data: when we transit the model to noise-sensitive period earlier, it is more likely that most of the memorized interactions are clean, which benefits the following self-guided learning. On the contrary, when we delay the memorization point, the memorized data tends to contain more noisy samples, incurring suboptimal performance.

*4.3.3 **The Design of Adaptive Denoising Scheduler.*** We also investigate the design of adaptive denoising scheduler. Specifically, we propose three different approaches to evaluate the denoising contributions of each memorized data: i) Rank memorized data according to their sums of two factors (*i.e.,* normalized gradient similarity and loss value), and simply pick top-F memorized data as informative denoising signals. To keep the picked data clean, we set F as the half size of memorized data. ii) Choose a one-layer MLP as the scheduler, and train the scheduler with the strategy presented in Section 3.2.2. iii) Select the LSTM as the scheduler, and leverage the historical factors to predict the sampling probabilities. We report the performance of three sampling approaches with Recall@20 and NDCG@20 in Table 5, while Recall@5 and NDCG@5 are omitted due to limited space. We have the following observations:

- By simply choosing top-F memorized data with the highest sums as denoising signals, the performance drops and becomes even worse than the normal training. We attribute such degradation to the non-linear correlation of two factors: the large loss value may not necessarily mean the memorized sample is beneficial to denoising, as it can also be noisy if the gradient similarity is small. Directly summing up the two factors is unable to properly

capture the characteristic of memorized data, and thus fails to provide reliable denoising signals for self-guided learning.

- The scheduler with LSTM consistently achieves the best performance cross the datasets. This is because LSTM can leverage the historical factors information to predict more accurate and stable sampling probability for memorized data. The results are consistent with previous robust learning studies [4, 48].

4.4 Learned Weights of SGDL (RQ3)

In this section, we visualize the learned weights of SGDL to offer an intuitive impression of our denoising performance. Specifically, we train NeuMF with SGDL on the MovieLens dataset, and plot the learned weights distribution *w.r.t.* clean and noisy interactions as well as their loss values in Figure 5. We find that:

- The left of Figure 5 indicates that almost all large weights belong to clean interactions, and the weights of noisy interactions are much smaller than that of clean ones, meaning that the SGDL can differentiate clean implicit feedback from the noisy one.
- The learned weighting function of SGDL in the right of Figure 5 demonstrates that when the loss has relatively small values, the weighting function inclines to increase the weight together with loss, indicating that it tends to emphasize more on clean data for learning user preference; while as the loss gradually becomes larger, the weighting function first remains unchanged and then begins to dramatically decrease its weight, implying that it tends to highlight the hard yet clean interactions with large weights first and then suppress noise interactions. Therefore, SGDL is able to automatically locate hard interactions for better learning user preferences.

5 RELATED WORK

Existing recommender systems are typically trained with implicit feedback. Recently, some studies [25, 28, 39, 42] have noticed that implicit feedback could be easily corrupted by different factors (*e.g.,* popularity bias [7] and unawareness of users' behaviors [24]), and the inevitable noise would dramatically degrade the recommendation performance [37, 39, 41, 42]. As a result, some efforts have been dedicated to solving the noisy implicit feedback problem, which can be categorized into sample selection methods [12, 16, 32, 42, 50] and sample re-weighting methods [23, 39, 41].

Sample Selection. A simple idea to denoise implicit feedback is to select clean and informative samples only, and train the recommendation model with them. For example, WBPR [16] considers that the missing interactions of popular items are highly likely to be real negative examples, and hence assigns higher sampling probabilities to them. IR [42] interactively generates pseudo-labels for user preferences based on the difference between labels and predictions, to discover the noisy-positive and noisy-negative examples. Nonetheless, their performance has high variance since they heavily depend on the sampling distribution [51].

Sample Re-weighting. On the other hand, loss-based methods focus on the learning process of models (*e.g.,* loss values and predictions) to distinguish noisy interactions from clean data. For instance, T-CE [39] dynamically assigns lower weights to high-loss samples since it has been shown that noisy examples would have larger loss values. DeCA [41] develops an ensemble method to minimize the KG-divergence between the two models' predictions, under the assumption that different models make relatively similar predictions on clean examples. Although these methods achieve promising results without additional data, they heavily rely on the predefined loss function and hyperparameters, incurring poor generalization for different recommendation models.

Other Directions. There are some recent studies that consider using additional information [3, 46, 49] or designing model-specific structures [5, 43, 47] to improve the robustness of recommender systems. For instance, DFN [46] proposes a feedback interaction component to extract clean and useful information from noisy feedback with additional explicit feedback (*e.g.,* like and dislike). SGCN [5] treat user-item interactions as a bipartite graph, and devise a learnable regularization module to preserve the sparsity and low rank of graph structure. SGL [43] advances graph-based recommender systems with self-supervised learning by employing graph structure augmentations. However, these methods suffer from poor generalization, since they either need additional information (*e.g.,* explicit feedback) as guidance to denoise implicit feedback [3, 46], or are only applicable to specific data structures (*e.g.,* user-item bipartite graph) [5, 43, 47].

Difference from Existing Work. Our work can be seen as a variant of self-training [52], which leverages self-labeled memorized data to enhance the robustness of training process. Compared with existing work, SGDL is in an orthogonal direction, opening up a new research line of denoising implicit feedback for recommendations. The most relevant work with ours is MORPH [36]. However, it tackles the classification problem in computer vision, and only uses the memorized data for training, which is not feasible in ranking-based recommendations since the hard yet clean data may not be memorized without clear guidance. Compared with previous methods, SGDL is a truly model-agnostic framework, which can be easily applied to *any* learning-based recommendation models with *any* ranking loss functions, and does not need to define *any* weighting functions.

6 CONCLUSION AND FUTURE WORK

In this paper, we present a new denoising paradigm, called SGDL, which leverages the memorization effect of recommendation models and designs two training phases to exploit the self-labeled memorization data as guidance for denoising learning. Extensive experiments conducted on three real-world datasets with four representative recommendation models demonstrate the superiority and universality of SGDL. In the future, we plan to jointly explore the noise and bias existing in implicit feedback to develop a universal denoising and debiasing solution.

REFERENCES

[1] Eric Arazo, Diego Ortego, Paul Albert, Noel O'Connor, and Kevin McGuinness. 2019. Unsupervised label noise modeling and loss correction. In *ICML*. 312–321.

[2] Devansh Arpit, Stanisław Jastrzundefinedbski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. 2017. A Closer Look at Memorization in Deep Networks. In *ICML*. 233–242.

[3] Zhi Bian, Shaojun Zhou, Hao Fu, Qihong Yang, Zhenqi Sun, Junjie Tang, Guiquan Liu, Kaikui Liu, and Xiaolong Li. 2021. Denoising User-aware Memory Network for Recommendation. In *RecSys*. 400–410.

[4] Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. 2017. Active bias: Training more accurate neural networks by emphasizing high variance samples. In *NeurIPS*. 1002–1012.

[5] Huiyuan Chen, Lan Wang, Yusan Lin, Chin-Chia Michael Yeh, Fei Wang, and Hao Yang. 2021. Structured Graph Convolutional Networks with Stochastic Masks for Recommender Systems. In *SIGIR*. 614–623.

[6] Jiawei Chen, Hande Dong, Yang Qiu, Xiangnan He, Xin Xin, Liang Chen, Guli Lin, and Keping Yang. 2021. AutoDebias: Learning to Debias for Recommendation. In *SIGIR*. 21–30.

[7] 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. In *arXiv preprint arXiv:2010.03240*.

[8] Pengfei Chen, Ben Ben Liao, Guangyong Chen, and Shengyu Zhang. 2019. Understanding and utilizing deep neural networks trained with noisy labels. In *ICML*. 1062–1070.

[9] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys*. 39–46.

[10] Balázs Csanád Csáji. 2001. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary* 24, 48 (2001).

[11] Jingtao Ding, Yuhan Quan, Xiangnan He, Yong Li, and Depeng Jin. 2019. Reinforced Negative Sampling for Recommendation with Exposure Data.. In *IJCAI*. 2230–2236.

[12] Jingtao Ding, Guanghui Yu, Xiangnan He, Fuli Feng, Yong Li, and Depeng Jin. 2019. Sampler design for bayesian personalized ranking by leveraging view data. *TKDE* 33 (2019), 667–681.

[13] Yuntao Du, Xinjun Zhu, Lu Chen, Ziquan Fang, and Yunjun Gao. 2022. MetaKG: Meta-learning on Knowledge Graph for Cold-start Recommendation. *TKDE* (2022).

[14] Yuntao Du, Xinjun Zhu, Lu Chen, Baihua Zheng, and Yunjun Gao. 2022. HAKG: Hierarchy-Aware Knowledge Gated Network for Recommendation. In *SIGIR*.

[15] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*. 1126–1135.

[16] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2012. Personalized ranking for non-uniformly sampled items. In *KDD Cup*. 231–247.

[17] Zhabiz Gharibshah, Xingquan Zhu, Arthur Hainline, and Michael Conway. 2020. Deep learning for user interest and response prediction in online display advertising. *Data Science and Engineering* 5, 1 (2020), 12–26.

[18] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. 249–256.

[19] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*. 639–648.

[20] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.

[21] 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.

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

[23] Kaixi Hu, Lin Li, Qing Xie, Jianquan Liu, and Xiaohui Tao. 2021. What is Next When Sequential Prediction Meets Implicitly Hard Interaction?. In *CIKM*. 710–719.

[24] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM*. 263–272.

[25] 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.

[26] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *ICLR*.

[27] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *ICLR*.

[28] Dongha Lee, SeongKu Kang, Hyunjun Ju, Chanyoung Park, and Hwanjo Yu. 2021. Bootstrapping User and Item Representations for One-Class Collaborative Filtering. In *SIGIR*. 317–326.

[29] Junnan Li, Richard Socher, and Steven CH Hoi. 2020. DivideMix: Learning with noisy labels as semi-supervised learning. In *ICLR*.

[30] Tzu-Heng Lin, Chen Gao, and Yong Li. 2019. Cross: Cross-platform recommendation for social e-commerce. In *SIGIR*. 515–524.

[31] Tongliang Liu and Dacheng Tao. 2015. Classification with noisy labels by importance reweighting. *PAMI* 38, 3 (2015), 447–461.

[32] Dae Hoon Park and Yi Chang. 2019. Adversarial sampling and training for semi-supervised information retrieval. In *WWW*. 1443–1453.

[33] Zhaochun Ren, Shangsong Liang, Piji Li, Shuaiqiang Wang, and Maarten de Rijke. 2017. Social Collaborative Viewpoint Regression with Explainable Recommendations. In *WSDM*. 485–494.

[34] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. 452–461.

[35] 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*. 1919–1930.

[36] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. 2021. Robust Learning by Self-Transition for Handling Noisy Labels. In *KDD*. 1490–1500.

[37] Yanchao Tan, Carl Yang, Xiangyu Wei, Ziyue Wu, and Xiaolin Zheng. 2022. Partial Relaxed Optimal Transport for Denoised Recommendation. In *arXiv preprint arXiv:2204.08619*.

[38] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep knowledge-aware network for news recommendation. In *WWW*. 1835—-1844.

[39] Wenjie Wang, Fuli Feng, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2021. Denoising implicit feedback for recommendation. In *WSDM*. 373–381.

[40] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*. 165–174.

[41] Yu Wang, Xin Xin, Zaiqiao Meng, Xiangnan He, Joemon Jose, and Fuli Feng. 2022. Learning Robust Recommenders through Cross-Model Agreement. In *WWW*.

[42] Zitai Wang, Qianqian Xu, Zhiyong Yang, Xiaochun Cao, and Qingming Huang. 2021. Implicit Feedbacks are Not Always Favorable: Iterative Relabeled One-Class Collaborative Filtering against Noisy Interactions. In *MM*. 3070–3078.

[43] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *SIGIR*. 726–735.

[44] Lijun Wu, Fei Tian, Yingce Xia, Yang Fan, Tao Qin, Lai Jian-Huang, and Tie-Yan Liu. 2018. Learning to teach with dynamic loss functions. In *NeurIPS*. 6466–6477.

[45] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *WSDM*. 153–162.

[46] Ruobing Xie, Cheng Ling, Yalong Wang, Rui Wang, Feng Xia, and Leyu Lin. 2020. Deep Feedback Network for Recommendation.. In *IJCAI*. 2519–2525.

[47] Yonghui Yang, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2021. Enhanced Graph Learning for Collaborative Filtering via Mutual Information Maximization. In *SIGIR*. 71–80.

[48] Huaxiu Yao, Yu Wang, Ying Wei, Peilin Zhao, Mehrdad Mahdavi, Defu Lian, and Chelsea Finn. 2021. Meta-learning with an Adaptive Task Scheduler. In *NeurIPS*.

[49] Xing Yi, Liangjie Hong, Erheng Zhong, Nanthan Nan Liu, and Suju Rajan. 2014. Beyond clicks: dwell time for personalization. In *RecSys*. 113–120.

[50] Wenhui Yu and Zheng Qin. 2020. Sampler design for implicit feedback data by noisy-label robust learning. In *SIGIR*. 861–870.

[51] Fajie Yuan, Xin Xin, Xiangnan He, Guibing Guo, Weinan Zhang, Tat-Seng Chua, and Joemon M Jose. 2018. fBGD: Learning Embeddings From Positive Unlabeled Data with BGD. In *UAI*.

[52] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin Dogus Cubuk, and Quoc Le. 2020. Rethinking Pre-training and Self-training. In *NeurIPS*. 3833–3845.