Distributionally Robust Sequential Recommendation

Rui Zhou
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China
ruizhou@ruc.edu.cn

Xian Wu
Tencent Jarvis Lab
Shenzhen, China
kevinxwu@tencent.com

Zhaopeng Qiu
Tencent Jarvis Lab
Shenzhen, China
zhaopengqiu@tencent.com

Yefeng Zheng
Tencent Jarvis Lab
Shenzhen, China
yefengzheng@tencent.com

Xu Chen*
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China
successcx@gmail.com

ABSTRACT

Modeling user sequential behaviors have been demonstrated to be effective in promoting the recommendation performance. While previous work has achieved remarkable successes, they mostly assume that the training and testing distributions are consistent, which may contradict with the diverse and complex user preferences, and limit the recommendation performance in real-world scenarios. To alleviate this problem, in this paper, we propose a robust sequential recommender framework to overcome the potential distribution shift between the training and testing sets. In specific, we firstly simulate different training distributions via sample reweighting. Then, we minimize the largest loss induced by these distributions to optimize the "worst-case" loss for improving the model robustness. Considering that there can be too many sample weights, which may introduce too much flexibility and be hard to optimize, we cluster the training samples based on both hard and soft strategies, and assign each cluster with a unified weight. At last, we analyze our framework by presenting the generalization error bound of the above minimax objective, which help us to better understand the proposed framework from the theoretical perspective. We conduct extensive experiments based on three real-world datasets to demonstrate the effectiveness of our proposed framework. To reproduce our experiments and promote this research direction, we have released our project at https://anonymousrsr.github.io/RSR/.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Sequential Recommendation, Robust Learning, Distribution Shifts

∗ Xu Chen is the corresponding author.

SIGIR '23, July 23–27, 2023, Taipei, Taiwan

ACM ISBN 978-1-4503-9408-6/23/07…$15.00
https://doi.org/10.1145/3539618.3591668

ACM Reference Format:
Rui Zhou, Xian Wu, Zhaopeng Qiu, Yefeng Zheng, and Xu Chen*. 2023. Distributionally Robust Sequential Recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3539618.3591668

1 INTRODUCTION

Sequential recommendation has recently attracted increasing attention from both industry and academic communities. It basically aims to learn the user dynamic preference by modeling the item chronological correlations. For example (see Figure 1(a)), the user may click the camera before interacting with the tripod. The purchase of a mobile phone may trigger the interaction with the phone case. To capture such correlations, in recent years, many promising models have emerged. For example, FPMC [29] models the transitions between successive behaviors by assuming that the user current decision is only influenced by the most recent action. With the help of the strong expressive power of deep learning, GRU4Rec [16] introduces more historical behaviors to estimate user preference. NARM and SASRec [17, 19] further discriminate different behavior importance to refine the histoory information.

While the above models have achieved remarkable successes, they mostly learn and evaluate the item correlations in the same domains (*i.e.*, with similar sample distributions). However, in real-world scenarios, the user preferences can be diverse and complex, which may result in different training and testing distributions. As exemplified in Figure 1(a), in early days, most of the user interactions are about digital products. However, as the time passed, the user becomes more interested in the sports items, which are more frequently observed in her recent interactions. If the recommender models are optimized to mainly learn item transition/correlation patterns among the digital products, then they may not perform well when the testing sets contain more sports products. Similar phenomenons can also be observed in Figures 1(b) and 1(c), where the training and testing sample distributions are different on the item brand and color. The distribution shift problem may widely exist in real-world scenarios, which should be carefully considered in an effective sequential recommender model.

To achieve the above goal, we propose a **r**obust **s**equential **r**ecommender framework (called **RSR** for short) in this paper. It is expected to perform well on the testing sets which are distributionally different from the training ones. Specifically, we firstly assign

Figure 1: (a) Example of the user preference shift from digital products to sports items. (b-c) Examples of the user preference changes on the item brands and colors, respectively.

each training sequence with a unique weight. From the probabilistic perspective, different weight combinations can simulate different training distributions. As show in Figure 1(a), although there are more digital products in the training set, if we assign much higher weights on the sports items and lower the weights of the digital products, then the model is similar to the one trained on a dataset containing more sports items. To make the model perform well, a straightforward idea is adjusting the sample weights to make the training distribution close to the testing one. However, in general, we do not know the testing distribution. To solve this problem, we firstly find the "worst-case" training distribution by learning the sample weights which can lead to the largest loss. Then, we fix the sample weights, and optimize the model parameters by minimizing the loss function. By this strategy, we bypass the real testing distribution, and minimize its upper bound to improve the model robustness. While this idea seems to be promising, assigning each sequence with a unique weight may result in too many parameters, which are hard to optimize. Intuitively, the diversity of the user preferences usually exist on high-level aspects. For example, in Figure 1(b), the user preferences vary on the item brand, while in Figure 1(c), the interacted items have diverse colors. Each brand or color may corresponds to a set of sequences instead of a specific one. Inspired by this intuition, we propose to reduce the number of weights by clustering the sequences. In specific, we firstly learn the sequence representations adaptively from the training data, and then the sequences are clustered based on their representations by K-means [22]. At last, we assign the sequences in the same cluster with the same weight. Such method uses hard strategy, where the samples are clustered not necessarily for improving the recommendation performance. To back-propagate the recommendation signals for guiding the clustering process, we soften the above strategy to jointly optimize the clustering and recommendation parameters. In addition, we theoretically analyze our framework by deriving its generalization error bound. In the experiment, we apply our framework to different sequential models, and evaluate it based on three real-world datasets to demonstrate its effectiveness.

In a summary, the main contributions of this paper are as follows: (1) we propose a distributionally robust framework for sequential recommendation, which, to the best of our knowledge, is the first attempt in the recommendation domain. (2) To achieve the above goal, we firstly design a basic model by tuning the sequence weights. Then, we improve the basic model by clustering the training sequences based on both soft and hard strategies. (3) We provide theoretical analysis on the designed framework based on the probably approximately correct (PAC) learning theory. (4) We conduct extensive experiments to evaluate our framework, and to promote this research direction, we have released our project at https://anonymousrsr.github.io/RSR/.

2 PRELIMINARIES

2.1 Sequential Recommendation

Suppose we have a user set $\mathcal{U}$ and an item set $\mathcal{V}$. The interactions between the users and items are organized into a set $O = \{\mathcal{V}_u | u \in \mathcal{U}\}$, where $\mathcal{V}_u = \{v_u^1, v_u^2, ..., v_u^{n_u}\}$ is the set of items sequentially interacted by user u. Given $\{\mathcal{U}, \mathcal{V}, O\}$, sequential recommendation aims to learn a model f, such that, given a user u and the history of interacted items $\{v_u^1, v_u^2, ..., v_u^{n_u}\}$, f can accurately predict the next item. Usually, f is learned based on the following loss function:

$$L_s = \sum_{u \in \mathcal{U}} \underbrace{\sum_{t=2}^{n_u} \sum_{v \in \{v_u^t\} \cup S_u^t} y_v \log f(v|u, H_u^{t-1}) + (1-y_v) \log[1 - f(v|u, H_u^{t-1})]}_{L_u(\Theta_f)}$$

where $L_u(\Theta_f)$ is the binary cross entropy loss for interacted items of user u. The recommender model f is parameterized by Θ_f. S_u^t is the set of negative samples for v_u^t, and $y_v = 1$ if $v = v_u^t$, otherwise, $y_v = 0$. $H_u^{i-1} = \{v_u^1, v_u^2, ..., v_u^{i-1}\}$ is the history information. The output layer of f is usually implemented with the sigmoid[1] activation function.

In the past few years, the field of sequential recommendation has been significantly advanced from the model perspective, where people have designed a lot of promising models. However, an important problem has received little attention, that is, due to the diversity of the user preferences, the training and testing sample distributions can be different, which may lower the final recommendation performance. In this paper, we try to advance sequential recommendation from the data perspective, and design a robust framework to alleviate the distribution shift problem.

2.2 Robust Optimization

Robust optimization [1, 11] has been widely studied in the machine learning community. In general, there are two types of methods to improve the recommendation's robustness, that is, adversarial robust optimization [9, 23, 30] and distributionally robust optimization (DRO) [7, 8, 24, 34]. Our framework follows the research line of DRO. Generally speaking, DRO firstly assigns each training sample with a tunable weight, and then it finds the worst-case loss by adjusting the sample weights. At last, the model is optimized based on the weights determined in the above step. Formally, suppose we have a training dataset $D = \{x_i, y_i\}_{i=1}^N$, and the model to be learned

[1]It should be noted that there are also many models with softmax output layers. For these models, the loss is usually cross entropy without negative sampling.

is h, which is parameterized by Θ. Then, the objective of DRO is:

$$\min_{\Theta} \max_{\boldsymbol{w} \in \Delta^{N-1}} \sum_{i=1}^{N} w_i \delta(h(x_i), y_i),$$

where Δ^{N-1} is the $(N-1)$-dimensional simplex. $\boldsymbol{w} = \{w_i\}$ is the set of sample weights. Previously, DRO has been successfully applied to the fields of neural language processing [25, 35], computer vision [26, 32] and user behavior modeling [15]. However, there is little work on leveraging it to improve sequential recommendation for learning robust item correlations.

3 THE RSR FRAMEWORK

To build a robust sequential recommender framework, in this section, we firstly design a basic model based on DRO. Then, to reduce the number of sample weights, we firstly cluster the sequences and then optimized the model by unifying the weights of the sequences in the same cluster. At last, we soften the clustering process, and optimize the clustering and recommendation parameters in a joint manner. In the following, we detail our framework.

3.1 Basic Robust Sequential Model

The key of sequential recommendation lies in the accurate item correlation modeling. However, due to the dynamic nature of the user preference, the item correlations may have to be learned and evaluated with different sample distributions. For example, in Figure 1(a), the model mainly learns about the correlations of digital products, but in the later testing stage, we need to make more inferences on the sports items. In the digital product, purchasing a Mac product may lead to the interaction with another Mac item. However, in the sports domain, one may successively interact with the Apple watch and Nike clothing. Inconsistent item correlations can decrease recommendation performance.

To solve this problem, an intuitive idea is to adjust the training distributions to approach the testing ones. However, we cannot access the testing distribution. Thus, we find and optimize the "worst-case" sample distribution, which guarantees that the learned model is is not too bad for any testing distribution. To find the worst distribution, we firstly assign each user interaction sequence with a unique weight, and then learn the sequence weights by maximizing the loss function. At last, we fix the learned weights, and optimize the model parameters. Formally, we have the following objective:

$$\min_{\Theta_f} \max_{\boldsymbol{w} \in \Delta^{M}} \{L_{\text{basic}} := \sum_{u \in \mathcal{U}} w_u L_u(\Theta_f)\},$$

where $\boldsymbol{w} = \{w_u\}$, $M = |\mathcal{U}| - 1$, and Δ^M is an M-dimensional simplex, that is, $\Delta^M = \{\boldsymbol{w} | \sum_{u \in \mathcal{U}} w_u = 1, w_u \geq 0, \forall u \in \mathcal{U}\}$. In practice, $\boldsymbol{w} \in \Delta^M$ can be implemented by imposing softmax operator on a set of free parameters as follows:

$$w_l = \frac{\exp(\kappa_l / \tau)}{\sum_{l'=1}^{M+1} (\kappa_{l'} / \tau)},$$

where $\boldsymbol{\kappa} = \{\kappa_l\}$ are the actual parameters for optimization. τ is a temperature parameter determining the sharpness of the softmax operator. Smaller τ makes $\boldsymbol{w}$ more like a one-hot vector, while if τ is too large, $\boldsymbol{w}$ represents a uniform distribution. Basically, the above objective is a "min-max" game. By the "max" operation, we aim to find the worst $\boldsymbol{w}$ which can lead to the largest loss. With the "min" operation, the model parameters Θ_f are optimized with fixed $\boldsymbol{w}$.

3.2 Robust Optimization with Hard Clustering

In the above basic method, the number of sample weights can be too large, which may bring difficulties for model optimization. In real-world scenarios, the user preferences are usually diverse on high-level feature space, for example, on the item brands or colors. Different sequences may have various similarities in such feature space. The similarity information can actually be leveraged to reduce the number of sample weights. Straightforwardly, one can firstly cluster the user interaction sequences, and then merge the sample weights in the same cluster. However, without sufficient knowledge, it is hard to accurately determine the features for clustering. To alleviate this problem, we firstly learn an embedding for each sequence adaptively from the data, and then directly cluster the embeddings without indicating any specific feature. Such data-driven strategy is expected to discover the most representative latent features for characterizing the user behavior patterns.

Formally, for each user sequence, we firstly feed it into the sequential model f, and use the output as its representation, which is denoted by $\boldsymbol{e}_u$. Then, we cluster the sequences based on the following objective:

$$\min_{\boldsymbol{c}, \boldsymbol{\mu}} \{L_{\text{cl}} := \sum_{u \in \mathcal{U}} \sum_{i=1}^{K} \mathbf{1}(c_u = i) ||\boldsymbol{e}_u - \boldsymbol{\mu}_i||_2^2\},$$

where $\boldsymbol{c} = \{c_u\}$, c_u is the cluster index of the interaction sequence of user u. $\boldsymbol{\mu} = \{\boldsymbol{\mu}_i\}$ is the set of cluster centers. K is the number of clusters. After clustering the samples, we unify the sample weights in the same cluster, and design the following objective:

$$\min_{\Theta_f} \max_{\boldsymbol{w} \in \Delta^{K-1}} \{L_{\text{hard}} := \sum_{u \in \mathcal{U}} w_{c_u} L_u(\Theta_f)\},$$

where the number of weights is reduced from $|\mathcal{U}| - 1$ to K and w_{c_u} is the cluster-level weight for the sequence of user u. The clustering and model optimization processes are alternatively conducted to mutually enhance each other. On one hand, by learning more accurate recommender models, the samples can be better represented to improve the clustering process. On the other hand, if the samples are well clustered, the semantically similar samples are assigned with the same weights, which facilitates more effective distribution simulation and better final performance.

3.3 Robust Optimization with Soft Clustering

The clustering method mentioned above can reduce the number of sample weights, but it still has several limitations. User-item interactions are only used to derive sequence representations, and the clustering process is not directly associated with the recommender model, meaning that the samples can be partitioned in a sub-optimal manner, which may constrain recommendation performance. Additionally, the above method clusters sequences with a hard assignment, but many sequences may not strictly belong to only one cluster. For example, a sequence may contain both digital and sports products. Forceing each sequence to be assigned to a fixed cluster can be less accurate and may limit the flexibility of the downstream distribution simulation process.

To overcome the above limitations, we propose to cluster the sequences in a soft manner [2], and connect the clustering and recommender model optimization processes in a fully differentiable manner. Formally, we firstly represent each sequence by averaging the embeddings of all its items, which is denoted by $\boldsymbol{h}_u$. Then we feed $\boldsymbol{h}_u$ into a neural model as follows:

$$\boldsymbol{h}_u^* = V_2\sigma(V_1\boldsymbol{h}_u + b_1) + b_2,$$

where V_2, V_1, b_2 and b_1 are learnable parameters. σ is the sigmoid activation function. Based on $\boldsymbol{h}_u^*$, we introduce the following objective for soft clustering:

$$L_{\text{sc}} = \sum_{u\in\mathcal{U}} ||\boldsymbol{h}_u^* - \sum_{i=1}^{K} c_{u,i}\boldsymbol{\mu}_i||_2^2, \tag{1}$$

where $\boldsymbol{c}_u = \{c_{u,i}\}_{i=1}^K \in \Delta^{K-1}$ is the cluster assignment distribution. $c_{u,i}$ represents the probability that the sequence is grouped into the ith cluster. $\boldsymbol{\mu}_i$ is the center of cluster i. For better sequence representations, we introduce the following reconstruction loss:

$$L_{\text{rec}} = \sum_{u\in\mathcal{U}} ||\boldsymbol{h}_u - \widetilde{\boldsymbol{h}}_u||_2^2, \tag{2}$$

where $\widetilde{\boldsymbol{h}}_u = V_3\sigma(V_4\boldsymbol{h}_u^* + b_4) + b_3$ is the reconstructed sequence embedding. V_3, V_4, b_3 and b_4 are learnable parameters.

To connect the above soft clustering process with the recommender model optimization, we design the following target:

$$\min_{\Omega,\Theta_f,\boldsymbol{c},\boldsymbol{\mu}} \max_{\boldsymbol{w}\in\Delta^{K-1}} \{L_{\text{soft}} := \sum_{u\in\mathcal{U}}\sum_{i=1}^{K} c_{u,i}w_iL_u(\Theta_f) + \lambda(L_{\text{sc}} + L_{\text{rec}})\},$$

where $\boldsymbol{c} = \{\boldsymbol{c}_u\}$ is the set of cluster assignment distributions $\Omega = \{V_1, V_2, V_3, V_4, b_1, b_2, b_3, b_4\}$ is the set of soft clustering parameters. λ is a hyper-parameter to balance the importances between different objectives. By this objective, the clustering and recommendation processes are bridged via $\boldsymbol{c}$, which means that the sample cluster assignments are encouraged to better reduce the recommendation and clustering losses simultaneously.

Remark. By comparing L_{cl} with L_{sc}, we can see, $\boldsymbol{c}_u$ in L_{cl} can be seen as a special cluster assignment distribution, where the probabilities are all zero except for only one cluster. If we constrain $\boldsymbol{c}_u$ to be a one-hot vector, then L_{sc} is reduced to L_{cl}. Similar relations can also be found between L_{hard} and the first term of L_{soft}.

In summary, we present the complete learning algorithm of our framework in Algorithm 1. In specific, for the basic method, we firstly obtain the optimal sample weights in terms of maximizing the loss function. Then the model parameters are updated by minimizing the loss with fixed sample weights. For the hard clustering method, the cluster assignment distributions $\boldsymbol{c}$ and cluster center $\boldsymbol{\mu}$ are firstly obtained based on L_{cl}. Then we update the sample weights and model parameters in the same way as the basic method. For the soft clustering method, we firstly obtain the sample weights by maximizing L_{soft}, and then Θ_f, Ω, $\boldsymbol{c}$ and $\boldsymbol{\mu}$ are jointly optimized by minimizing L_{soft}.

3.4 Further Analysis

3.4.1 Computational Complexity. Suppose, in each iteration, the time cost for learning $\boldsymbol{w}$ is $O(C_w)$, updating $\boldsymbol{\Theta_f}$ costs $O(C_f)$. In the hard clustering method, deriving $\boldsymbol{c}$ and $\boldsymbol{\mu}$ needs $O(C_{clu})$. In the soft clustering method, updating $\boldsymbol{s}$ costs $O(C_s)$. Then the total time cost for training our framework is $O(T_eT_iC_wX)$, where $X = C_f$ for the basic method, $X = C_{clu} + C_f$ for the hard clustering method and $X = C_s$ for the soft clustering method. In the recommendation domain, compared to the training complexity, the inference cost can be more important, as it directly impacts the user online experience. Our framework does not bring additional inference cost, since the original model architectures remain unchanged.

Algorithm 1: Learning algorithm for RSR

```
Initialize the number of training epochs T_e.
Initialize the iteration number T_i.
Initialize the learning rate α.
Initialize the parameters Θ_f, Ω, w, c and μ.
for e=1 to T_e do
    Basic Method:
    for i=1 to T_i do
        Obtaining w by maximizing L_basic.
        Update Θ_f by: Θ_f ← Θ_f − α∇_{Θ_f} L_basic.
    end
    Or Hard Clustering Method:
    for i=1 to T_i do
        Obtaining c and μ by minimizing L_cl.
        Obtaining w by maximizing L_hard.
        Update Θ_f by: Θ_f ← Θ_f − α∇_{Θ_f} L_hard.
    end
    Or Soft Clustering Method:
    for i=1 to T_i do
        Obtaining w by maximizing L_soft.
        for s in [Θ_f, Ω, c, μ] do
            Update s by: s ← s − α∇_s L_soft.
        end
    end
end
```

3.4.2 Theoretical Understandings. In this section, we theoretically analyze our framework by relating it with the ideal expectation loss based on the theory of probably approximately correct (PAC) learning [12]. For easy analysis, we focus on the hard clustering method, and rewrite L_{hard} as follows: $L_{\text{hard}} = \sum_{i=1}^K w_i\delta_i(\Theta_f)$, where $\delta_i(\Theta_f) = \sum_{u\in C_i} L_u(\Theta_f)$. $\boldsymbol{C}_i$ is the set of users whose sequences are in the ith cluster, that is, $\boldsymbol{C}_i = \{u|c_u = i\}$. Then, we have the following theory:

Theorem 1. *Suppose the Rademacher complexity of $\delta_i(\Theta_f)$ is $\mathcal{R} = E_{C_i}[E_{\epsilon_v}[\sup_{\Theta_f}\sum_{(H_u^{t-1},v_u^t)\in C_i}\epsilon_v L_{ut}(\Theta_f)]]$, then for any testing distribution, given $\kappa \in (0,1)$, the following inequality holds with probability at least $1-\kappa$:*

$$L_{hard}^*(\Theta_f^*) \le \min_{\Theta_f}\max_{\boldsymbol{w}\in\Delta^{K-1}} L_{hard} + Const,$$

where $L_{hard}^*(\Theta_f^*) = \sum_{i=1}^K w_i\delta_i^*(\Theta_f^*)$, and δ_i^* is the expectation loss of δ_i. Θ_f^* is the optimal solution. Const is a constant.

Proof. To begin with, according to the basic PAC learning theory [12] on Rademacher complexity, we have: $\delta_i^*(\Theta_f) \le \delta_i(\Theta_f) +$

$2\mathcal{R}+B\sqrt{\frac{\log(1/\kappa')}{|C_i|}}$ holds with probobility at least $1-\kappa'$, where B is a constant. Considering that $L^*_{\text{hard}}(\Theta^*_f)=\sum_{i=1}^K w_i\delta_i^*(\Theta^*_f)$, we have:

$$
\begin{aligned}
&P(\max_{\boldsymbol{w}\in\Delta^{K-1}}\sum_{i=1}^K w_i\left[\delta_i(\Theta^*_f)+2\mathcal{R}+B\sqrt{\frac{\log(1/\kappa')}{|C_i|}}\right]\geq\sum_{i=1}^K w_i\delta_i^*(\Theta^*_f))\\
\geq&P(\sum_{i=1}^K w_i\left[\delta_i(\Theta^*_f)+2\mathcal{R}+B\sqrt{\frac{\log(1/\kappa')}{|C_i|}}\right]\geq\sum_{i=1}^K w_i\delta_i^*(\Theta^*_f))\\
\geq&P(\delta_1(\Theta^*_f)+2\mathcal{R}+B\sqrt{\frac{\log(1/\kappa')}{|C_1|}}\geq\delta_1^*(\Theta^*_f)\cap\dots\\
&\delta_i(\Theta^*_f)+2\mathcal{R}+B\sqrt{\frac{\log(1/\kappa')}{|C_K|}}\geq\delta_K^*(\Theta^*_f))\\
=&1-P(\delta_1(\Theta^*_f)+2\mathcal{R}+B\sqrt{\frac{\log(1/\kappa')}{|C_1|}}\geq\delta_1^*(\Theta^*_f)\cup\dots\\
&\delta_i(\Theta^*_f)+2\mathcal{R}+B\sqrt{\frac{\log(1/\kappa')}{|C_K|}}\geq\delta_K^*(\Theta^*_f))\\
\geq&1-\sum_{i=1}^K P(\delta_i(\Theta^*_f)+2\mathcal{R}+B\sqrt{\frac{\log(1/\kappa')}{|C_i|}}\geq\delta_i^*(\Theta^*_f))\\
\geq&1-K\kappa'.
\end{aligned}
$$

Here, $C=2\mathcal{R}+B\sqrt{\frac{\log(K/\kappa)}{|C_i|}}$. The first and second inequalities hold because if $A\Rightarrow B$, then $P(A)\leq P(B)$. The third inequality holds because $P(A\cup B)=P(A)+P(B)-P(AB)\leq P(A)+P(B)$.

Denoting $K\kappa'=\kappa$, then we have the following inequality holds with probobility at least $1-\kappa$:

$$
\begin{aligned}
L^*_{\text{hard}}(\Theta^*_f)&\leq\max_{\boldsymbol{w}\in\Delta^{K-1}}\sum_{i=1}^K w_i\delta_i(\Theta^*_f)+Const\\
&=\min_{\Theta_f}\max_{\boldsymbol{w}\in\Delta^{K-1}}\sum_{i=1}^K w_i\delta_i(\Theta_f)+Const\\
&=\min_{\Theta_f}\max_{\boldsymbol{w}\in\Delta^{K-1}}L_{\text{hard}}+Const,
\end{aligned}
$$

where $Const=2\mathcal{R}+B\sqrt{\frac{\log(K/\kappa)}{|C_i|}}$ is a constant. □

This theorem builds a connection between our actual optimization target and the ideal expectation loss, which reveals the theoretical foundations of our framework.

4 RELATED WORK

Our paper aims to improve the robustness of sequential recommender models. In the following, we review the related work on sequential recommendation and robust optimization, respectively.

4.1 Sequential Recommendation

Sequential recommendation [3–5, 17, 20, 21] has been demonstrated to be a promising direction due to its effectiveness on modeling user dynamic preference and better recommendation performance. In early days, the models usually focused on capturing the item-to-item transition patterns based on Markov chains [13, 27]. For example, FPMC [29] combines the advantages of Markov chains and matrix factorization for modeling the user general and sequential preferences simultaneously. With the recent advances of the neural network, many deep sequential models have been proposed. They mostly aim to incorporate longer user history information for predicting the next item. For example, Caser [31] leverages convolutional neural network to capture the item transition patterns. GRU4Rec [16] uses recurrent neural network to compress the user history interactions into an embedding, which is leveraged to predict the next item. To better leverage the history information, NARM [19] designs attention mechanism to discriminate items with different importances. SASRec [17] further leverages self-attention to capture the long-range item correlations, which is expected to enhance the significance of the items interacted longer before. Beyond specific model architectures, various researchers have also tried to improve the training process. For example, S^3Rec [36] builds many auxiliary contrastive tasks to improve the learning of the user and item representations. Inspired by contrastive learning, ICLRec [6] designs sequence- and feature-level contrastive tasks for building more effective sequential recommender frameworks. While the above models have achieved remarkable successes, they mostly assume that the training and testing environments are similar, *e.g.*, having the same sample distributions. In our work, we relax this assumption, and try to build a more practical sequential recommender framework, which can effectively handle the distribution shift between the training and testing sets.

4.2 Robust Optimization

Robust optimization has been widely investigated in the machine learning community. There are usually two types of approaches, that is, adversarial robustness and distributional robustness. Our paper is more related with distributional robustness, which aims to handle the distribution shift between the training and testing sets. Along this research line, [26] proposed a hierarchical method for improving the model distributional robustness. [10] designed a dynamic importance weighting mechanism to improve the model performance. In the recommendation domain, [33] developed a streaming-DRO framework to reduce the loss variances of the recommendation objectives. [15] designed an invariant representation learning method based on causal inference to improve the model robustness. In our paper, we use DRO to build a more robust sequential recommender framework, a novel approach that has not been widely explored in the recommendation domain. Additionally, we design both hard and soft clustering methods to further enhance the effectiveness of our framework. Lastly, we theoretically analyze our framework based on the PAC learning theory, where we find that, in our framework, the minimax objective of DRO can upper bound the expected loss under the ideal distribution.

5 EXPERIMENTS

In this section, we conduct extensive experiments to demonstrate the effectiveness of our framework, where we mainly focus on the following important research questions: **RQ1**. Whether our framework can improve the robustness of the sequential recommender models? **RQ2**. How different distribution shift influence the performance of our framework? **RQ3**. How different hyper-parameters influence the performance of our framework? **RQ4**. How different components in our framework contribute the final performance? In

Table 1: Statistics of the experiment datasets. ASL is short for "average sequence length".

Dataset	# User	# Item	# Interactions	ASL	Sparsity
Sports	35,598	18,357	286,207	8.02	99.95%
Toys	19,412	11,924	156,072	8.04	99.93%
Yelp	30,431	20,033	282,399	9.28	99.95%

the following, we firstly introduce the experiment setup, and then present and analyze the results to answer the above questions.

5.1 Experiment Setup

5.1.1 Datasets. Our experiments are conducted based on the following three real-world datasets: **Sports** and **Toys** are e-commerce datasets[2], which are collected from Amazon and contain the user purchasing records on the sports and toy products, respectively. **Yelp** is a restaurant review dataset[3], which contains a large amount of comments from the users. We can see they can cover different application domains, and for each dataset, we keep the users and items with more than five interactions. The dataset statistics are summarized in Table 1.

5.1.2 Baselines. To demonstrate the effectiveness and generality of our framework, we apply it to the following representative sequential recommender models for evaluation: **GRU4Rec** [16] is an early neural sequential recommender model based on the gated recurrent unit (GRU). **NARM** [19] is an attentive sequential recommender model, where the history items are merged with different attention weights. **SASRec** [17] is a state-of-the-art sequential recommender model, where any two history items can be directly interacted by the self-attention mechanism. We use B-X, H-X and S-X to represent the models obtained by applying our basic method, hard clustering method and soft clustering method on the base sequential recommender model X, where X can be GRU4Rec, NARM or SASRec. For reference, we also introduce the following well-known general recommender models as baselines: **BPR** [28] is a widely used recommender model for capturing user implicit feedback. **NCF** [14] is a neural model, which generalizes MF by stacking multiple non-linear layers on the user/item embeddings.

5.1.3 Implementation details. In the experiments, the last and second last items in each user interactions are used for testing and validation, and the other ones are left for model training. Since we aim to evaluate the model robustness, we deliberately perturb the training sets by adding noises on each sample. Specifically, we firstly introduce many fake items, which will not appear in the testing set. And then, for each training sample, we replace $\beta\%$ items with the fake ones to simulate distribution shifts. Larger β means larger distribution shift. In the experiments, we tune $\beta\%$ to study the performance of our framework under different distribution shifts. The widely used metrics including *Recall* and *NDCG* are leveraged for model evaluation, and we recommend twenty items to compare with the ground truth. The hyper-parameters of our framework are determined by grid search. In specific, the learning rate and batch size are tuned in the ranges of $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$ and $\{32, 64, 128, 256, 512, 1024\}$, respectively. The number of clusters K is searched in $\{4, 8, 16, 32, 64, 128\}$. We empirically set the item embedding size as 50, and for SASRec, the numbers of self-attention blocks and attention heads are both set as 2. We limit the maximum sequence length by 20. We use Adam [18] to optimize all the base sequential recommender models. For the other baseline parameters, we tune them in the same ranges as our methods. All the experiments are conducted on a single NVIDIA A40 GPU. For more implementation details, we refer the readers to our project, which has been released at https://anonymousrsr.github.io/RSR/.

[2]http://jmcauley.uscd.edu/data/amazon/
[3]https://www.yelp.com/dataset

5.2 Overall Performance

The overall comparison results are presented in Table 2, where we can see: among the baselines, the sequential recommender models can usually achieve better performance than the general ones, which agrees with the previous work [16], and confirms the importance of incorporating the history information for user preference prediction. Between different sequential models, we find that SASRec can always achieve better performance than NARM and GRU4Rec. The reason can be that, in SASRec, any two items can be associated with each other by the self-attention mechanism. Such architecture ensures that even if an important item appears at a distant position, it can also directly influence the next item prediction, which facilitates better recommendation performance.

Encouragingly, by applying our framework on the base sequential recommender models, the performances have been improved, and the results are consistent across different datasets, evaluation metrics and base models. In specific, the best of our framework can, on average, improve the base model by about 3.59% and 3.61% on Recall and NDCG, respectively, which demonstrates the effectiveness and generality of our framework under different settings.

Among our models, the hard clustering method can outperform the basic one in more cases. As we have mentioned before, in the basic method, there are too many sample weights, directly optimizing them can be less efficient for simulating the distributions and further alleviating the distribution shift problem. In addition, the user preferences can be diverse on high-level features. Assigning each sequence with a unique weight (like the basic method) may fail to leverage the feature-level sequence similarity information, and letting the model to automatically learn such information is too flexible. As a result, we can observe improved performance of the hard clustering method. Not surprisingly, we find that the soft clustering method can lead to the best performance on all the datasets and evaluation metrics. Comparing with the hard clustering method, the soft one can effectively leverage the recommendation signals, that is, the clustering process can be well supervised by the user-item interactions, and the cluster assignment distribution $\boldsymbol{c}_u$ is learned by jointly considering the clustering and recommendation losses. However, in the hard clustering method, the clustering and recommendation objectives are separated, and the samples can be clustered in a sub-optimal manner, which can be even harmful to the recommendation performance.

5.3 Influence of Different Distribution Shifts

The main focus of this paper is to develop a distributionally robust sequential recommender framework. Readers can be interested in whether our framework can be generally effective for different

Table 2: Overall comparison between our framework and the baselines. In this experiment, the noisy item ratio β is set as 5%. We use bold fonts to label the best performance for each dataset, evaluation metric and base model. The performance improvements of our framework against the base model are significant under paired-t test.

Methods	Sports			Toys			Yelp		
	Recall@20	NDCG@20	MRR@20	Recall@20	NDCG@20	MRR@20	Recall@20	NDCG@20	MRR@20
BPR	0.0186	0.0069	0.0038	0.0266	0.0091	0.0043	0.0211	0.0090	0.0041
NCF	0.0165	0.0065	0.0038	0.0212	0.0072	0.0034	0.0228	0.0093	0.0056
GRU4Rec	0.0425	0.0179	0.0111	0.0524	0.0240	0.0160	0.0640	0.0268	0.0161
B-GRU4Rec	0.0427	0.0181	0.0115	0.0529	0.0241	0.0159	0.0629	0.0255	0.0155
H-GRU4Rec	0.0430	0.0181	0.0115	0.0525	0.0240	0.0161	0.0646	0.0267	0.0163
S-GRU4Rec	**0.0439**	**0.0189**	**0.0120**	**0.0559**	**0.0253**	**0.0170**	**0.0652**	**0.0272**	**0.0165**
NARM	0.0372	0.0153	0.0091	0.0466	0.0211	0.0140	0.0649	0.0273	0.0166
B-NARM	0.0380	0.0155	0.0095	0.0468	0.0207	0.0133	0.0652	0.0272	0.0167
H-NARM	0.0381	0.0158	0.0097	0.0477	0.0216	0.0143	0.0653	0.0277	0.0170
S-NARM	**0.0391**	**0.0161**	**0.0103**	**0.0495**	**0.0220**	**0.0147**	**0.0657**	**0.0281**	**0.0176**
SASRec	0.0483	0.0198	0.0129	0.0827	0.0369	0.0235	0.0724	0.0377	0.0284
B-SASRec	0.0477	0.0191	0.0124	0.0822	0.0366	0.0233	0.0720	0.0380	0.0286
H-SASRec	0.0500	0.0205	0.0132	0.0814	0.0370	0.0238	0.0737	0.0383	0.0289
S-SASRec	**0.0508**	**0.0206**	**0.0134**	**0.0838**	**0.0375**	**0.0241**	**0.0736**	**0.0384**	**0.0294**

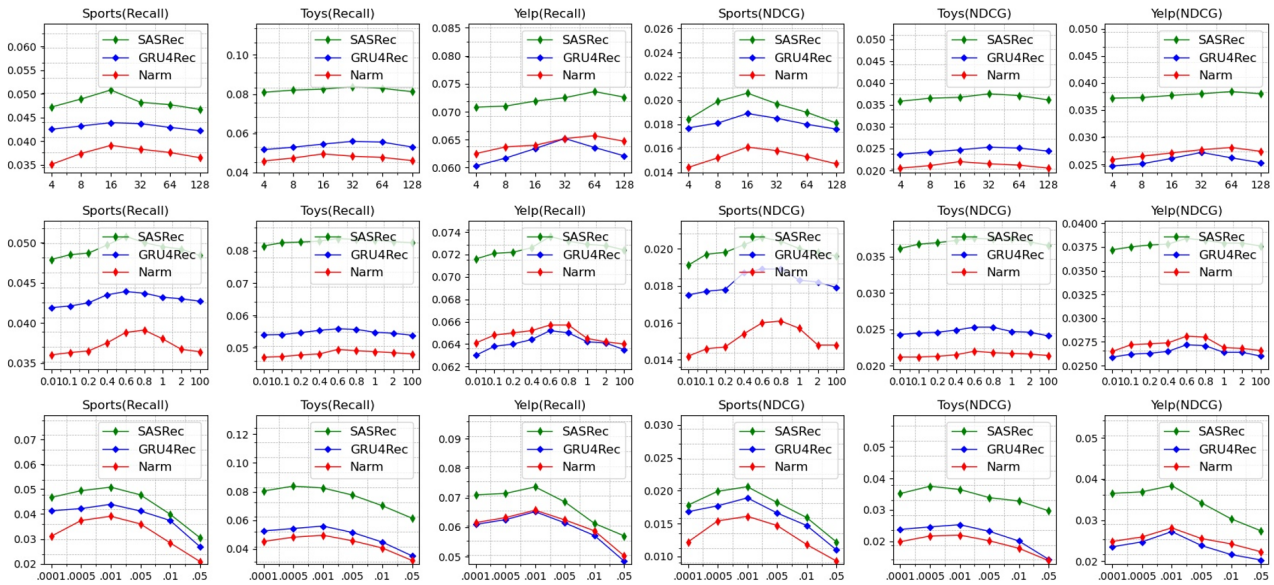

Figure 2: Experiment results on the hyper-parameter influences. In the first row, we present the influence of the number of clusters K. In the second row, we present the influence of the temperature parameter τ. In the third row, we present the influence of the balancing weight λ.

distribution shifts. To study this problem, in this section, we remain the testing set, and build different training samples by tuning the noisy item ratio β in the range of $\{0, 5, 10, 20\}$, where a larger β corresponds to a larger distribution shift. In the experiment, we evaluate different models based on Recall@20 and NDCG@20. We focus on the improvement of our framework on sequential models, and the general recommendation baselines are omitted in the final results. The model parameters are set as their optimal values tuned in Table 2, and the other settings follow the above experiment.

From the results presented in Table 3, we can see: as the noisy ratio β becomes larger, the performances of our framework and the base models are lowered in most cases. This is as expected, since the noisy items can mislead the models to learn incorrect item transition patterns. Since we do not change the testing set, the testing items are still correlated according to the original rules, which makes the models not perform well. For each noisy ratio, our framework can always achieve better performance than the base models. In addition, for any sequential model X, S-X can usually perform better than H-X, which can further improve the performance of B-X

Table 3: Performance comparison under different distribution shifts. Similar to Table 2, we use bold fonts to label the best performance. The performance improvements are significant under paired-t test. For simplicity, we remove "@20" in the metrics, and use "G", "N" and "S" to represent GRU4Rec, NARM and SASRec, respectively.

Dataset	Noisy	Metric	G	B-G	H-G	S-G	N	B-N	H-N	S-N	S	B-S	H-S	S-S
Sports	0%	Recall	0.0445	0.0451	0.0456	**0.0457**	0.0390	0.0395	0.0398	**0.0401**	0.0529	0.0527	0.0533	**0.0537**
		NDCG	0.0186	0.0190	0.0198	**0.0202**	0.0159	0.0167	0.0167	**0.0169**	0.0217	0.0214	0.0219	**0.0219**
	5%	Recall	0.0425	0.0427	0.0430	**0.0439**	0.0372	0.0380	0.0381	**0.0390**	0.0483	0.0477	0.0500	**0.0508**
		NDCG	0.0179	0.0181	0.0181	**0.0189**	0.0153	0.0155	0.0158	**0.0161**	0.0198	0.0191	0.0205	**0.0206**
	10%	Recall	0.0411	0.0419	0.0421	**0.0425**	0.0335	0.0363	0.0361	**0.0367**	0.0465	0.0472	0.0491	**0.0494**
		NDCG	0.0172	0.0178	0.0176	**0.0186**	0.0135	0.0146	0.0144	**0.0148**	0.0188	0.0191	0.0199	**0.0200**
	20%	Recall	0.0342	0.0340	0.0346	**0.0353**	0.0255	0.0287	0.0279	**0.0288**	0.0383	0.0393	0.0399	**0.0402**
		NDCG	0.0141	0.0140	0.0145	**0.0148**	0.0102	0.0121	0.0116	**0.0121**	0.0153	0.0158	0.0155	**0.0159**
Toys	0%	Recall	0.0549	0.0532	0.0538	**0.0574**	0.0511	0.0504	0.0516	**0.0523**	0.0884	0.0885	0.0863	**0.0890**
		NDCG	0.0252	0.0239	0.0245	**0.0261**	0.0222	0.0223	0.0225	**0.0231**	0.0378	0.0382	0.0369	**0.0382**
	5%	Recall	0.0524	0.0529	0.0525	**0.0559**	0.0466	0.0468	0.0477	**0.0495**	0.0827	0.0822	0.0814	**0.0838**
		NDCG	0.0240	0.0241	0.0240	**0.0253**	0.0211	0.0207	0.0216	**0.0220**	0.0369	0.0366	0.0370	**0.0375**
	10%	Recall	0.0444	0.0455	0.0471	**0.0486**	0.0422	0.0430	0.0443	**0.0462**	0.0785	0.0796	0.0798	**0.0801**
		NDCG	0.0199	0.0209	0.0216	**0.0219**	0.0190	0.0192	0.0194	**0.0202**	0.0334	0.0339	0.0340	**0.0341**
	20%	Recall	0.0355	0.0342	0.0353	**0.0376**	0.0273	0.0291	**0.0326**	0.0312	0.0634	0.0655	0.0643	**0.0657**
		NDCG	0.0162	0.0156	0.0159	**0.0165**	0.0118	0.0125	**0.0142**	0.0139	0.0271	0.0279	0.0276	**0.0279**
Yelp	0%	Recall	0.0663	0.0647	0.0678	**0.0680**	0.0660	0.0670	0.0668	**0.0672**	0.0741	0.0749	0.0733	**0.0752**
		NDCG	0.0272	0.0265	0.0277	**0.0280**	0.0274	0.0281	0.0280	**0.0282**	0.0394	0.0396	0.0390	**0.0399**
	5%	Recall	0.0640	0.0629	0.0646	**0.0652**	0.0649	0.0652	0.0653	**0.0657**	0.0724	0.0720	**0.0737**	0.0736
		NDCG	0.0268	0.0255	0.0267	**0.0272**	0.0273	0.0272	0.0277	**0.0281**	0.0377	0.0380	0.0383	**0.0384**
	10%	Recall	0.0602	0.0575	0.0597	**0.0611**	0.0621	0.0623	0.0609	**0.0627**	0.0676	0.0692	0.0685	**0.0704**
		NDCG	0.0259	0.0235	0.0244	**0.0255**	0.0250	0.0260	0.0257	**0.0264**	0.0371	0.0372	0.0372	**0.0375**
	20%	Recall	0.0440	0.0436	**0.0465**	0.0453	0.0472	0.0425	0.0475	**0.0486**	0.0659	0.0632	0.0643	**0.0671**
		NDCG	0.0168	0.0163	**0.0182**	0.0175	0.0192	0.0172	0.0194	**0.0200**	0.0355	0.0346	0.0351	**0.0362**

in most cases. These results are consistent with our observations in Table 2, which demonstrates that our framework is effective for different distribution shifts, and the performance rankings among S-X, H-X, and B-X are highly consistent. Our framework can be used in various scenarios without worrying too much about the distribution shift levels. Comparing the performance improvement of our framework for different β values, we find that the improvement tends to be more significant on datasets with larger distribution shifts. In specific, the performance improvement for $\beta = 0$, $\beta = 5$, $\beta = 10$, $\beta = 20$ is 2.27%, 3.59%, 8.78%, 15.51% on Recall and 3.51%, 3.61%, 8.91%, 16.36% on NDCG, respectively.

5.4 Hyper-parameter Analysis

Hyper-parameters are usually important in recommender models. In this section, we would like to study the influences of the hyper-parameters in our framework. In specific, we focus on three important hyper-parameters, that is, the number of clusters K, the temperature parameter τ and the balancing weight λ. When studying one parameter, we fix the other ones. We focus on the soft clustering method due to its better performance, and the noisy item ratio is set as 5%. The other settings follow the above experiment.

5.4.1 Influence of the cluster number K. In our framework, K indicates the number of clusters. Intuitively, if K is too small, the samples can be not well separated, which may bring difficulties for clearly approximating the distributions. If K is too large, there are too many sample weights, which is hard to optimize. To study the influence of K, we tune it in the range of $\{4, 8, 16, 32, 64, 128\}$. From the results presented in the first row of Figure 2, we can see: for different datasets and evaluation metrics, the performances of our framework continue to rise as K increases. After reaching the optimal points, the performances go down if we still enlarge K. The best performances are usually achieved when $K \in [8, 32]$. This observation is not surprising, since when K is a small value, many items with different categories are grouped into the same cluster. If the distribution shift exactly happens on these categories, then our framework can be less effective, since the sample weights in these categories are the same, and we cannot tune the weight ratio between them. However, if K is too large, many similar items can be grouped into different clusters, this enhances the hardness for distribution simulation, since the models have to automatically learn the similarity information. Based on the above analysis, we observe that the performances are usually better when K is moderate.

Table 4: The results of comparing our framework with its two variants. We use the same abbreviations and fonts as those of Table 3. "↓" means that the performance of the variant is lowered.

Dataset	Metric	S-G	S-G(-Rec)	S-G(-Sc)	S-N	S-N(-Rec)	S-N(-Sc)	S-S	S-S(-Rec)	S-S(-Sc)
Sports	Recall	**0.0439**	0.0402 ↓	0.0384 ↓	**0.0391**	0.0330 ↓	0.0314 ↓	**0.0508**	0.0483 ↓	0.0467 ↓
	NDCG	**0.0189**	0.0171 ↓	0.0163 ↓	**0.0161**	0.0137 ↓	0.0124 ↓	**0.0206**	0.0197 ↓	0.0185 ↓
Toys	Recall	**0.0559**	0.0515 ↓	0.0503 ↓	**0.0495**	0.0439 ↓	0.0421 ↓	**0.0838**	0.0817 ↓	0.0804 ↓
	NDCG	**0.0253**	0.0218 ↓	0.0210 ↓	**0.0220**	0.0184 ↓	0.0167 ↓	**0.0375**	0.0357 ↓	0.0351 ↓
Yelp	Recall	**0.0652**	0.0611 ↓	0.0597 ↓	**0.0657**	0.0617 ↓	0.0604 ↓	**0.0736**	0.0708 ↓	0.0689 ↓
	NDCG	**0.0272**	0.0240 ↓	0.0229 ↓	**0.0281**	0.0249 ↓	0.0241 ↓	**0.0384**	0.0363 ↓	0.0354 ↓

5.4.2 Influence of the temperature parameter τ. The temperature parameter τ determines the sharpness of the distribution represented by $\boldsymbol{w}$. If $\tau \to 0$, then $\boldsymbol{w}$ is a one-hot vector, that is, there is only one "1", and all the other elements are "0". On the other hand, if $\tau \to \infty$, then $\boldsymbol{w}$ represents a uniform distribution, that is, all the elements are the same. To study the influence of τ, we tune it in the range of $\{0.01, 0.1, 0.2, 0.4, 0.6, 0.8, 1, 2, 100\}$. From the results shown in the second row of Figure 2, we can see: for different datasets, evaluation metrics and base models, the best performances are always achieved when τ falls between 0.4 and 0.8. We speculate that, if τ is set too large, then different clusters are assigned with similar weights. In such a scenario, our framework may fail to simulate the testing distributions that are significantly different from the training ones. If τ is set too small, only one cluster is highlighted in the optimization process. However, if the testing distribution is materially different from the training distribution and does not align with this cluster, it may limit the model's performance. Therefore, we find that a moderate τ value tends to yield better results.

5.4.3 Influence of the balancing parameter λ. In our framework, λ balances the importances between the recommendation and soft clustering losses. If λ is set as a small value, then the model focuses more on the recommendation performance, otherwise, the model cares more about the clustering effect. To study its influences, we tune it in the range of $\{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05\}$. From the results presented in the third row of Figure 2, we can see: for different settings, the best performances of our framework are usually achieved when λ takes a relatively small value. The reason can be that, when λ is too large, then the objective focuses more on the clustering loss. However, the evaluation is conducted based on the recommendation performance. Thus, the learned parameters can be sub-optimal. We have also noticed that if λ is too small, then the performance is also not satisfactory. We speculate that too small λ may lead to inaccurate item clustering, which may fail to accurately capture the sequence similarity information and limit the capability of our framework for distribution simulation.

5.5 Ablation Studies

In all the above experiments, our framework is evaluated as a whole. In this section, we would like to study how different components in our framework contribut to the final performance. Similar to the above experiment, we focus on the soft clustering method. In specific, we compare the final framework with its two variants: in S-X(-Rec), we remove the reconstruction loss L_{rec} in L_{soft}. In S-X(-Sc), we remove the soft clustering loss L_{sc} in L_{soft}. "X" is the base model, which can be GRU4Rec, NARM or SASRec. The other settings of the experiment follow the above section.

From the results presented in Table 4, we can see: the reconstruction loss is important, since for all the base models, datasets and evaluation metrics, the performances of S-X(-Rec) are worse than S-X . This observation confirms our design of introducing the reconstruction loss for enhancing the sample representations. The soft clustering loss is also significant to our framework, which is evidenced by the lowered performance of S-X(-Sc) as compared with S-X. This observation is as expected, since without the clustering loss, the items are clustered in a random manner, which may fail to accurately capture the sequence similarity information and effectively simulate the distributions. By combining L_{rec} and L_{sc}, our final framework can always achieve the best performance, which suggests that both of these terms are necessary.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we propose to build a distributionally robust sequential recommender framework by optimizing loss with the "worst" sample distribution, which is expected to guarantee that the loss with the real testing distribution is not too bad. Additionally, we introduce two improved models that cluster the samples in hard and soft manners, respectively, to alleviate the challenge of determining sample weights. Moreover, we provide theoretical analysis on our proposed framework, which helps to justify its effectiveness. This paper made an initial step toward distributionally robust sequential recommendation with an abundance of opportunities for further improvement in the future. For instance, the optimal number of clusters in our framework is a crucial parameter that significantly impacts the performance of the model. Nevertheless, determining it using grid search may not be feasible in practical scenarios. Hence, an intriguing possibility for future research is the design of automatic strategies to learn the number of clusters jointly with the model. Moreover, the incorporation of auxiliary information to cluster the samples has the potential to significantly improve the accuracy and precision of our proposed framework. Additionally, extending our framework to graph-based recommendation systems presents a more comprehensive and challenging task, but holds great promise for enhancing the versatility and applicability of our proposed approach to a wider range of applications.

7 ACKNOWLEDGEMENT

This work is supoorted in part by National Key R&D Program of China 2022ZD0120103 and National Natural Science Foundation of China (No. 62272467 and 62102420).

REFERENCES

[1] Dimitris Bertsimas, Vishal Gupta, and Nathan Kallus. 2018. Data-driven robust optimization. *Mathematical Programming* 167 (2018), 235–292.

[2] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. 2018. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*. 132–149.

[3] Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. 2021. Sequential recommendation with graph neural networks. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 378–387.

[4] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 108–116.

[5] Yongjun Chen, Jia Li, Chenghao Liu, Chenxi Li, Markus Anderle, Julian McAuley, and Caiming Xiong. 2021. Modeling Dynamic Attributes for Next Basket Recommendation. *arXiv preprint arXiv:2109.11654* (2021).

[6] Yongjun Chen, Zhiwei Liu, Jia Li, Julian McAuley, and Caiming Xiong. 2022. Intent Contrastive Learning for Sequential Recommendation. In *Proceedings of the ACM Web Conference 2022*. 2172–2182.

[7] Erick Delage and Yinyu Ye. 2010. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations research* 58, 3 (2010), 595–612.

[8] John Duchi and Hongseok Namkoong. 2018. Learning models with uniform performance via distributionally robust optimization. *arXiv preprint arXiv:1810.08750* (2018).

[9] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. 2019. Adversarial robustness as a prior for learned representations. *arXiv preprint arXiv:1906.00945* (2019).

[10] Tongtong Fang, Nan Lu, Gang Niu, and Masashi Sugiyama. 2020. Rethinking importance weighting for deep learning under distribution shift. *Advances in Neural Information Processing Systems* 33 (2020), 11996–12007.

[11] Bram L Gorissen, İhsan Yanıkoğlu, and Dick den Hertog. 2015. A practical guide to robust optimization. *Omega* 53 (2015), 124–137.

[12] David Haussler. 1990. *Probably approximately correct learning*. University of California, Santa Cruz, Computer Research Laboratory Santa ....

[13] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 191–200.

[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[15] Yue He, Zimu Wang, Peng Cui, Hao Zou, Yafeng Zhang, Qiang Cui, and Yong Jiang. 2022. CausPref: Causal Preference Learning for Out-of-Distribution Recommendation. In *Proceedings of the ACM Web Conference 2022*. 410–421.

[16] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[17] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*. IEEE, 197–206.

[18] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[19] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.

[20] Yang Li, Tong Chen, Peng-Fei Zhang, and Hongzhi Yin. 2021. Lightweight Self-Attentive Sequential Recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 967–977.

[21] Zhiwei Liu, Yongjun Chen, Jia Li, Philip S Yu, Julian McAuley, and Caiming Xiong. 2021. Contrastive self-supervised sequential recommendation with robust augmentation. *arXiv preprint arXiv:2108.06479* (2021).

[22] J MacQueen. 1967. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*. 281–297.

[23] Pratyush Maini, Eric Wong, and Zico Kolter. 2020. Adversarial robustness against the union of multiple perturbation models. In *International Conference on Machine Learning*. PMLR, 6640–6650.

[24] Hongseok Namkoong and John C Duchi. 2016. Stochastic gradient methods for distributionally robust optimization with f-divergences. *Advances in neural information processing systems* 29 (2016).

[25] Yonatan Oren, Shiori Sagawa, Tatsunori B Hashimoto, and Percy Liang. 2019. Distributionally robust language modeling. *arXiv preprint arXiv:1909.02060* (2019).

[26] Qi Qian, Juhua Hu, and Hao Li. 2020. Hierarchically robust representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7336–7344.

[27] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International conference on data mining*. IEEE, 995–1000.

[28] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[29] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. 811–820.

[30] Uri Shaham, Yutaro Yamada, and Sahand Negahban. 2018. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing* 307 (2018), 195–204.

[31] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*. 565–573.

[32] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. 2020. Measuring robustness to natural distribution shifts in image classification. *Advances in Neural Information Processing Systems* 33 (2020), 18583–18599.

[33] Hongyi Wen, Xinyang Yi, Tiansheng Yao, Jiaxi Tang, Lichan Hong, and Ed H Chi. 2022. Distributionally-robust Recommendations for Improving Worst-case User Experience. In *Proceedings of the ACM Web Conference 2022*. 3606–3610.

[34] Wolfram Wiesemann, Daniel Kuhn, and Melvyn Sim. 2014. Distributionally robust convex optimization. *Operations Research* 62, 6 (2014), 1358–1376.

[35] Chunting Zhou, Daniel Levy, Xian Li, Marjan Ghazvininejad, and Graham Neubig. 2021. Distributionally robust multilingual machine translation. *arXiv preprint arXiv:2109.04020* (2021).

[36] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1893–1902.