Learning Recommender Systems with Implicit Feedback via Soft Target Enhancement

Mingyue Cheng[1†], Fajie Yuan[3,5†], Qi Liu[1,2*], Shenyang Ge[4],
Zhi Li[1], Runlong Yu[2], Defu Lian[1,2], Senchao Yuan[2], Enhong Chen[1,2]
[1]Anhui Province Key Lab of Big Data Analysis and Application,
School of Data Science, University of Science and Technology of China, Hefei, China
[2]School of Computer Science and Technology, University of Science and Technology of China, Hefei, China
[3]Westlake University, Hangzhou, China; [4]JD Inc, Beijing, China; [5] Tencent Inc, Shenzhen, China
{mycheng,zhili03,yrunl, yuansc}@mail.ustc.edu.cn, fajieyuan@westlake.edu.cn, {qiliuql, liandefu, cheneh}@ustc.edu.cn

ABSTRACT

One-hot encoder accompanied by a softmax loss has become the default configuration to deal with the multiclass problem, and is also prevalent in deep learning (DL) based recommender systems (RS). The standard learning process of such methods is to fit the model outputs to a one-hot encoding of the ground truth, referred to as the hard target. However, it is known that these hard targets largely ignore the ambiguity of unobserved feedback in RS, and thus may lead to sub-optimal generalization performance.

In this work, we propose SoftRec, a new RS optimization framework to enhance item recommendation. The core idea is that we add additional supervisory signals - well-designed soft targets - for each instance so as to better guide the recommender learning. Meanwhile, we carefully investigate the impacts of specific soft target distributions by instantiating the SoftRec with a series of strategies, including item-based, user-based, and model-based. To verify the effectiveness of SoftRec, we conduct extensive experiments on two public recommendation datasets by using various deep recommendation architectures. The experimental results show that our methods achieve superior performance compared with the standard optimization approaches. Moreover, SoftRec could also exhibit strong performance in cold-start scenarios where user-item interaction has higher sparsity.

CCS CONCEPTS

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

KEYWORDS

Implicit Feedback; Item Recommendation; Soft Target

†Equal contribution. This work was done when Fajie worked at Tencent (past affiliation) and Westlake University (current affiliation).
*Qi Liu is corresponding author.

SIGIR '21, July 11–15, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3462863

ACM Reference Format:
Mingyue Cheng[1†], Fajie Yuan[3,5†], Qi Liu[1,2*], Shenyang Ge[4],, Zhi Li[1], Runlong Yu[2], Defu Lian[1,2], Senchao Yuan[2], Enhong Chen[1,2]. 2021. Learning Recommender Systems with Implicit Feedback via Soft Target Enhancement. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3404835.3462863

1 INTRODUCTION

Online consumptions (e.g., purchasing items, watching videos.) have become increasingly popular with the rapid development of Internet services. Meanwhile, people repeatedly encounter the choice problem because of Information Overload [17]. Recommender systems (RS) have become an important tool to address such problems. The key of personalized RS is to model users' preferences on items based on their past interactions, known as collaborative filtering (CF). There are two common tasks in the recommendation literature: rating prediction with explicit feedback [27, 28, 48] and item recommendation from implicit feedback [29, 50]. To be specific, rating prediction is to estimate unseen ratings based on user rating history while item recommendation is to predict a personalized ranking on a set of candidate items. In the past decade, the research on RS has shifted from explicit feedback problem to implicit feedback given that most signals in user actions are implicit [26, 49].

Deep learning (DL) has made massive strides in many research areas, continually achieving breakthrough performance in recent years. Meanwhile, the integration of DL in recommendation [47, 59] has demonstrated the advantages of complex network architectures over traditional factorization models in user/item representation learning. Many newly proposed deep learning architectures, such as attention-based networks [24, 44], convolution neural networks (CNNs) [42, 53], graph neural networks (GNNs) [45], have been successfully applied in personalized recommender systems.

In parallel, a remaining challenge for implicit RS is how to formulate the loss function over implicit feedback and how to perform general optimizations for various deep recommender models. In general, there are three popular ways that cast the item recommendation problem into a supervised machine learning (ML) problem [36], namely, pointwise loss [33], pairwise loss [7, 34, 37] and softmax loss [6, 9, 14, 22, 53]. In particular, softmax loss that includes a softmax activation plus a default cross-entropy loss is becoming increasingly popular due to its simplicity, ease of use for

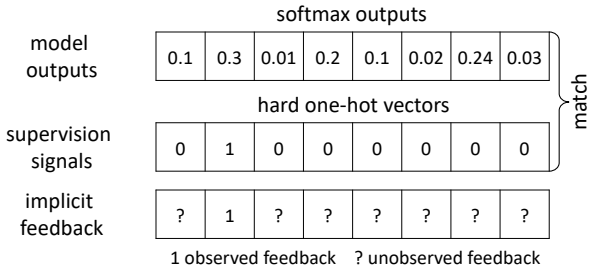

Figure 1: An example of multiclass optimization.

DL models, and wide applications in many other research domains, such as computer vision (CV) [55] and natural language processing (NLP) [20]. The core idea of softmax loss is to interpret the item recommendation as the multiclass or multinomial classification problem — i.e., by fitting the softmax output into the one-hot vector space, as shown in Figure 1. Concretely, the optimization of it is to find a group of model parameters to force the position of the target item (or class) in the one-hot vector to be a value of 1 and all other positions with values of 0 [43]. In fact, the way of using this multiclass softmax loss in RS has no difference with that in other fields, where, for example, in the NLP models the classes are words, and in the CV models the classes are categories. It is perhaps for this reason that so far very few work are devoted to exploring a fundamental research question: *whether such 'standard' multiclass classification practice is optimal for item recommendations with implicit feedback?*

In this paper, we argue that the typical softmax loss has an obvious drawback when dealing with implicit feedback. To be specific, such multiclass classification methods tend to ignore the ambiguity of missing feedback due to leveraging one-hot vectors as supervisory signals. That is, the unobserved items can be just missing data rather than a signal of dislike. Actually, this unobserved preference problem is not at all a new research question for the recommendation task, for example, in [33, 51] where authors solve it by considering the pointwise loss, and in [37, 52] where they combine it with pairwise loss. However, to the best of our knowledge, this problem has never been investigated when using the softmax loss, which is as important and influential as the pointwise and pairwise loss functions.

Targeting at this problem, in this paper, we propose a novel optimization framework, termed as SoftRec, to enhance item recommendation from implicit feedback when using the softmax loss during optimization. The core idea of SoftRec is to incorporate additional prior of soft targets to smooth the original hard targets (i.e., one-hot vectors) so as to guide the process of recommender training. Specifically, we introduce three different strategies, including item-based, user-based, and model-based, to instantiate our framework for fully exploring the influence of specific target distributions. Each one is based on its own concern, i.e., item-based strategies focus on leveraging the dataset prior knowledge; user-based strategies aim to extract soft targets from the neighborhood information; model-based strategies mainly explore the generation of the soft targets via knowledge distillation from the same architecture or different architectures. On two benchmark datasets, we evaluate these strategies over four representative recommendation models: GRU4Rec [15], Caser [42], NFM [13] and Youtube Recommender (YoutubeDNN for short) [9]. The experimental results show SoftRec yields significant improvements over the standard multiclass optimization approach using only the softmax loss.

The contributions of this paper are summarized as follows:

- We explore a fundamental optimization problem for deep recommender models with implicit feedback, i.e., the widely used multiclass classification method with the softmax loss might result in only sub-optimal performance for the item recommendation problem due to the use of hard targets as supervisory signals.
- We formulate a more effective optimization framework namely SoftRec, which incorporates the priors of soft targets as additional supervisory signals. To instantiate the SoftRec framework, we explore a series of soft target generating strategies to instantiate the SoftRec from three perspectives: item-based, user-based and model-based.
- We compare SoftRec with the standard optimization method with four well-known DL-based recommendation architectures on two benchmark datasets. The results consistently demonstrate that SoftRec can achieve appealing performance compared with their counterparts. Further, we also show that the proposed optimization of SoftRec can significantly improve the performance of the cold-start issues, where user-item interactions are highly scarce.

2 PRELIMINARIES

In this paper, we consider a generic deep recommender model as multiclass classifier $h_\theta(c, i) : C \rightarrow \mathcal{I}$, in which the relevance of a context $c \in C$ to an item $i \in \mathcal{I}$ can be computed by a score function $h_\theta(c, i)$. Here, C and $\mathcal{I}$ denote two set of entries. The set C denotes the set of M contexts while set $\mathcal{I} = \{o_1, o_2, ..., o_N\}$ represents the set of N items. Note, to be generic, we use context to represent user, time, location, sequence of previous selected items and so on. Assuming the deep recommender model $h = g \circ f$ can be decomposed of a user embedding function $f_\varphi : C \rightarrow Z$, which transforms the input context information into logit vectors, and a *softmax* output layer is widely adopted as $g_\vartheta : Z \rightarrow \mathcal{I}$.

2.1 Multiclass Optimization for Item Recommendation

As mentioned before, item recommendation can often be formulated as the multiclass classification problem by fitting the outputs of the recommender model to the one-hot supervisory signals, where class labels here are the items in the systems. Correspondingly, each input context needs to be assigned with the label of the item. To do this, a softmax layer is usually placed on the final layer of the deep recommender model, which is a default choice for transforming the model output to a probabilistic distribution. This naturally aligns with the sentiment that the label with the highest score matches the context. Without loss of generality, cross entropy loss is used as the loss function by computing the distance between softmax output and target item distribution. In the standard optimization manner, the supervisory signals of context c are usually represented by only a one-hot vector $q(i|c)$ over a set of items, with the coordinate corresponding to the target item is set to 1. Mathematically, for a

given context item pair (c, i), the optimization objective can then be formulated as:

$$\mathcal{H}(q, p) = -\sum_{i \in \mathcal{I}} q(i|c) \log p(i|c), \tag{1}$$

where $p(i|c) = h_\theta(c) = \text{softmax(z)}$, $z = f_\theta(c)$. As we know, minimizing the cross-entropy is equivalent to minimize the negative log-likelihood objective [43]. In this way, we could obtain following the maximum likelihood estimation (MLE) objective,

$$\begin{aligned} \mathcal{L}(c, q) &= -\log h_\theta(c), \\ &= -h_\theta(c, i) + \log \sum_{i' \in \mathcal{I}} \exp(h_\theta(c, i')), \end{aligned} \tag{2}$$

where $h_\theta(c, i)$ denotes the user preference score between context c and item i.

The goal of the above optimization strategy is to find a group of network parameters so that the position of the target item is forced to 1 and all others are forced to 0. Although such an optimization manner has become a standard practice in the ML community for multiclass classification, we argue that it might be too strict for the recommendation task given that it simply treats all unobserved data as negative feedback, where the ambiguity of missing feedback is largely neglected. With such an objective, preferences of the huge unobserved items cannot be well exploited. That is, it might not be suitable to solely maximize the predicted score of the ground-truth item. In contrast, it is more reasonable by allowing some potential matched items to have non-zero supervisory signals. Unfortunately, the standard one-hot encoding of target items fails to exploit the difference of unobserved feedback. Hence, we argue that such standard practice by using the softmax loss and hard target ground truth could be too strict and lead to sub-optimal results for item recommendation from implicit feedback.

3 METHOD

The former section has demonstrated that the one-hot encoding would lead to sub-optimal results for learning actual user preference on unobserved feedback. Motivated by this, we set our goal to assign more *informative* supervisory signals for training examples so as to improve recommendation accuracy.

This section elaborates our proposed methods. Specifically, we first give a formal definition of the well-informed soft targets and present some practical solutions under the guidance of the definition. Finally, we provide a summary and remark for the proposed optimization framework.

3.1 The SoftRec Optimization Framework

As discussed above, we are aware that the one-hot label might not be an appropriate supervision signal for guiding recommender training due to simply ignoring the ambiguity of missing feedback problems. Under this circumstance, a natural idea is to use additional soft targets as supervisory signals for each instance. To be more specific, we need to assign unobserved feedback a non-zero value signal, which denotes the *preference level* between the user and this candidate item. Likewise, the principle for designing soft targets should also follow the important ranking constraint: positive (or target) items are ranked higher than other unobserved

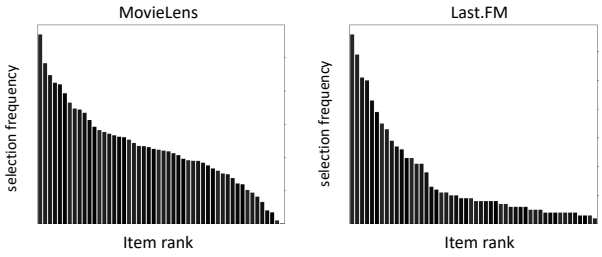

Figure 2: Item popularity in two example datasets.

feedback [37]. Thus, we develop the following formal definition of *well-informed soft targets*.

Definition 1. *We define the soft target $d = \{d^{o_1}, d^{o_2}, ..., d^{o_N}\}$, is subject to $\sum d^{o_i} = 1$ and $d^{o_i} \in [0, 1]$, for each instance. In the whole soft targets, each unlabeled item is assigned a non-zero supervisory signal value, which denotes the user's preference for the item. Particularly, the argmax output of the soft target should be equal to the position of target item in one-hot vectors for each instance.*

Take inspirations from label smoothing regularization [41, 55], we smooth the original hard targets with the soft targets and obtain new target q'. Mathematically, the label fusing process can be formulated as

$$q'(i|c) = (1 - \alpha) q(i|c) + \alpha d, \tag{3}$$

where the controlling parameter $\alpha \in [0, 1]$ denotes the coefficient to trade off two types of supervisory signals. With the mixed target q', we reformulate the objective Eq (1) and get a new optimization framework as follows,

$$\begin{aligned} \mathcal{L}(c, q') &= -\sum_{i \in \mathcal{I}} q'(i|c) \log p(i|c) \\ &= (1 - \alpha) \mathcal{H}(q(i|c), p(i|c)) + \alpha \mathcal{H}(d, p(i|c)). \end{aligned} \tag{4}$$

From a density estimation point of view, minimizing cross-entropy is equivalent to optimizing the Kullback-Leibler (KL) divergence $\mathcal{D}_{KL}$, i.e., $\mathcal{D}_{KL} = \mathcal{H}(d, p(i|c)) - \mathcal{H}(d)$, where the entropy $\mathcal{H}(d)$ is a constant for a fixed distribution. Thus, we can further reformulate Eq (4) to

$$\begin{aligned} \mathcal{L}(c, q') &= (1 - \alpha) \mathcal{H}(q(i|c), p(i|c)) + \alpha (\mathcal{D}_{KL}(d, p(i|c)) + \mathcal{H}(d)) \\ &= (1 - \alpha) \mathcal{H}(q(i|c), p(i|c)) + \alpha \mathcal{D}_{KL}(d, p(i|c)). \end{aligned} \tag{5}$$

We name the newly proposed optimization objective as SoftRec. In the new optimization framework, If $\alpha = 0$, the optimization objective is reduced back to the original multiclass cross entropy objective. Otherwise, this new optimization framework not only penalizes errors relevant to the target item but also errors relevant to these unobserved data so that the recommendation model can find a compromise between two types of supervisory signals. In this way, the new optimization framework allows us to further explore the unobserved item preference by assigning unobserved data with non-zero value supervisory signals.

3.2 Instantiations of the SoftRec

According to the above description, incorporating high-quality soft targets could have an important impact on the optimization

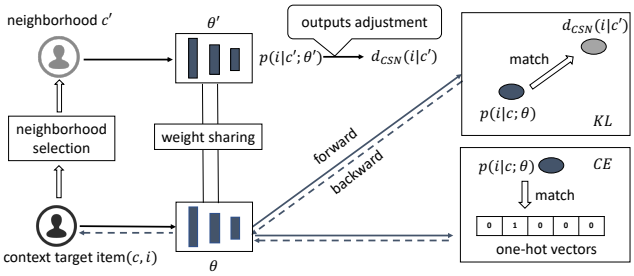

Figure 3: Collaborative Siamese Network.

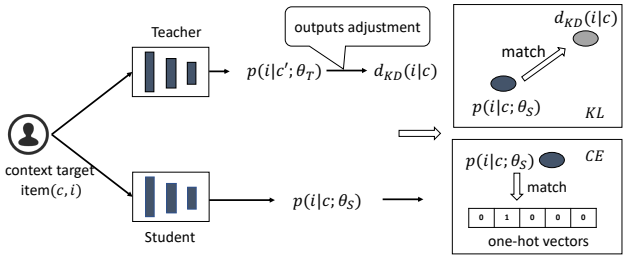

Figure 4: Teacher-Student Training Network.

performance. Though the optimal well-informed soft targets are hard to achieve, we can still try to simulate it by leveraging the rich information from the data and model. In the following, we will present a series of generating strategies to explore the specific soft target distribution d, which includes item-based, user-based, and model-based.

3.2.1 ***Item-Based Strategy.*** In this part, we attempt to add the soft targets for each instance from the item side. Prior literature [52] has pointed out that when training recommender models from implicit feedback, an item popularity-based negative sampler usually exceeds the random negative sampler. Inspired by this, we plan to generate soft targets by leveraging item popularity. To be specific, we denote item popularity distribution with $d_{pop} = \{f^{o_1}, f^{o_2}, ..., f^{o_N}\}$, where f^{o_i} represents the clicking frequency of item i for all users. It has been recognized that item popularity distribution in most real-world recommendation datasets has a non-uniform distribution, following an approximate power-law or exponential distribution. We have also depicted the popularity distribution of our datasets, as shown in Figure 2. Generally, the more popular an item is, the more times it acts as a positive one since popular items are more likely to be suggested by recommender systems. In this way, generating soft targets by normalizing popularity distribution should be beneficial.

However, one important problem is that the *argmax* of normalized popularity distribution might not match with the one-hot vectors. As a result, it would violate the basic constraint in Definition 1 as is illustrated in Section 3.1 if we directly adopt such a normalized distribution as soft targets. Hence, we further take an adjustment for the original normalized popularity as follows,

$$d_{pop+}(i|c) = \frac{1}{2}(\text{softmax}(d_{pop}/\mathcal{T}) + q(\text{i|c})), \tag{6}$$

where softmax$(\cdot)$ is used to normalize the popularity distribution, and we also add temperature $\mathcal{T}$ to softmax so as to scale the overall popularity distribution. Note that temperature $\mathcal{T}$ should be carefully set or scheduled. Combining the formulated SoftRec optimization framework before, the popularity-based instantiation strategies can be given as

$$\mathcal{L}_{pop+} = (1-\alpha)\mathcal{H}(q(i|c), p(i|c)) + \alpha\mathcal{D}_{KL}(d_{pop+}, p(i|c)). \tag{7}$$

It means that the deep recommender is not only guided by the original hard targets (i.e., one-hot vectors) but also supervised by the popularity-based soft targets with KL divergence. Here, we name such item-based instantiation manner as POP+ strategy.

3.2.2 ***User-Based Strategy.*** Now, we turn to instantiate the SoftRec framework from the user side. That is, we are committed to providing each instance with personalized soft targets. Past works have proven that neighborhood, as a type of local information, plays an important role in RS for complementing the user-item interaction data [2]. Hence, an intuitive idea is to leverage neighborhood information to help generate soft targets. However, it is non-trivial for deep recommendation model to achieve this purpose. We have to solve the following two issues: (1) how to obtain the neighborhood of each user (context), and (2) how to model such neighborhood information in deep recommendation model.

For the first question, we consider the historical interactions of each user as his/her pre-existing features, following [51]. We first embed each item into latent vector space by Item2vec [12] in bipartite graph, and then obtain the embedding of each user. In this paper, we choose the cosine similarity as the measurement to compute the similarity distance between contexts. However, it will be inefficient for comparing with all other context information due to a large number of context sets. To solve the inefficiency problem, we adopt the similar embedding search solution in [19], which can work well in terms of searching efficiency problem and achieve near-optimal performance.

For the second question, taking inspirations from metric learning methods [8, 32], we design a Collaborative Siamese Network (CSN) to extract the knowledge from the neighborhood to generate soft targets. For a better illustration, we show the proposed CSN model in Figure 3. The key point of the CSN network is to feed pairs of examples, including current context information and its neighborhood, into a single network. Then, we consider matching the prediction results between the two examples.

For a given context c and its neighborhood c', we can respectively obtain two prediction results $p(i|c;\theta_{CSN})$ and $p(i|c';\theta'_{CSN})$, where θ'_{CSN} is a fixed copy of parameter θ_{CSN}. It should be noted that the gradient is not propagated through θ'_{CSN} to avoid the model collapse issue [30, 56]. Next, one may wonder that *argmax* of the prediction results in terms of neighborhood c' might not keep pace with the original one-hot label $q(i|c)$. Hence, we smooth the prediction results of the neighborhood with one-hot label $q(i|c)$, ensuring the soft distribution to satisfy the basic constraint in Definition 1. Formally, the fusing process can be written as

$$d_{CSN} = \frac{1}{2}(p(i|c';\theta'_{CSN},\mathcal{T}) + q(i|c)), \tag{8}$$

where θ'_{CSN} denotes the copy version of θ_{CSN}. Note that we also adopt the temperature softmax to control the whole distribution.

Revisiting the formulated SoftRec framework in Section 3.1, the total training loss of proposed CSN network can be defined as

$$\mathcal{L}_{CSN} = (1-\alpha)\mathcal{H}(q(i|c), p(i|c;\theta_{CSN})) + \alpha\mathcal{D}_{KL}(d_{CSN}, p(i|c;\theta_{CSN})). \tag{9}$$

In this way, the loss can enforce consistent predictive distributions between context c and its neighborhood c'. In addition, the whole algorithm is explained in Algorithm 1.

Algorithm 1 Collaborative Siamese Network

1: Initialize parameters θ_{CSN}.
2: **while** θ_{CSN} has not converged **do**
3: Sample a batch (c, q) from the training datasets.
4: Sample a batch c' from the neighborhood of batch c.
5: Update parameters θ_{CSN} by computing the gradients of the proposed loss function in Eq (9).
6: **end while**

In summary, we can generate additional soft targets for each instance by fully utilizing the localized information, i.e., neighborhood. For each context, each instance is supervised by two types of supervisory signals including both the hard one-hot target and the prediction results of its neighborhood.

3.2.3 ***Model-Based Strategy.*** Knowledge distillation (KD) [16], which extracts the dark knowledge from a teacher network to guide the learning process of a student network, has become an important technique for model compression and transfer learning. It has been shown that the student model can achieve comparable or sometimes even higher accuracy than its teacher network by KD [10, 55, 57]. Inspired by this, we aim to leverage the KD framework to implement the proposed SoftRec, since the core idea of KD well matches our demand for SoftRec. Here, we denote the teacher network as T and the student network as S, as shown in Figure 4.

By leveraging KD, we can treat the outputs of a pre-trained teacher network $p(i|c;\theta_T)$ as soft targets for each training instance, where θ_T denotes the parameters in the pre-trained teacher. To do this, one question may arise: the maximum prediction in the teacher's softmax outputs might not be consistent with the position of value 1 in the one-hot encoding of target items. Since these incorrect predictions may mislead the learning of student model, we hence smooth teacher outputs with the one-hot vectors $q(i|c)$. Formally, such a process can be represented as

$$d_{KD} = \frac{1}{2}(p(i|c;\theta_T, \mathcal{T}) + q(i|c)). \tag{10}$$

After obtaining the informative soft target information. Based on above formulated SoftRec framework, we formulate the specific optimization goal as

$$\mathcal{L}_{KD} = (1-\alpha)\mathcal{H}(q(i|c), p(i|c;\theta_S)) + \alpha\mathcal{D}_{KL}(d_{KD}, p(i|c;\theta_S)), \tag{11}$$

where θ_S denotes the parameters of the student model. Specifically, we explore the KD from two different perspectives. On the one hand, we allow a student network to learn from the same architecture, assuming that the teacher network can provide extra information [1]. We name such a knowledge distillation strategy as self knowledge distillation (SKD). On the other hand, holding that it keeps a higher diversity between different architectures, we hence propose encouraging the student network to mimic the teacher network crossing architectures to explore more ability. We name such a knowledge transfer process as Cross-architecture Knowledge Distillation (CKD). In summary, we distill the soft targets by encouraging the student network to mimic the teacher network including both the same architecture and different architectures.

3.3 Summary and Remarks

It should be noted that in this paper we only focus on formulating the training optimization strategy, which is applicable to different deep recommender models equipped with the softmax function in the last layer. We instantiate SoftRec with a series of strategies, which deeply explore the impacts of a specific soft target distribution for implicit recommender systems. Although the proposed three instantiation strategies are intuitively simple, they are effective in boosting the recommendation performance. That is, our proposals are not the only solution in practice, and we encourage more effective generation approaches could be explored and further improve our SoftRec framework [11].

4 EXPERIMENTS

The key contribution of this work is to design effective soft targets so as to provide more informative supervisory signals for implicit feedback when using the standard multiclass softmax loss. To evaluate SoftRec, we use two public datasets and explore multiple deep recommendation architectures. In general, we aim to answer the following research questions:

- **RQ1**: Whether the proposed soft targets help to enhance the accuracy of deep recommender models from implicit feedback?
- **RQ2**: Is the SoftRec framework generic to various recommendation architectures?
- **RQ3**: Is SoftRec also well-suited to the sampled softmax loss [18] which is an efficient alternative for the standard full softmax loss?
- **RQ4**: How does SoftRec perform under a very difficult recommendation setting, e.g., for the cold-start problem?

4.1 Experimental Settings

4.1.1 ***Datasets.*** Since the core contributions of this paper do not focus on the exploration of various features, we simply treat user historical interactions as context features.

- **MovieLens**[1]: To alleviate the impact of cold users and items, we perform the basic pre-processing by filtering out interactions (rates) with less than 5-score and users with less than 10 items, following [37]. Then, we define the maximum length of the interaction sequence as 30. Sessions longer than 30 is split into additional sequences, while sequences shorter than 30 is padded with zero in the beginning of the sequence to reach 30, similar to [40].
- **Last.FM**[2]: We randomly pick 199 thousand songs from the original Last.FM data. We define the session length as 20, and extract 20 successive items as input sequence. This is done

[1] http://files.grouplens.org/datasets/movielens/
[2] http://www.dtic.upf.edu/ocelma/MusicRecommendationDataset/lastfm-1K.html

Table 1: Statistics of the datasets.

DATA	# sequences	# items	length
MovieLens	858K	18K	30
Last.FM	535K	199K	20

by sliding a window of both size and stride of 20 over the whole data. We ignore sessions where the time span between the last two items is longer than 2 hours.

The statistics of our datasets after the basic pre-processing are described in Table 1.

4.1.2 ***Evaluation Protocols.*** We evaluate all models by employing three popular top-K metrics, namely MRR@K (Mean Reciprocal Rank), NDCG@K (Normalized Discounted Cumulative Gain), and Recall@K. K is set to 10 for comparison. Following previous works [21, 39, 58], we apply the *leave-one-out* strategy for evaluation. Specifically, for each user sequence, the last item is used as the test data, the item before the last one is used as the validation data, and the remaining data is used as training set. In addition, we evaluate each method on the whole item set rather than the sampled metrics (e.g., with 100 randomly selected negative items) which are questioned by [23].

4.1.3 ***Testing Recommenders and Compared Methods.*** To verify whether the SoftRec framework is generic, we apply the four on four representative deep recommender models that are proven to yield strong performances. Specifically we select two sequential-aware algorithms (**GRU4Rec** [15], **Caser** [42]) and two feature-based algorithms, **NFM** [13] and Youtube Deep Neural Network (**YoutubeDNN** for short) [9]. Meanwhile, we treat the standard softmax loss (short for **Base**) and label smoothing (**LS**) [35] method as our competitive baselines. Note we emphasize that the purpose of our study is not to judge which loss — binary pointwise, binary pairwise, or multiclass softmax loss — performs the best, since they were shown different strengths and weaknesses under various hyper-parameter settings or on different datasets [4, 14, 14, 36]. The purpose is rather to assess whether there is still significant room to improve the 'standard' softmax loss widely recognized and used for deep recommender models with implicit feedback data.

- **Base** denotes the original multiclass classification optimization framework by optimizing the cross-entropy loss between model outputs and the hard one-hot target.
- **LS** employs the label smoothing regularization strategy to smooth the hard one-hot target as soft supervisory signals for each instance during training.
- **POP+** is our proposed **item-based** strategy by adjusting item popularity distribution according to our principle 1,2 as soft supervisory signals for model training.
- **CSN** is our proposed **user-based** strategy using the proposed Collaborative Siamese Network (CSN) to implement the our proposed optimization framework.
- **SKD** is our proposed **model-based** strategy by distilling knowledge from the same model architecture.
- **CKD**: also denotes the **model-based** strategy using the knowledge distillation framework but with different types of architectures as the teacher model. That is, for GRU4Rec, we use Caser as its teacher, and for NFM, we use YoutubeDNN as its teacher, and vice versa.

4.1.4 ***Implementation Details.*** All models were trained on GPUs using Tensorflow. To ensure a fair evaluation, hyper-parameter are fine-tuned on the validation set and shared for the base model and SoftRec given they use exactly the same backbone architecture. To be specific, we randomly initialize model parameters using Gaussian distribution. We set the embedding dimension e to 128 for all models. The hidden dimensions are set the same value as the embedding dimension. Though methods with other e ($e.g., e = 64, 256, 512$) yield different results, the performance trend keeps similar. The learning rate is set to 0.001 in this paper. We use Adam as the optimizer. The learning rates for Adam with 0.001 to 0.0001 show consistent trends. Batch size is set 128 for all models. The controlling parameter α is searched from $\{0.1, 0.2, .., 1.0\}$ while temperature $\mathcal{T}$ is searched from $\{1, 2, ..., 10\}$.

4.2 Experimental Results

4.2.1 ***Recommendation Performances (RQ1 & RQ2).*** To show the effectiveness of SoftRec, we evaluate it by specifying four types of deep recommendation architectures, namely GRU4Rec, Caser, NFM, and YoutubeDNN. We report the overall results in Table 2. First, we observe consistent performance gains when comparing the proposed SoftRec (POP+, CSN, SKD, CKD) with the base optimization strategy, which demonstrates the effectiveness by using soft targets as an additional supervisory signal. For example, CSN, by considering the prediction results of neighborhoods, the SoftRec framework can achieve obvious improvements in most situations. The main reason behind such phenomenon is because the deep recommender optimized by our proposed SoftRec framework can not only capture the relation between user context and target items, but also learn the implicit relations between these candidate items by fitting the soft target. Further, LS by label smoothing also improves the base optimization. However, LS in general underperforms SoftRec because the newly incorporated soft target distribution is simply assigned with uniform supervisory signal without taking any types of user preference into account. In addition, we observe that sequential recommender models (i.e., GRU4Rec and Caser) largely surpass the feature-based models (i.e., NFM and YoutubeDNN), similar to the finding in [46]. The reason is that sequential models can capture the dynamics of user behaviors while the non-sequential models only treat these interactions as common features. In brief, these extensive experimental results well back up our claims that (1) using soft targets instead of simply adopting one-hot targets can enhance the item recommendation performance to a large extent. (2) SoftRec is not specialized to a specific network architecture given the consistent improvements for all base models.

4.2.2 ***SoftRec with Sampled Softmax (RQ3).*** In practice, the item size in a large-scale recommender systems could be very huge, leading to prohibitive computation if using the standard (full) softmax loss. As such, the sampled softmax loss [6, 18] is even more popular for the item recommendation task [53]. Here we want to evaluate SoftRec by using sampled softmax.

We report main results in Figure 5. As shown, we can see that recommender models with the sampled softmax loss in general

Table 2: Item recommendation performance based on the full softmax loss using two benchmark datasets on four deep recommender. The best performance methods are denoted in bold.

Recommender	Datasets	Measures	Base	LS	POP+	CSN	SKD	CKD
GRU4Rec	MovieLens	MRR@10	0.0502	0.0508	**0.0584**	0.0538	0.0512	0.0526
		NDCG@10	0.0666	0.0681	**0.0759**	0.0713	0.0688	0.0694
		Recall@10	0.1212	0.1251	**0.1338**	0.1290	0.1273	0.1253
	Lastfm.FM	MRR@10	0.2464	0.2568	**0.2718**	0.2686	0.2767	0.2686
		NDCG@10	0.2611	0.2716	**0.2877**	0.2840	0.2918	0.2840
		Recall@10	0.3076	0.3180	**0.3381**	0.3322	0.3394	0.3322
Caser	MovieLens	MRR@10	0.0474	0.0504	0.0596	0.0565	0.0594	**0.0604**
		NDCG@10	0.0627	0.0665	0.0780	0.0747	0.0776	**0.0787**
		Recall@10	0.1133	0.1195	0.1386	0.1348	0.1378	**0.1389**
	Lastfm.FM	MRR@10	0.2294	0.2361	**0.2670**	0.2402	0.2411	0.2559
		NDCG@10	0.2488	0.2564	**0.2923**	0.2619	0.2632	0.2771
		Recall@10	0.3098	0.3204	**0.3709**	0.3301	0.3325	0.3434
NFM	MovieLens	MRR@10	0.0264	0.0267	0.0298	0.0290	**0.0315**	0.0300
		NDCG@10	0.0374	0.0381	0.0427	0.0415	**0.0438**	0.0433
		Recall@10	0.0742	0.0766	0.0859	0.0830	**0.0849**	0.0877
	Lastfm.FM	MRR@10	0.1012	0.1025	0.1103	**0.1228**	0.1132	0.1177
		NDCG@10	0.1205	0.1242	0.1298	**0.1407**	0.1348	0.1371
		Recall@10	0.1822	0.1930	0.1920	**0.1970**	0.2038	0.1986
YoutubeDNN	MovieLens	MRR@10	0.0221	0.0232	0.0238	**0.0255**	0.0244	0.0242
		NDCG@10	0.0328	0.0342	0.0352	**0.0362**	0.0356	0.0352
		Recall@10	0.0688	0.0708	0.0734	**0.0722**	0.0733	0.0722
	Lastfm.FM	MRR@10	0.0845	0.0866	0.0929	**0.1008**	0.0980	0.0998
		NDCG@10	0.1045	0.1101	0.1134	**0.1210**	0.1186	0.1218
		Recall@10	0.1690	0.1856	0.1791	**0.1855**	0.1838	0.1918

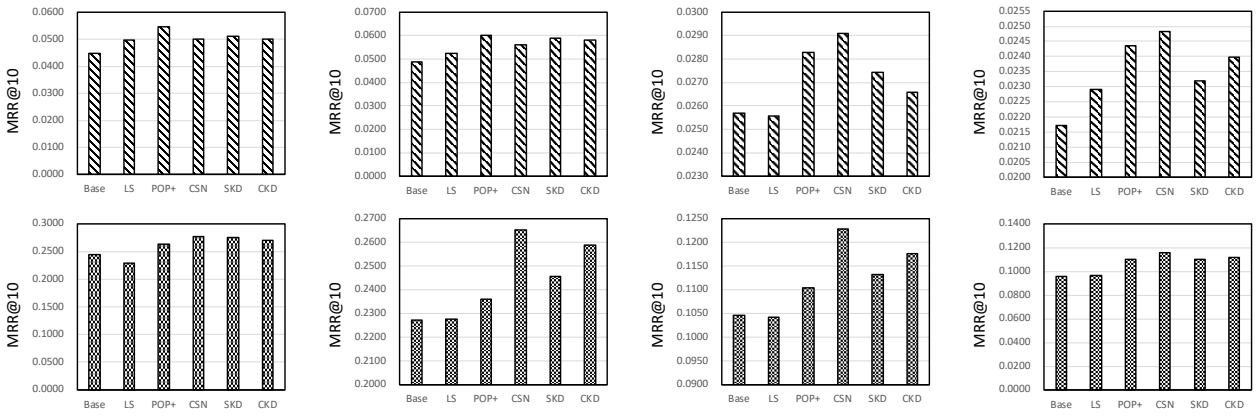

Figure 5: Item recommendation performance on the basis of sampled softmax loss with 20% sampling ratio using MovieLens (first line) and Last.FM (second line) on four recommender architectures.

perform a bit worse than the full softmax loss. Despite that, the performance improvements of SoftRec over the base methods are highly consistent. In fact, with sampled softmax loss, SoftRec sometimes could perform better than itself with full softmax. We believe this is reasonable since it is hard to judge whether the full soft target distributions generated by our three strategies are better or not than its sampled versions.

To conclude, item recommendation based on such sampled softmax loss can also be improved by our soft target strategies. That is because there is no fundamental difference between the full softmax and its sampled version in both intuition and theory. As a result, such sampled softmax has exactly the same hard issue as the standard softmax loss.

Table 3: Recommendation performances of new users cold-start on two datasets using GRU4Rec and Caser architecture.

	MovieLens						Last.FM					
	MRR@10		NDCG@10		Recall@10		MRR@10		NDCG@10		Recall@10	
	GRU4Rec	Ours	GRU4Rec	Ours	GRU4Rec	Ours	GRU4Rec	Ours	GRU4Rec	Ours	GRU4Rec	Ours
t1	0.0010	0.0042	0.0016	0.0064	0.0037	0.0139	0.0002	0.0007	0.0003	0.0011	0.0007	0.0023
t2	0.0019	0.0051	0.0031	0.0076	0.0069	0.0163	0.0003	0.0012	0.0004	0.0017	0.0009	0.0036
t3	0.0028	0.0063	0.0043	0.0095	0.0095	0.0204	0.0004	0.0019	0.0007	0.0028	0.0014	0.0058
t4	0.0104	0.0133	0.0156	0.0197	0.0330	0.0413	0.0034	0.0153	0.0045	0.0206	0.0082	0.0384
t5	0.0437	0.0502	0.0592	0.0667	0.1104	0.1215	0.1652	0.1873	0.1755	0.1995	0.2083	0.2384
	Caser	Ours	Caser	Ours	Caser	Ours	Caser	Ours	Caser	Ours	Caser	Ours
t1	/	0.0063	0.0001	0.0091	0.0001	0.0183	0.0015	0.0065	0.0020	0.0083	0.0035	0.0142
t2	0.0001	0.0128	0.0002	0.0185	0.0006	0.0373	0.0036	0.0127	0.0048	0.0167	0.0086	0.0299
t3	0.0004	0.0174	0.0008	0.0248	0.0020	0.0495	0.0047	0.0120	0.0064	0.0159	0.0122	0.0287
t4	0.0010	0.0307	0.0018	0.0425	0.0044	0.0815	0.0054	0.0326	0.0074	0.0400	0.0141	0.0641
t5	0.0458	0.0546	0.0604	0.0712	0.1087	0.1261	0.1318	0.1895	0.1454	0.2071	0.1884	0.2624

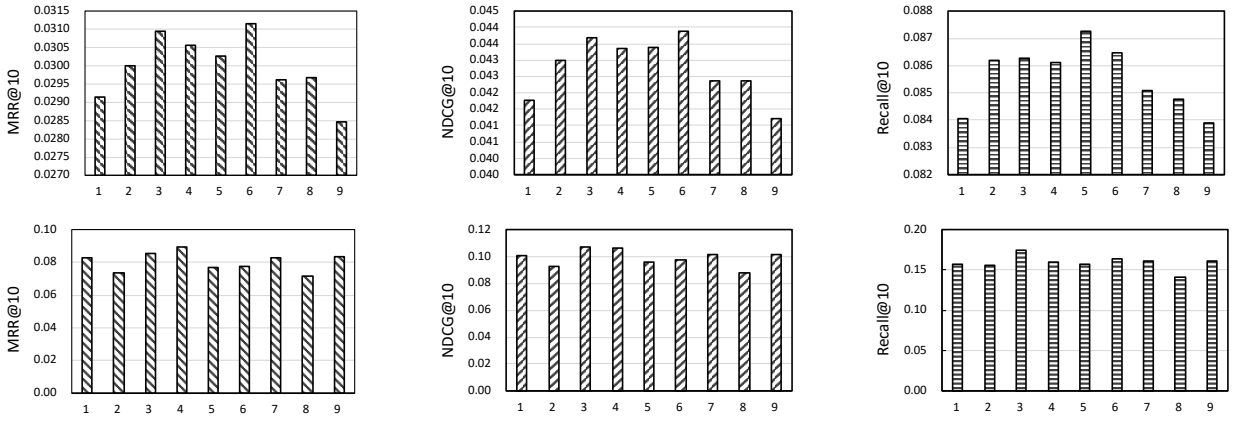

Figure 6: The effect of temperature $\mathcal{T}$ for the POP+ strategy with the NFM recommender on MovieLens (first line) and Last.FM (second line).

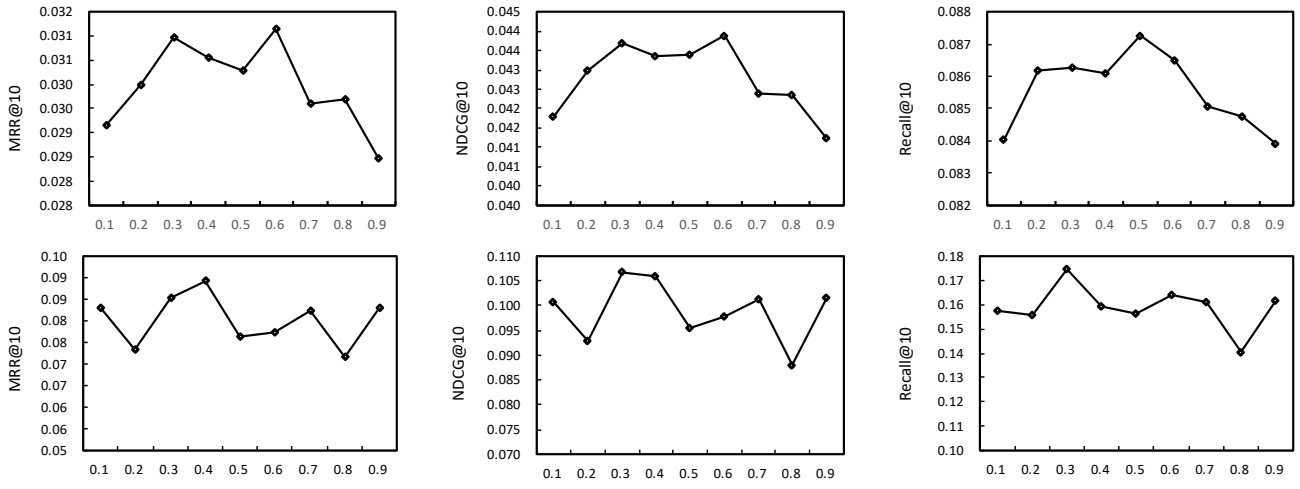

Figure 7: The effect of trade-off parameter α for the POP+ strategy with the Caser recommender on MovieLens (first line) and Last.FM (second line).

4.2.3 ***Cold Start Analysis (RQ4).*** Cold start is a common problem in recommender systems (RS) when new users or items have not yet obtained sufficient interactions [38]. Considering the soft targets as one type of prior knowledge, we turn to analyze the performance of SoftRec in the setting of the cold start problem. We report SoftRec with GRU4Rec as the base model following the same experimental settings in [25]. Specifically, we extract a number of items (denote as M) from the beginning of the original training sequence and train both models using the new dataset. Let $M = \{5, 15, 20, 25, 30\}$ for {t1, t2, t3, t4, t5} on MovieLens, and $M = \{4, 8, 12, 16, 20\}$ for {t1, t2, t3, t4, t5} on Last.FM, as shown in Table 3. Note that results on NFM and YoutubeDNN are highly consistent but are simply omitted for space reason.

We report results for the standard softmax loss (i.e., Base) and item-based SoftRec (i.e., POP+). We find that SoftRec significantly outperforms the baseline approach on all metrics, indicating the effects of SoftRec in the cold start setting. In particular, SoftRec outperforms the baseline with a large margin with t1. Then, by growing the number of users' interactions (i.e., t2, t3, t4 and t5), the improvement decreases gradually. Such results indicate SoftRec optimization strategy poses powerful capacity to handle the cold-start challenges of new/cold users. In view of this, we conclude that SoftRec can work very well for the new user recommendation problem. The results suggest that it is possible to transfer knowledge by a prior distribution while learning recommender models. We hope our findings could bring a new insight for alleviating such cold-start issues.

4.2.4 ***Hyper-parameter Sensitivity Analysis.*** In this section, we study how the hyper-parameter affect the performance. Our proposed SoftRec incorporate two important hyper-parameters, i.e., trade-off coefficient α and scaling parameter $\mathcal{T}$. For clarity and saving space purpose, we only report the results of NFM trained with item-based soft target strategy (i.e., POP+) in Figure 6 and Figure 7. Other strategies have the similar patterns. From the experimental results, we find that: 1) the recommender is relatively sensitive to α since the trade-off α decides how much the hard one-hot encoding will be changed by the newly added soft target loss; 2) SoftRec is slightly sensitive to the temperature $\mathcal{T}$ in Last.FM dataset but more sensitive to the data of MovieLens. Such experimental phenomenon might be caused by the different scales of the item set since $\mathcal{T}$ can make the soft target distribution either spiky (with low temperature) or uniform (high temperature). These findings provide insights on how to tune the hyper-parameters of SoftRec in practice.

5 RELATED WORK

Learning recommender systems from implicit feedback has become a research hotspot for over a decade [33, 37]. In general, one need to consider two aspects for generating high-quality recommendations, namely, designing a highly expressive neural network architecture as the scoring function and formulating a loss function over implicit feedback. In this paper, we pay our attention to the latter one.

Three popular loss functions have been proposed to cast the item recommendation problem into a supervised machine learning problem. They are pointwise loss (binary) [33, 51], pairwise loss (binary) [7, 37, 52], and softmax loss (multiclass) [6, 53, 54]. To be specific, pointwise loss function cast learning from implicit feedback as a classification or regression problem, where observed feedback is treated as positive, and unobserved (i.e., missing) feedback is often treated as weak negatives. To formulate the specific loss, one could assign $y = 1$ for the positive context-item (c, i) pair, while for unobserved instance (c, j), where $j \in \mathcal{I}$, one could assign $y = 0$ from the regression perspective, and $y = -1$ from the classification perspective. A weight can be assigned to each (c, i, j) tuple so as to represent the confidence of the training case [36]. Pairwise (a.k.a. BPR) loss is an another typical loss function for the implicit feedback problem, which was first introduced in [37]. The key motivation for pairwise loss is that for item recommendation, it is not of primary interest if a score $\hat{y}$ is 1 or 0. Instead, the relative ordering of items under a given context is a more important concern. Thereby, pairwise losses are mostly designed under a basic assumption that the relevance of a positive (c, i) pair is supposed to be higher than other unobserved (c, j) pairs. A key drawback of pairwise loss function is that all unobserved feedback is treated equally, while in practice some of them are harder to be learned than others. To address the hard negative examples, a series of negative sampling techniques have been proposed [7, 26, 52]. In addition to the above two losses, softmax loss [9, 53] can be regarded as a third option, and becomes particularly popular in recent years because of its wide applications in deep learning models. In domains like NLP and CV, softmax loss has even become a default choice for the classification problem [36]. Unlike pointwise and pairwise loss, softmax loss cast the item recommendation problem as the multinomial classification problem [9, 53]. In practice, an efficient sampling-based softmax [18], kernel-based softmax [23], hierachical softmax [31], and a two-pass sampler [3] are often used as alternatives for the full softmax, especially when the classes (i.e., items) in the system have a very large scale [5]. For future work, it is interesting to explore how to apply SoftRec for these different variants, beyond the full and sampled softmax studied in this paper.

6 CONCLUSIONS

In this work, motivated by the drawback of the standard multiclass optimization strategy for implicit recommender systems, we proposed a simple but very effective optimization framework, called SoftRec, in which informative soft target loss is designed to complement the standard hard target loss. To be specific, we contributed three specific instances: item-based, user-based, and model-based. These methods are very general and can be applied to different types deep recommender models. We conduct thorough experiments on two public datasets to show the effectiveness of SoftRec. Furthermore, we also empirically evaluated SoftRec in cold-start settings and confirmed that it could considerably improve the recommendation results.

Acknowledgements. This research was partially supported by grants from the National Key Research and Development Program of China (No. 2018YFC0832101), and the National Natural Science Foundation of China (Grants No. 61922073, U20A20229, 61976198 and 62022077), and the Fundamental Research Funds for the Central Universities.

REFERENCES

[1] Zeyuan Allen-Zhu and Yuanzhi Li. 2020. Towards Understanding Ensemble, Knowledge Distillation and Self-Distillation in Deep Learning. *arXiv preprint arXiv:2012.09816* (2020).

[2] Ting Bai, Ji-Rong Wen, Jun Zhang, and Wayne Xin Zhao. 2017. A neural collaborative filtering model with interaction-based neighborhood. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1979–1982.

[3] Yu Bai, Sally Goldman, and Li Zhang. 2017. Tapas: Two-pass approximate adaptive sampling for softmax. *arXiv preprint arXiv:1707.03073* (2017).

[4] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. 2017. A generic coordinate descent framework for learning from implicit feedback. In *WWW*. 1341–1350.

[5] Yoshua Bengio, Jean-Sébastien Senécal, et al. 2003. Quick Training of Probabilistic Neural Nets by Importance Sampling.. In *AISTATS*. 1–9.

[6] Guy Blanc and Steffen Rendle. 2018. Adaptive sampled softmax with kernel based sampling. In *International Conference on Machine Learning*. PMLR, 590–599.

[7] Mingyue Cheng, Runlong Yu, Qi Liu, Vincent W Zheng, Hongke Zhao, Hefu Zhang, and Enhong Chen. 2019. Alpha-Beta Sampling for Pairwise Ranking in One-Class Collaborative Filtering. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1000–1005.

[8] Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 1. IEEE, 539–546.

[9] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[10] Tommaso Furlanello, Zachary C Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born again neural networks. *arXiv preprint arXiv:1805.04770* (2018).

[11] Xin Geng. 2016. Label distribution learning. *IEEE Transactions on Knowledge and Data Engineering* 28, 7 (2016), 1734–1748.

[12] Marco Gori, Augusto Pucci, V Roma, and I Siena. 2007. Itemrank: A random-walk based scoring algorithm for recommender engines.. In *IJCAI*, Vol. 7. 2766–2771.

[13] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR*. 355–364.

[14] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 843–852.

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

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[17] Folasade Olubusola Isinkaye, YO Folajimi, and Bolande Adefowoke Ojokoh. 2015. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal* 16, 3 (2015), 261–273.

[18] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007* (2014).

[19] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* (2019).

[20] Rafal Jozefowicz, Oriol Vinyals, and Schuster et al. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410* (2016).

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

[22] Weiwei et al Kong. 2020. Rankmax: An Adaptive Projection Alternative to the Softmax Function. *Advances in Neural Information Processing Systems* 33 (2020).

[23] Walid Krichene and Steffen Rendle. 2020. On sampled metrics for item recommendation. In *ACM SIGKDD*. 1748–1757.

[24] 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.

[25] Zhi Li, Hongke Zhao, Qi Liu, Zhenya Huang, Tao Mei, and Enhong Chen. 2018. Learning from history and present: Next-item recommendation via discriminatively exploiting user behaviors. In *ACM SIGKDD*. 1734–1743.

[26] Defu Lian, Qi Liu, and Enhong Chen. 2020. Personalized Ranking with Importance Sampling. In *Proceedings of The Web Conference 2020*. 1093–1103.

[27] Qi Liu, Enhong Chen, Hui Xiong, Chris HQ Ding, and Jian Chen. 2011. Enhancing collaborative filtering by user interest expansion via personalized ranking. *IEEE TSMC, Part B (Cybernetics)* 42, 1 (2011), 218–233.

[28] Qi Liu, Yong Ge, Zhongmou Li, Enhong Chen, and Hui Xiong. 2011. Personalized travel package recommendation. In *2011 IEEE ICDM*. IEEE, 407–416.

[29] Qi Liu, Zhenya Huang, Yu Yin, Enhong Chen, Hui Xiong, Yu Su, and Guoping Hu. 2019. Ekt: Exercise-aware knowledge tracing for student performance prediction. *IEEE Transactions on Knowledge and Data Engineering* 33, 1 (2019), 100–115.

[30] Takeru et al Miyato. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence* 41, 8 (2018), 1979–1993.

[31] Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model.. In *Aistats*, Vol. 5. Citeseer, 246–252.

[32] Mohammad Norouzi, David J Fleet, and Russ R Salakhutdinov. 2012. Hamming distance metric learning. In *NIPS*. 1061–1069.

[33] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 502–511.

[34] Weike Pan and Li Chen. 2013. Gbpr: Group preference based bayesian ranking for one-class collaborative filtering. In *IJCAI*.

[35] Gabriel et al Pereyra. 2017. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548* (2017).

[36] Steffen Rendle. 2021. Item Recommendation from Implicit Feedback. *arXiv preprint arXiv:2101.08769* (2021).

[37] Steffen Rendle and Freudenthaler et al. 2009. BPR: Bayesian personalized ranking from implicit feedback. *UAI* (2009).

[38] F. Ricci, L. Rokach, and Bracha Shapira. 2011. Introduction to Recommender Systems Handbook. In *Recommender Systems Handbook*.

[39] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *ACM CIKM*. 1441–1450.

[40] Yang Sun, Fajie Yuan, Min Yang, Guoao Wei, Zhou Zhao, and Duo Liu. 2020. A Generic Network Compression Framework for Sequential Recommender Systems. In *Proceedings of the 43rd International ACM SIGIR*. 1299–1308.

[41] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE CVPR*. 2818–2826.

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

[43] Ugo Tanielian and Flavian Vasile. 2019. Relaxed softmax for PU learning. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 119–127.

[44] Md Mehrab Tanjim, Congzhe Su, Ethan Benjamin, Diane Hu, Liangjie Hong, and Julian McAuley. 2020. Attentive Sequential Models of Latent Intent for Next Item Recommendation. In *WWW*. 2528–2534.

[45] Hongwei et al Wang. 2018. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *ACM CIKM*. 417–426.

[46] Jiachun Wang and Fajie et al Yuan. 2020. StackRec: Efficient Training of Very Deep Sequential Recommender Models by Layer Stacking. *arXiv preprint arXiv:2012.07598* (2020).

[47] Le Wu, Xiangnan He, Xiang Wang, Kun Zhang, and Meng Wang. 2021. A Survey on Neural Recommendation: From Collaborative Filtering to Content and Context Enriched Recommendation. *arXiv preprint arXiv:2104.13030* (2021).

[48] Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Zi Huang. 2014. A temporal context-aware model for user behavior modeling in social media systems. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 1543–1554.

[49] Runlong Yu, Qi Liu, Yuyang Ye, Mingyue Cheng, Enhong Chen, and Jianhui Ma. 2020. Collaborative List-and-Pairwise Filtering from Implicit Feedback. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[50] Runlong Yu, Yunzhou Zhang, Yuyang Ye, Le Wu, Chao Wang, Qi Liu, and Enhong Chen. 2018. Multiple pairwise ranking with implicit feedback. In *Proceedings of the 27th ACM CIKM*. 1727–1730.

[51] Fajie Yuan, Xin Xin, Xiangnan He, Guibing Guo, Weinan Zhang, Chua Tat-Seng, and Joemon M Jose. 2018. fbgd: Learning embeddings from positive unlabeled data with bgd. (2018).

[52] Fajie et al Yuan. 2016. Lambdafm: learning optimal ranking with factorization machines using lambda surrogates. In *ACM CIKM*. 227–236.

[53] Fajie et al Yuan. 2019. A simple convolutional generative network for next item recommendation. In *ACM WSDM*. 582–590.

[54] Fajie et al Yuan. 2020. Parameter-Efficient Transfer from Sequential Behaviors for User Modeling and Recommendation. In *ACM SIGIR*. 1469–1478.

[55] Li Yuan, Francis EH Tay, and et al Li. 2019. Revisit knowledge distillation: a teacher-free framework. *arXiv preprint arXiv:1909.11723* (2019).

[56] Sukmin Yun, Jongjin Park, Kimin Lee, and Jinwoo Shin. 2020. Regularizing class-wise predictions via self-knowledge distillation. In *CVPR*. 13876–13885.

[57] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. 2018. Deep mutual learning. In *IEEE CVPR*. 4320–4328.

[58] Wayne Xin Zhao, Junhua Chen, Pengfei Wang, Qi Gu, and Ji-Rong Wen. 2020. Revisiting Alternative Experimental Settings for Evaluating Top-N Item Recommendation Algorithms. In *Proceedings of the 29th ACM CIKM*. 2329–2332.

[59] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with negative feedback via pairwise deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD*. 1040–1048.