Differentiable Ranking Metric Using Relaxed Sorting for Top-K Recommendation

HYUNSEUNG LEE\textsuperscript{1}, SANGWOO CHO\textsuperscript{2}, YEONGJAE JANG\textsuperscript{2}, JAEKWANG KIM\textsuperscript{3}, AND HONGUK WOO\textsuperscript{4}, (Member, IEEE)

1Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon 16447, South Korea
2Department of Mathematics, Sungkyunkwan University, Suwon 16419, South Korea
3School of Convergence, Sungkyunkwan University, Suwon 03063, South Korea
4Department of Computer Science and Engineering, Sungkyunkwan University, Suwon 16419, South Korea

Corresponding author: Honguk Woo (hwoo@skku.edu)

This work was supported in part by the Institute for Information and Communications Technology Planning and Evaluation (IITP) (Adaptive Federated Learning in Dynamic Heterogeneous Environment) under Grant 2021-0-00900, in part by the ICT Creative Consilience Program supervised by the IITP under Grant IITP-2020-0-01821, in part by the National Research Foundation of Korea (NRF) funded by the Korean Government (MSIT) under Grant 2019R1C1C1008174, and in part by Samsung Electronics.

ABSTRACT Most recommenders generate recommendations for a user by computing the preference score of items, sorting the items according to the score, and filtering top-\(K\)-items of high scores. Since sorting is not differentiable and is difficult to optimize with gradient descent, it is nontrivial to incorporate it in recommendation model training despite its relevance to top-\(K\) recommendations. As a result, inconsistency occurs between existing learning objectives and ranking metrics of recommenders. In this work, we present the Differentiable Ranking Metric (DRM) that mitigates the inconsistency between model training and generating top-\(K\) recommendations, aiming at improving recommendation performance by employing the differentiable relaxation of ranking metrics via joint learning. Using experiments with several real-world datasets, we demonstrate that the joint learning of the DRM objective and existing factor based recommenders significantly improves the quality of recommendations.

INDEX TERMS Recommender systems, learning to rank, top-K recommendation, differentiable ranking metric.

I. INTRODUCTION

To make top-\(K\) item recommendations, most recommender systems generate the relevance score of items with respect to a user and filter the top-\(K\) items of high scores. Thus, ranking, or equivalently, sorting items serves an important role in top-\(K\) recommendations. However, conventional model training procedures for recommenders have a limitation to accurately reflect ranking natures of top-\(K\) recommendations because the sorting operation is not differentiable, and incorporating it into end-to-end model training is challenging. Instead, most model based recommenders usually exploit surrogate objectives such as mean squared error or log-likelihood, which do not take into account ranking metrics. As has been noted in several works [1]–[3], optimizing such objectives that do not consider ranking natures of top-\(K\) recommendations does not always guarantee the best performance.

There are several ranking-oriented objectives, including pairwise and listwise objectives; however, neither objectives are known to fit well with top-\(K\) recommendations. Pairwise objectives [4]–[7] enable a recommender to learn users’ preferences by casting a recommendation task as a binary classification, predicting whether a user prefers an item to another item. For top-\(K\) recommendations, classifying top-\(K\) ranked items is important, but the pairwise objectives do not emphasize the relative difference between top-\(K\) ranked items and non-top-\(K\) ranked items. While several listwise objectives [3], [8], [9] have been recently proposed to model ranking natures of recommendations, adopting listwise objectives is limited in practice especially in large-scale recommenders due to their high computational complexity caused by modeling permutations with Plackett-Luce distribution [10].

In this work, we address such limitations in existing ranking-oriented objectives for model based recommenders, aiming to bridge the gap between the existing learning objectives commonly used for training recommenders and the ranking nature of top-\(K\) recommendation tasks. Inspired
by differential programming [11], [12], we present the
differentiable relaxation scheme of ranking metrics such as Prec@K
or Recall@K. By employing the differentiable relaxation
scheme for sorting operation [11], DRM expedites direct
optimization of ranking metrics for recommendation models.
To do so, we first reformulate the ranking metrics in terms
of permutation matrix arithmetic forms and then relax the
nondifferentiable permutation matrix in the arithmetic forms
to a differentiable row-stochastic matrix. This reformulation
and relaxation allows us to optimize ranking metrics in a
differentiable form of DRM. Using DRM as an optimization
objective renders end-to-end recommendation model training
highly consistent with ranking metrics.

Moreover, DRM can be readily incorporated into existing
recommenders via joint learning with their objectives without
modifying their model structure, thus preserving their benef-
fits. Specifically, we adopt two state-of-the-art model based
recommenders, WARP [5], [13] and CML [6], wherein the
joint learning can be used.

Our experiments demonstrate that the DRM objective
significantly improves the performance of top-K recommenda-
tions on several real-world datasets in terms of ranking
metrics, compared with several other recommendation models.

Furthermore, we demonstrate how to adapt other rank-
ing metric driven objective (RMDO) schemes, e.g., Approx-
NDCG, for the model based recommenders via joint learning.
While RMDO schemes have been studied by the information
retrieval community [7], [8], [14], [15], the joint learning
approach by those has not been fully investigated for factor
based recommenders. To the best of our knowledge, our work
is the first to incorporate joint learning with relaxed ranking
metrics to factor based recommenders.

The contribution of this work is summarized as follow:

• We propose the DRM objective that alleviates the mis-

alignment between the training objective and evaluation
metrics in top-K recommendation models,

• We present a joint learning approach to train factor based
recommenders using the DRM objective.

• Through extensive experiments, we empirically show that
(1) our approach outperforms other state-of-the art

top-K recommendation models and (2) the DRM object-

ive fits better than other ranking metric driven objectives
for top-K recommendation tasks.

II. PRELIMINARIES

Given a set of \( M \) users \( U = \{1, 2, \ldots, M\} \), set of \( N \) items \( V = \{1, 2, \ldots, N\} \), and a set of interactions \( y_{u,i} \) for all users \( u \) in
\( U \) and all items \( i \) in \( V \), a recommendation model is learned to predict
preference or score \( \hat{y}_{u,i} \in \mathbb{R} \) of user \( u \) to item \( i \). We use
predicted preference and predicted score interchangeably to
denote \( \hat{y}_{u,i} \). We use binary implicit feedback \( y_{u,i} \) such that
\( y_{u,i} = 1 \) if user \( u \) has interacted with item \( i \), and 0 otherwise.
Note that we only consider this binary feedback format in this
work, while the approach can be generalized for other implicit
feedback settings.

For user \( u \), we use \( i \) to represent positive items that \( u \)
has interacted with, and \( j \) to represent negative items that \( u \)
has not. In addition, we use \( Y_u \) to represent a set of posi-
tive items for user \( u \), and use \( y_u \) for its bag-of-words nota-
tion, i.e., a column vector \([y_{u,1}, y_{u,2}, \ldots, y_{u,n}]^T\). Similarly,
we use \( \hat{y}_u \) to represent the vector of predicted scores of items,
i.e., \([\hat{y}_{u,1}, \hat{y}_{u,2}, \ldots, \hat{y}_{u,n}]^T\).

A. OBJECTIVES FOR RECOMMENDATION MODELS

In general, the objectives of recommenders are categorized
into pointwise, pairwise, and listwise.

Pointwise objectives maximize the prediction accuracy
independently from the errors of item rankings. While several
pointwise objectives, such as mean squared error and cross-
entropy, are commonly used, those objectives are known to
have limitations in that small errors of the objectives do not
always lead to high-quality recommendations [16].

In the recommender system domain, pairwise objectives
have gained popularity because they are more closely related
to top-K recommendations than pointwise objectives. Model
training with pairwise objectives enables a recommender to
learn users’ preferences by casting the recommendation task
as a binary classification, predicting whether user \( u \) prefers
item \( i \) to item \( j \). For example, Bayesian personalized rank-
ing [4] minimizes negative log-likelihood of the probability
that user \( u \) prefers item \( i \) to item \( j \)

\[
L_{BPR} = - \sum_{u \in U} \sum_{i \in V_u} \sum_{j \in V - V_u} \log \sigma(\hat{y}_{u,i} - \hat{y}_{u,j})
\]  

(1)

where \( \sigma(\cdot) \) is a sigmoid function, and \( \sigma(\hat{y}_{u,i} - \hat{y}_{u,j}) \) is inter-

preted as the probability user \( u \) prefers item \( i \) to item \( j \).

Another popular pairwise objective is weighted hinge loss,

\[
L_{hinge} = \sum_{u \in U} \sum_{i \in V_u} \sum_{j \in V - V_u} \Phi_{ul}[\mu - \hat{y}_{u,i} + \hat{y}_{u,j}]_+
\]  

(2)

where \([x]_+ = \max(0, x) \), and \( \mu \) is a margin value. The
weight \( \Phi_{ul} \) introduced in [5], [13] enables pairwise objectives

to emphasize the loss of positive items at lower ranks; the
value of \( \Phi_{ul} \) is chosen to be larger if the approximated rank is
lower for a positive item \( i \). Some choices of \( \Phi_{ul} \) are known to
make the optimizing hinge loss closely related to maximizing
discounted cumulative gain [17], [18].

For top-K recommendations, listwise objectives have been
recently explored by a few research works [3], [8], [19].
In general, listwise objectives are based on Plackett-Luce
probability distribution of list permutations, i.e.,

\[
\pi(\hat{y}) = \prod_{j=1}^{\min(K,N)} \frac{\phi(\hat{y}_{\pi_j})}{\sum_{j'=1}^{N} \phi(\hat{y}_{\pi_{j'}})}
\]  

(3)

where \( \phi(\cdot) \) is an arbitrary smoothing function, e.g., \( \phi(\cdot) = \exp(\cdot) \). These listwise objectives aim to maximize the
probability of correctly ordered permutations by minimizing the
log-likelihood or cross-entropy. However, they have a limitation of high computational complexity in that the complexity of calculating the permutation probability grows exponentially as the number of items in the dataset increases.

**B. RANKING METRICS FOR TOP-K RECOMMENDATIONS**

Here, we introduce common evaluation metrics for top-K recommendations. To explain ranking metrics, we represent the list of items sorted by the predicted scores for user \( u \) as \( \pi_u \), and the item at rank \( k \) as \( \pi_u(k) \). In addition, we define the Hit\((u, k)\) function that specifies whether \( \pi_u(k) \) is in the validation dataset \( \mathcal{V}_u \). Specifically,

\[
\text{Hit}(u, k) = \mathbb{I}[\pi_u(k) \in \mathcal{V}_u]
\]  

(4)

where \( \mathbb{I}[\text{statement}] \) is the indicator function, yielding 1 if the statement is true and 0 otherwise.

- \( K \)-Truncated Precision (Prec\(@K\)) and Recall (Recall\(@K\)) are two of the most widely used evaluation metrics for top-K recommendations. Prec\(@K\) specifies the fraction of hit items in \( \mathcal{V}_u \) among recommended items. Recall\(@K\) specifies the fraction of recommended items among the items in \( \mathcal{V}_u \). Notice that both metrics do not take into account the difference in the ranking of recommended items, while they collectively emphasize top-K-ranked items by counting only the items up to \( K \)-th ranks.

\[
\text{Prec}@K(u, \pi_u) = \frac{1}{K} \sum_{k=1}^{K} \text{Hit}(u, k)
\]

(5)

\[
\text{Recall}@K(u, \pi_u) = \frac{1}{|\mathcal{V}_u|} \sum_{k=1}^{K} \text{Hit}(u, k)
\]

(6)

On the other hand, \( K \)-Truncated Normalized Discounted Cumulative Gain (NDCG\(@K\)) and Average Precision (AP\(@K\)) take into account the relative ranks of items by weighting the impact of Hit\((u, k)\). NDCG\(@K\) specifies a normalized value of DCG\(@K\), which is divided by the ideal DCG IDC@K = \( \max_{\pi_u} \text{DCG}@K(u, \pi_u) \).

\[
\text{NDCG}@K(u, \pi_u) = \frac{\text{DCG}@K(u, \pi_u)}{\text{IDCG}@K}
\]

where

\[
\text{DCG}@K(u, \pi_u) = \sum_{k=1}^{K} \frac{\text{Hit}(u, k)}{\log_2(k + 1)}
\]

(7)

The \( k \)-truncated AP is defined as

\[
\text{AP}@K(u, \pi_u) = \sum_{k=1}^{K} \frac{\text{Prec}@K(u, \pi_u)\text{Hit}(u, k)}{\min(K, |\mathcal{V}_u|)}
\]

(8)

where AP can be viewed as a weighted sum of Hit for each rank \( k = 1, 2, \ldots, K \), weighted by Prec@k.

We can represent the aforementioned metrics in a unified way as \( \mathcal{O}(K) \) conditioned on weight functions \( w(k, K) \).

\[
\mathcal{O}(K) = \sum_{k=1}^{K} w(k, K) \text{Hit}(u, k)
\]

III. PROPOSED METHOD

In this section, we briefly review a factor based recommender with weighted hinge loss and a ranking metric based objective in terms of vector arithmetic as well as how to relax the aforementioned ranking metrics to be differentiable, which can be optimized using gradient descent for training factor based recommenders. We call this relaxed approach DRM. We then describe our joint learning approach to train factor based recommenders using DRM. We also show that this learning approach can be used to incorporate other ranking metric driven objectives in training factor based recommenders.

**A. FACTOR BASED RECOMMENDERS WITH HINGE LOSS**

Factor based recommenders represent users and items in a latent vector space \( \mathbb{R}^d \), and then formulate the preference score of user \( u \) to item \( i \) as a function of two vectors, user vector \( \alpha_u \) and item vector \( \beta_i \). Dot product is a common method for mapping a pair of user and item vectors to a predicted preference \([4], [20], [21]\). In \([6]\), the collaborative metric learning (CML) embeds users and items in the Euclidean space and defines its score function using the negative value of L2 distance of two factors.

Our model uses either the dot product or L2 distance of user vector \( \alpha_u \) and item vector \( \beta_i \) as a score function.

\[
\hat{y}_{ui} = \begin{cases} 
\alpha_u^T \beta_i & \text{Dot product} \\
-\|\alpha_u - \beta_i\|^2 & \text{L2 distance} 
\end{cases}
\]

(10)

Note that \( \| \cdot \| \) is the L2 norm. Having the score functions above, we update our model using weighted hinge loss in Equation (2). Following \([6]\), we calculate the weight \( \Phi_{ui} \) of hinge loss by

\[
\Phi_{ui} = \log \left(1 + \frac{N}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} [1 - \hat{y}_{ui,j} + \hat{y}_{uj,j} \geq 0]\right).
\]

(11)

For each update, \( |\mathcal{J}| \) negative items are sampled from \( \mathcal{V} - \mathcal{V}_u \) and used to estimate \( \Phi_{ui} \).

**B. DIFFERENTIABLE RANKING METRICS**

Sorting and ranking items can be considered as permutation of items. An \( N \)-dimension permutation corresponds to a vector \( p = [p_1, p_2, \ldots, p_N]^T \) where \( p_i \in \{1, 2, \ldots, N\} \) and \( p_i \neq p_j \) if \( i \neq j \). For each vector \( p \), we then have its permutation matrix \( \mathcal{P} \in [0, 1]^{N \times N} \) and its element can be described as

\[
P_{ij} = \begin{cases} 
1 & \text{if } j = p_i \\
0 & \text{otherwise}
\end{cases}
\]

(12)
For example, a permutation matrix \( P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \) maps a score vector \( v = [3, 5, 1]^T \) to \( P v = [5, 3, 1]^T \).

In [11], they propose continuous relaxation of sorting (namely NeuralSort), which represents a sorting operation in a permutation matrix, and then relaxes the matrix into a continuous form. Specifically, sorting a vector \( s = [s_1, s_2, \ldots, s_N]^T \) in descending order can be represented in a permutation matrix such as

\[
P^{(s)}_{i,j} = \begin{cases} 1 & \text{if } j = \text{argmax} [(n + 1 - 2i)s - A_i1] \\ 0 & \text{otherwise} \end{cases}
\]

where \( 1 \) is the column vector having 1 for all elements and \( A_i \) is the matrix such that \( A_{i,j} = |s_j - s_i| \). Then, softmax is used instead for relaxing the permutation matrix, i.e.,

\[
\tilde{P}^{(s)}_k = \text{softmax} \left( \tau^{-1} ((n + 1 - 2k)s - A_i1) \right)
\]

where \( \tau > 0 \) is a temperature parameter. Larger \( \tau \) values make each row of the relaxed matrix flatter. This transformation of NeuralSort renders the permutation matrix in Equation (13) relaxed into a unimodal row-stochastic matrix, realizing the differentiation operation for sorting of real-value elements. Equation (14) is continuous everywhere and differentiable nearly everywhere with respect to the elements of \( s \). Furthermore, as \( \tau \to 0^+ \), \( \tilde{P}^{(s)} \) reduces to the permutation matrix \( P^{(s)} \).

The \( k \)-th row \( P^{(s)}_k \) of the permutation matrix \( P^{(s)} \) is equal to the one-hot vector representation of the \( k \)-th ranked item. Thus, we can reformulate Hit (Equation (4)) using the dot product of \( y_u \) and \( P^{(s)}_k \).

\[
\text{Hit}(u, k) = y_u^T P^{(s)}_k
\]

We then obtain the representation of ranking metrics in Equation (9) in terms of vector arithmetic using Equation (15).

\[
O(K, w) = \sum_{k=1}^{K} w(k, K) y_u^T P^{(s)}_k
\]

By replacing \( P^{(s)}_k \) in Equation (16) with differentiable \( \tilde{P}^{(s)}_k \), we obtain the differentiable relaxed objective, which can be used for optimization using gradient descent.

\[
\tilde{O}(K, w) = \sum_{k=1}^{K} w(k, K) y_u^T \tilde{P}^{(s)}_k
\]

We empirically find that the below equation, namely DRM (objective), is more stable in model training.

\[
\mathcal{L}_{\text{DRM}} = \sum_{u \in \mathcal{U}} \| y_u - \sum_{k=1}^{K} w(k, K) \tilde{P}^{(s)}_k \|_2^2
\]

Note that minimizing Equation (18) is equivalent to maximizing Equation (17) because

\[
\tilde{O} = y_u^T a_u = \frac{1}{2} \left[ (y_u^T y_u + a_u^T a_u) - \| y_u - a_u \|_2^2 \right]
\]

\[
\geq - \frac{1}{2} \| y_u - a_u \|_2^2
\]

where \( a_u = \sum_{k=1}^{K} w(k, K) \tilde{P}^{(s)}_k \) for each user \( u \).

**C. JOINT LEARNING WITH DRM**

We incorporate the proposed objective in Equation (18) into the model learning structure via joint learning of two objectives,

\[
\mathcal{L} = \mathcal{L}_{\text{model}} + \lambda \mathcal{L}_{\text{DRM}} + \Omega_{\text{model}}
\]

\[
= \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{V}_u \cup \mathcal{V}_u^{\neg}} \Phi_u[iu - \hat{y}_{u,i} + \bar{y}_{u,i}] + \lambda \sum_{u \in \mathcal{U}} \| y_u - a_u \|_2^2 + \Omega_{\text{model}}
\]

where \( \lambda \) is a scaling parameter for controlling the effect of \( \mathcal{L}_{\text{DRM}} \) and \( \Omega_{\text{model}} \) is a regularization term of model parameters. \( \mathcal{L}_{\text{model}} \) depends on the chosen base model. Specifically, we adopt two pairwise, hinge loss, factor based models, WARP and CML. Therefore, \( \mathcal{L}_{\text{model}} \) corresponds to a hinge loss \( \mathcal{L}_{\text{hinge}} \).

Two regularization schemes are adapted. We first keep latent factors of users and items within the unit hypersphere, i.e., \( \| \theta \| \leq 1 \) where \( \theta \) is either \( \alpha \) or \( \beta \). For the model exploiting negative L2 distance as a score function, we also adopt covariance regularization [22] between all pairs of latent factors using a matrix \( C_{ij} = 1 - \sum_{\theta \in \Theta} (\theta_i - \mu_i)(\theta_j - \mu_j)^T \) where \( \Theta = \{\alpha_1, \alpha_2, \ldots, \alpha_M, \beta_1, \beta_2, \ldots, \beta_N\} \) and \( \mu \) is the average vector of all user and item factors. We define a regularization term \( \mathcal{L}_C \) as \( \mathcal{L}_C = \| C \|_F - \| \text{diag}(C) \|_F \) where \( \| \cdot \|_F \) is the Frobenius norm. In the case that L2 distance is used, we add this regularization to Equation (20) with a control parameter \( \lambda_C \), i.e., \( \Omega_{\text{model}} = \lambda_C C_{\text{C}} \). The learning procedure for DRM is summarized in Algorithm 1.

Similar to negative sampling [4], [6], [23], [24] used in factor based models using gradient descent updates, we follow a sampling procedure for positive items. We construct each training sample to contain a user \( u \), a set of positive items \( \mathcal{I} \) of size \( \rho \) and a set of negative items \( \mathcal{J} \) of size \( \eta \). Specifically, we construct a list of items \( y_u \) of size \( \rho + \eta \) where first \( \rho \) elements are from \( \mathcal{I} \), and next \( \eta \) elements are from \( \mathcal{J} \). Using Equation (10), the list of predicted scores \( \hat{y}_u \) is constructed similarly.

\[
y_u = [y_{u,1}, y_{u,2}, \ldots, y_{u,\rho}, y_{u,\rho+1}, \ldots, y_{u,\rho+\eta}]^T
\]

\[
\hat{y}_u = [\hat{y}_{u,1}, \hat{y}_{u,2}, \ldots, \hat{y}_{u,\rho}, \hat{y}_{u,\rho+1}, \ldots, \hat{y}_{u,\rho+\eta}]^T
\]

The DRM objective has quadratic complexity to the number of items to build a relaxed permutation matrix in Equation (14) for a user. Model training can become too time consuming if all items in \( \mathcal{V} \) are used in the objective in Equation (20). Instead, we sample a fixed number of positive and negative items for each user. Negative sampling draws \( n_{\text{neg}} \) items along with one positive item, as normally employed for learning recommendation models [4], [6], [23], [24]. Similarly, we sample \( n_{\text{pos}} \) positive items and \( n_{\text{neg}} \) items.
to construct a list of items containing both positive and negative items. In this work, we set $n_{\text{neg}} = 17$ and $n_{\text{pos}} = 3$.

**D. JOINT LEARNING WITH OTHER OBJECTIVES**

Optimizing ranking metrics is considered an important problem in information retrieval [25], and a number of ranking metric driven objectives (RMDOs) have been proposed [7], [8], [14], [15]. ApproxNDCG and LambdaLoss employ the Bradley-Terry pairwise rank probability to have the differentiable proxy for ranks in NDCG. ListNet uses Plackett-Luce [10] distribution to represent distributions of ranks in NDCG. These prior works commonly view item ranks as nondifferentiable variables and relax them using probability distributions, while DRM treats item ranks as matrices and relaxes the matrices to be differentiable.

Besides, incorporating RMDOs into existing recommendation models without modifying the model structure has not gained much attention. We show that the model update strategy in Section III-C can be used for that purpose. For example, ApproxNDCG [7] relaxes the rank of an item $i$ (rank($i$)) to be continuous and differentiable, i.e.,

$$\text{rank}(i) = 1 + \sum_{j \in \mathcal{V} \setminus \{i\}} \sigma(\tau^{-1}(\hat{y}_i - \hat{y}_j))$$  \hspace{1cm} (22)

where $\sigma(\cdot)$ is a sigmoid function. Note that $\tau$ is a temperature parameter as in Equation (14). Relaxation of ranking metrics is realized by plugging the relaxed rank, rank($i$), into ranking metrics (Equation (9)). In the same vein of the DRM-driven joint learning in Equation (20), it is possible to use aggregated relaxed NDCG over users as training objectives in joint learning.

$$\mathcal{L} = \mathcal{L}_{\text{model}} + \lambda \mathcal{L}_{\text{ApproxNDCG}} + \Omega_{\text{model}}$$  \hspace{1cm} (23)

Similarly, other RMDOs with gradient update can be applied for the joint learning scheme.

**IV. EMPIRICAL EVALUATION**

In this section, we evaluate our DRM-driven joint learning scheme in comparison with state-of-the-art recommendation models and other joint learning objectives.

**A. EXPERIMENTAL SETUP**

We validate our approach and baselines on four real-world user-item interaction datasets.

- **SketchFab** contains users’ likes on 3D models. We treat each like as a user-item interaction.
- **Epinion** [26] contains 5-star rating reviews of customers on products from a web service. We view each rating as a user-item interaction.
- **ML-20M** [27] contains the movie ratings from a movie recommendation service. Rating ranges from 0.5 to 5.0 in 0.5 increments. We treat ratings as positive interactions and exclude the ratings less than four.
- **Melon** contains playlists from a music streaming service. To be consistent with the implicit feedback setting, we treat each playlist as a user, and songs in a playlist as a list of positive items of the user.

The dataset statistics are summarized in Table 1.

**TABLE 1. Dataset statistics:** # users, # items, and # interactions denote the number of users, items, and interactions, respectively; avg. row and avg. col denote the average number of items that each user has interacted with, and the average number of users who have interacted with each item, respectively. Density denotes the interaction matrix density (i.e., Density = # interactions / (# users × # items)).

| Dataset | # users | # items | # interactions | Avg. row | Avg. col | Density |
|---------|---------|---------|----------------|----------|----------|---------|
| SketchFab | 135K    | 21.3K   | 136K           | 70.0K    | 31.9K    | 0.12%   |
| Epinion  | 28.8K   | 59.2K   | 15.5K          | 32.9K    | 31.9K    | 0.04%   |
| ML-20M   | 558K    | 631K    | 9.98M          | 4.10M    | 39.2     | 0.47%   |
| Melon    | 35.9    | 29.4    | 73.0           | 39.2     | 50.5     | 0.04%   |

**1) EVALUATION PROTOCOL**

We split the dataset into training, validation, and test datasets in 70%, 10%, and 20% portions, respectively. We train each model using the training dataset to find the best hyperparameter settings, evaluating with the validation dataset. We use Recall@50 for model validation. We then train the model five times with the best hyperparameter settings using the training and validation datasets, evaluate the model using the test dataset, and report the average performance in evaluation metrics for all experiments. We use mean AP@10
(MAP@10), NDCG@10, Recall@50, and NDCG@50 averaged for all users with fewer than five interactions in the training dataset. We also conduct Welch’s T-test [28] on our results and denote the best results with p-value lower than 0.01 in boldface.

2) BASELINE MODELS

We compare DRM with the following baselines. Note that except for SLIM, all baselines fall in the category of factor based recommenders.

- **SLIM** [29] is an item-item collaborative filtering algorithm in which the item-item similarity matrix is represented as a sparse matrix. It poses L1 and L2 regularization on the item-item similarity matrix.
- **CDAE** [30] can be seen as a factor based recommender where user factors are generated via an encoder. The encoder takes embeddings of the items the user has consumed and the embedding of the user as input and returns the latent factor of the user. We use a neural network with no hidden layers as described in the original paper implementation.
- **WMF** [21] uses the mean squared error as the objective function and minimizes it using alternating least squares.
- **BPR** [4] exploits the pairwise sigmoid objective.
- **SQLRank** [9] views a recommendation problem as approximating a value function of log-likelihood of probabilities of correctly sorted lists.
- **SRRMF** [31] is a factor based recommender that interpolates unobserved feedback scores to be nonzero to treat negative items differently.
- **WARP** [5], [13] is trained using hinge loss with approximated rank based weights.
- **CML** [6] models user-item preferences as a negative value of the distance between user and item vectors.

3) HYPERPARAMETER SETTINGS

We tune hyperparameters of the baselines and DRM using grid-search. For SLIM, we set the search space to be \{0.1, 0.5, 1.0, 3.0, 5.0, 10.0, 20.0\} for both L1 regularization and L2 regularization. All other baselines are factor based recommenders, thus we set the search space of dimension of latent factors to be \{16, 32, 64, 128\}. For CDAE, BPR, WMF, and SRRMF which exploit L2 regularization, we set the search space of the learning rate to be \{10^{-4}, 0.001, 0.003, 0.01, 0.03, 0.1\}. We set the confidence \(c\) in the WMF and SRRMF, we used search space of \{1.0, 3.0, 5.0, 10.0, 20.0\}. We used the size of negative sampling for WARP, CML, and WARP to be \{10, 20, 30, 40, 50\}. We use margin in the WARP, CML, and DRM variants to be 1.0, temperature \(\tau\) and scaling parameter \(\lambda\) of DRM parameters are determined through searching in \{0.1, 0.3, 0.5, 1.0, 3.0, 10.0\} and \{0.1, 0.5, 1.0, 2.0, 3.0, 5.0, 10.0, 30.0\}, respectively.

We use Adagrad optimizer for CDAE, WARP, CML, and use stochastic gradient descent for BPR as mentioned in the original papers. We use Adagrad optimizer for training DRM_{dot} and DRM_{L2}.

We use open source implementations of models and objectives: ApproxNDCG, LambdaLoss, and ListNet [8], WMF and BPR, SLIM, SQLRank and SRRMF. We implement CDAE, CML, WARP and DRM using Python 3.7.3 and PyTorch 1.5.0. We run our experiment on a machine with an Intel(R) Xeon(R) CPU E5-2698, 160G memory, and an NVIDIA Tesla V100 GPU with CUDA 10.1.

### B. OVERALL EVALUATION

Table 2 shows the recommendation performance of the baseline and DRM variants (i.e., DRM_{dot}, DRM_{L2}) on different...
TABLE 3. The effects of regularization term $\lambda$ on recommendation performance: The best results with $p \leq 0.01$ on the paired T-test are boldfaced.

|       | DRM$_{dot}$ |   | 0    | 0.1  | 1.0  | 3.0  | 5.0  | 10.0 | 30.0 | $\infty$ |
|-------|-------------|---|------|------|------|------|------|------|------|-------|
|       | Recall@50   | NDCG@50 |      |      |      |      |      |      |      |       |
| ML-20M| 0.4450      | 0.3584 | 0.4091| 0.4898| 0.4941| 0.4994| 0.4912| 0.4873|      |       |
| Melon | 0.2964      | 0.3088 | 0.3166| 0.3272| 0.3322| 0.3433| 0.3471| 0.3270|      |       |

|       | DRM$_{L2}$  |   | 0    | 0.1  | 1.0  | 3.0  | 5.0  | 10.0 | 30.0 | $\infty$ |
|-------|-------------|---|------|------|------|------|------|------|------|-------|
|       | Recall@50   | NDCG@50 |      |      |      |      |      |      |      |       |
| ML-20M| 0.4874      | 0.4902 | 0.4949| 0.4951| 0.4938| 0.4952| 0.4881| 0.4582|      |       |
| Melon | 0.3361      | 0.3540 | 0.3700| 0.3554| 0.3531| 0.3531| 0.3465| 0.2920|      |       |

datasets. We record the performance of SQLRank only upon SketchFab and Epinion datasets, because it is hardly possible for us to train SQLRank successfully with large datasets, ML-20M and Melon, due to the huge training time.

When comparing models with the same structure, which use the same score function or share the objectives, we observe that DRM consistently outperforms the respective models with the same structure. For example, WARP and DRM$_{dot}$ use the same score function and share hinge loss, but DRM$_{dot}$ consistently outperforms WARP for all datasets. This pattern recurs for CML and DRM$_{L2}$, whose score function is L2 distance and shares hinge loss. This clarifies that the proposed DRM objective leads factor based models to make better recommendations by exploiting the top-$K$ recommendation nature.

We observe DRM$_{L2}$ significantly improves performance over the baselines by up to 15.5% on SketchFab, Epinion, and Melon datasets in most cases except for Recall@50 on SketchFab and Epinion. Additionally, we observe that the performance gain is significant in NDCG@10 and MAP@10, except for ML-20M. DRM tends to perform better than the baselines, especially compared with other models with a similar structure (e.g., DRM$_{dot}$ outperforms WMF and WARP, all exploiting dot score function).

Notably, among the models using hinge loss, the models using dot product as a score function (WARP and DRM$_{dot}$) show better performance than the models using L2 distance (CML and DRM$_{L2}$) on SketchFab and Epinion datasets, which are relatively small datasets. In contrast, the models exploiting L2 distance as a score function outperform the models using dot product on larger datasets, ML-20M and Melon.

**C. BENEFITS OF DRM OBJECTIVE**

In Table 3, we show the effect of the DRM objective in Equation (20) by adjusting the regularization term $\lambda$ that controls the intensity of the DRM objective. If $\lambda$ is set to be 0, DRM$_{L2}$ shares the same learning objective and score function with CML, and DRM$_{dot}$ becomes WARP. When $\lambda$ is $\infty$, we represent the models trained only with the DRM objective without a pairwise hinge loss.

Our DRM objective achieves significant performance enhancement, especially when the dot score function is used (e.g., the performance gain is up to 10% for Recall@50 and NDCG@50 in ML-20M dataset). It is worth noting that the joint learning does not require any additional training data or structural modification of the models. These results clarify the benefits of employing the proposed DRM objective.

Although the DRM objective is helpful for the model performance, its effect decreases when the $\lambda$ becomes too large; the performance tends to degrade since $\lambda$ becomes greater than 10 for both models. The DRM models trained without the pairwise hinge objective perform quite well compared to the other baselines; however, the best performance is achieved via joint learning using both pairwise hinge loss and listwise DRM loss.

The proposed objective gives the models a more aligned objective with the goal of model training. However, the proposed objective alone yields suboptimal performance. Our model balances between pairwise hinge objective and listwise objective for optimization, via a form of joint learning. This structure of joint learning makes our DRM scheme readily applicable for modern recommenders, requiring no modification on the model structure.

**D. ABLATION STUDY**

The training procedure of DRM requires sampling $\rho$ positive items. In Figure 1, we experiment on SketchFab and Melon datasets to see the relation between the number of positive samples $\rho$ and recommendation performance. As the number of positive items $\rho$ increases, NDCG@10 improves; however, this positive effect often decreases gradually for large $\rho$ beyond some point, e.g., $\rho = 3$ for SketchFab. We verify that sampling a fraction of positive items and negative items is effective for fitting DRM models, without requiring entire user history.

In Figure 2, we evaluate the models jointly learned with different RMDOs including our proposed DRM objective and other RMDOs such as ApproxNDCG [7], LambdaLoss [14], and ListNet [8] on ML-20M dataset, as described in Section III-D. We observe that the models with such RMDOs other than the DRM objective can have performance gain.
over the respective base models, while our DRM model yields better performance than those. This observation is consistent with the performance benefit by Cofactor joint learning [32] combined with a given model objective without using additional data.

In Table 4, we also evaluate the training time of the joint learning approach. We represent the average time of 20 runs of model training on ML-20M. While the joint learning approach usually increases the training time compared to the original model (WARP or CML in the dashed column), DRM performs faster in training than the other RMDOs for joint learning. As shown in Figure 2 and Table 4, DRM demonstrates not only its performance improvement but training efficiency in joint learning compared to other RMDOs.

V. RELATED WORK

The optimization of ranking metrics is considered an important problem in the domain of information retrieval [25]. A common approach to the problem is to optimize nondifferentiable ranking metrics using differentiable surrogates, such as the upper bound of the ranking metrics. In [7], ApproxNDCG was introduced to derive the relaxation of Hit for AP and NDCG with a unified view of ranking metrics and achieve the optimization of relaxed ranking metrics. Furthermore, ApproxNDCG was adopted in deep feed-forward networks for search engine optimization in [34]. In recommender systems, we present a new ranking metric driven objective DRM that is efficiently trained and yields competitive recommendation performance.

Listwise Collaborative Filtering [3] addresses the misalignment issue between the cost and objective of K-Nearest neighbors recommenders. Specifically, this work exploits the similarity between two lists for K-Nearest recommenders. Our work is complementary since we support factor based models that are commonly used in recent user services.

SQLRANK [9] is a factor based recommender minimizing the upper bound of negative log-likelihood of the permutation probability. Despite its theoretical soundness, we were not able to achieve high-performance models in our experiments with SQLRANK. In [31], a similar pattern to our experiments is also found. Specifically, SRRMF [31] shows that merely treating missing ratings as zero values leads to suboptimal behaviors and thus it exploits respective approximated ranks to smooth negative feedbacks to nonzero values. It is conjectured that DRM achieves competitive performance since it makes use of both pairwise and listwise objectives, differently from SQLRANK that focuses on the theoretical foundation of listwise objective based model learning.
Cofactor [32] is the most similar model to ours in that a new objective was explored to train factor based models through joint learning without additional context information. Specifically, Cofactor uses word2vec-like [33], [35] embedding techniques to incorporate item co-occurrence information into a matrix factorization model. Unlike Cofactor with such word2vec based objective, our work concentrates on directly optimizing ranking metrics such as Precision through differentiable relaxation of the ranking metric itself.

VI. CONCLUSION

While learning based recommender systems are popular, their performance in terms of personalized ranking might be suboptimal because they are not directly optimized for top-K recommendation tasks. In this work, we proposed DRM that enables sorting-embedded end-to-end training, and presented the joint learning of the DRM objective with existing factor based recommenders for improving top-K recommendation performance. DRM utilizes the relaxation of sorting to continuous operation, which is suitable for a high-performance objective that can directly maximize ranking metrics. We experimentally demonstrate that our approach to several real-world datasets achieves better recommendation performance than other state-of-the-art recommender methods.

Our future work is to apply direct ranking optimization to deep learning based recommendation models such as Autoencoder for various recommendation scenarios.

REFERENCES

[1] C. J. C. Burges, R. Ragno, and Q. V. Le, “Learning to rank with nonsmooth cost functions,” in Proc. 20th Conf. Neural Inf. Process. Syst., 2006, pp. 193–200.
[2] J. Xu, T.-Y. Liu, M. Lu, H. Li, and W.-Y. Ma, “Directly optimizing evaluation measures in learning to rank,” in Proc. 31st Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. New York, NY, USA: Association for Computing Machinery, 2008, pp. 107–114.
[3] S. Huang, S. Wang, T.-Y. Liu, J. Ma, Z. Chen, and J. Veijaalanen, “Listwise collaborative filtering,” in Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., Aug. 2015, pp. 343–352.
[4] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, “BPR: Bayesian personalized ranking from implicit feedback,” in Proc. 21st Conf. Uncertainty Artif. Intell., 2009, pp. 452–461.
[5] J. Weston, S. Bengio, and N. Usunier, “WSABIE: Scaling up to large vocabulary image annotation,” in Proc. 22nd Int. Joint Conf. Artif. Intell., 2011, pp. 2764–2770.
[6] C.-K. Hsieh, L. Yang, Y. Cui, T.-Y. Lin, S. Belongie, and D. Estrin, “Collaborative metric learning,” in Proc. 26th Conf. Neural Inf. Process. Syst., Apr. 2017, pp. 193–201.
[7] T. Qin, T.-Y. Liu, and H. Li, “A general approximation framework for direct optimization of information retrieval measures,” Inf. Retr., vol. 13, no. 4, pp. 375–397, Aug. 2010.
[8] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, “Learning to rank: From pairwise approach to listwise approach,” in Proc. 24th Int. Conf. Mach. Learn. (ICML), 2007, pp. 129–136.
[9] L. Wu, C. Hsieh, and J. Sharpnack, “SQL-Rank: A listwise approach to collaborative ranking,” in Proc. 35th Int. Conf. Mach. Learn., 2018, pp. 5311–5320.
[10] J. Guiver and E. Snelson, “Bayesian inference for Plackett-Luce ranking models,” in Proc. 26th Annu. Int. Conf. Mach. Learn. (ICML), 2009, pp. 377–384.
[11] A. Grover, E. Wang, A. Zweig, and S. Ermon, “Stochastic optimization of sorting networks via continuous relaxations,” in Proc. 7th Int. Conf. Learn. Represent., 2019, pp. 1–23.
[12] M. Blondel, O. Teboul, Q. Berthet, and J. Djolonga, “Fast differentiable sorting and ranking,” in Proc. 37th Int. Conf. Mach. Learn., Oct. 2020, pp. 950–959.
[13] J. Weston, H. Yee, and R. J. Weiss, “Learning to rank recommendations with the k-order statistic loss,” in Proc. 7th ACM Conf. Rec. Syst., Oct. 2013, pp. 245–248.
[14] X. Wang, C. Li, N. Golbandi, M. Bendersky, and M. Najork, “The LambdaLoss framework for ranking metric optimization,” in Proc. 27th ACM Int. Conf. Inf. Knowl. Manage., Oct. 2018, pp. 1313–1322.
[15] Q. Wu, C. Burges, K. Svore, and J. Gao, “Adapting boosting for information retrieval measures,” Int. Retr., vol. 13, no. 3, pp. 254–270, 2010.
[16] P. Cremonesi, Y. Koren, and R. Turrin, “Performance of recommender algorithms on top-N recommendation tasks,” in Proc. 4th ACM Conf. Rec. Syst. (RecSys), 2010, pp. 39–46.
[17] D. Lim, J. McAuley, and G. Lanckriet, “Top-N recommendation with missing implicit feedback,” in Proc. 9th ACM Conf. Rec. Syst., Sep. 2015, pp. 309–312.
[18] J. Liang, J. Hu, S. Dong, and V. Honavar, “Top-N: A scalable listwise ranking method for recommender systems,” in Proc. IEEE Int. Conf. Big Data (Big Data), Dec. 2018, pp. 1052–1058.
[19] Y. Shi, M. Larson, and A. Hanjalic, “List-wise learning to rank with matrix factorization for collaborative filtering,” in Proc. 4th ACM Conf. Rec. Syst. (RecSys), 2010, pp. 269–272.
[20] R. Salakhutdinov and A. Mnih, “Probabilistic matrix factorization,” in Proc. 21st Conf. Neural Inf. Process. Syst., 2007, pp. 1257–1264.
[21] Y. Hu, Y. Koren, and C. Volinsky, “Collaborative filtering for implicit feedback datasets,” in Proc. 8th IEEE Int. Conf. Data Mining, Dec. 2008, pp. 263–272.
[22] M. Cogswell, F. Ahmed, R. Girshick, L. Zitnick, and D. Batra, “Reducing overfitting in deep networks by decorrelating representations,” 2015, arXiv:1511.06068. [Online]. Available: http://arxiv.org/abs/1511.06068
[23] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, “Neural collaborative filtering,” in Proc. 26th Int. Conf. World Wide Web, 2017, pp. 173–182.
[24] C. C. Johnson, “Logistic matrix factorization for implicit feedback data,” in Proc. Adv. Neural Inf. Process. Syst., 2014, vol. 27, no. 76, pp. 1–9.
[25] T.-Y. Liu, “Learning to rank for information retrieval,” Found. Trends Inf. Retr., vol. 3, no. 3, pp. 225–331, 2009.
[26] J. Tang, H. Liu, H. Gao, and A. Das Sarmas, “ETrust: Understanding trust evolution in an online world,” in Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD), 2012, pp. 253–261.
[27] F. M. Harper and J. A. Konstan, “The movielens datasets: History and context,” ACM Trans. Interact. Intell. Syst., vol. 5, no. 4, pp. 19:1–19:19, Jan. 2016.
[28] B. L. Welch, “The generalization of ‘student’s’ problem when several different population variances are involved,” Biometrika, vol. 34, nos. 1–2, pp. 25–35, 1947.
[29] X. Ning and G. Karypis, “SLIM: Sparse linear methods for top-N recommendation algorithms on top-N recommendation tasks,” in Proc. 4th ACM Conf. Rec. Syst., 2010, pp. 39–46.
[30] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, “Collaborative denoising auto-encoders for top-N recommender systems,” in Proc. IEEE 11th Int. Conf. Data Mining, Dec. 2011, pp. 497–506.
[31] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, “Collaborative denoising auto-encoders for top-N recommender systems,” in Proc. 9th ACM Int. Conf. Web Search Data Mining, Feb. 2016, pp. 153–162.
[32] J. Chen, D. Liao, and K. Zheng, “Improving one-class collaborative filtering via ranking-based implicit regularizers,” in Proc. AAAI Conf. Artif. Intell., 2019, pp. 37–44.
[33] D. Liang, J. Atsalas, L. Charlin, and D. M. Blei, “Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence,” in Proc. 10th ACM Conf. Rec. Syst., Sep. 2016, pp. 59–66.
[34] O. Levy and Y. Goldberg, “Neural word embedding as implicit matrix factorization,” in Proc. 27th Int. Conf. Neural Inf. Process. Syst., vol. 27, 2014, pp. 2177–2185.
[35] S. Bruch, M. Zoghi, M. Bendersky, and M. Najork, “Revisiting approximate metric optimization in the age of deep neural networks,” in Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., Jul. 2019, pp. 1241–1244.
[36] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in Proc. 26th Int. Conf. Neural Inf. Process. Syst., 2013, pp. 3111–3119.
HYUNSUNG LEE was born in Seoul, South Korea. He received the B.S. degree in computer engineering from Sungkyunkwan University, Suwon, South Korea, in 2019, where he is currently pursuing the master’s degree in electrical and computer engineering. His research interests include recommendation systems, cluster orchestration, and reinforcement learning.

SANGWOO CHO is currently pursuing the bachelor’s degree with the Department of Mathematics, Sungkyunkwan University, Suwon, South Korea. His research interests include multi-armed bandit, theoretical machine learning, and differential programming.

YEONGJAE JANG received the B.S. degree in mathematics from Sungkyunkwan University, Suwon, South Korea, in 2017, where he is currently pursuing the master’s degree with the Department of Mathematics. He has also been working as a Data Engineer, since April 2021. His research interests include theoretical machine learning and service optimization.

JAEEKWANG KIM was born in Seoul, South Korea, in 1980. He received the B.S., M.S., and Ph.D. degrees from Sungkyunkwan University, Suwon, South Korea, in 2004, 2006, and 2010, respectively. He is currently an Assistant Professor with the Department of Computing, School of Convergence, Sungkyunkwan University. His research interests include networks, security, and intelligent systems.

HONGUK WOO (Member, IEEE) was born in Seoul, South Korea. He received the B.S. degree in computer science from Korea University, Seoul, in 1995, and the M.S. and Ph.D. degrees in computer sciences from The University of Texas at Austin, Austin, TX, USA, in 2002 and 2008, respectively.

From 2008 to 2018, he worked with Samsung Research, Samsung Electronics, as a Principal Engineer and the Vice President. Since 2018, he has been an Assistant Professor with the Department of Computer Science and Engineering, Sungkyunkwan University, Suwon, South Korea. His research interests include intelligent application, data-driven monitoring, cloud computing, and networked cyber-physical systems.

* * *