LambdaMF: Learning Nonsmooth Ranking Functions in Matrix Factorization Using Lambda

Guang-He Lee
Department of Computer Science and Information Engineering
National Taiwan University
b00902055@csie.ntu.edu.tw

Shou-De Lin
Department of Computer Science and Information Engineering
National Taiwan University
sdlin@csie.ntu.edu.tw

***Abstract*—This paper emphasizes optimizing ranking measures in a recommendation problem. Since ranking measures are non-differentiable, previous works have been proposed to deal with this problem via approximations or lower/upper bounding of the loss. However, such mismatch between ranking measures and approximations/bounds can lead to non-optimal ranking results. To solve this problem, we propose to model the gradient of non-differentiable ranking measure based on the idea of virtual gradient, which is called lambda in learning to rank. In addition, noticing the difference between learning to rank and recommendation models, we prove that under certain circumstance the existence of popular items can lead to unlimited norm growing of the latent factors in a matrix factorization model. We further create a novel regularization term to remedy such concern. Finally, we demonstrate that our model, LambdaMF, outperforms several state-of-the-art methods. We further show in experiments that in all cases our model achieves global optimum of normalized discount cumulative gain during training. Detailed implementation and supplementary material can be found at (http://www.csie.ntu.edu.tw/~b00902055/).**

I. Introduction

With the prosperous emergence of e-commerce, recommendation system has played a big role in online stores. In general, users might prefer a recommendation system that selects a subset of candidate items for them to choose. In this sense the top-N recommended list is commonly adopted to evaluate a ranking-oriented recommendation system. Such evaluation consists of an Information Retrieval (IR) measure, i.e. ranking measure, and a cut-off value N. For example, Precision@5 specifies the precision value given top 5 recommended items.

However, the commonly used ranking measures are either indiscriminate or discontinuous over certain model space. Fig. 1 illustrates this behavior: adding a small Δs to the score of document c, $s(c)$ cannot alter the ranking of c. Thus a derivative equal to 0 appears in the measuring objective function. However, adding a larger value $\Delta s'$ to the score of c can cause the ranking of document c to immediately jump from 3^{rd} to 2^{nd}, which leads to discontinuity on the ranking measure due to the swapping of two documents. The property of zero or discontinuity gradient discourages the usage of common gradient descent optimization techniques, which usually leads to non-optimal solutions.

To handle such drawback, researchers have designed novel loss function for optimization. They are usually derived from a smooth approximation [1] or a bound [2] of the original ranking measures. However, there is a concern that the mismatch between the loss function and the ranking measure would result in a gap between the optimal and learned models [3].

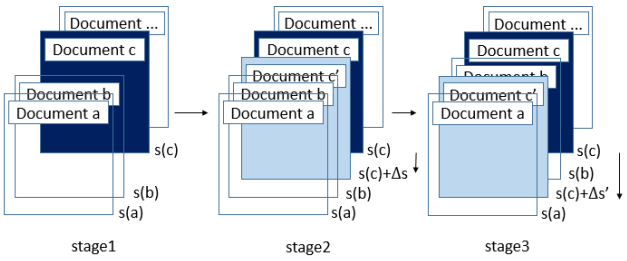

Figure 1. Ranking based on different model score

In the field of learning to rank, lambda-based methods [4] have been proposed to bypass converting challenging ranking measures into loss functions, by using a proxy of the gradient for ranking measures. The required ranking and sorting in ranking measures are then incorporated into the model by calculating the virtual gradient, a.k.a lambda, after sorting. Inheriting the idea of lambda, we design a new recommendation model named Lambda Matrix Factorization (LambdaMF) to combine the idea of lambda with the highly-successful matrix factorization model in collaborative filtering. To our knowledge, LambdaMF is the first ranking matrix factorization model to achieve global training optimum of a ranking measure: the Normalized Discount Cumulative Gain (NDCG). Our experiment also shows that LambdaMF directly optimize NDCG 98.4% of time during the optimization process.

Moreover, we have observed an interesting effect while learning LambdaMF. Sometimes the norm of some latent parameters can grow unlimitedly due to the lack of suppression power to constrain the effect of popular items during learning. In this paper we prove that such phenomenon can always happen with the existence of some popular items and some ranking-oriented matrix factorization models that satisfy certain criteria. We further purpose a novel regularization term to combat norm growing and empirically show the numerical stability of our solution.

1550-4786/15 $31.00 © 2015 IEEE
DOI 10.1109/ICDM.2015.108

Finally, we conduct experiments with MovieLens and Netflix datasets. Empirical comparison shows that our method outperforms several state-of-the-art models. In a nutshell, our main contribution in this paper can be listed as follows:

- We create a new matrix factorization algorithm modeling the gradient of ranking measure. To our best knowledge, LambdaMF is the first model to show that it empirically achieves global optimum of training data.
- We prove that some ranking-oriented matrix factorization model would suffer from unlimited norm growing due to popular item effect.
- We create a novel regularization term to deal with the popular item effect on recommendation systems.
- We conduct experiments to show that the model is scalable to large recommendation datasets and outperforms several state-of-the-art models.

The remaining sections are organized as follows. Section II presents several related works and comparative discussion. In Section III, the proposed LambdaMF and some related discussion are presented. Afterwards, experiment results and conclusion are in Section IV and Section V.

II. RELATED WORK

The designed model is strongly related to researches on both collaborative filtering and learning to rank. We discuss them briefly below.

A. Learning to Rank

In the context of Learning to Rank (LTR), the problem is usually formulated with queries Q, documents D and relevance levels R for documents in respect to each query. A feature vector is often given for each document-query pair as the focus is on optimizing the ranking measure for each query. The essential difference from a recommendation model is thus revealed: there is no queries or features in a collaborative filtering task. Nevertheless, people usually evaluate the results of Top-N recommendation by correlating users to queries and items to documents. To be general, we would use the same user to query and item to document analogy to describe the LTR task in the remaining paragraphs.

There are several different LTR measures that can be simply divided into explicit feedback and implicit relevance settings. For explicit feedback, for each observed item, users explicitly provide a specific score for it. In this scenario, the performance is often evaluated using normalized discounted cumulative gain (NDCG) with a cut off value K. Given relevance level R_i and predicted ranking l_i for every recommended item i in the sorted list based on sorted model score, the $NDCG@K$ is generated as follows:

$$NDCG@K = \frac{DCG@K}{max(DCG@K)}, DCG@K = \sum_{l_i=1}^{K} \frac{2^{R_i} - 1}{\log_2 l_i + 1}$$

B. Collaborative Filtering

Classically, the recommendation task is equivalent to the task of rating prediction. Well-known examples can be found in competitions such as the Netflix Prize. Common methods can be divided into memory based and model based methods. The major distinction between memory based and model based methods lies in whether the process of learning a model is conducted. Both methods can be regarded as collaborative filtering methods, as they meet the basic assumption that similar users tend to share similar interests.

However, recently the ranking-oriented CF methods [1], [5]–[8] attract more attention because ranking prediction can better capture the activity of personal preference ordering of items. Example can be found in CLiMF [1], optimizing an approximation of mean reciprocal rank (MRR), and CoFi Rank [2], optimizing a bound of NDCG. However, in CLiMF, the $\frac{1}{ranking}$ of an item is approximated as the sigmoid function of its model score. It only considers the actual value of the model score and rather than the ordinal ranking of the model score. In contrast, LambdaMF directly deals with actual ranking due to is incorporation of sorting. For CoFi Rank, the loss function is derived from a bound of NDCG. However, there is no guarantee about the distance between the bound and actual real ranking function. Conversely, LambdaMF directly formulate the ranking function into its gradient and therefore there is no need to worry about the similar extent of approximation or bound. Comparing to the above models, LambdaMF is more general; it presents a template model which can be fine-tuned for different types of ranking functions given sparse data.

III. LAMBDA MATRIX FACTORIZATION

This section first introduces the background of lambda, then discusses how lambda gradient can be incorporated into a matrix factorization model. To reflect the difference between learning to rank and recommendation system, a novel regularization term is created to address the concern of overflow from using lambda. Finally, a time complexity analysis is presented to demonstrate the efficiency of the model

A. RankNet and LambdaRank

In a series of researches on learning to rank [4], RankNet [9] was first proposed to handle the problem using gradient. In this model, given a query q, for every pair of documents i and j with different relevance level $R_{q,i} > R_{q,j}$, RankNet aims to promote the relative predicted probability of these pair in terms of their model score s_i, s_j. Then the relative probability, document i ranked higher than document j, generated from this model is defined as (1)

$$P^q_{i,j} = \frac{1}{1 + \exp(-\sigma(s_i - s_j))} \tag{1}$$

σ is simply a parameter controlling the shape of sigmoid function. Finally, the loss function is defined as the cross entropy between the pair-wise preference ordering of the training data and the relative probability of the model. Given the same assumption with relevance level $R_{q,i} > R_{q,j}$, the

probability of pairwise real preference ordering between i, j is defined as $P^q_{i,j} = 1$. Finally, the loss function and gradient for the query q, document i, j is defined as (2), (3)

$$\begin{aligned} C^q_{i,j} &= -\bar{P^q_{i,j}} \log P^q_{i,j} - (1 - \bar{P^q_{i,j}}) \log(1 - P^q_{i,j}) \\ &= \log(1 + \exp(-\sigma(s_i - s_j))) \end{aligned} \quad (2)$$

$$\frac{\partial C^q_{i,j}}{\partial s_i} = -\frac{\sigma}{1 + \exp(\sigma(s_i - s_j))} = -\frac{\partial C^q_{i,j}}{\partial s_j} \quad (3)$$

Based on the idea of RankNet, formulating pair-wise ranking problem into gradient descent for a pair of documents, LambdaRank [3] is proposed to formulate gradient of a list-wise ranking measure, which is called λ. Several possible λs have been tested in [3], while the authors report the best λ used for optimizing NDCG as (4) with $|\Delta NDCG|$ defined as the absolute NDCG gain from swapping documents i and j

$$\frac{\partial C^q_{i,j}}{s_i} = \lambda^q_{i,j} = \frac{\sigma |\Delta NDCG|}{1 + \exp(\sigma(s_i - s_j))} = -\frac{\partial C^q_{i,j}}{s_j} \quad (4)$$

$$\frac{\partial C^q_{i,j}}{w} = \frac{\partial C^q_{i,j}}{\partial s_i}\frac{\partial s_i}{\partial w} + \frac{\partial C^q_{i,j}}{\partial s_j}\frac{\partial s_j}{\partial w} = \lambda^q_{i,j}\frac{\partial s_i}{\partial w} - \lambda^q_{i,j}\frac{\partial s_j}{\partial w}$$

B. Lambda in Matrix Factorization

To introduce λ into a collaborative filtering model, the first thing to notice is that we do not assume the existences of user or item features due to privacy concern. Hence, we adopt MF as our latent factor model representing features for users and items. Given M users and N items, there is a sparse rating matrix $R_{M,N}$, in which each row represents a user while each column represents an item. The values in the matrix are only available when the corresponding ratings exist in the training dataset, and the task of MF is to guess the remaining ratings. Similar to Singular Value Decomposition, the idea behind MF is to factorize the rating matrix $R_{M,N}$ into two matrices $U_{M,d}$ and $V_{N,d}$ with pre-specified latent factor dimension d. Given $a \in [1, M]$ and $b \in [1, N]$, U_a represents the latent factor of the a^{th} user while V_b represents the latent factor of the b^{th} item. Finally, the rating prediction can be formulated as the matrix product below. The inner-product of U_u, V_i represents the predicted score for each (u, i) pair:

$$\bar{R} = UV^T$$

Using MF as the basis of our latent model, now we are ready to introduce LambdaMF. Suppose the goal is to optimize a ranking measure function f, we first construct users and items latent factors of dimension d as our model basis. Then we assume that there is a virtual cost function C optimizing f, and the gradient of C in respect to model score is defined as λ,

$$\frac{\partial C^u_{i,j}}{\partial s_i} = \lambda^u_{i,j} \quad (5)$$

However, to obtain the gradient in respect to model parameters, we need to first discuss the differences between LambdaRank and LambdaMF. First, given a user u, when a pair of items (i, j) is chosen, unlike in LTR which only model weight vector is required to be updated, in a recommendation task we have item latent factors of i and j and user latent factor of u to update. Second, in the original LambdaRank model, the score of the pair (u, i) in the model is generated from the prediction outputs of a neural network, while in MF such score is generated from the inner-product of U_u and V_i. Hence, we have $\frac{\partial s_i}{\partial U_u} = V_i$ and $\frac{\partial s_i}{\partial V_i} = U_u$.

Here we are ready to describe how to apply SGD to learn model parameters. Given the first difference described above, it is apparent that there are 2 item latent factors and 1 user latent factor to be considered when generating the stochastic gradient. Moreover, given the second difference, we can exploit the chain rule during the derivation of gradient. Finally, given a user u, a pair of item i and j, with $R_{u,i} > R_{u,j}$, the gradient can be computed as (6):

$$\begin{aligned} \frac{\partial C^u_{i,j}}{V_i} &= \frac{\partial C^u_{i,j}}{\partial s_i}\frac{\partial s_i}{\partial V_i} = \lambda^u_{i,j} U_u \\ \frac{\partial C^u_{i,j}}{V_j} &= \frac{\partial C^u_{i,j}}{\partial s_j}\frac{\partial s_j}{\partial V_j} = -\lambda^u_{i,j} U_u \\ \frac{\partial C^u_{i,j}}{U_u} &= \frac{\partial C^u_{i,j}}{\partial s_i}\frac{\partial s_i}{\partial U_u} + \frac{\partial C^u_{i,j}}{\partial s_j}\frac{\partial s_j}{\partial U_u} = \lambda^u_{i,j}(V_i - V_j) \end{aligned} \quad (6)$$

The specific definition of $\lambda^u_{i,j}$ is not given in this section. In contrast, we want to convey that the design of $\lambda^u_{i,j}$ can be simple and generic. In [10], to deal with LTR problems, several λ have been proposed for optimizing corresponding metrics. In our opinion, any function with positive range can be viewed as an legitimate lambda as long as it can promote the preferred item and penalize the less relevant item. Here we emphasize the positive values of lambda as positive gradient which can always enlarge the gap between each pair of items with different relevance levels and leads to faster optimization.

C. Popular Item Effect

As would be shown later, we found the existence of popular items can cause a serious problems while trying to apply the concept of Lambda in MF. Moreover, we will show that the phenomena is universal for MF models with positive gradient $\frac{\partial C^u_{i,j}}{\partial s_i} > 0$ for all j with $R_{u,i} > R_{u,j}$ such as BPRMF [5]. Suppose that there exists a popular item $\hat{i}$ liked by all users who have rated it, then for each user u among them, the rating for item $\hat{i}$ would keep climbing during the optimization process as $\frac{\partial C^u_{i,j}}{\partial s_i}$ is positive: it pushes the score of $U_u V_i^T$ upwards. Moreover, if the latent factors among all users observing the item are similar, the increasing of predicted score of $\hat{i}$ for every user u observing $\hat{i}$ would not cause the decrease of predicted score of $\hat{i}$ for the other users who observe $\hat{i}$. Hence, the latent factor of item $\hat{i}$ would keep growing and soon cause overflow.

Formally speaking, let us first denote the set of users observing $\hat{i}$ in the training set as $Rel(\hat{i})$, the rating of $\hat{i}$ for user k as $R_{k,\hat{i}}$, the latent factors in the iteration t as U^t/V^t. Then the above discussion yields the Theorem 1, whose proof is shown in supplementary materials.

Theorem 1 (Popular Item Theorem): If there exists an item $\hat{i}$, such that for all users $k \in Rel(\hat{i})$, $R_{k,\hat{i}} \geq R_{k,j}$ for all other observed item j of user k. Furthermore, if after certain iteration τ, latent factors of all users $k \in Rel(\hat{i})$ converge to certain extent. That is, there exists a vector $\bar{U}^t$ such that for all

$k \in Rel(\hat{i})$ in all iteration $t > \tau$, inner-product$(U_k^t, \bar{U^\tau}) > 0$. Then the norm of $V_{\hat{i}}$ will eventually grow to infinity for any MF model satisfying the constraint that $\frac{\partial C_{\hat{i},j}^u}{\partial s_{\hat{i}}} > 0$ for all j with $R_{u,\hat{i}} > R_{u,j}$, as shown below:

$$\lim_{n\to\infty} \|V_{\hat{i}}^n\|^2 = \infty$$

Note that the existence of item $\hat{i}$ can be easily satisfied since the set size of $Rel(\hat{i})$ can be arbitrary (even with only one user). The other condition in Theorem 1, namely the convergence of latent factors of users observing $\hat{i}$, can also happen frequently in CF since we generally believe that users with similar transaction history, i.e. buying the same item $\hat{i}$, shall have similar latent factors. Furthermore, since the theorem only requires $\frac{\partial C_{\hat{i},j}^u}{\partial s_{\hat{i}}} > 0$ for all j with $R_{u,\hat{i}} > R_{u,j}$, it is not restricted to LambdaMF but applies to all other matrix factorization models satisfying such constraint. Empirical evidences will also be shown in the experiment section.

D. Regularization in LambdaMF

To address the above concern, an intuitive solution is to add a regularization term to the original cost C (we call this new cost $\dot{C}$). For example, a commonly used L2 regularization term, i.e. weight decay, can confine the norm of the matrix U and V and thus restrict the complexity of model. Then the new cost function $\dot{C}$ can be formulated as

$$\dot{C} = \sum_{all(u,i,j)\in data} C_{i,j}^u - \frac{\alpha}{2}(\|U\|_2^2 + \|V\|_2^2) \tag{7}$$

However, since the norm of $V_{\hat{i}}$ can grow very fast, the L2 regularization term may not effectively stop the norm of $V_{\hat{i}}$ from going to infinity when the effect of weight decay is not significant. On the other extreme, if α is large, the capability of the model is strongly restricted even when the norm of all parameters are small. Hence, we argue that the utilization of norm restriction based method is blind; it cannot adapt its regularization intensity for different magnitudes of parameter norms.

Here, we propose another regularization to confine the inner-product of $U_u V_i^T$ (which is the prediction outcome of the model) to be as close to the actual rating $R_{u,i}$ for every observed rating u, i as possible, as defined in (8)

$$\hat{C} = \sum_{all(u,i,j)\in data} C_{i,j}^u - \frac{\alpha}{2} \sum_{all(u,i)\in data} (R_{i,j} - U_u V_i^T)^2 \tag{8}$$

$$\frac{\partial \hat{C_{i,j}^u}}{\partial V_i} = \lambda_{i,j}^u U_i + \alpha(R_{u,i} - U_u V_i^T)U_u \tag{9}$$

$$\frac{\partial \hat{C_{i,j}^u}}{\partial V_j} = -\lambda_{i,j}^u U_i + \alpha(R_{u,j} - U_u V_j^T)U_u$$

$$\frac{\partial \hat{C_{i,j}^u}}{\partial U_u} = \lambda_{i,j}^u (V_i - V_j) + \alpha(R_{u,i} - U_u V_i^T)V_i + \alpha(R_{u,j} - U_u V_j^T)V_j$$

We choose the square error between $R_{u,i}$ and $U_u V_i^T$ because it is differentiable and thus can be optimized easily. This equation is known as mean-square-error (MSE), which is often adopted in a rating prediction task as an objective function [11]. There are two advantages in adopting the MSE regularization term.

1) *Intensity-adaptive regularization term:* It can be derived from (8) that the gradient of C with respect to parameters is (9). Such regularization can adapt its intensity to the difference between the predicted score and the real rating using $(R_{u,i} - U_u V_i^T)$. When the predicted score varies, the regularization power also adjust linearly. In conclusion, we would like to point out that the adaptation power of MSE regularization is automatic with no manipulation of value α needed.
2) *Inductive transfer with MSE:* Inductive transfer [12] is a technique to use some source task T_S to help the learning of target task T_T. In our scenario, T_S corresponds to MSE while T_T is corresponds to the ranking measure f. Moreover, MSE presents a way to model learning to rank with point-wise learning. An optimal MSE=0 also suggests an optimal NDCG=1. Hence, the proposed regularization can be viewed as adopting an inductive transfer strategy to enhance the performance of LambdaMF.

E. 3.5 Algorithm and Complexity Analysis

Once the lambda gradient and regularization term are derived, we are ready to define the algorithm and analyze its time complexity. In the previous sections we have provided a general definition of λ, but how to choose λ remains unsolved. In [9], the λ is defined as (4); however, we argue that the multiplication with RankNet gradient is irrelevant to the target function. In addition, the computation of RankNet gradient includes exponential function, which is time consuming. Therefore, given a user u, ranking measure f, and an item ranking list ξ, for every pair of items i and j with $R_{u,i} > R_{u,j}$, we adopt the absolute ranking measure difference between ξ and $\xi\prime$, the same list as ξ except that the rankings of i and j are swapped. That is,

$$\lambda_{i,j}^u = |f(\xi) - f(\xi\prime)| \tag{10}$$

The ranking measure can be applied to certain ranking measures. To be concrete, applying NDCG as the ranking measure and denoting the ranking of i as l_i, (10) will be:

$$|f(\xi) - f(\xi\prime)| = |\frac{(2^{R_i} - 2^{R_j})(\frac{1}{\log(1+l_i)} - \frac{1}{\log(1+l_j)})}{max(DCG)}| \tag{11}$$

Let us assume that the computation of inner-product of latent factors takes $O(1)$ time to simplify the following derivation. Given $\bar{N}$ observed items for a user u, the computation of ξ takes $O(\bar{N}\log\bar{N})$ time since it requires sorting, so as the computation of $\lambda_{i,j}^u$. Hence, if SGD is applied, updating a pair of items for a user takes total $O(\bar{N}\log\bar{N})$ time, which is expensive considering only a single pair of item is updated.

To overcome such deficiency, we propose to conduct the mini-batch gradient descent. After sorting the observed item list for a user, we compute the gradient for all observed items. As a result, $O(\bar{N}\log\bar{N})$ time is still needed to do

the sorting and $max(DCG)$. Finally, the computation of $\lambda^u_{i,j}$ takes constant time with $O(\bar{N}^2)$ pairs. Hence, the total computation of updating $O(\bar{N}^2)$ pairs takes $O(\bar{N}^2)$ time. Effectively, updating each pair of items takes $O(1)$ time as the sorting time is shadowed with mini-batch learning structure. To sum up, the whole algorithm is depicted in **Algorithm 1**.

Algorithm 1: Learning LambdaMF

input : $R_{M,N}$, learning rate η, α, latent factor dimension d, # of iterations n_iter, and target ranking measure f

Initialize $U_{M,d}$,$V_{N,d}$ randomly;
for $t \leftarrow 1$ **to** n_iter **do**
  for $u \leftarrow 1$ **to** M **do**
    $\frac{\partial \hat{C^u}}{\partial U_u} \leftarrow 0$;
    for *all observed* $i \in R_u$ **do**
      $\frac{\partial \hat{C^u}}{\partial V_i} \leftarrow 0$;
    ξ = observed item list of u sorted by its predicted scores
    for *all observed pair* $i, j \in R_u, R_{u,i} > R_{u,j}$ **do**
      $\xi\prime$ = ξ with i, j swapped
      $\frac{\partial \hat{C^u}}{\partial U_u} += \frac{\partial C^u_{i,j}}{\partial U_u}$ as Eqation(9),(10)
      $\frac{\partial \hat{C^u}}{\partial V_i} += \frac{\partial C^u_{i,j}}{\partial V_i}$ as Eqation(9),(10)
      $\frac{\partial \hat{C^u}}{\partial V_j} += \frac{\partial C^u_{i,j}}{\partial V_j}$ as Eqation(9),(10)
    $U_u += \eta \frac{\partial \hat{C^u}}{\partial U_u}$
    for *all observed* $i \in R_u$ **do**
      $V_i += \eta \frac{\partial \hat{C^u}}{\partial V_i}$
return U, V

Since there are enormous amount of items in a recommender system, the $O(\bar{N}^2)$ time complexity seems to be unignorable. However, the sparsity of data does compensate this drawback. Due to data sparsity, the number of observed items per user $\bar{N}$ is negligible compared to the number of total items N. For example, with 99% sparsity, the computation of all pairs takes $10^{-4}N^2 \ll N^2$. In [7], each λ is computed through all N items, leading to the failure of applying lambda.

IV. EXPERIMENT

In this section, we evaluate the performance of LambdMF comparing to other state-of-the-art recommendation methods. Furthermore, we demonstrate the optimality of LambdaMF and the stability of regularization.

A. Comparison Methods

In this work, we adopt NDCG as the target ranking function to demonstrate the performance of LambdaMF; applying LambdaMF to other ranking functions is also possible but not included in this paper. We compare LambdaMF with several state-of-the-art LTR recommendation system methods.

Table I
STATISTICS OF DATASETS

	Netflix	MovieLens
# of users	480,189	943
# of items	17,770	1,682
sparsity	98.82%	93.70%

- *PopRec*: Popular Recommendation (PopRec) is a strong non-CF baseline. The idea of PopRec is to give users non-personalized recommendation according to the popularity of items in the training set. That is, the more times an item appears in training, the higher ranking the item gets.
- *CoFi Rank*: CoFi Rank is a state-of-the-art MF method minimizing a convex upper bound of (1-NDCG) [2]. CoFi Rank presents the choice of formulating an upper bound as the learning function, which differs from our choice of formulating gradient. In their work, several other loss functions are also evaluated for NDCG. We compare with the NDCG loss function (denoted as CoFi NDCG) and the loss function with the best performance (denoted as CoFi Best) reported in the original paper.
- *ListRank MF*: ListRank MF is another state-of-the-art MF method optimizing cross-entropy of top-one probability [6], which is implemented using softmax function. Though the top-one probability seems to be irrelevant to NDCG, the experiment shows the superiority over CoFi Rank in some experiments.

B. Experiment Results

We conduct experiments on MovieLens 100K and Netflix dataset as in [2]. The statistics are illustrated in Table I. For each dataset, we apply weak generalization [2], [6] to produce the experimental data. That is, we first remove users who rated less than 20, 30, and 60 items (such as movies). Then for the remaining users, we sample $N = 10, 20, 50$ ratings for each user in the training dataset and put the remaining ratings in testing dataset. As a result, a user will have at least 10 relevant items in testing set, which favors the usage of $NDCG@10$ as our evaluation function.

Similar to [2], [6] which fixed all model parameters in their experiment, we tune 5 set of parameters in MovieLens 100K $N = 20$ dataset and set $\eta = 0.001, \alpha = 0.5$ and $n\_iter = 250$ for our experiments. For $N = 10, 20, 50$, we construct different experimental dataset 10 times for each N to avoid bias. Finally, in each experimental dataset, we run LambdaMF 10 times for MovieLens 100K and once for Netflix, following the same setting as [2]. Finally, we report average result of $NDCG@10$ across 10 experimental datasets for $N = 10, 20, 50$.

The experiment results for MovieLens 100K and Netflix are shown in Table II and Table III, respectively. Note that for LambdaMF, we report two experiment results of adopting different regularization terms, L2 and MSE. For L2 regularization, we apply the weight decay for each related latent factors once while updating a single user. For all experiment,

Table II
(MEAN,STANDARD DEVIATION) OF $NDCG@10$ IN MOVIELENS100K

	N=10	N=20	N=50
PopRec	(0.5995,0.0000)*	(0.6202,0.0000)*	(0.6310,0.0000)*
CoFi NDCG	(0.6400,0.0061)*	(0.6307,0.0062)*	(0.6076,0.0077)*
CoFi Best	(0.6420,0.0252)*	(0.6686,0.0058)*	(0.7169,0.0059)
ListRank MF	(0.6943,0.0050)*	(0.6940,0.0036)*	(0.6881,0.0052)*
LambdaMF L2	(0.5518,0.0066)*	(0.5813,0.0074)*	(0.6471,0.0074)*
LambdaMF MSE	(**0.7119,0.0005**)	(**0.7126,0.0008**)	(**0.7172,0.0013**)

Table III
(MEAN) OF $NDCG@10$ IN NETFLIX; THE RESULTS OF COFI RANK FOR N=50 AND LISTRANK MF IS NOT AVAILABLE SINCE THE AUTHORS DO NOT REPORT ITS RESULTS AND DISCLOSE THE COMPLETE PARAMETERS

	N=10	N=20	N=50
PopRec	(0.5175)	(0.5163)	(0.5293)
CoFi NDCG	(0.6081)	(0.6204)	(Not available)
CoFi Best	(0.6082)	(0.6287)	(Not available)
LambdaMF L2	(**0.7550**)	(**0.7586**)	(**0.7464**)
LambdaMF MSE	(0.7171)	(0.7163)	(0.7218)
LambdaMF MSE best	(**0.7558**)	(**0.7586**)	(**0.7664**)

we found that LambdaMF with MSE regularization achieves superior performance comparing to all other competitors. In Netflix, as LambdaMF with L2 regularization yields better result than MSE regularization, it seems to be due to the tolerability of strong regularization power in large Netflix dataset, but MSE regularization performs better in general for all experiments. Moreover, we found that LambdaMF with MSE regularization using fewer iterations achieves the best result in all Netflix experimental settings using the same parameters (LambdaMF MSE best). Finally, we conduct unpaired t test on the LambdaMF MSE with the other models. With two tailed p value smaller than 0.0001, we add * to show extremely significant improvement. Besides, we also run experiments for LambdaMF without regularization, and it lead to overflow after certain iterations in all cases.

C. Optimality of LambdaMF

Even with superior performance, one may still argue that the learning of LambdaMF is only an approximation of optimizing ranking measure, but not a real optimization of ranking measure. The question has once been answered in [10] using Monte-Carlo hypothesis testing of the weighting vectors of the underlined model. The result of testing shows that LambdaRank has achieved a local optimum with 99% confidence. However, in this work, we notice that LambdaMF with MSE regularization is capable of achieving a global optimum in all our experimental datasets given long enough iterations. That is, the training performance of $NDCG@10$ achieves 1.0. Furthermore, the $NDCG@10$ is nondecreasing for all iterations until convergence in Netflix dataset and nondecreasing for at least 98.4% of the iterations in MovieLens 100K dataset before reaching an optimum.

V. CONCLUSION

In this work, we purpose a novel framework for lambda in a recommendation scenario. After deriving a model incorporating MF and lambda, we further prove that ranking-based MF model can be ineffective due to overflow in certain circumstances. To deal with this problem, an adjustment of our model using regularization is proposed. Note that such regularization is not emphasized in a typical LTR problem. In addition, the update step for each pair of items takes effectively $O(1)$ time in LambdaMF. We empirically show that LambdaMF outperforms the state-of-the-arts in terms of $NDCG@10$. Extended experiments demonstrate that LambdaMF exhibits global optimality and directly optimizes target ranking function. The insensitivity of MSE regularization in the experiments also demonstrates the stability of MSE regularization.

ACKNOWLEDGMENT

This work is supported by Taiwan government MOST funding number 103-2221-E-002 -104 -MY2 and 102-2923-E-002 -007 -MY2

REFERENCES

[1] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic, "Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering," in *Proceedings of the sixth ACM conference on Recommender systems*. ACM, 2012, pp. 139–146.

[2] M. Weimer, A. Karatzoglou, Q. V. Le, and A. Smola, "Maximum margin matrix factorization for collaborative ranking," *Advances in neural information processing systems*, 2007.

[3] C. Quoc and V. Le, "Learning to rank with nonsmooth cost functions," *Proceedings of the Advances in Neural Information Processing Systems*, vol. 19, pp. 193–200, 2007.

[4] C. J. Burges, "From ranknet to lambdarank to lambdamart: An overview," *Learning*, vol. 11, pp. 23–581, 2010.

[5] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2009, pp. 452–461.

[6] Y. Shi, M. Larson, and A. Hanjalic, "List-wise learning to rank with matrix factorization for collaborative filtering," in *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010, pp. 269–272.

[7] W. Zhang, T. Chen, J. Wang, and Y. Yu, "Optimizing top-n collaborative filtering via dynamic negative item sampling," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2013, pp. 785–788.

[8] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010, pp. 39–46.

[9] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 89–96.

[10] P. Donmez, K. M. Svore, and C. J. Burges, "On the local optimality of lambdarank," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2009, pp. 460–467.

[11] P.-L. Chen, C.-T. Tsai, Y.-N. Chen, K.-C. Chou, C.-L. Li, C.-H. Tsai, K.-W. Wu, Y.-C. Chou, C.-Y. Li, W.-S. Lin *et al.*, "A linear ensemble of individual and blended models for music rating prediction," *KDDCup*, 2011.

[12] S. J. Pan and Q. Yang, "A survey on transfer learning," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 22, no. 10, pp. 1345–1359, 2010.