List-wise Learning to Rank with Matrix Factorization for Collaborative Filtering

Yue Shi
Multimedia Information Retrieval Lab
Delft University of Technology
Delft, The Netherlands
y.shi@tudelft.nl

Martha Larson
Multimedia Information Retrieval Lab
Delft University of Technology
Delft, The Netherlands
m.a.larson@tudelft.nl

Alan Hanjalic
Multimedia Information Retrieval Lab
Delft University of Technology
Delft, The Netherlands
a.hanjalic@tudelft.nl

ABSTRACT

A ranking approach, ListRank-MF, is proposed for collaborative filtering that combines a list-wise learning-to-rank algorithm with matrix factorization (MF). A ranked list of items is obtained by minimizing a loss function that represents the uncertainty between training lists and output lists produced by a MF ranking model ListRank-MF enjoys the advantage of low complexity and is analytically shown to be linear with the number of observed ratings for a given user-item matrix. We also experimentally demonstrate the effectiveness of ListRank-MF by comparing its performance with that of item-based collaborative recommendation and a related state-of-the-art collaborative ranking approach (CoFiRank).

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Information Filtering*

General Terms

Algorithms, Performance, Experimentation

Keywords

Recommender systems, collaborative filtering, learning to rank, matrix factorization, recommendation

1. INTRODUCTION

Recommender systems attract research attention because of their importance for handling the unprecedentedly large amount of content, e.g., movies, music, books, currently available online to users [1]. Collaborative filtering (CF) has been regarded as one of the most successful recommender techniques. It is based on the concept that a user would like an item if the item is liked by the users with similar preferences. The ultimate purpose of a recommender system is to provide users with a ranking or recommendation list and this objective has been argued to be more crucial than the accuracy of rating prediction [1][5][7][13]. For this reason, we focus on ranking or recommendation (also referred as top-N recommendation [3]) in this paper, rather than rating prediction.

We propose a scalable extension to the matrix factorization approach to collaborative filtering called ListRank-MF. Our approach makes use of a list-wise learning-to-rank technique to rank items for each user, in which users and items are represented as latent features learned using matrix factorization (MF). The key contribution of our approach is twofold: it provides a recommendation performance improvement over the existing state-of-the-art matrix factorization approach and it keeps the complexity linear with the number of observed ratings in the given user-item matrix, meaning that it can scale-up to be used in very large collections.

Learning to rank (LTR) is supervised machine learning approach that automatically constructs a ranking model/function from training data [10]. Recently LTR has been the subject of intensive research effort, e.g., within the Yahoo! LTR Challenge[1]. LTR research benefits both information retrieval (IR) and recommendation, both fields that focus on the task of returning a rank list of items in response to an information need, expressed explicitly as a query (IR) or implicitly in the user profile (recommendation). Although much work has been carried out on LTR techniques for systems that return documents in response to a query [10], little effort has been devoted to exploiting LTR in recommender systems, or specifically, collaborative filtering.

In order to apply LTR to CF several challenges must be faced. First, user profiles and items in the recommendation setting are not easy to represent in terms of explicit features. In this way, recommendation contrasts with IR applications, where queries and documents can be explicitly represented, e.g., using term weights derived from occurrence frequencies. Researchers in recommender systems frequently represent users and items with their rating vectors, which are essentially different from query and document representations in that they have no direct relationship to user and item characteristics. To address this issue, we make use of matrix factorization to represent users and items by latent features. Second, not all LTR approaches are inherently suitable for CF application. Liu et al. [10] categorizes LTR into point-wise, pair-wise and list-wise. Point-wise approaches predict ranking scores for individual documents and can thus be regarded as conceptually equivalent to rating prediction problem in CF. A ranked list is created by ordering documents according to these scores. As discussed in the literature, the accuracy measure of rating prediction is difficult to interpret as a measure of ranking quality [11][13]. Pair-wise LTR makes a prediction for every pair of documents concerning their relative ordering in the final list. This approach is computationally intensive and this disadvantage prevents it from scaling well to large data collections typical for CF scenarios. Approaches to CF that are conceptually similar to pair-wise LTR have been presented in the literature [11][12][14], but a scalable solution to the complexity problem remains elusive. For this reason, we turn to list-wise LTR.

[1] http://learningtorankchallenge.yahoo.com/index.php

The remainder of the paper is structured as follows. In the next section, we summarize related work and position our approach with respect to it. Then, we present the ListRank-MF algorithm and validate it experimentally. Finally, we sum up the key aspects of ListRank-MF and briefly mention directions for future work.

2. RELATED WORK

This section briefly summarizes the existing approaches to collaborative filtering and related LTR approaches.

2.1 Collaborative Filtering

A comprehensive survey of collaborative filtering approaches can be found in [1][7][18]. Collaborative filtering can be memory-based or model-based. In general, memory-based approaches make recommendations on the basis of similarities between users (user-based) [6], or on the basis of similarities between items (item-based) [3][16]. Further modification and enhancement has been devoted to improving either user-based approaches [4][17] [22], or item-based approaches [21]. Model-based approaches first fit prediction models based on training data and then use the model to predict users' preference on items, e.g., latent semantic model [8]. Matrix factorization (MF) techniques have attracted much research attention, due to the advantages of scalability and accuracy [9][15], especially for large-scale data, as exemplified by the Netflix contest. Generally, MF techniques learn latent features of users and items from the observed ratings in the user-item matrix, which are further used to predict unobserved ratings.

Recently, research attention in the area of CF has shifted away from the rating prediction problem and places more focus on the quality of the ranking or recommendation list that the recommender system produces [13]. Many approaches have been investigated, including investigation of pair-wise preference between items for users, e.g., EigenRank [11], probabilistic latent preference analysis [12] and Bayesian probabilistic ranking [14]. These approaches all suffer from expensive computation which limits their scalability. In contrast, ListRank-MF proposed in this paper is of low complexity, i.e., complexity linear with number of the observed ratings in a given user-item rating matrix.

2.2 Learning to Rank

A comprehensive survey of LTR can be found in [10]. The accumulated body of LTR literature is sizable and here we focus on list-wise LTR approaches as most relevant to our ListRank-MF approach. Under a list-wise approach, an individual training example is an entire list of items, rather than individual items or item pairs [2]. Loss functions for list-wise LTR are formulated to reflect the distance between the reference list and the output list from the ranking model. Various algorithms are applied to learn the optimal or local optimal ranking model. The permutation probability was proposed to represent the ranked list, which could be further simplified to top one probability [2]. In this paper, we also take the concept of top one probability to represent the recommendation list, making our work closest to list-wise LTR. CoFiRank [20] was proposed to directly optimize ranking effectiveness metrics for collaborative ranking, motivated also mainly by LTR and is also close to the work reported here.

3. THE ALGORITHMS

In this section, we introduce the list-wise learning to rank with matrix factorization (ListRank-MF). We first present the two key components related to ListRank-MF, i.e., probabilistic matrix factorization (PMF) framework and top one probability. Then, we present the formulation of ListRank-MF as a loss function and develop corresponding learning process for local optimal solution.

3.1 Probabilistic Matrix Factorization

The PMF framework was proposed in [15], where matrix factorization is formulated from probabilistic inference of conditional distribution of observed ratings, user rating priors and item rating priors. And the ultimate framework is formulated as:

$$U,V=\arg\min_{U,V}\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{N}I_{ij}\left(R_{ij}-g(U_i^TV_j)\right)^2+\frac{\lambda_U}{2}\|U\|_F^2+\frac{\lambda_V}{2}\|V\|_F^2 \qquad (1)$$

Supposing the user-item rating matrix R consists of M users and N items, PMF seeks to represent the user-item rating matrix R with two low-rank matrices, U and V. A d-dimensional set of latent features is used to represent both users (in U) and items (in V). Note that we use U_i to denote a d-dimensional column feature vector of user i and V_j to denote a d-dimensional column feature vector of item j. R_{ij} denotes the user i's rating on item j. I_{ij} is an indicator function that equals 1 when $R_{ij} > 0$, and 0 otherwise. Finally, λ_U and λ_V are regularization coefficients. We usually set $\lambda_U=\lambda_V=\lambda$ for simplicity. The $g(x)$ is a logistic function to bound the range of $U_i^TV_j$, i.e., $g(x)=1/(1+exp(-x))$.

Note that PMF, and also some other matrix factorization approaches as in [9], are rating prediction oriented. Although we can use the predicted ratings to rank the items, the quality of ranking is not directly related to the purpose of PMF (minimizing the error of rating prediction). In analogy, PMF is equivalent to a point-wise ranking model, while not modeling ranking directly.

3.2 Top One Probability

As proposed in [2], the top one probability for an item rated R_{ij} in user i's ranking list l_i (e.g., with K items) can be expressed as:

$$P_{l_i}\left(R_{ij}\right)=\frac{\varphi\left(R_{ij}\right)}{\sum_{k=1}^{K}\varphi\left(R_{ik}\right)} \qquad (2)$$

where $\varphi(x)$ can be any monotonically increasing and strictly positive function. For simplicity, we take the same form as used in [2], i.e., the exponential function for $\varphi(x)$. The top one probability indicates the probability of an item being ranked in the top position for a given ranking list.

3.3 ListRank-MF

We formulate the ListRank-MF by using the cross-entropy of top-one probabilities of the items in the training example lists and the ranking lists from the ranking model (MF) as the loss function:

$$\begin{aligned}L(U,V)&=\sum_{i=1}^{M}\left\{-\sum_{j=1}^{N}P_{l_i}\left(R_{ij}\right)\log P_{l_i}\left(g\left(U_i^TV_j\right)\right)\right\}+\frac{\lambda}{2}\left(\|U\|_F^2+\|V\|_F^2\right)\\&=\sum_{i=1}^{M}\left\{-\sum_{j=1}^{N}I_{ij}\frac{\exp\left(R_{ij}\right)}{\sum_{k=1}^{N}I_{ik}\exp\left(R_{ik}\right)}\log\frac{\exp\left(g\left(U_i^TV_j\right)\right)}{\sum_{k=1}^{N}I_{ik}\exp\left(g\left(U_i^TV_j\right)\right)}\right\} \qquad (3)\\&+\frac{\lambda}{2}\left(\|U\|_F^2+\|V\|_F^2\right)\end{aligned}$$

The training example lists consist of the training-set items in the profiles of each user. The output of the recommendation model is a recommendation list for each user i and is generated by ranking items in collection in descending order according to the value U_i^TV. Items already contained among the training examples in the user i's profile are removed.

The loss function reflects the uncertainty between the training lists and the output lists from the ranking model, i.e., MF here.

The regularization term serves to reduce over-fitting. The optimal ranking model should perform least uncertainty between lists of training ratings and lists of output predictions list-wisely. Note that here the MF is not optimized for rating prediction, but for ranking positions of items in the users' lists. As the loss function is not convex jointly over *U* and *V*, we choose to use gradient descent with alternatively fixed *U* and *V*, from which a local minimum can be obtained. The gradients of *L(U,V)* with respect to *U* and *V* can be computed as:

$$\frac{\partial L}{\partial U_i} = \sum_{j=1}^{N} I_{ij} \left(\frac{\exp\left(g\left(U_i^T V_j\right)\right)}{\sum_{k=1}^{N} I_{ik} \exp\left(g\left(U_i^T V_k\right)\right)} - \frac{\exp\left(R_{ij}\right)}{\sum_{k=1}^{N} I_{ik} \exp\left(R_{ik}\right)} \right) g'\left(U_i^T V_j\right) V_j + \lambda U_i \qquad (4)$$

$$\frac{\partial L}{\partial V_j} = \sum_{i=1}^{M} I_{ij} \left(\frac{\exp\left(g\left(U_i^T V_j\right)\right)}{\sum_{k=1}^{N} I_{ik} \exp\left(g\left(U_i^T V_k\right)\right)} - \frac{\exp\left(R_{ij}\right)}{\sum_{k=1}^{N} I_{ik} \exp\left(R_{ik}\right)} \right) g'\left(U_i^T V_j\right) U_i + \lambda V_j \qquad (5)$$

Note that *g'(x)* denotes the derivative of *g(x)*.

3.4 Complexity Analysis

Exploiting the sparseness of user-item matrix, the computation of the loss function in Eq. (3) is of complexity *O(2dS+d(M+N))*, where *S* denotes the number of observed ratings in a given user-item matrix. The gradients in Eq. (4) and Eq. (5) are *O(2dS+dM)* and *O(dS+pdS+dN)*, respectively, where *p* denotes the average number of items rated per user and usually is a very small value compared to *S* in collaborative filtering. Considering we often have *S >> M, N*, the total complexity in one iteration is *O(dS+pdS)*, which is linear in the number of observed ratings in the matrix. This analysis indicates that ListRank-MF is computationally efficient and can be applied to large-scale cases.

4. EXPERIMENTS AND EVALUATION

In this section, we present the preliminary experiments to evaluate ListRank-MF. We first investigate the impact of regularization coefficient on the ListRank-MF and further validate its effectiveness that optimizing the loss function contributes to ranking performance. We finally demonstrate that ListRank-MF could be not only efficient in learning, but also promising in performance which could improve the state-of-the-art approach CoFiRank.

4.1 Experimental Setup

Our experiments are conducted on the MovieLens dataset, which consists of 100K ratings (scale 1–5) assigned by 943 users to a collection of 1682 items [6]. We adopted the evaluation protocol referred to as "weak generalization" in the CoFiRank evaluation in [20]. We randomly select 10, 20 and 50 rated items for each user for training and use the remaining rated items in the user profile for testing. For each condition, users with less than 20, 30, or 60 rated items are removed in order to ensure we can evaluate on at least 10 rated items per user. We report the average performance attained across all users and ten runs of this procedure. We choose the evaluation metric as Normalized Discounted Cumulative Gain (NDCG), which is more sensitive to the relevance of higher ranked items. As in CoFiRank [20], we focus on NDCG@10. Note that we use latent feature dimension of 5 and learning rate 0.01 in our implementation for ListRank-MF in all the experiments.

4.2 Impact of Regularization Coefficient λ

The regularization coefficient λ in ListRank-MF influences the convergence of the loss function and controls for overfitting. To investigate the impact of λ, we use one data fold, from the condition in which 10 randomly selected rated items from each user are chosen for training. Fig. 1 illustrates the relationship between λ and the loss and show the danger of overfitting arises when λ is lower than the level of 0.001. Note that for illustrative purposes, the loss has been normalized for each setting of λ. Since no over-fitting effect is evident with λ equal or larger than 0.01, we set λ to 0.01 in all the following experiments for consistency.

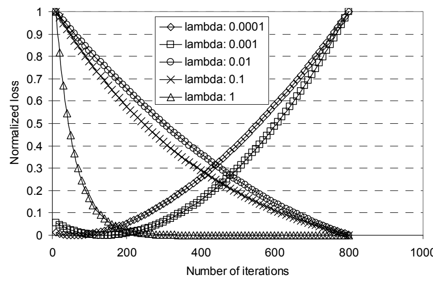

Fig. 1 The impact of regularization coefficient on the convergence of loss in learning process

4.3 Effectiveness and Efficiency

The optimization of the loss function leads to minimizing the loss. However, investigation whether the minimizing of the loss could indeed lead to a good property of ranking is still needed, e.g., NDCG@10 used in this paper. For this reason, we demonstrate the development of the loss and NDCG@10 simultaneously during the iterations of optimization, as shown in Fig. 2. As observed, it is effective that the ranking performance becomes both optimal and convergent when optimizing the loss function. Moreover, it indicates the ListRank-MF is efficient as the NDCG@10 becomes approximately optimal after around 250 iterations.

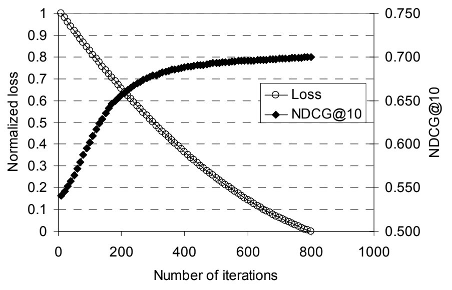

Fig. 2 The effectiveness of ListRank-MF to achieve optimal NDCG@10 by minimizing the loss

4.4 Performance Comparison

In this section we compare the performance of ListRank-MF with the well-known item-based collaborative recommendation (ItemCR) [3] and the state-of-the-art CoFiRank [20]. Since we share exactly same experimental protocol as used in CoFiRank [20], we can compare results directly, i.e., CoFiRank-NDCG and CoFiRank-Best reported in [20]. Note that CoFiRank-NDCG represents an approach that directly optimizes NDCG as a loss function and CoFiRank-Best represents the options that lead to best results by CoFiRank in each condition. As shown in Table 1, ListRank-MF achieves a performance improvement of ca. 15% over ItemCR and ca. 10% over CoFiRank-NDCG. The improvement is significant for all the conditions, according to Wilcoxon signed rank test with $p < 0.05$. ListRank-MF also achieves ca. 5% improvement over CoFiRank-Best (significantly) for experimental conditions with 10 and 20 rated items per user for training, while it is slightly outperformed for the condition with 50 rated items per user for training.

Table 1. NDCG@10 (mean ±standard deviation across 10 runs) comparison between ItemCR, CoFiRank and ListRank-MF, with "^" denoting significant improvements over others.

	Rated items per user in training set		
	10	**20**	**50**
ItemCR	0.5779 ±0.0017	0.5805 ±0.0026	0.5715 ±0.0063
CoFiRank-NDCG	0.6400 ±0.0061	0.6307 ±0.0062	0.6076 ±0.0077
CoFiRank-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

Note that since we do not have performance of CoFiRank for each user in each run, the significance test involving CoFiRank was conducted by generating 10 samples from a Gaussian distribution with the corresponding mean and deviation in each condition, which are further compared with 10 average performances in 10 runs by ListRank-MF and ItemCR.

5. DISCUSSION AND CONCLUSIONS

In this paper, we have introduced the ListRank-MF, an approach that exploits a list-wise learning to rank technique in order to realize improvement over the performance of the state-of-the art matrix factorization techniques for the task of recommendation. The experiments demonstrate the ListRank-MF outperforms item-based collaborative recommendation and the state-of-the-art CoFiRank, significantly in most cases. We also analyze the computational complexity of ListRank-MF and find it to be linear in the number of observed ratings in the given user-item matrix. Because ListRank-MF is computationally inexpensive, it can be scaled up for use on large-scale real-world collections.

Future work involves several interesting directions. First, the ListRank-MF in this paper is based on the concept of top one probability. From the point of view of the recommender system, for which the top items are critical, this provides a sensible representation of the list. However, we would like to investigate whether performance improvement can be achieved via representations that reflect information along the entire list without an undue increase in computational complexity. Second, the proposed ListRank-MF approach, like most of the current CF recommendation algorithms, could be regarded as a variational recommendation approach, where the common evaluation metrics, e.g., mean reciprocal rank, mean average precision and NDCG, are not directly related to the model. CoFiRank [20] has made the first attempt to address this issue, and recent developments in this area in LTR research [10] [19] could be further exploited for improving recommendation performance by directly optimizing for evaluation metrics.

6. ACKNOWLEDGEMENTS

The research leading to these results was carried out within the PetaMedia Network of Excellence and has received funding from the European Commission's 7th Framework Program under grant agreement n° 216444.

7. REFERENCES

[1] Adomavicius G., and Tuzhilin, A., 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE TKDE, 17, 6, 734-749.

[2] Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F. and Li, H., 2007. Learning to rank: From pairwise approach to listwise approach. Technical Report, MSR-TR-2007-40, Microsoft Research.

[3] Deshpande, M., and Karypis, G., 2004. Item-based top-N recommendation algorithms. ACM TOIS, 22, 1, 143-177.

[4] Ding, S., Zhao, S., Yuan, Q., Zhang, X., Fu, R. and Bergman, L., 2008. Boosting collaborative filtering based on statistical prediction errors. In RecSys '08, 3-10.

[5] Gunawardana A., and Shani, G., 2009. A survey of accuracy evaluation metrics of recommendation tasks. JMLR, 10, 2935-2962.

[6] Herlocker, J., Konstan, J., Borchers, A., and Riedl, J., 1999. An algorithmic framework for performing collaborative filtering. In SIGIR '99, 230-237.

[7] Herlocker, J., Konstan, J., Terveen, L. G., and Riedl, J. 2004. Evaluating collaborative filtering recommender systems. ACM TOIS, 22, 1, 5-53.

[8] Hofmann, T., 2004. Latent semantic models for collaborative filtering. ACM TOIS, 22, 1, 89-115.

[9] Koren, Y., Bell, R., and Volinsky, C., 2009. Matrix factorization techniques for recommender systems. IEEE Computer, 42, 8, 30-37.

[10] Liu, T. -Y., 2009. Learning to rank for information retrieval. Foundations and Trends in Information Retrieval, 3, 3, 225-331.

[11] Liu, N. N., and Yang, Q., 2008. EigenRank: a ranking-oriented approach to collaborative filtering. In SIGIR '08, 83-90.

[12] Liu, N. N., Zhao, M., and Yang, Q., 2009. Probabilistic latent preference analysis for collaborative filtering. In CIKM '09, 759-766.

[13] McNee, S. M., Riedl, J., and Konstan, J. A., 2006. Being accurate is not enough: How accuracy metrics have hurt recommender systems. In extended abstracts of CHI '06, 1097-1101.

[14] Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme L., 2009. BPR: Bayesian personalized ranking from implicit feedback. In UAI '09, 452-461.

[15] Salakhutdinov, R., and Mnih, A., 2008. Probabilistic matrix factorization. In NIPS '08, 20.

[16] Sarwar, B., Karypis, G., Konstan, J., and Reidl, J., 2001. Item-based collaborative filtering recommendation algorithms. In WWW '01, 285-295.

[17] Shi, Y., Larson, M., and Hanjalic, A., 2009. Exploiting user similarity based on rated-item pools for improved user-based collaborative filtering. In RecSys '09, 125-132.

[18] Su, X. and Khoshgoftaar, T. M., 2009. A survey of collaborative filtering techniques. Advances in Artificial Intelligence, no. 421425, 19 pages.

[19] Volkovs, M. N., and Zemel, R. S., 2009. BoltzRank: learning to maximize expected ranking gain. In ICML '09, 1089-1096.

[20] Weimer, M, Karatzoglou, A., Le, Q., and Smola, A., 2007. CoFi rank-maximum margin matrix factorization for collaborative ranking. In NIPS '07, 20, 1593-1600.

[21] Yildirim, H., and Krishnamoorthy, M. S., 2008. A random walk method for alleviating the sparsity problem in collaborative filtering. In RecSys '08, 131-138.

[22] Zhang, J. and Pu, P., 2007. A recursive prediction algorithm for collaborative filtering recommender systems. In RecSys '07, 57-64.