xCLiMF: Optimizing Expected Reciprocal Rank for Data with Multiple Levels of Relevance

Yue Shi[a], Alexandros Karatzoglou[b], Linas Baltrunas[b], Martha Larson[a], Alan Hanjalic[a]
[a]Delft University of Technology, Netherlands; [b]Telefonica Research, Spain
[a]{y.shi, m.a.larson, a.hanjalic}@tudelft.nl, [b]{alexk, linas}@tid.es

ABSTRACT

Extended Collaborative Less-is-More Filtering *xCLiMF* is a learning to rank model for collaborative filtering that is specifically designed for use with data where information on the level of relevance of the recommendations exists, e.g. through ratings. *xCLiMF* can be seen as a generalization of the Collaborative Less-is-More Filtering (*CLiMF*) method that was proposed for top-N recommendations using binary relevance (implicit feedback) data. The key contribution of the *xCLiMF* algorithm is that it builds a recommendation model by optimizing Expected Reciprocal Rank, an evaluation metric that generalizes reciprocal rank in order to incorporate user feedback with multiple levels of relevance. Experimental results on real-world datasets show the effectiveness of *xCLiMF*, and also demonstrate its advantage over *CLiMF* when more than two levels of relevance exist in the data.

Categories and Subject Descriptors

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

Keywords

Collaborative filtering, expected reciprocal rank, graded relevance, top-N recommendation, ranking

1. INTRODUCTION

The purpose of a recommender system is, typically, to generate a ranked item list (or top-N item list) that is tailored to the preferences of each individual user [1, 8]. Collaborative Filtering (CF) is one of the most successful techniques for recommender systems [6], the core concept of which is to provide recommendations to a user based on the common interest between this user and other users.

Learning to rank techniques originated in the Information Retrieval (IR) and Machine Learning communities [3]. One main branch of learning to rank techniques is to directly optimize a IR measure such as Mean Average Precision (MAP), Area Under the ROC Curve (AUC), Mean Reciprocal Rank (MRR) , or Normalized Discounted Cumulative Gain (NDCG), e.g., [2, 15]. Learning to rank techniques have been recently used to optimize CF models. CF models for ranking binary relevance data are based on optimizing a smooth version of binary relevance measures for IR such as AUC in the case of Bayesian Personalized Ranking (BPR) [10], MAP for the TFMAP algorithm [12] and MRR for *CLiMF* [13]. Note that in this paper we use the term binary relevance to refer to data where users provide implicit feedback about their preferences (e.g., clicks, purchases, views etc.) *not* the case where there is explicit feedback on a binary scale (e.g., like/dislike). When multiple levels of relevance exist in the data, e.g., rating levels assigned by users can be productively treated as grades of relevance, these methods cannot be easily used since that would require an arbitrary threshold to binarize the relevance. In this way though fine-grained rating information is thrown out that actually may be helpful.

Ranking methods for CF have been also developed for multiple level relevance data which typically come in the form of ratings, for example, CoFiRank [14] optimized an upper bound of a NDCG loss function. One of the main disadvantage of NDCG is the underlying assumption that the usefulness of a document at rank i is independent of the usefulness of the documents at rank less than i. Thus a single perfectly relevant item in the top of a recommendation list might get less NDCG score than a list with several lower scored relevant items in the top positions.

The *Expected Reciprocal Rank* (ERR) [4] measure was recently introduced to deal with these shortcomings. ERR is a generalized version of RR designed to be used with multiple relevance level data (e.g., ratings). It has similar properties to RR in that it strongly emphasizes the relevance of results returned at the top of the list. Importantly, ERR, like RR, imposes the assumption that the user scans the results list from top to bottom, and stops when a result is found that fully satisfies the user's information need. This model of user behavior provides a particularly good fit with top-N recommender use scenarios. In such scenarios, the user is often looking for a single item/product and is completely satisfied after having found it in the list. This work builds upon our previous contribution of *CLiMF* [13], which aimed at optimizing Reciprocal Rank (RR), for top-N recommendation in domains with binary relevance data. It extends our initial experimentation with exploiting graded relevance for

RecSys'13, October 12–16, 2013, Hong Kong, China.
Copyright 2013 ACM 978-1-4503-2409-0/13/10 ...$15.00.
http://dx.doi.org/10.1145/2507157.2507227.

top-N recommendation [11]. Its critical contribution is the focus on the benefits of ERR within top-N recommendation use scenarios. We propose a new CF approach, *Extended Collaborative Less-is-More Filtering (xCLiMF)*, which aims at directly optimizing ERR, for top-N recommendation in domains with graded relevance data. We derive an optimization procedure that is linear to the number of relevance data (i.e., ratings), and show that *xCLiMF* outperforms a few baseline methods on two datasets.

The remainder of the paper is structured as follows. In the next section, we present the technical detail of *xCLiMF*, after which we demonstrate the experimental evaluation in Section 3. Finally, Section 4 summarizes the key aspects of this work and briefly addresses the directions for future work

2. EXTENDED CLIMF

Using the definition of ERR in [4], we can express ERR for a ranked item list of user m as follows:

$$ERR_m = \sum_{i=1}^{N} \frac{r_{mi}}{R_{mi}} \prod_{j=1}^{N} \big(1 - r_{mj}\mathbb{I}(R_{mj} < R_{mi})\big) \quad (1)$$

$\mathbb{I}(\cdot)$ is an indicator function, which is equal to 1 if the condition is true, and otherwise 0. R_{mi} denotes the rank position of item i for user m, when all items are ranked in descending order of the predicted relevance values. r_{mi} denotes the probability that user m finds the item i is relevant. We use a mapping function similar to the one used in [4], to convert ratings (or levels of relevance in general) to probabilities of relevance, as shown below:

$$r_{mi} = \begin{cases} \frac{2^{y_{mi}-1}}{2^{y_{max}}}, & I_{mi} > 0 \\ 0, & I_{mi} = 0 \end{cases} \quad (2)$$

where y_{mi} denotes the rating given by user m to item i, and y_{max} is the highest rating. Note that $I_{mi} > 0$ ($I_{mi} = 0$) indicates that user m's preference to item i is known (unknown). We employ factor models and in particular matrix factorization where the relevance between user m and item i is modeled by the inner product of U_m and V_i (i.e., the latent factors of user m and item i), as follows:

$$f_{mi} = \langle U_m, V_i \rangle = \sum_{d=1}^{D} U_{md} V_{id} \quad (3)$$

Note that the value of rank R_{mi} depends on the value of f_{mi}. For example, if the predicted relevance value f_{mi} of item i is the second highest among all the items for user m,then we will have $R_{mi} = 2$.

Similar to other ranking measure such as RR, ERR is also non-smooth with respect to the latent factors of users, items, i.e., U and V, it is thus impossible to optimize ERR directly using conventional optimization techniques. We thus employ smoothing techniques that where also used in *CLiMF* [13], to attain a smoothed version of ERR. In particular we approximate the rank-based terms $1/R_{mi}$ and $\mathbb{I}(R_{mj} < R_{mi})$ in Eq. (1) by smooth functions with respect to the model parameters U and V, as shown below

$$\mathbb{I}(R_{mj} < R_{mi}) \approx g(f_{mj} - f_{mi}) \quad (4)$$

$$\frac{1}{R_{mi}} \approx g(f_{mi}) \quad (5)$$

where $g(x)$ is a logistic function, i.e., $g(x) = 1/(1 + e^{-x})$.

Substituting Eq. (4) and (5) into Eq. (1), we obtain a smoothed approximation of ERR_m:

$$ERR_m = \sum_{i=1}^{N} r_{mi} g(f_{mi}) \prod_{j=1}^{N} \big(1 - r_{mj} g(f_{m(j-i)})\big) \quad (6)$$

Note that for notation convenience, we make use of the substitution $f_{m(j-i)} := \langle U_m, V_j \rangle - \langle U_m, V_i \rangle$.

Using Jensen's inequality and exploiting the concavity of the *log* function in a similar manner to [13], we can derive a lower bound of ERR_m in Eq.(6) :

$$L(U_m, V) = \sum_{i=1}^{N} r_{mi} \big[\ln g(f_{mi}) + \sum_{j=1}^{N} \ln \big(1 - r_{mj} g(f_{m(j-i)})\big)\big] \quad (7)$$

Using the lower bound we can optimize a factor model with respect to ERR. Taking into account all M users and using the Frobenius norm of the latent factors for regularization, we obtain the objective function of *xCLiMF*:

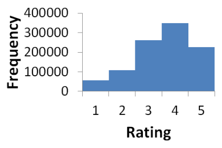
$$\begin{aligned} F(U, V) = \sum_{m=1}^{M} \sum_{i=1}^{N} r_{mi} \big[& \ln g(f_{mi}) \\ & + \sum_{j=1}^{N} \ln \big(1 - r_{mj} g(f_{m(j-i)})\big)\big] \\ & - \frac{\lambda}{2}(\|U\|^2 + \|V\|^2) \end{aligned} \quad (8)$$

We can now maximize the objective function (8) with respect to the latent factors $argmax_{U,V} F(U, V)$. Note that $F(U, V)$ represents an approximation of the mean value of ERR across all the users, we can thus remove the constant coefficient $1/M$, since it has no influence on the optimization of $F(U, V)$. Since the objective function is smooth we can use gradient ascent for the optimization. The gradients can be derived in a similar manner to *CLiMF* [13], as shown below:

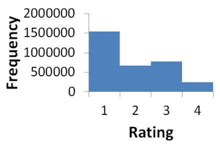
$$\begin{aligned} \frac{\partial F}{\partial U_m} = & \sum_{i=1}^{N} r_{mi} \big[g(-f_{mi}) V_i \\ & + \sum_{k=1}^{N} \frac{r_{mk} g'(f_{mk} - f_{mi})}{1 - r_{mk} g(f_{mk} - f_{mi})} \big(V_i - V_k\big)\big] - \lambda U_m \end{aligned} \quad (9)$$

$$\begin{aligned} \frac{\partial F}{\partial V_i} = & r_{mi} \big[g(-f_{mi}) \\ & + \sum_{k=1}^{N} r_{mk} g'(f_{mi} - f_{mk}) \big(\frac{1}{1 - r_{mk} g(f_{mk} - f_{mi})} \\ & - \frac{1}{1 - r_{mi} g(f_{mi} - f_{mk})} \big)\big] U_m - \lambda V_i \end{aligned} \quad (10)$$

Note that *xCLiMF* can be seen as a generalized version of the *CLiMF* algorithm. We can derive *CLiMF* from the *xCLiMF* algorithm as a special case when r_{mi} is binary, i.e., 1 for relevant items and 0 otherwise. The overall complexity of the optimization procedure is in the order of $O(d\tilde{n}^2 M)$, where $\tilde{n}$ the average number of items per user. Note that we have $\tilde{n}M = S$, in which S denotes the number of non-zeros in the user-item matrix. The complexity of the learning algorithm is then $O(d\tilde{n}S)$. Since we usually have $\tilde{n} << S$,



(a) ML1m



(b) TVid

Figure 1: The distributions of ratings in ML1m and TVid datasets.

the complexity is $O(dS)$ even in the case that $\tilde{n}$ is large, i.e., being linear to the number of non-zeros (i.e. relevant observations in the data).

3. EXPERIMENTAL EVALUATION

In this section we present a series of experiments for the evaluation of *xCLiMF*. We are trying to address the following two research questions:

1. Does *xCLiMF* outperform state-of-the-art CF approaches for top-N recommendation in the domains with explicit feedback data?

2. How does *xCLiMF* perform compared to *CLiMF*?

3.1 Experimental Setup

3.1.1 Datasets

We use two datasets for the experiments: The first is the MovieLens 1 million dataset[1] (ML1m) [7], which contains ca. 1M ratings (1-5 scale) from ca. 6K users and 3.7K movies. The sparseness of the ML1m dataset is 95.53%. The second dataset was collected from the video watching data in the Tuenti social network[2]. In the experiments, we convert the count of video-watching into four levels: 1 if a user did not watched a video completely; 2 if a user watching a video only once; 3 if a user watching a video 2-5 times; 4 if a user watched a video more than 5 times. Those levels (similar to ratings) are defined to indicate different levels of user preference to videos, and used as explicit feedback data in the experiments. In the following, we denote this dataset as TVid, it contains ca. 3.2 million rating levels from ca. 55.5K users and 51.1K videos with 99.89% data sparseness. The distributions of ratings in the two datasets are presented in Figure 1.

3.1.2 Experimental Protocol

For each dataset, we randomly selected 5 rated items (movies or videos) for each user to form a test set. We then randomly selected a varying number of rated items from the rest to form a training set. For example, under the condition of "Given 5", we randomly select 5 rated items (disjoint to the items in the test set) for each user in order to generate a training set. We investigated a variety of "Given" conditions for the training sets, i.e., 5, 10 and 15 for the ML1m dataset and 5, 15, 25 for the TVid dataset. Generated recommendation lists for each user are compared to the ground truth in the test set in order to measure the performance. Apart from ERR, we also measure the recommendation performance by NDCG, a well-known evaluation metric for measuring the performance of ranked results with graded judgments. Since in recommender systems the user's satisfaction is dominated by only a few items on the top of the recommendation list, our evaluation in the following experiments focuses on the performance of top-5 recommended items, i.e., ERR@5 and NDCG@5. Note that in the evaluation we cannot treat all the unrated items as irrelevant to a given user. For this reason, we apply a widely-used practical strategy, as suggested in literature [5, 9], which is to first randomly select 1000 unrated items (which are assumed to be irrelevant to the user) in addition to the ground truth (i.e., rated items) for each user in the test set, and then evaluate the performance of the recommendation list that consists of only the selected unrated items and the ground truth items. In addition, considering the domination effect of top popular items [13], we exclude the top-3 most popular items in each dataset from recommendations for the evaluation in the test set. The parameters in *xCLiMF* are tuned so as to yield the best balance between performance and computational cost based on a validation set of each dataset, i.e., D=10, λ=0.001 and learning rate 0.001.

[1]http://www.grouplens.org/node/73
[2]https://www.tuenti.com/

3.2 Performance Comparison

We compare the performance of *xCLiMF* with that of two baseline algorithms. The approaches we compare against are listed below:

- **PopRec**. A naive baseline that recommends items based on the number of times of being rated.

- **CofiRank**. A state-of-the-art CF approach that optimizes the NDCG measure [14] for domains with explicit feedback data (ratings). The implementation is based on the publicly available software package from the authors[3].

The results of the experiments are shown in Table 1 and 2, from which we obtain two observations.

First, *xCLiMF* outperforms the baseline approaches in terms of both ERR and NDCG in most of the cases. The improvements against baselines are all statistically significant according to the Wilcoxon signed rank test with p<0.05. The results also show that the improvement of ERR aligns consistently with the improvement of NDCG, indicating that optimizing ERR would not degrade the utility of recommendations that are captured by the NDCG measure.

Second, the relative performance of *xCLiMF* improves as the number of observed ratings from the users increases. This result indicates that *xCLiMF* can learn better top-N recommendation models if more observations of the graded relevance data from users can be used. The results also reveal that it is difficult to model user preferences encoded in multiple levels of relevance with limited observations, in particular, when the number of observations is lower than the number of relevance levels.

3.3 xCLiMF vs. CLiMF

We compare the performance between *xCLiMF* and *CLiMF*. We use *CLiMF* on the explicit feedback datasets by binarizing the rating values with different thresholds. On the ML1m dataset, we take rating 4 and 5 (the highest two relevance levels), respectively, as the relevance threshold for

[3]http://www.cofirank.org/

Table 1: **Performance comparison of *xCLiMF* and the baseline approaches on the ML1m dataset**

	Given 5		Given 10		Given 15	
	NDCG	ERR	NDCG	ERR	NDCG	ERR
PopRec	0.024	0.023	0.030	0.029	0.025	0.025
CofiRank	0.031	0.037	0.035	0.040	0.036	0.042
xCLiMF	**0.035**	**0.038**	**0.037**	**0.044**	**0.043**	**0.054**

Table 2: **Performance comparison of *xCLiMF* and the baseline approaches on the TVid dataset**

	Given 5		Given 15		Given 25	
	NDCG	ERR	NDCG	ERR	NDCG	ERR
PopRec	0.120	0.059	0.160	0.079	0.203	0.107
CofiRank	**0.148**	**0.097**	0.169	0.096	0.205	0.124
xCLiMF	0.121	0.063	**0.229**	**0.164**	**0.337**	**0.241**

training the *CLiMF* model and measure the performance using Mean Reciprocal Rank (MRR). On the TVid dataset, we take rating 3 and 4 (the highest two relevance levels), respectively, as the relevance threshold. The results are shown in Table 3 and 4. We observe that, although *CLiMF* is trained to optimize RR, it suffers from the additional data sparseness as a result of the binarization of the multi-level relevance data. *CLiMF* is outperformed by *xCLiMF* in terms of both MRR and ERR, across all the settings of relevance thresholds and datasets. The results indicate that the information loss from binarizing multi-level relevance data would inevitably make recommendation models based on binary relevance data, such as *CLiMF*, suboptimal for the use cases with explicit feedback data. We can thus conclude that, *xCLiMF* is addressing the top-N recommendation problem for multiple-level relevance data.

4. CONCLUSIONS AND FUTURE WORK

We have presented *xCLiMF*, an extended version of *CLiMF* that can be used with multi-level relevance data for the purpose of top-N recommendation. Compared to *xCLiMF*, which optimizes RR, *xCLiMF* learns latent factors of users and items by optimizing ERR, a measure of quality of ranked results with multiple relevance levels. The experiments demonstrate the competitive performance of *xCLiMF* for top-N recommendation in the domains with multi-level relevance data. We also demonstrate the specific advantage of *xCLiMF* compared to *CLiMF* when used on multi-level relevance data.

There are several possibilities for future work. One is to investigate the relationship between ERR optimization and diversification. Another interesting topic is to compare the recommendation models that are designed for optimizing different ranking measures, and to investigate the possibility of optimizing recommendation results for multiple criteria.

5. ACKNOWLEDGEMENTS

This work is funded as part of a Marie Curie Intra European Fellowship for Career Development (IEF) award (CARS, PIEF-GA-2010-273739) held by Alexandros Karatzoglou.

6. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Kowledge and Data Engineering*, 17(6):734–749, 2005.

Table 3: ***xCLiMF* vs. *CLiMF* on the ML1m dataset**

	Threshold 4		Threshold 5	
	MRR	ERR	MRR	ERR
CLiMF	0.088	0.032	0.062	0.043
xCLiMF	**0.104**	**0.054**	**0.064**	**0.054**

Table 4: ***xCLiMF* vs. *CLiMF* on the TVid dataset**

	Threshold 3		Threshold 4	
	MRR	ERR	MRR	ERR
CLiMF	0.234	0.161	0.065	0.133
xCLiMF	**0.348**	**0.241**	**0.080**	**0.241**

[2] C. J. C. Burges, R. Ragno, and Q. V. Le. Learning to rank with nonsmooth cost functions. NIPS '06, pages 193–200, 2006.

[3] O. Chapelle, Y. Chang, and T.-Y. Liu. Future directions in learning to rank. *Journal of Machine Learning Research - Proceedings Track*, 14:91–100, 2011.

[4] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 621–630, New York, NY, USA, 2009. ACM.

[5] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. RecSys '10, pages 39–46. ACM, 2010.

[6] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan. Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction*, 4(2):81–173, 2011.

[7] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. SIGIR '99, pages 230–237. ACM, 1999.

[8] J. Konstan and J. Riedl. Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction*, 22:101–123, 2012. 10.1007/s11257-011-9112-x.

[9] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. KDD '08, pages 426–434. ACM, 2008.

[10] S. Rendle, C. Freudenthaler, Z. Gantner, and S.-T. Lars. Bpr: Bayesian personalized ranking from implicit feedback. UAI '09, pages 452–461. AUAI Press, 2009.

[11] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, and A. Hanjalic. GAPfm: Optimal top-n recommendations for graded relevance domains. CIKM '13. ACM, 2013.

[12] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, and N. Oliver. TFMAP: optimizing map for top-n context-aware recommendation. SIGIR '12, pages 155–164. ACM, 2012.

[13] 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. RecSys '12, pages 139–146. ACM, 2012.

[14] M. Weimer, A. Karatzoglou, Q. Le, and A. Smola. Cofirank - maximum margin matrix factorization for collaborative ranking. NIPS'07, pages 1593–1600, 2007.

[15] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. SIGIR '07, pages 271–278. ACM, 2007.