AdaMF:Adaptive Boosting Matrix Factorization for Recommender System

Yanghao Wang, Hailong Sun, and Richong Zhang

School of Computer Science and Engineering, Beihang University
Beijing, 100191, China
{wangyh,sunhl,zhangrc}@act.buaa.edu.cn

Abstract. Matrix Factorization (MF) is one of the most popular approaches for recommender systems. Existing MF-based recommendation approaches mainly focus on the prediction of the users' ratings on unknown items. The performance is usually evaluated by the metric Root Mean Square Error (RMSE). However, achieving good performances in terms of RMSE does not guarantee a good performance in the top-N recommendation. Therefore, we advocate that treating the recommendation as a ranking problem. In this study, we present a ranking-oriented recommender algorithm AdaMF, which combines the MF model with AdaRank. Specifically, we propose an algorithm by adaptively combining component MF recommenders with boosting methods. The combination shows superiority in both ranking accuracy and model generalization. Normalized Discounted Cumulative Gain (NDCG) is chosen as the parameter of the coefficient function for each MF recommenders. In addition, we compare the proposed approach with the traditional MF approach and the state-of-the-art recommendation algorithms. The experimental results confirm that our proposed approach outperforms the state-of-the-art approaches.

Keywords: Recommender System, Matrix Factorization, Learning to Rank.

1 Introduction

Recommender systems are widely adopted by online websites to discover the implicit needs of consumers. In recent years, existing approaches have focused on the rating prediction for users on unobserved items, in which Root Mean Square Error (RMSE) is usually chosen as the performance evaluation metric. We refer these approaches as rating-oriented recommender systems. One of the popular approaches for solving this problem is Matrix Factorization (MF) [1] which can also be called Latent Factor Model. However in real recommender systems, instead of rating prediction performance, top-N recommendation precision may attract more attention as it generates a ranked list which is directly presented to users. The uncertainty of the prediction in rating-oriented recommender system brings instability in the ranking performance. Moreover, other factors such as

F. Li et al. (Eds.): WAIM 2014, LNCS 8485, pp. 43–54, 2014.

the position information that measure the ranking performance are not taken in to consideration in rating-oriented recommender systems. Paolo et.al show that lower RMSE does not always translate into a better ranking result in [2]. Thus, instead of achieving better rating prediction, directly optimizing the relevance of the ranking list can offer user better experience. In recent years, Learning to Rank (LTR) [3] has becoming a powerful technique to solve ranking problem by using machine learning. For existing LTR-based recommender systems [4,5], in practice, the single model cannot be adjusted adaptively to the shifty real world data and the ranking performance evaluation metrics, such as NDCG, are not usually taken into the account when building the objective function.

In this paper, we propose AdaMF, an adaptive boosting approach for ranking-oriented recommender systems, by combining a state-of-the-art ranking model AdaRank [6] with MF. AdaRank creates a *strong ranker* by composing *weak rankers* linearly and the coefficient of each weak ranker is calculated from the weight function of a certain performance metric. Different with the traditional LTR problem, recommendation problem usually does not have explicit features in dataset to build model-based weak rankers. However Matrix Factorization can allocate latent factors for each user and item only based on user history data. Therefore, we choose Matrix Factorization to build the *component recommender* and component recommenders are combined to the *ensemble recommender* linearly according to the recommendation performance. In our approach we choose Normalized Discounted Cumulative Gain (NDCG) as the parameter for the coefficient calculation, which can measure the effect of the overall relevance of the recommendation lists.

In summary, the contributions of this work are three folded:

- We introduce a ranking-oriented approach for recommender system AdaMF, which is an adaptive boosting algorithm that combines component recommenders built by simple Matrix Factorization Models. The combination insures the ranking performance and decreases the generalization error.
- We propose a gradient descend algorithm for training component recommenders. The algorithm is ease in implementation and efficient in choosing parameters.
- We conduct a validation towards the efficiency and performance of AdaMF. The experimental results confirm the effectiveness and show an outstanding increase from the state-of-the-art approaches.

This paper is organized as follows: In Section 2, we introduce some related works for our work. In Section 3 we present our AdaMF algorithm. Some experiment and comparison are described in Section 4. We conclude our study of AdaMF at Section 5.

2 Related Work

2.1 Matrix Factorization

Matrix Factorization model uses low dimensional vectors to represent the feature for users and items. The MF model has high scalability and accuracy, and gets

high success in Netflix challenge [1]. During the competition, more elements are considered to enhance the factor vector. For example, traditional neighbor based collaborative filtering is combined with MF model [7]. Temporal dynamic of users' taste and items' timeliness is also important in modeling. In temporal approaches of MF model, function based temporal dynamic factors have proven its effectiveness by presenting the latent factor vector as linear function of the time stamp [8]. MF can be viewed as graphical models in some probabilistic approaches of recommender system [9]. Latent Dirichlet Allocation [10] has been used to build hidden topic feature variables of users and items, thus Probabilistic Matrix Factorization model can be combined to improve performance and solve cold start problem [11].

2.2 Learning to Rank

Traditional ranking algorithm generates a list by sorting the relevance function of query and document. With the development of linguistics model and more accumulation of labeled data from search engine, researchers try to make machines 'learn' the ranking model.

LTR is a series of supervised learning algorithm to learn better ranking model from the labeled documents and feedbacks of users. LTR methods are categorized as Point-wise, Pair-wise and List-wise [3] models. Point-wise approaches predict the score of a document to the query by regression methods in machine learning. Pair-wise ranking models use classification to learn the relative preference of each item pair, classification methods like SVM [12] can be adapted into solving the problem. In recent years List-wise algorithms have drawing much attentions. The key point of List-wise LTR is to define loss function based on the whole result list for each query. The simple intuition is to use performance metrics in information retrieval such as MAP, NDCG, and so on, but these metrics are associated with the rank of documents thus they are discontinuous with model's parameters, therefore different smoothing technology for metrics lead to various approaches [13].

Based on the direct optimization of the performance metric, AdaRank [6] uses boosting technique to solve the problem. The basic idea of AdaRank is to employ an exponential loss function of performance metrics. It generates a strong ranker which linearly combines weak rankers, where the coefficients are computed by the performance of each ranker. So that different weak rankers are correlated according to each one's performance. To create different weak rankers, AdaRank maintains a distribution of weights over the queries in the training data. The distribution is determined according to the performance of strong ranker on each query. AdaRank increases the weights on the queries that are not ranked well by the strong ranker. For each round in the training, AdaRank trains a new weak ranker according to the weight distribution, and the newly trained weak ranker is appended to the strong ranker.

2.3 Ranking-Oriented Recommender System

Learning to Rank has made contribution to a wide variety of applications. One of the mainly used areas is in recommender system [4,5]. In Ranking-Oriented recommender systems, several ranking-based model have been built. In [4], Balakrishnan et.al. propose to take the trained latent factor of each user and item as feature vector. The extra model parameters are added for building point-wise and pair-wise model. They denote as Collaborative Ranking. List-wise LTR algorithm was also used to build MF model. Shi et.al. propose ListRank-MF [5], which optimize the loss function which is conducted based on the top one probability of item when given a predicted score. However, Collaborative Ranking proposes an EM-like training process, which cost more time on model training. Ranking based loss function in ListRank-MF also increases the complexity in gradient descent algorithm.

3 Adaptive Boosting Matrix Factorization

In this section we present our model AdaMF which combines Matrix Factorization with AdaRank, that provides a boosting technique for conducting the ranking-oriented recommender system. Firstly, we present our problem definition. Secondly, we introduce Matrix Factorization for recommender system and the framework of our AdaMF model. Finally, we discuss the advantages of combining Matrix Factorization with AdaRank.

3.1 Problem Definition

The objective of the recommender system is to learn the preference model from users' rating history to generate a top-N recommendation list. Suppose that there are N ratings given by U users to I items. For each user u, R_u is the set of rated items $\{R_{ui}\}_{i=1}^{N_u}$ where N_u is the size of R_u. The average of N_u is simplified to n. The rating given by user u to item i is denoted as R_{ui}. The goal of our algorithm is to create a recommender $f(u, i)$ to rank the unrated items for users.

3.2 Matrix Factorization

In the MF model [1], the prediction of the unobserved rating of item i given by user u is formulated as:

$$\hat{r}_{ui} = \sum_{k=0}^{K} P_{uk} Q_{ik} \tag{1}$$

where P_u is the latent factor vector of user u to reflect the interests of user u to each latent factor, similarly Q_i shows the relevance of item i with the latent factors, and K is the dimension of the latent factor.

The traditional approach of MF model is built for the prediction of missing ratings. To measure the performance of the accuracy of prediction, RMSE is one

of the most commonly used evaluation metric. The RMSE-based loss function is defined as:

$$L = \sum_{u=1}^{U} \sum_{i \in R_u}^{N_u} (R_{ui} - \hat{r}_{ui})^2 + \frac{\lambda}{2}(||P||_F + ||Q||_F) \tag{2}$$

where $\hat{r}_{ui}$ denotes the predicted rating given by (1). In the loss function, $||\cdot||_F$ is Frobenius norm, which is used as the normalized term to generalize the model. Once the latent vectors have been learned, the unobserved ratings of each user can be estimated by (1). Intuitively, the recommendation list is generated by sorting $\hat{r}_{ui}$ for unrated items of each user.

3.3 AdaMF

Similar to AdaBoost, the AdaMF maintains a weight distribution on each user in the training set. The training process contains T rounds in all. In the t^{th} round of training, the algorithm builds a component recommender P_u^t, Q_u^t by MF according to the weight distribution on users. The distribution reflects the focal point of the component recommender. During the learning, weights on the users with low training performance are increased so that the component recommender of next round would be forced to focus on users with bad fitting. A coefficient α_t of the component recommender is calculated according to the performance metric. Then the newly trained recommender is integrated to the ensemble recommender $f(u, i)$. New distribution for each user is updated after testing the performance of the ensemble recommender $f(u, i)$ on the training set.

In recommender system, the user's rating on items can be labeled by different level, such as one to five star scale. A metric that measures both multi-level relevance and position information is required to measure the ranking performance. For the reason of better representing the ranking performance, we use NDCG [14] measurement to measure the performance of the recommenders. Specifically, NDCG is defined as:

$$NDCG(u)@m = Z_u \sum_{j=1}^{m} \frac{2^{r_{u,j}} - 1}{log(1 + j)} \tag{3}$$

where $r_{u,j}$ is the real rating data given by user u to the item at the j^{th} position of recommendation list. Z_u is the normalized parameter for the user u, so that the perfect ranker will get NDCG of 1. In the information retrieval domain, $r_{u,j}$ stands for the relevance of document on j^{th} position to a certain query. For our recommender problem, the rating of user given to the item can be viewed as the relevance. From the definition of NDCG, we can summarize that both various relevance level and position information is taken into consideration. Thus, we use NDCG for the calculation of the coefficient α_t. The AdaMF algorithm is shown in Algorithm 1.

AdaMF builds a strong recommender $f(u, i) = \sum_{t=1}^{T} \alpha_t P_u^{tT} Q_i^t$. The recommender list can be created by sorting $f(u, i)$ of the unrated items for each

Algorithm 1. AdaMF

Input: Rating history $\{r_{ui}\}$
Output: MF ranker $f(u,i)$

1. Initialize $D_1(u) = 1/U, f^0(u,i) = 0$
2. **for** $t = 1$,$t <= T$,$t++$ **do**
3. $P_u^t, Q_u^t = ComponentRecommender(D_t(u), \{r_{ui}\})$;
4. Sort the observed items for each user by the component recommender $\hat{r}_{ui} = \sum_{k=0}^{K} P_{uk}^t Q_{ik}^t$;
5. Calculate the current component recommender performance $NDCG(u)@m_c^t$
6. Choose α_t:

$$\alpha_t = \frac{1}{2} * ln \frac{\sum_{u=1}^{U} D_t(u)(1 + NDCG(u)@m_c^t)}{\sum_{u=1}^{U} D_t(u)(1 - NDCG(u)@m_c^t)} \tag{4}$$

7. Create ensemble recommender $f^t(u,i)$:

$$f^t(u,i) = f^{t-1}(u,i) + \alpha_t P_u^{tT} Q_i^t \tag{5}$$

8. Sort the observed items for each user by the ensemble recommender $f(u,i)$
9. Calculate the ensemble recommender performance $NDCG(u)@m_e^t$
10. Update $\mathbf{D_{t+1}}$ with $NDCG(u)@m_e^t$ of the ensemble recommender:

$$D_{t+1}(u) = \frac{exp\{-NDCG(u)@m_e^t\}}{\sum_{j=1}^{U} exp\{-NDCG(u)@m_e^t\}} \tag{6}$$

11. **end for**
12. $f(u,i) = f^T(u,i)$
13. **return** $f(u,i)$;

user. Theoretical analysis shows that the adaptive boosting approaches can continuously improve the performance on the training set until reach the upper bound [6].

3.4 Component Recommender

In the original AdaRank, similar to AdaBoost, the weak ranker is conducted with a simple approach, only one feature is chosen to build the weak ranker. This approach can build the weak models which generate relatively low performances. However as mentioned by Li et.al. in [15], AdaBoost can get better generalization result with correlated *strong classifiers*. They present AdaBoostSVM by taken Support Vector Machine as the component classifier. We note that SVM is viewed as a strong classifier. In collaborative filtering, we utilize Matrix Factorization as the component recommender model. Matrix Factorization can easily achieve a good performance in rating prediction and recommender ranking. We can view MF as a relatively *strong recommender*.

One of the problem when building recommender model is the generalization problem. For matrix factorization, the single model would meet overfitting easily

when the training gets deeper, as in usual, the latent factor vector's dimension would be close to the real user's rating number. The number of model parameter is larger than traditional model-based approaches. The common solution for solving generalization problem is to add the normalized term on the loss function, as illustrated in (2). This would solve the generalization problem to some extent. However, user's multi-faceted interest cannot be easily modeled by single model with normalization. AdaMF provides an adaptive voting mechanism for different component recommenders to determine the final result together. Different faceted models are correlated. The generalization is maintained by the combination of different component recommenders.

Similar with AdaBoost's robustness toward overfitting, AdaMF can rarely meet dilemma in generalization when more component recommenders are combined. Moreover, if the component recommender is generalized by the additional normalized term, the component recommender cannot reflect the preference over the adjusted user weight distribution. Therefore the component recommenders need to prune the normalized term.

AdaMF gives weight distribution on users which depends on the performance of current component recommender, the next component recommender model must fit the distribution. The current model should fit users with higher weights, i.e. minimize the errors on higher weighted users.

In summary we define the loss function of component recommender as:

$$L(P,Q) = U \sum_{u=1}^{U} D(u) \sum_{i \in R_u}^{N_u} (R_{ui} - P_u^T Q_i)^2 \tag{7}$$

From this loss function, we can generate derivatives of each P_u and Q_i, then the parameters can be learned by Stochastic Gradient Descend (SGD). The training algorithm of component recommender is shown in Algorithm 2. η stands for the learning rate during the training, we set $\eta = 0.01$ in our study. Notice that if D(u) equals to $\frac{1}{U}$ for every user u, the component recommender equals to the basic MF model.

Algorithm 2. ComponentRecommender

Input: Distribution $\mathbf{D_t}$, Rating history $\{r_{ui}\}$
Output: Latent Factor Vector P_u, Q_i

1. Initialize P_u^0, Q_i^0 randomly, initialize learning rate η, iteration rounds I
2. **for** $n = 1, n <= I, n++$ **do**
3. **for** each r_{ui} **do**
4. $err = r_{ui} - P_u^{n-1T} Q_i^{n-1}$;
5. $P_u^n := P_u^{n-1} + \eta(UD(u)err * Q_i^{n-1})$;
6. $Q_i^n := Q_i^{n-1} + \eta(UD(u)err * P_u^{n-1})$;
7. **end for**
8. **end for**
9. **return** P_u^I, Q_i^I;

3.5 Discussion

AdaMF is easy to be implemented and can be trained fast with high accuracy. Since the complexity of training component recommender is $O(K \cdot T \cdot (I \cdot N + U \cdot m \cdot nlogn))$. The algorithm's cost linearly increases with the scale of case. One of the advantage of AdaMF is that it combines component recommenders with various focal point to prevent the overfitting problem. We think that in real world data with more rating history, different aspects of users' preference are covered, in this occasion our model may show its effectiveness.

4 Experiment

In this section, we conduct experiments to evaluate the effectiveness of our method and make comparison with two baseline methods.

4.1 Experimental Setup and Datasets

We use MovieLens-100K[1] dataset to conduct our experiment. MovieLens-100K dataset contains 100K ratings(scale 1-5) given by 943 users to 1682 movies. In our conduction, 10, 20, 50 ratings are randomly chosen from each user to build a training set, and the remaining ratings are gathered for the testing set. For each condition, the chosen user must have 20, 30, 60 ratings so that we can test NDCG@10 on the testing set. In addition, we build a mixed length dataset. In reality, users' rating number varies in a large range, and only a few active users would contribute most of the rating feedbacks. Therefore, to involve the different activeness from different user, we select 50, 20, 10 ratings for users with more than 60, 30, 20 ratings then put the remain ratings into testing sets. In this way we can evaluate the adaptiveness of our method to various users' activeness and stickiness. In the following sections, we use 10-set, 20-set, 50-set, mix-set to denote these datasets representatively.

Table 1. Data sets statistic

	10-set	20-set	50-set	mix-set
Training Rating number	9430	14480	24850	31780
Test Rating number	90570	85520	75150	68220
User number	943	724	493	943

[1] `http://www.grouplens.org/node/73`

Table 2. Statistic on test set

	10-set	20-set	50-set	mix-set
Users' rating number in [0,20]	213	106	51	356
Users' rating number in [20,50]	236	192	85	226
Users' rating number in [50,100]	172	144	135	135
Users' rating number more than 100	322	302	226	226

4.2 Improvement by Iteration

Since AdaMF combines different weak recommenders, the overall performance would get improved with the increasing rounds of combination. In this experiment we test our AdaMF on four training sets to validate the improvement of NDCG@10 when increasing the iteration rounds number T. In specially, we choose T in 1, 2, 3, 4, 5, 7, 10, 15, 20. For each weak recommender we set the latent factor dimension K to be 10, learning rate as 0.01 and iteration upper bound T_w to be 5 rounds. For each condition, we run the algorithm for 10 times and compare the mean and variance of NDCG@10.

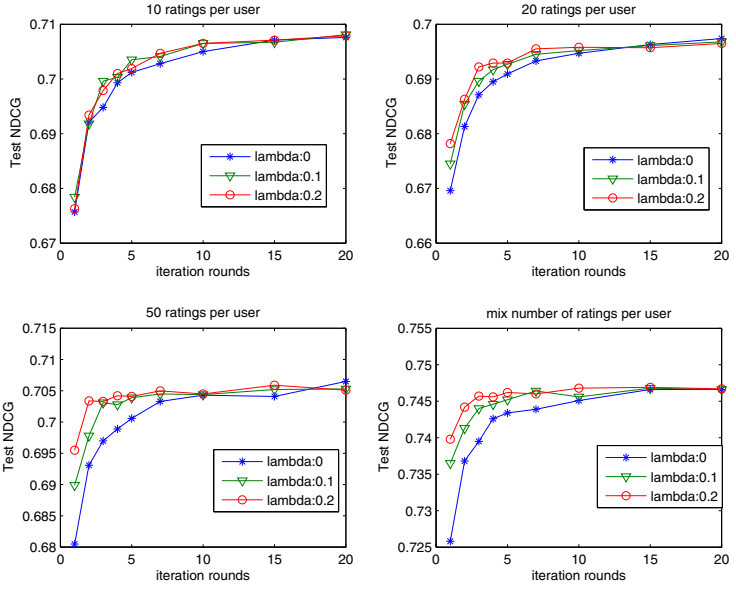

Fig. 1. The performance curve of AdaMF

Fig.1 shows the performance of AdaMF. It can be observed that after the combination of weak recommenders, the original MF model is enhanced in terms of accuracy. It is obvious from the curve that AdaMF reaches its best performance after about 10 rounds of iteration.

In traditional MF model, the regularization coefficient λ controls the degree of overfitting on the training set. For the single MF model, good performance could be achieved by selecting parameters from Cross-Validation, and this may take a great quantity of works. However as expressed in Fig.1, when setting different parameters in a certain range,, AdaMF would converge to the best performance, thus we do not need to pay too much attentions to the parameter selection.

Not only does AdaMF improve the accuracy of recommendation, it also decreases the uncertainty caused by randomly initialized factor value. We use the variance of the results in 10 time running to express the uncertainty. From Fig.2, it is evident to observe that the variance of NDCGs decreases with the increasing of iteration number T. The result confirms the expectation that AdaMF reduces the uncertainty from the initial value of the model when increasing the round of iterations.

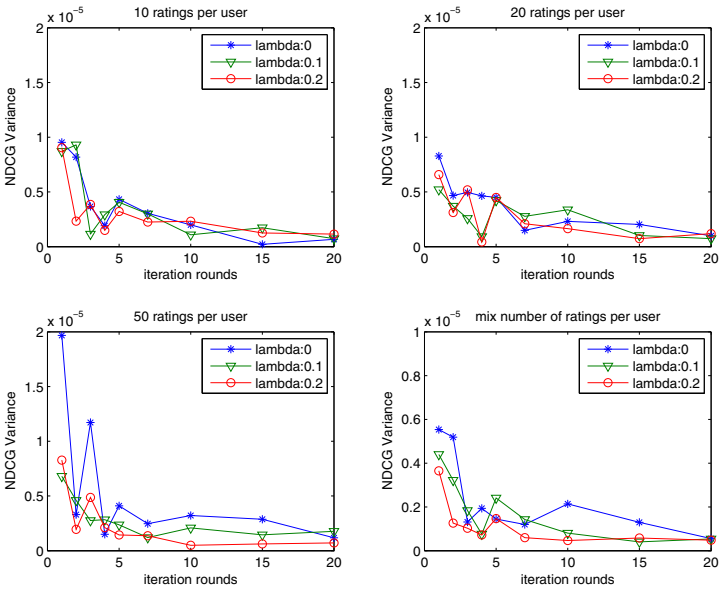

Fig. 2. The variance curve of AdaMF

4.3 Ranking Result Comparison

In this subsection we evaluate the ranking performance by making a comparison for AdaMF with two baseline methods. We choose ListRank-MF [5] and the basic MF model to be the baseline. ListRank-MF is a state-of-the-art ranking-oriented recommendation model based on List-wise learning to rank approach, which uses top one probability for the items to build the loss function. We choose the parameters of baselines by reach their best performance. For each algorithm, we run for 10 times on different dataset to compare the average NDCG@10 metric.

From the results presented in Table 3, we observe that AdaMF performs better on all of four datasets. Especially for the mix-set, AdaMF shows a gratifying improvement in comparison to the ListRank-MF model and the basic MF model.

Table 3. The NDCG@10 comparison between ListRank-MF, Basic MF and AdaMF

	10-set	20-set	50-set	mix-set
ListRank-MF	0.6968	0.6861	0.6875	0.7309
Basic MF	0.6784	0.6745	0.6899	0.7365
AdaMF	0.7065	0.6961	0.7043	0.7464

It can be manifested from Table 3 that ListRank-MF model performs better on 10-set and 20-set. MF shows its disadvantage when meeting a sparse rating matrix, thus the model would easily get overfitted. However AdaMF delivers a 1.43% increase to ListRank-MF. Such result reflects the superiority of AdaMF when facing the sparsity. The composition of weak MF models can prevent overfitting on the training set.

With the grow of matrix density, higher RMSE can somehow transform into better ranking result, therefore basic MF presents a similar result with ListRank-MF on the 50-set and mix-set. AdaMF can take the advantage of different MF models which fit different users better, so that AdaMF can enhance the performance of single MF model. For the 50-set and the mix-set, AdaMF improves the basic MF model by 1.73%.

5 Conclusions and Future Work

In this paper, we have introduced the AdaMF, a boosting algorithm for ranking-oriented recommender system. Optimization for NDCG is done by the boosting of the component recommenders which trained by different weight distribution for each user. The model offers several advantages: ease in implementation, low complexity in training and high performance in ranking. Experimental results demonstrate the effectiveness and accuracy of our model. In comparison with ListRank-MF and basic MF model, we show the superiority of AdaMF. In the future, we are going to focus on the the distributed training algorithm by adaptively composing component recommenders that are trained on different processers in parallel, so that the training process can be easily extended to the multi-processer environment.

Acknowledgments. This work was supported partly by National Natural Science Foundation of China (No. 61300070, No. 61103031), partly by China 863 program (No.2013AA01A213), China 973 program (No.2014CB340305), partly by the State Key Lab for Software Development Environment (SKLSDE-2013ZX-16), partly by A Foundation for the Author of National Excellent Doctoral Dissertation of PR China(No. 201159), partly by Beijing Nova Program(2011022) and partly by Program for New Century Excellent Talents in University.

References

1. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer 42(8), 30–37 (2009)
2. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In: Proceedings of the Fourth ACM Conference on Recommender Systems, RecSys 2010, pp. 39–46. ACM, New York (2010)
3. Liu, T.Y.: Learning to rank for information retrieval. Found. Trends Inf. Retr. 3(3), 225–331 (2009)
4. Balakrishnan, S., Chopra, S.: Collaborative ranking. In: Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, WSDM 2012, pp. 143–152. ACM, New York (2012)
5. Shi, Y., Larson, M., Hanjalic, A.: List-wise learning to rank with matrix factorization for collaborative filtering. In: Proceedings of the Fourth ACM Conference on Recommender Systems, RecSys 2010, pp. 269–272. ACM, New York (2010)
6. Xu, J., Li, H.: Adarank: a boosting algorithm for information retrieval. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2007, pp. 391–398. ACM, New York (2007)
7. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2008, pp. 426–434. ACM, New York (2008)
8. Koren, Y.: Collaborative filtering with temporal dynamics. Commun. ACM 53(4), 89–97 (2010)
9. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In: NIPS (2007)
10. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. 3, 993–1022 (2003)
11. Agarwal, D., Chen, B.C.: flda: matrix factorization through latent dirichlet allocation. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, WSDM 2010, pp. 91–100. ACM, New York (2010)
12. Herbrich, R., Graepel, T., Obermayer, K.: Large margin rank boundaries for ordinal regression. MIT Press, Cambridge (2000)
13. Chapelle, O., Wu, M.: Gradient descent optimization of smoothed information retrieval metrics. Inf. Retr. 13(3), 216–235 (2010)
14. Järvelin, K., Kekäläinen, J.: Ir evaluation methods for retrieving highly relevant documents. In: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2000, pp. 41–48. ACM, New York (2000)
15. Li, X., Wang, L., Sung, E.: AdaBoost with SVM-based component classifiers. Engineering Applications of Artificial Intelligence 21(5) (2008)