Proceedings of the 3rd International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/6/20230342

AutoDebias based on NeuralCF

Meng Li[1,5,+], **Zhuoya Gu**[2,6,+], **Haolun Zheng**[3,7,+] **and Yening He**[4,8,+]

[1]Department of Science,University of Auckland,Auckland,1010, New Zealand
[2]Department of Computer Science,Xidian University,Xi'an, 710126, China
[3]School of Communication Engineering,Xidian University, Xi'an, 710126, China
[4]College of Computer Science and Technology,College of Software,Zhejiang University of Technology,Hang Zhou, 310023,China

[5]mli561@hotmail.com
[6]yadeemail@163.com
[7]hz2016@hw.ac.uk
[8]1521310090@qq.com
[+]These authors contributed equally to this work and should be considered co-first authors

Abstract. Recommendation systems rely on user behavior data (ratings, clicks, etc.) to build personalized models. Nevertheless, the data collected was always observational rather than experimental, which can lead to a variety of biases. Data deviations will directly affect the operation results of the recommendation system. Users may not favor the items recommended by the system to them, so this is an imperative problem that must be resolved as soon as possible. The majority of current efforts to debias recommendation systems focus only on one or two specific biases and are not universally applicable. As a result, Chen et al. and their group developed a general debiasing framework (AutoDebias) containing most of the debias strategies, which overcomes the problem of low flexibility of most of today's debiasing techniques. However, the model still has room for improvement. Chen et al. chose Matrix Factorization (MF) as the benchmark model for AutoDebias. In our study, however, we found that NeuralCF model may be a better alternative to MF. Therefore, the goal of our work is to replace the MF base model in AutoDebias with the NeuralCF model, and to conduct relevant experiments in order to validate our hypothesis. According to experiments, when using NLL as evaluation metrics, NeuralCF has better performance than MF, which improved 14.3% in final results.

Keywords: AutoDebias, MF Model, NeuralCF Model.

1. Introduction

Recommendation systems (RS) have been throughout used in a wide variety of online utilizations due to their ability to make personalized suggestions for each individual user. Recently, the field of recommendation systems has developed rapidly, and most relevant research focuses on inventing machine learning models to fit user behavior data [1,2,3]. In practice, however, data tend to be observational rather than experimental, which results in bias in the data. Bias in data can be broadly divided into four types: selection bias [4,5], exposure bias [6,7], conformity bias [8,9] and position bias

[10,11]. It is possible to make predictions that are very different from reality if we fit the data blindly without taking into consideration these biases.

Existing work to resolve the recommendation model bias can be divided into three types:

- Data imputation [4,12]: To reduce variance, missing data is assigned pseudo notations.
- Inverse propensity scoring (IPS) [7,13]: For expectation-unbiased learning, collected data is reweighted in a counterfactual manner.
- Generative modeling [14,15]: Bias is reduced by assuming the process of data generation.

These efforts, however, have two limitations:

- Lack Universality: In spite of the fact that these methods are widely applicable to specific scenarios that contain one or two biases, they are insufficient for data that contain multiple bias types.
- Lacking Adaptivity: In order for these methods to be effective, the bias configuration must be specified. However, it is actually quite difficult to obtain the correct bias configuration. Additionally, the optimal configuration changes over time as the data is continually updated. Manual adjustment is extremely inefficient.

To address these issues, Chen et al. developed a universal debiasing approach that is not only applicable to manifold biases and their assortments, but also allows users to self-identify biases and modify configurations according to their own learning abilities. Chen et al. presented two findings from their review of common bias types and de-biasing methods:

- It is possible to formulate all types of deviations as the difference between empirical and real risks, due to the difference between the distribution for which training data are gathered and the distribution used for unbiased testing.
- Recent debias methods depend on offsetting differences in model training.

On the basis of these findings, Chen et al. proposed a general debias framework by reducing risk differences, which provides an opportunity to develop a general recommendation debias solution by learning the debias parameters of the framework to achieve automatic debias.

As a result, optimizing the depolarization parameters becomes the key challenge. Based on two-level optimization, Chen et al. used the depolarization function as the hyperparameter of the learning recommendation model and optimized the depolarization parameters through meta-learning [16].

Chen et al. used the MF model as the benchmark for their AutoDebias algorithm. However, our team found that in the case of NLL as a benchmark, using NCF rather than MF might be a better choice as a benchmark model. We replaced MF with NCF by modifying the code and demonstrated that the self-debias algorithm based on NCF has high debias ability under the NLL evaluation benchmark.

2. MF model

2.1. Basic concepts of MF

According to the MF algorithm, a hidden vector is generated for each user and video, which is used to locate the user and video in the representation space of the hidden vector (see Figure 1). Similar distances between users and videos indicate similar interest characteristics. It is recommended that videos with similar distances be recommended to target users during the recommendation process. Accordingly, if you want to recommend videos to the user Victor in Figure 1, and you discover that the two video vectors closest to Victor are "Forrest Gump" and "The Truman Show", you can generate a recommendation list for Victor based on the order of near to far.

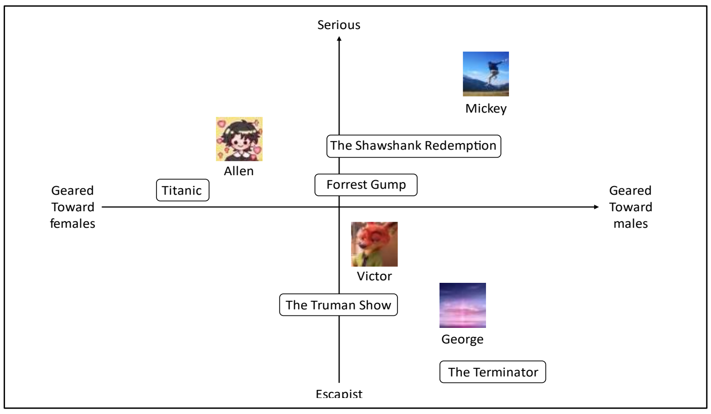
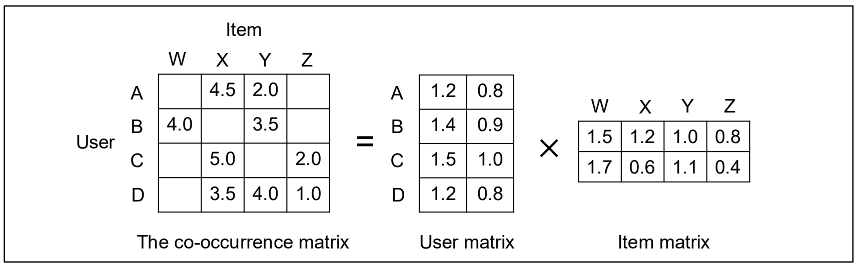


Figure 1. Schematic diagram of MF (matrix factorization) algorithm.

Keep similar users and items close to each other using hidden vectors to represent users and items. It sounds like a good idea, but the key challenge is how to obtain such implicit vectors.

According to the algorithm framework of "MF", hidden vectors of users and items are obtained through a co-occurrence matrix generated by decomposition and collaborative filtering [17] (as shown in Figure 2), which is also the explanation for the name "MF (matrix factorization).



$$\begin{array}{c|cccc} & W & X & Y & Z \\ \hline A & & 4.5 & 2.0 & \\ B & 4.0 & & 3.5 & \\ C & & 5.0 & & 2.0 \\ D & & 3.5 & 4.0 & 1.0 \end{array} = \begin{array}{c|cc} A & 1.2 & 0.8 \\ B & 1.4 & 0.9 \\ C & 1.5 & 1.0 \\ D & 1.2 & 0.8 \end{array} \times \begin{array}{cccc} W & X & Y & Z \\ \hline 1.5 & 1.2 & 1.0 & 0.8 \\ 1.7 & 0.6 & 1.1 & 0.4 \end{array}$$

Figure 2. The process of marix factorization.

The MF algorithm decomposed the $m \times n$ dimension co-occurrence matrix R into $m \times k$ dimension user matrix U and $k \times n$ dimension item matrix V. Where m is the number of users, n is the number of items, and k is the dimension of the hidden vector. The value of k determines the expression ability of implicit vector. The smaller the value of k is, the less information the implicit vector contains, and the higher the generalization degree of the model is. On the contrary, the larger the value of k is, the stronger the expression ability of implicit vector is, but the generalization degree decreases accordingly. In addition, the value of k is directly related to the solution complexity of matrix decomposition. In the specific application, the value of k needs to find a balance between the recommendation effect and the project cost through several tests.

2.2. *Gradient descent*

A method of MF is Gradient Descent, which was used in AutoDebias. (1) is the objective function for solving the matrix decomposition. The objective of this objective function is to minimize the difference between the original score r_{ui} and the product $q_i^T p_u$ of the user vector and the item vector, so as to preserve the original information of the co-occurrence matrix to the maximum extent.

$$\min_{q^*,p^*} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 \tag{1}$$

Where K is the collection of all user rating samples. The objective function after adding the regularization term [18] is shown in (2) in order to reduce the overfitting phenomenon [19].

$$\min_{q^*,p^*} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 + \lambda\left(||q_i||^2 + ||p_u||^2\right) \tag{2}$$

Gradient descent can be used to solve the objective function shown in (2).

Step 1: As shown in (2), determine the objective function.

Step 2: In order to determine the direction and amplitude of gradient descent, take the partial derivative of the objective function.

According to the partial derivative of (2) with respect to q_i, the result is:

$$2\left(r_{ui} - q_i^T p_u\right)p_u - 2\lambda q_i \tag{3}$$

The partial derivative with respect to p_u is:

$$2\left(r_{ui} - q_i^T p_u\right)q_i - 2\lambda p_u \tag{4}$$

Step 3: Using the results of Step 2, the parameter is updated in the opposite direction of the gradient.

$$q_i \leftarrow q_i - \gamma(\left(r_{ui} - q_i^T p_u\right)p_u - \lambda q_i) \tag{5}$$

$$p_u \leftarrow p_u - \gamma(\left(r_{ui} - q_i^T p_u\right)q_i - \lambda p_u) \tag{6}$$

Where, γ is the learning rate.

Step 4: When the number of iterations exceeds the upper limit n or the loss is lower than the threshold θ, the training is ended, otherwise the Step 3 of the cycle is completed.

The hidden vectors for all users and items are obtained following the matrix decomposition process. The implicit vector of a user and the implicit vector of all items can be used to perform inner product operations one by one to obtain the user's score prediction for all items, and order successively to generate the final recommendation list.

2.3. Review of MF model from the perspective of deep learning

A primary function of embedding is to transform sparse vectors into dense vectors. When viewed from a deep learning perspective, a matrix decomposition model makes sense. Therefore, the implicit vectors of the matrix decomposition layer can be considered to be embedded vectors. The final "Scoring layer" is the inner product operation of the user's hidden vector and the item's hidden vector to calculate "similarity", where "similarity" is the prediction of the score. Figure 3 illustrates the framework of the matrix decomposition model using deep learning network graphs.

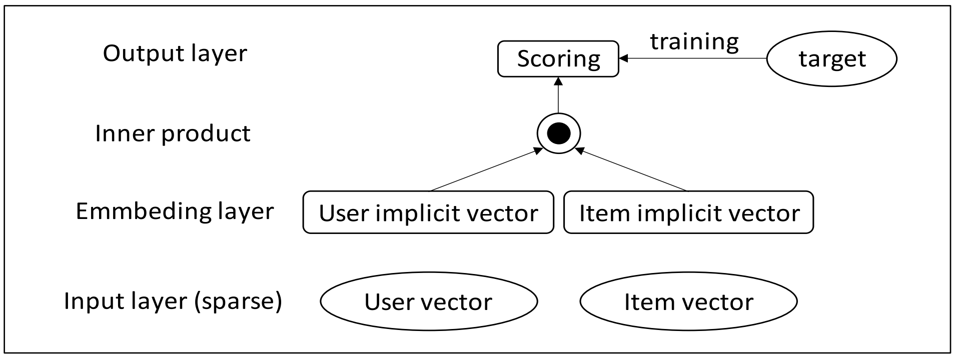


Figure 3. Networked representation of matrix decomposition.

As MF is trained and evaluated, it is often found that it is prone to underfitting. MF is relatively simple in its model structure, especially the "output layer" (also known as the "Scoring layer"), which cannot effectively fit the optimization target. In order to accomplish this, the model must be more expressive. This motivation led researchers at the National University of Singapore to propose the NeuralCF (NCF) model.

2.4. NCF model

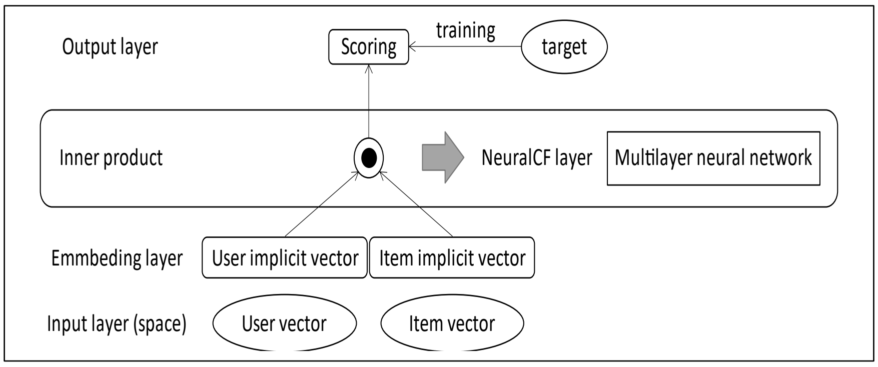


Figure 4. From MF to NCF.

NeuralCF replaces the simple inner product operations of the matrix factorization model with a "multilayer neural network + output layer" structure. It is intuitive that this would be beneficial. The first benefit is that the user vector and the item vector can be crossed more fully, and more valuable feature combination information can be obtained. Secondly, the model is made more expressive by adding more nonlinear features.

As a result, the interoperable layer of user and item vectors can be replaced by any form of interoperable form, which is referred to as Generalized Matrix Factorization.

In the original matrix decomposition, "inner product" is used to allow users to interact with vectors of items. For a further cross-section of vectors in all dimensions, the element-wise product method can be used (two vectors with the same length are multiplied to obtain a vector of the same length). The final prediction target is then fitted using logistic regression and other output layers. As a generalized form of interoperability, neural networks are used to fit interoperable functions in NeuralCF.

Moreover, feature vectors obtained through different interoperable networks can be spliced together and delivered to the output layer for target fitting. As a result, the model has a stronger feature combination and nonlinearity.

A model framework is presented in the NeuralCF model. As a result of the two embedded layers of the user vector and object vector, different interop layers are used to combine features and to flexibly splice different interop layers. Thus, we can see the advantages of deep learning when it comes to building recommendation models, namely the ability of neural networks to fit any function theoretically, to combine different features flexibly, and to increase or decrease the complexity of a model as required.

3. Experiment

In this section, we conducted experiments to evaluate the performance of our proposed AutoDebias with NCF.

3.1. Experimental setup

To test the universality of AutoDebias based on NeuralCF, we examine three types of data: explicit feedback, implicit feedback, and feedback on recommendation lists.

3.1.1. Dataset

As part of our experiments, we used two publicly available datasets (Yahoo!R3 and Coat) as sources of explicit feedback. There was a biased set of data included in both datasets that was based on the normal interactions of users on the platform. Additionally, they contained a small set of unbiased data derived from a stochastic experiment in which items were assigned at random. Biased data was referred to as the training set DT, while unbiased data was divided into three parts: 5% for uniform set DU to assist in the training, 5% for validation set DV to tune hyperparameters, and 90% for test set DT e to assess the model. Furthermore, we employed a synthetic dataset simulation to test whether AutoDebias is capable of dealing with both positional and selectional biases.

3.1.2. Evaluation metrics

We adopted the following metrics to evaluate recommendation performance:

- NLL
- AUC
- NDCG@k

3.1.3. Implementation details

MF and NCF had been selected as the two benchmark recommendation models for experiments. During the optimization of the base model, SGD was used, while Adam was adopted for optimizing the meta model. The best hyper-parameters were identified using grid search based on the performance on the validated set. We optimized the base model with SGD and meta model with Adam. The search space of learning rate and weight decay are [1e-4,1e-3,1e-2,1e-1].

3.2. Performance comparison on explicit feedback

3.2.1. Baseline

- MF(biased), MF(uniform) and MF(combine): the basic matrix factorization model that trained on DT , DU , and $DT + DU$, respectively;
- Inverse propensity score (IPS): An analysis of the collected data reweighed in a counterfactual manner. We calculate the propensity with naive bayes;
- Doubly robust (DR): Inverse propensity score combined with data imputation;
- KD-Label: the state-of-the-art method that transfers the unbiased information with a teacher model. The best distillation based on label is selected for comparison.
- CausE: Using an additional alignment term to distill unbiased information.
- AutoDebias(MF): Using AutoDebias algorithm based on MF model.

3.2.2. Performance comparison

Table 1. Comparison of performance based on explicit feedback data. The bold font indicates the winner in each column.

Explicit feedback			
Yahoo!R3			
Method	NLL	AUC	NDCG@5
MF(biased)	-0.587	0.727	0.547
MF(uniform)	-0.503	0.568	0.436
MF(combine)	-0.579	0.729	0.551
IPS	-0.450	0.725	0.556
DR	-0.471	0.725	0.557
CausE	-0.584	0.729	0.554
KD-Label	-0.636	0.740	0.576
AutoDebias(MF)	-0.422	0.740	0.640
AutoDebias(NCF)	-0.416	0.698	0.567

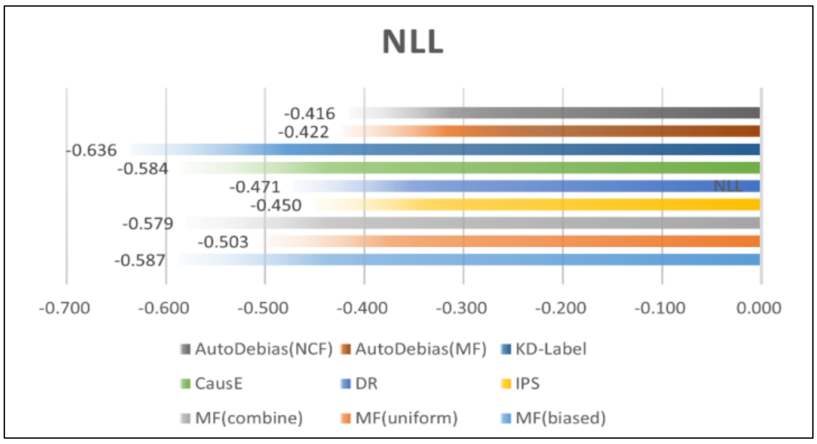


Figure 5. Recommendation performance of the compared methods in terms of NLL.

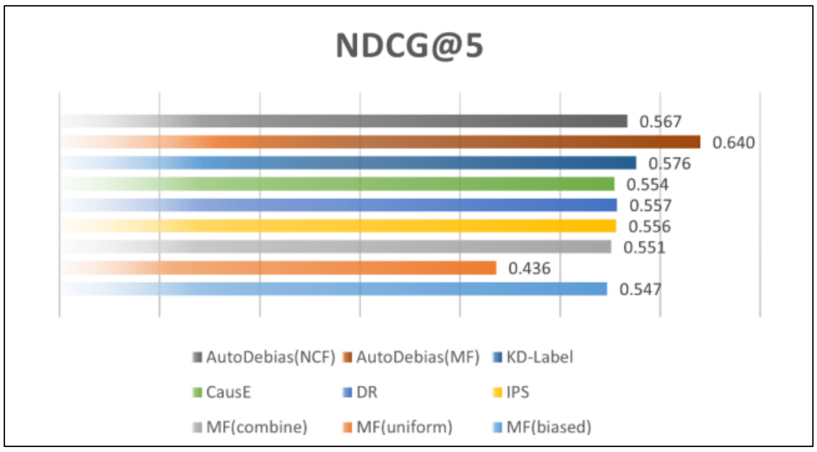


Figure 6. Recommendation performance of the compared methods in terms of NDCG@5.

According to Table 1, Figure 4 and 5, the recommendation performance of the compared methods is described in terms of three evaluation metrics. The boldface font denotes the winner in that column. We have the following observations:

- Among all compared methods, AutoDebias(NCF) outperforms all others in terms of NLL across all datasets. In the dataset Yahoo! R3, the improvement is rather impressive — 14.3% in terms of NLL, but also a bit fall in NDCG and AUC. Our result confirms the ability of AutoDebias with NCF to learn better debiasing configurations than other methods.
- The improvement in NLL is due to the introduction of a two-layer neural network. Instead of matrix factorization (MF), NCF can capture the nonlinear relationship between grading and embedding. Thus, it is reasonable to focus on the improvement of NLL rather than the decline of AUC and NDCG.
- Moreover, the setup of a two-layer network structure prevents the over-fitting problem, thereby improving prediction accuracy.

3.3. *Ablation study*

We conducted an ablation study to determine whether the three parameters $w^{(1)}$, $w^{(2)}$, and m [15] for AutoDebias based on NCF affect the final result. From Table2, we can conclude that the introduction of $w^{(1)}$ and m improves results significantly, even better than AutoDebias which is based on MF. In addition, we can see that introducing $w^{(1)}$ alone negatively affects results, so imputation(m) plays a significant role in determining results. A further improvement is brought about by the introduction of $w^{(2)}$.

Table 2. Ablation Study in terms of NLL.

Method	W1	m	W2	Yahoo!R3
NCF	×	×	×	-0.580
AutoDebias(NCF)-w1	√	×	×	-0.666
AutoDebias(NCF)-w1m	√	√	×	-0.419
AutoDebias(NCF)	√	√	√	-0.415

3.4. Sensitivity test

To determine the sensitivity of our model, we oscillate the three parameters by 10% and 20%. Taking a look at Figure 7, we can see that the model seems to be highly sensitive to imputation, but it is robust to both $w^{(1)}$ and $w^{(2)}$.

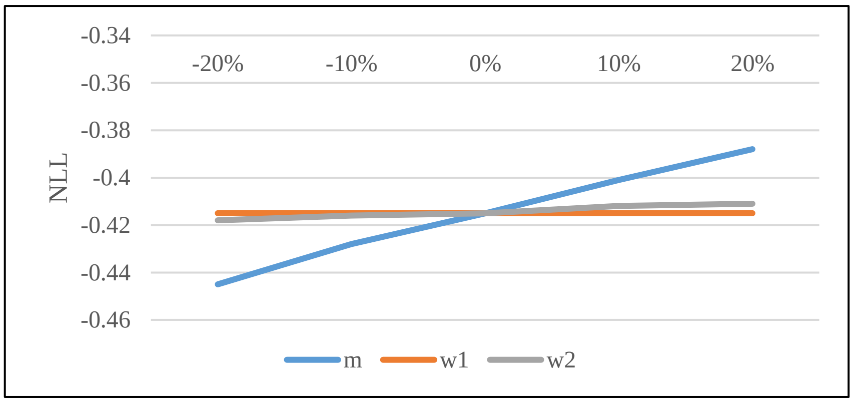


Figure 7. Comparison of performance across different parameter ranges.

4. Conclusion and future work

We tried to replace the MF model in AutoDebias with NCF model and conducted experiments and evaluations. Finally, it is found that under the condition of taking NLL as the evaluation benchmark, the efficiency of Autodebias algorithm taking NCF model as the benchmark model is significantly better than the AutoDebias with MF model.

In the future, we will try to use more evaluation criteria to test the performance of AutoDebias with NCF and compare and analyze it with AutoDebias based on MF. We will continue to study and consider whether there is a better scheme to improve the performance of AutoDebias.

Acknowledgement

Meng Li, Zhuoya Gu, Haolun Zheng and Yening He contributed equally to this work and should be considered co-first authors.

References

[1] He X, Deng K, Wang X, et al. Lightgcn: Simplifying and powering graph convolution network for recommendation[C]//Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval. 2020: 639-648.

[2] Sun F, Liu J, Wu J, et al. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer[C]//Proceedings of the 28th ACM international conference on information and knowledge management. 2019: 1441-1450.

[3] Yuan F, He X, Karatzoglou A, et al. Parameter-efficient transfer from sequential behaviors for user modeling and recommendation[C]//Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2020: 1469-1478.

[4] Hernández-Lobato J M, Houlsby N, Ghahramani Z. Probabilistic matrix factorization with non-random missing data[C]//International Conference on Machine Learning. PMLR, 2014: 1512-1520.
[5] Marlin B, Zemel R S, Roweis S, et al. Collaborative filtering and the missing at random assumption[J]. arXiv preprint arXiv:1206.5267, 2012.
[6] Ovaisi Z, Ahsan R, Zhang Y, et al. Correcting for selection bias in learning-to-rank systems[C]//Proceedings of The Web Conference 2020. 2020: 1863-1873.
[7] Wang X, Bendersky M, Metzler D, et al. Learning to rank with selection bias in personal search[C]//Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. 2016: 115-124.
[8] Krishnan S, Patel J, Franklin M J, et al. A methodology for learning, analyzing, and mitigating social influence bias in recommender systems[C]//Proceedings of the 8th ACM Conference on Recommender systems. 2014: 137-144.
[9] Liu Y, Cao X, Yu Y. Are you influenced by others when rating? improve rating prediction by conformity modeling[C]//Proceedings of the 10th ACM conference on recommender systems. 2016: 269-272.
[10] Joachims T, Granka L, Pan B, et al. Accurately interpreting clickthrough data as implicit feedback[C]//Acm Sigir Forum. New York, NY, USA: Acm, 2017, 51(1): 4-11.
[11] Joachims T, Granka L, Pan B, et al. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search[J]. ACM Transactions on Information Systems (TOIS), 2007, 25(2): 7-es.
[12] Kamishima T, Akaho S, Asoh H, et al. Correcting Popularity Bias by Enhancing Recommendation Neutrality[C]//RecSys Posters. 2014.
[13] Schnabel T, Swaminathan A, Singh A, et al. Recommendations as treatments: Debiasing learning and evaluation[C]//international conference on machine learning. PMLR, 2016: 1670-1679.
[14] Liang D, Charlin L, McInerney J, et al. Modeling user exposure in recommendation[C]//Proceedings of the 25th international conference on World Wide Web. 2016: 951-961.
[15] Chen J, Dong H, Qiu Y, et al. AutoDebias: Learning to debias for recommendation[C]//Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2021: 21-30.
[16] Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks[C]//International conference on machine learning. PMLR, 2017: 1126-1135.
[17] Koren Y, Rendle S, Bell R. Advances in collaborative filtering[J]. Recommender systems handbook, 2022: 91-142.
[18] Goodfellow I, Bengio Y, Courville A. Regularization for deep learning[J]. Deep learning, 2016: 216-261.
[19] Rice L, Wong E, Kolter Z. Overfitting in adversarially robust deep learning[C]//International Conference on Machine Learning. PMLR, 2020: 8093-8104.