Recommendation Algorithms for Optimizing Hit Rate, User Satisfaction and Website Revenue

Xin Wang, Yunhui Guo, Congfu Xu*
Institute of Artificial Intelligence, College of Computer Science
Zhejiang University
cswangxinm@gmail.com, {gyhui, xucongfu}@zju.edu.cn

Abstract

We usually use hit rate to measure the performance of item recommendation algorithms. In addition to hit rate, we consider additional two important factors which are ignored by most previous works. First, we consider whether users are satisfied with the recommended items. It is possible that a user has bought an item but dislikes it. Hence high hit rate may not reflect high customer satisfaction. Second, we consider whether the website retailers are satisfied with the recommendation results. If a customer is interested in two products and wants to buy one of them, it may be better to suggest the item which can help bring more profit. Therefore, a good recommendation algorithm should not only consider improving hit rate but also consider optimizing user satisfaction and website revenue. In this paper, we propose two algorithms for the above purposes and design two modified hit rate based metrics to measure them. Experimental results on 10 real-world datasets show that our methods can not only achieve better hit rate, but also improve user satisfaction and website revenue comparing with the state-of-the-art models.

1 Introduction

Recommender systems have been widely applied to online shopping websites for better user experience. Traditional recommendation algorithms try to leverage item aspects and historical user behaviors to exploit the potential user preferences. Broadly, a customer's behavior can be divided into two types: explicit feedback and implicit feedback. The former, such as rating scores, reveals the preference degree of buyers, and the latter, such as browsing and clicking information, records valuable customer actions. Based on them, most well-known methods utilize the collaborative filtering (CF) technique, which learns the correlations of users and items beneath the feedback, in order to recognize potential user-item associations [Koren *et al.*, 2009]. To measure recommendation performance, researches have also considered designing good metrics. For rating prediction, we usually evaluate the difference between predicted rating values and actual rating values by computing root mean square error (RMSE), mean square error (MSE) or mean absolute error (MAE). For item recommendation, we consider whether the most relevant items are recommended with the highest priority by measuring a ranked item list for each user in terms of precision, recall, normalized discounted cumulative gain (NDCG), area under the curve (AUC), expected reciprocal rank (ERR), etc. [Weimer *et al.*, 2007; Rendle *et al.*, 2009; Shi *et al.*, 2013; Pan and Chen, 2013].

Despite those algorithms and metrics, it is still challenging to comprehensively measure the performance of a particular recommendation algorithm. The rating prediction metrics (e.g., RMSE, MAE, MSE) or some item recommendation metrics on the whole item lists (e.g., NDCG, AUC, ERR) may not directly reflect system performance in applications, where only top-K items are provided. On the other hand, the hit rate based metrics (e.g. precision and recall) cannot convey preference degree information. Hence, in this paper, we consider two other important factors for designing and evaluating recommendation algorithms.

User Satisfaction We define user satisfaction as how much a user satisfies with an item (s)he has already bought. The models with higher hit rate performance may recommend more items that users tend to buy, but they do not guarantee that the proposed items suit the customers. An extreme example is that many users are not pleased with the bought items that recommended by the system. In this case, although the models achieve high hit rate, it is even worse than not recommending the items to these users, because they are disappointed with the items and even distrust the recommender systems in the future.

Website Revenue Another destination of recommender systems is to help website retailers earn more revenue (which is narrowly defined as *retailer satisfaction* for convenience). Unfortunately, it is ignored by almost all the related algorithms. Improving hit rate can indeed help improve sales, but it is limited and indirect. Suppose a user intends to buy one of two selected items, recommending any one of them will gain the same hit rate, but it is better to choose the item with higher profit for it can help the website earn more money without reducing users' satisfaction.

To illustrate the impact of these factors, we show a simple example in Table 1. It contains three similar items that the

*Corresponding author

Table 1: A customer's wish list.

	item1	item2	item3
satisfaction of u	3	2	5
revenue ($)	5	10	9
alg1 (algorithm1)	hit	-	-
alg2 (algorithm2)	-	hit	-
alg3 (algorithm3)	-	-	hit

customer u is interested in. The *satisfaction* denotes the rating value that u will give to each item if s(he) buys it, and the *revenue* is related to item price. The following three lines show the performance of three recommendation algorithms (denoted as alg1, alg2 and alg3), and the *hit* means that u will buy the item if it is recommended. From the table, we immediately know that both alg1, alg2 and alg3 achieve similar performance in terms of hit rate (i.e., the same prediction, recall and F1 score), because they both hit one item. However, those methods have different performance when we consider the other two factors. If u buys the item recommended by alg1, he will give 3 stars to the item, and the retailer will get 5 dollars. If u buys the item recommended by alg2 instead, the user satisfaction value will drop 1 score, but the retailer can get double revenue. To the item recommended by alg3, the user gives the highest score, and the website retailer can get approximate optimal revenue. Therefore, we think that alg3 is the best algorithm because it satisfies both users and the retailer, and alg2 may be better than arg1 because it helps to double the revenue with the cost of only losing 1 level of user satisfaction.

To conclude, a good recommendation algorithm should not only improve hit rate, but also consider optimizing user satisfaction and website revenue, or at least achieve a balance among them. As a response, in this paper, we propose two novel algorithms to optimize them simultaneously. To test the performance of our methods in terms of user and retailer satisfaction, we design two simple but convincing metrics. The experimental results on 10 real-world datasets show that our models provide better overall performance than existing methods.

2 Background

In this section, we review several related studies and perform some analysis on real-world datasets.

The most popular techniques for collaborative filtering are matrix factorization(MF) models [Koren *et al.*, 2009], which are effective in fitting the rating matrix with low-rank approximations. MF uses the latent factor $U \in \mathbb{R}^{D\times N}$ to represent N users and $V \in \mathbb{R}^{D\times M}$ to represent M items, and it employs the D-$rank$ matrix $U^TV \in \mathbb{R}^{N\times M}$ for rating prediction. The latent factors can be trained by adopting Singular Value Decomposition(SVD) [Koren, 2008]. For example, Salakhutdinov et al. [2008b; 2008a] propose a probabilistic matrix factorization (PMF) model, in which the factors and ratings are assumed to be sampled from Gaussian distribution, and the latent factors are learned by maximum likelihood estimation, which equals to minimizing $\sum_{u,i\in T_o}(U_u^TV_i - s_{ui})^2 + \lambda_1\sum_u||U_u||^2 + \lambda_2\sum_i||V_i||^2$,

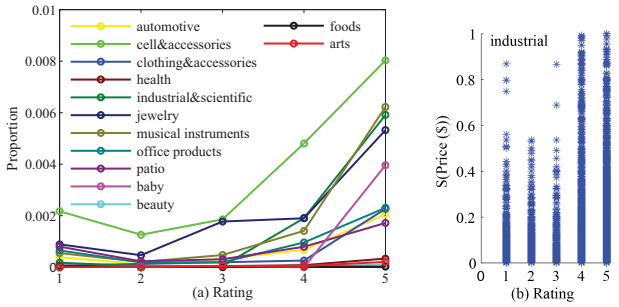

Figure 1: Statistics on real-world datasets.

where s_{ui} is the rating score that u gives to i; T_o denotes the set of observed (user, item) pairs; $||U||^2$ and $||V||^2$ are regularization terms for avoiding over-fitting.

Many MF models are originally proposed for rating prediction, hence they are weak for ranking tasks. To improve hit rate performance for implicit feedback, one-class collaborative filtering (OCCF) [Pan *et al.*, 2008] and implicit matrix factorization (iMF) [Hu *et al.*, 2008; Lin *et al.*, 2014] are proposed based on the assumption of *pointwise* preferences on items. Both of them treat users' feedback as an indication of positive and negative preferences with different levels (c_{ui}) and try to minimize $\sum_{u,i\in T_o} c_{ui}(1 - U_u^TV_i)^2 + \sum_{u,j\in T_{uo}} c_{uj}(0 - U_u^TV_j)^2$ for each (user,item) pair. In addition to these absolute preference assumptions, many *pairwise* algorithms such as BPR and some of its extensions are proposed [Rendle *et al.*, 2009; 2010; Krohn-Grimberghe *et al.*, 2012; Pan and Chen, 2013]. They directly optimize the observed and unobserved (user,item) pairs with relative preference assumptions. Although those algorithms are for implicit feedback, they can also be applied to ranking items with explicit information; the drawback is that the user satisfaction information (i.e., rating values) cannot be efficiently leveraged.

Nowadays, more and more methods try to utilize explicit information for item recommendation. For example, Weimer et al. [2007] propose a maximum margin matrix factorization (MMMF) method CofiRank to optimize ranking in terms of NDCG score. Similr to CofiRank, xCLiMF proposed by [Shi *et al.*, 2013] builds a recommendation model by generalizing collaborative less-is-more filtering assumption and optimizes expected reciprocal rank (ERR). ListRank [Shi *et al.*, 2010] combines a list-wise learning-to-rank algorithm with MF. It optimizes a loss function that represents the cross-entropy of the output rating lists and training lists. These algorithms perform well when the data is dense, but they may be unsuitable for sparse datasets, because few explicit ratings can be used to compare and learn users' preferences.

Beyond product ratings, directly optimizing website revenue is another interesting topic. Although a few researches have considered on how to make price with the help of recommender systems [Bergemann and Ozmen, 2006] or how to recommend discount items to users [Kamishima and Akaho, 2011], they do not consider optimizing the website revenue directly. A good recommender system should not only pro-

vide *potential items* to users that they *like*, but let the website earn *as much money as possible*. We study many real-world datasets and find it is feasible to both optimize user satisfaction and website revenue in applications. We briefly discuss it here by making a qualitative analysis of the relationship between user satisfaction and website revenue. Figure 1(a) shows our statistics on 13 Amazon datasets. The horizontal axis is user's satisfaction and the vertical axis represents the proportion of items which are above average price. Figure 1(b) records the rating values and related normalized item price of an example dataset. It is obvious that user satisfaction and website sales have some positive correlation and can be optimized simultaneously. Overall, we can draw the following conclusions: (1)For the products in the same category, item price has some impact on user satisfaction. The items with higher price may have good quality and hence tend to receive higher rating values. (2)Involving price information can not only benefit the website retailer, but also help to predict user satisfaction, because we can learn the acceptable price range of each user and avoid recommending the items that users do not accept.

3 Our Models and Metrics

In this section, we propose two different solutions to optimize Hit rate, user Satisfaction and website Revenue (HSR) simultaneously. We denote the probability (or the tendency) of a user u buying an item i by $P(b_{ui})$. We use s_{ui} to indicate the rating score that u gives to i and use $P(s_{ui})$ to represent how much u prefers i after (s)he has already bought it. Specifically, we assume the greater rating score that u gives to i in the dataset, the more likely u likes i. For the online retailer, we denote r_{ui} as the revenue it gets when u buys i. Therefore, $P(r_{ui})$ can be regarded as retailer satisfaction. In this paper, the revenue information is considered as the only factor to influence retailer satisfaction and is determined by hit rate and item price. The more sales the retailer achieves, the higher value of $P(r_{ui})$ will be. With the above assumptions, our goal is to maximize both $P(b_{ui})$, $P(s_{ui})$ and $P(r_{ui})$.

3.1 Algorithm1: HSR-1

Since b_{ui} has two states, i.e., $b_{ui} \in \{\text{buy}, \text{not buy}\}$, we use Bernoulli distribution to model it for HSR-1. The probability of b_{ui} is defined as follows:

$$P(b_{ui}|\delta_{ui}) = \sigma(u,i)^{\delta_{ui}}(1-\sigma(u,i))^{1-\delta_{ui}} \tag{1}$$

where δ_{ui} is an indicator parameter. $\delta_{ui} = 1$ if u buys i in the training set, otherwise $\delta_{ui} = 0$. Here $\sigma(u,i)$ is the sigmoid funciton with $f_b(u,i)$ denoting the user-item association,

$$\sigma(u,i) = \frac{1}{1+e^{-f_b(u,i)+c}} \tag{2}$$

where c is a constant.

Our next task is to model user satisfaction and retailer satisfaction given b_{ui}. We note that some items, (1)items that users like but do not pretend to buy, and (2)items with high price but irrelevant to users, should not be over optimized due to the risk of reducing the hit rate. Hence, s_{ui} and r_{ui} are considered only when $\delta_{ui} = 1$. In other words, we optimize s_{ui} and r_{ui} when (user, item) pair is observed. Since the correlation of s_{ui} with r_{ui} is obviously positive based on our analysis in the previous section, we assume they are sampled from bivariate normal distribution, that is,

$$P(s_{ui}, r_{ui}|\delta_{ui}) = \left\{ K_1 \exp\left[-K_2 \left(\frac{\Delta_s^2}{\sigma_s^2} + \frac{\Delta_r^2}{\sigma_r^2} - \frac{2\rho\Delta_s\Delta_r}{\sigma_s\sigma_r} \right) \right] \right\}^{\delta_{ui}} \tag{3}$$

where $K_1 = 1/(2\pi\sigma_s\sigma_r\sqrt{1-\rho^2})$, $K_2 = 1/(2(1-\rho^2))$, $\Delta_s = (s_{ui} - f_s(u,i))$ and $\Delta_r = (r_{ui} - f_r(u,i))$. Here σ denotes standard deviation of each factor and ρ is the correlation between s_{ui} and r_{ui}. $f_s(u,i)$ and $f_r(u,i)$ can be depicted as satisfaction function for u and i.

Because s_{ui} and r_{ui} have different ranges, in applications, we standardize both of them to $[0, L]$. For example,

$$\tilde{r}_{ui} = \frac{r_{ui} - r_{Min}}{r_{Max} - r_{Min}} L \tag{4}$$

where r_{Max} and r_{Min} are supremum and infimum of item price set. We then formulate the maximum posterior to derive our optimization criterion:

$$OPT(\text{HSR-1}) \propto \ln \prod_{u,i \in T} P(b_{ui}|\delta_{ui}) P(\tilde{s}_{ui}, \tilde{r}_{ui}|\delta_{ui}) \tag{5}$$

where some model parameters are ignored for the sake of brevity. We assume b_{ui}, s_{ui} and r_{ui} are influenced by U_u and V_i, and a straightforward assumption is that $f_b(u,i) = f_s(u,i) = f_r(u,i) = U_u^T V_i$.

We add regularization terms $||U||^2$ and $||V||^2$ for avoiding over-fitting, and conduct stochastic gradient descent (SGD) algorithm to update U and V. Different from PMF, we update the latent factors with both observed and unobserved (user, item) pairs. Specifically, in each iteration, we sample two pairs from each class and update the latent factors in turn. In order to make a prediction, we compute the expected value $U_u^T V_i$ for each (u, i) pair after finishing the update process. For u, the item with higher $U_u^T V_i$ will rank higher and will be recommended with higher probability.

3.2 Algorithm2: HSR-2

Because the Gaussian assumption in HSR-1 may not be an optimal solution to modeling discrete values (i.e., s_{ui}), we use binomial distribution to model user satisfaction inspired by [Wu, 2009; Kim and Choi, 2014]. We further integrate b_{ui} into s_{ui} for consistency. Then the probability of s_{ui} can be written as:

$$\begin{aligned} P(s_{ui}) &= \mathcal{B}(s_{ui}|L, \sigma(u,i)) \\ &= \binom{L}{s_{ui}} \sigma(u,i)^{s_{ui}} (1-\sigma(u,i))^{L-s_{ui}} \end{aligned} \tag{6}$$

where $\mathcal{B}$ denotes binomial distribution, and L is the maximal rating score in the dataset.

Intuitively, binomial distribution is always unimodal, which is more reasonable than Gaussian models. HSR-2 has another advantage than HSR-1: $U_u^T V_i$ will not be limited by s_{ui} and r_{ui} when optimizing $\sigma(u,i)$.

We also use binomial distribution to model r_{ui}. Note that r_{ui} needs not to be an integer because the first term in Eq.(6) will be eliminated in the optimization process. Similar to HSR-1, we also assume user satisfaction and retailer satisfaction are correlated but conditional independent given U_u and V_i. Combining distributions of s_{ui} and r_{ui}, and formulating the maximum posterior, we obtain the following optimization criterion:

$$OPT(\text{HSR-2}) \propto \ln \prod_{u,i \in T} P(\tilde{s}_{ui}) P(\tilde{r}_{ui} | \delta_{ui}) \tag{7}$$

We also conduct SGD algorithm to update U and V for HSR-2 and the learning process is similar to HSR-1. In each iteration, the time complexity of our methods is $O(D)$, therefore the execution time is comparable to the existing pairwise item recommendation methods.

3.3 Discussion and Explanation

Both HSR-1 and HSR-2 rank items based on $U_u^T V_i$. This is because if $U_u^T V_i$ is greater, $P(b_{ui})$, $P(s_{ui})$ and $P(r_{ui})$ will also be greater, and hence u will be more likely to choose and prefer i, while the website retailer will get relatively higher revenue. It can be explained in Figure 2, where the horizontal axis denotes user satisfaction, vertical axis represents website revenue and the contour expresses $P(s_{ui})P(r_{ui})$ that HSR try to optimize. The black points represent the items that u wants to buy and the gray points are irrelevant items.

We now compare HSR with other related models and give an interpretation of their assumptions. For implicit feedback, the algorithms such as iMF and BPR aim to find the most relevant items but ignore horizontal and vertical directions. Therefore, they do not consider whether u likes the recommended items or not. The recommendation methods based on explicit feedback, e.g, CofiRank, ListRank and xCLiMF, try to search the potential items from direction A, in which the items that u most likes are supposed to be first recommended. However, because they do not try to avoid the irrelevant items when training the models, some points with high $U_u^T V_i$ may be irrelevant to u in the test set.

For HSR, we search the items from direction B, which guarantees both user satisfaction and retailer satisfaction. We also try to avoid the irrelevant items in a training set by aforementioned Bernoulli or binomial assumptions, so the cheap and relevant items can also be recommended with many opportunities. From Eq.(5) and Eq.(7) we find that by involving item price information, HSR-1 and HSR-2 can not only globally optimize the website revenue, but also try to learn the acceptable price range of each customer, which is beneficial to learning user satisfaction. Because of those reasons, HSR are more comprehensive than existing approaches.

We note that for some datasets, the item price and user satisfaction may not be correlated obviously. In this case, we still try to balance all the factors for it can improve overall performance. In practice, we can also adjust the weight of hit rate part, user satisfaction part and revenue part of Eq.(5) and Eq.(7) for different purposes.

3.4 The Metrics

To measure user satisfaction and website revenue, we design two modified hit rate based metrics - satisfaction recall (S-R) and price recall (P-R) as follows:

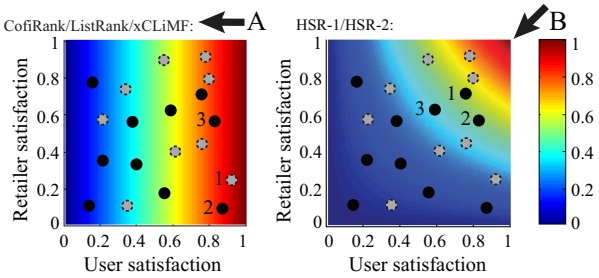

Figure 2: Model comparison.

$$\text{S-R} = M(s)R(b) \tag{8}$$

$$\text{P-R} = M(r)R(b) \tag{9}$$

where $R(b)$ is the recall metric on the test set. It can be represented as:

$$R(b) = \frac{\sum_u |L_u^{tr} \cap L_u^{te}| / |L_u^{te}|}{N} \tag{10}$$

where N is the number of users, L_u^{tr} denotes the recommended item set for u, and L_u^{te} is the observed item set. $M(s)$ and $M(r)$ are the mean rating score and mean price of hit items in the test set. For example,

$$M(r) = \frac{\sum_{u,i} r_{ui} \delta(i \in L_u^{tr} \cap L_u^{te})}{\sum_u |L_u^{tr} \cap L_u^{te}|} \tag{11}$$

From Eq.(8), we observe that if an algorithm achieves a high S-R score, it must guarantee both recall and mean rating score of hit items. This idea also applies to P-R, which balances both recall and mean price of hit items. Hence, our metrics describe overall system performance.

4 Experiments

4.1 Datasets

To fairly evaluate the performance of our models, we study 10 Amazon datasets provided by [McAuley and Leskovec, 2013]. Each dataset contains plenty of review information such as rating scores, item price, user comments and review time. We only consider user ratings and item price. Some related statistic information is summarized in Table 3. We preprocess the data by dropping the items without rating or price information, and subdivide them into training, validation and test sets, where 80% is used for training and the left 20% is evenly split between validation and test sets.

4.2 Baselines and Parameter Settings

We compare our models with 4 state-of-the-art methods for ranking: (1)CofiRank, (2)ListRank, (3)iMF and (4)BPR.

CofiRank [Weimer *et al.*, 2007] and ListRank [Shi *et al.*, 2010] are *listwise* methods for item ranking with explicit feedback (e.g, ratings). Both of them directly optimize a

Table 2: Prediction and recall performance of compared models on 10 real-world datasets.

metric	method	automotive	beauty	clothing& accessories	industrial& scientific	musical instruments	shoes	software	tools& home	toys& games	video games
Precision $K=5$	CofiRank	0.0009	0.0021	0.0039	0.0006	0.0003	0.0081	0.0018	0.0005	0.0005	0.0042
	ListRank	0.0001	0.0023	0.0072	0.0009	0.0013	0.0016	0.0022	0.0009	0.0006	0.0023
	iMF	0.0038	0.0102	0.0248	0.0102	0.0050	0.0613	0.0091	0.0022	0.0011	0.0023
	BPR	0.0057	0.0301	0.0449	0.0172	0.0062	0.0762	0.0080	0.0040	0.0015	0.0078
	HSR-1	**0.0072**	**0.0337**	0.0470	0.0193	**0.0073**	**0.0832**	0.0099	0.0028	0.0012	0.0084
	HSR-2	0.0064	0.0323	**0.0484**	**0.0239**	**0.0073**	0.0804	**0.0110**	**0.0050**	**0.0017**	**0.0087**
	improve	26.32%	11.96%	7.80%	38.95%	17.74%	9.19%	20.88%	25.00%	13.33%	11.54%
Recall $K=5$	CofiRank	0.0041	0.0102	0.0188	0.0023	0.0017	0.0272	0.0045	0.0022	0.0022	0.0152
	ListRank	0.0003	0.0108	0.0357	0.0012	0.0064	0.0075	0.0036	0.0040	0.0020	0.0012
	iMF	0.0155	0.0439	0.1215	0.0176	0.0218	0.2341	0.0426	0.0101	0.0053	0.0088
	BPR	0.0220	0.1327	0.2112	0.0641	0.0274	0.2890	0.0256	0.0188	0.0063	0.0329
	HSR-1	**0.0286**	**0.1501**	0.2199	0.0683	0.0322	**0.3124**	0.0436	0.0133	0.0060	0.0401
	HSR-2	0.0246	0.1446	**0.2271**	**0.0814**	**0.0325**	0.3044	**0.0514**	**0.0231**	**0.0082**	**0.0410**
	improve	30.00%	13.11%	7.53%	26.99%	18.61%	8.10%	20.66%	22.87%	30.16%	24.62%
F1	improve	27.06%	12.17%	7.75%	36.24%	17.90%	8.96%	20.84%	24.62%	16.22%	13.83%

Table 3: Basic statistics of the data.

dataset	#users	#items	avg rating	avg price ($)	sparsity
automotive	116898	40256	4.1641	56.0249	0.0035%
beauty	144771	19159	4.1712	23.9697	0.0077%
clothing&accessories	14849	2278	3.9649	23.0374	0.0571%
industrial&scientific	26381	17155	4.3450	42.1607	0.0192%
musical instruments	52665	9532	4.2403	59.8015	0.0132%
shoes	4762	1275	4.2541	42.0972	0.1966%
software	33362	3823	3.3448	54.5471	0.0334%
tools&home	235731	36784	4.0681	59.3773	0.0037%
toys& games	215373	30835	4.1607	46.1495	0.0045%
video games	174014	12257	3.9857	45.7538	0.0157%

list of items based on rating values. iMF [Hu *et al.*, 2008; Lin *et al.*, 2014] is a *pointwise* method that updates observed and unobserved pairs individually. BPR [Rendle *et al.*, 2009] adopts a *pairwise* ranking method with the assumption that the observed (user,item) pairs are more positive than the unobserved (user,item) pairs, and optimizes the comparison in each iteration. For iMF and BPR, we take a pre-processing step by keeping the ratings as observed positvie feedback and other (user, item) pairs as unobserved feedback. We conduct comparisons with iMF and BPR because they perform better in very sparse datasets than the other baseline methods.

We set the learning rate to 0.001 for iMF, ListRank and HSR-2 and to 0.01 for CofiRank, BPR and HSR-1. We fix latent dimension $D = 10$ and choose regularization coefficient from $\lambda \in \{0.001, 0.01, 0.1, 1\}$. The correlation ρ is set to 0.1 and $L = 5$. We adopt precision, recall and our proposed metrics S-R and P-R to test the performance. The length of recommendation lists is fixed at $K = 5$ for all metrics.

4.3 Analysis of Hit Rate

We show the hit rate performance comparison of our mothods with other baseline models. The precision and recall results are shown in Table 2, where the best performance is in bold font. For comprehensive comparison, we also list the improvement of F1 score in the last row. From the table, we have the following observations:

(1) Because our test sets are very sparse, the precision results are small comparing with recall when $K = 10$. We also observe that the models considering unobserved items (i.e., iMF, BPR, HSR-1 and HSR-2) perform much better than the models which only use rating information (i.e., CofiRank and ListRank). Extremely, when each user only has one observed item in the training set, CofiRank and ListRank will be next-to-no benefit for exploiting the relevant items, and that is the reason for their poor performance in our experiments.

(2) On average, BPR is better than iMF due to the pairwise assumption. Although iMF considers the unobserved pairs, it is based on an absolute assumption that the $s_{u,i}$ will be 1 when u buys i, and 0 otherwise. On the contrary, BPR, HSR-1 and HSR-2 try to take feedback as relative preferences rather than absolute ones.

(3) Both HSR-1 and HSR-2 are better than the other baseline methods in terms of both precision and recall results. They improve the precision by an average of more than 18% and improve the recall by an average of more than 20% comparing with the best baseline models. It is mainly because that HSR-1 and HSR-2 not only consider all possible (user, item) pairs, but also adopt explicit feedback information.

4.4 Analysis of User Satisfaction

We then analyse the performance of the compared models on S-R metric and list the improvement of our models on mean ratings of hit items (M(s)). The experimental results are shown in Table 4, from which we can have the following observations and conclusions:

(1) Some models with lower precision or recall values can achieve higher S-R scores. For example, on the *video games* dataset, HSR-1 has lower recall performance than HSR-2 but achieves better S-R due to higher $M(s)$.

(2) Both HSR-1 and HSR-2 are better than the other baseline models on S-R, which demonstrates the benefit of

Table 4: S-R and P-R performance of compared models on 10 real-world datasets.

metric	method	automotive	beauty	clothing& accessories	industrial& scientific	musical instruments	shoes	software	tools& home	toys& games	video games
S-R $k=5$	CofiRank	0.0174	0.0455	0.0891	0.0106	0.0071	0.1274	0.0148	0.0092	0.0094	0.0706
	ListRank	0.0012	0.0436	0.1493	0.0053	0.0265	0.0356	0.0123	0.0172	0.0086	0.0055
	iMF	0.0672	0.1861	0.5026	0.0724	0.0990	1.0332	0.1632	0.0444	0.0228	0.0400
	BPR	0.0967	0.5644	0.8861	0.2634	0.1220	1.2592	0.0886	0.0805	0.0271	0.1466
	HSR-1	**0.1249**	**0.6414**	0.9167	0.3021	0.1415	**1.3422**	0.1744	0.0546	0.0268	**0.1848**
	HSR-2	0.1093	0.6171	**0.9644**	**0.3443**	**0.1460**	1.2993	**0.2011**	**0.0992**	**0.0359**	0.1799
	improve	29.15%	13.64%	8.84%	30.70%	19.66%	6.59%	23.21%	23.24%	32.11%	26.05%
M(s)	improve	1.09%	0.53%	1.24%	7.33%	-1.14%	-2.65%	4.54%	-1.83%	3.22%	1.33%
P-R $k=5$	CofiRank	0.0303	0.0499	0.1507	0.0108	0.0176	0.3384	0.1180	0.0302	0.0346	0.1896
	ListRank	0.0053	0.0594	0.2434	0.0126	0.0615	0.0944	0.0824	0.0478	0.0238	0.0133
	iMF	0.1764	0.2392	0.7853	0.2481	0.1545	2.9148	1.2105	0.1478	0.0734	0.0989
	BPR	0.3458	0.8074	1.5440	0.8670	0.2786	3.7270	0.4779	0.3143	0.0717	0.3401
	HSR-1	**0.6205**	**1.0306**	1.6604	1.1300	**0.3436**	**4.4324**	1.3224	0.2676	0.0909	0.5586
	HSR-2	0.4901	0.9902	**1.7265**	**1.7384**	0.3343	4.3663	**1.5946**	**0.3550**	**0.1339**	**0.6510**
	improve	79.42%	27.65%	11.82%	100.51%	23.31%	18.93%	31.72%	12.95%	82.40%	91.40%
M(r)	improve	38.02%	12.92%	4.02%	51.16%	4.97%	11.25%	9.26%	19.72%	17.59%	41.19%

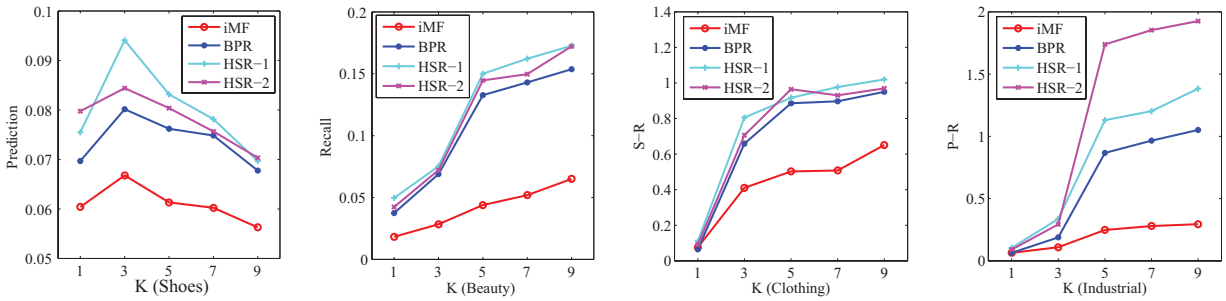

Figure 3: Performance comparison with different recommendation list size.

combining b_{ui} and s_{ui}. We also find that $M(s)$ is not improved in some datasets. This is mainly because HSR try to balance hit rate and user satisfaction instead of simply optimizing the latter one. Because HSR have much better performance than the other models on recall values, the influence of $M(s)$ is not obvious here.

4.5 Analysis of Revenue

We compare our models with other baselines on P-R metric, and list the improvement of HSR on mean price of hit items (M(r)). The results are summarized in the lower portion of Table 4, and we find that

(1) There is plenty of room to optimize the website revenue. Some algorithms with poor recall performance may achieve better P-R metric and $M(r)$. For example, on the *toys&games* dataset, iMF has nearly 19% lower recall score than BPR, but it performs better than BPR when we consider P-R metric, because it increases the average revenue a lot at the cost of decreasing a certain degree of the recall value.

(2) Our models achieve much better performance than any other models on both P-R metric and $M(r)$. For many datasets, the improvement is over 50%, and for some datasets, such as *industrial&scientific* and *video games*, the performance can even double the best baseline models, which shows the efficiency of HSR by integrating the price factor into the recommendation models. Because total revenue and profit are almost always correlated, our methods have high degree of confidence to help earn much more money for the online aggregators.

4.6 Analysis of Stability

Finally, to study the recommendation stability, we vary the recommendation size K to be values of $\{1, 3, 5, 7, 9\}$, and some results are shown in Figure 3. As can be seen, when K increases, the precision scores have general downward trend, and other metrics pretend to increase. HSR-1 and HSR-2 are consistently better than other models. In particular, when K increases, the advantages of our models on P-R become more and more obvious.

5 Conclusion and Future Work

In this paper, we consider three important factors, (1)hit rate, (2)user satisfaction, and (3)website revenue, to measure the performance of recommender systems. We then propose two algorithms, called HSR-1 and HSR-2, to optimize them. Finally, two simple but convincing metrics are designed to measure the comparison models. The experimental results show

that our models can achieve better hit rate and user satisfaction results and also significantly improve website revenue.

In the future, we intend to exploit more kinds of user behaviors and design some mechanisms for recommendation diversification. To better measure the overall recommendation performance, we will try to design more effective metrics.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (NSFC) No. 61272303 and Natural Science Foundation of Guangdong Province No. 2014A030310268. We also thank Dr. Weike Pan for helpful discussions.

References

[Bergemann and Ozmen, 2006] Dirk Bergemann and Deran Ozmen. Optimal pricing with recommender systems. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, pages 43–51. ACM, 2006.

[Hu *et al.*, 2008] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th International Conference on Data Mining*, ICDM'08, pages 263–272. IEEE, 2008.

[Kamishima and Akaho, 2011] Toshihiro Kamishima and Shotaro Akaho. Personalized pricing recommender system: Multi-stage epsilon-greedy approach. In *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec'11, pages 57–64. ACM, 2011.

[Kim and Choi, 2014] Yong-Deok Kim and Seungjin Choi. Bayesian binomial mixture model for collaborative prediction with non-random missing data. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys'14, pages 201–208. ACM, 2014.

[Koren *et al.*, 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[Koren, 2008] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, SIGKDD'13, pages 426–434. ACM, 2008.

[Krohn-Grimberghe *et al.*, 2012] Artus Krohn-Grimberghe, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. Multi-relational matrix factorization using bayesian personalized ranking for social network data. In *Proceedings of the 5th ACM International Conference on Web Search and Data Mining*, WSDM'12, pages 173–182. ACM, 2012.

[Lin *et al.*, 2014] Christopher H. Lin, Ece Kamar, and Eric Horvitz. Signals in the silence: Models of implicit feedback in a recommendation system for crowdsourcing. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, AAAI'14, pages 22–28, 2014.

[McAuley and Leskovec, 2013] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys'13, pages 165–172. ACM, 2013.

[Pan and Chen, 2013] Weike Pan and Li Chen. Gbpr: Group preference based bayesian personalized ranking for one-class collaborative filtering. In *Proceedings of the 23th International Joint Conference on Artificial Intelligence*, IJCAI'13, pages 2691–2697. AAAI Press, 2013.

[Pan *et al.*, 2008] Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *Proceedings of the 8th International Conference on Data Mining*, ICDM'08, pages 502–511. IEEE, 2008.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, pages 452–461. AUAI Press, 2009.

[Rendle *et al.*, 2010] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW'10, pages 811–820. ACM, 2010.

[Salakhutdinov and Mnih, 2008a] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th International Conference on Machine Learning*, ICML'08, pages 880–887. ACM, 2008.

[Salakhutdinov and Mnih, 2008b] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, volume 20 of *NIPS'08*, 2008.

[Shi *et al.*, 2010] Yue Shi, Martha Larson, and Alan Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In *Proceedings of the 4th ACM Conference on Recommender Systems*, RecSys'10, pages 269–272. ACM, 2010.

[Shi *et al.*, 2013] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, and Alan Hanjalic. xclimf: optimizing expected reciprocal rank for data with multiple levels of relevance. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys'13, pages 431–434. ACM, 2013.

[Weimer *et al.*, 2007] Markus Weimer, Alexandros Karatzoglou, Quoc Viet Le, and Alex Smola. Maximum margin matrix factorization for collaborative ranking. *Advances in Neural Information Processing Systems*, 2007.

[Wu, 2009] Jinlong Wu. Binomial matrix factorization for discrete collaborative filtering. In *Proceedings of the 9th International Conference on Data Mining*, ICDM'09, pages 1046–1051. IEEE, 2009.