Published in "Physica A: Statistical Mechanics and its Applications
2011, vol. 390, no. 23-24, p. 4486-4493"
which should be cited to refer to this work.

Negative ratings play a positive role in information filtering

Wei Zeng [a], Yu-Xiao Zhu [a], Linyuan Lü [b], Tao Zhou [a,c,*]

[a] *Web Sciences Center, School of Computer Science and Engineering, University of Electronic Science and Technology of China 610054 Chengdu, PR China*
[b] *Department of Physics, University of Fribourg, Chemin du Musee 3, CH-1700 Fribourg, Switzerland*
[c] *Department of Modern Physics, University of Science and Technology, Hefei Anhui 230026, PR China*

The explosive growth of information asks for advanced information filtering techniques to solve the so-called information overload problem. A promising way is the recommender system which analyzes the historical records of users' activities and accordingly provides personalized recommendations. Most recommender systems can be represented by user-object bipartite networks where users can evaluate and vote for objects, and ratings such as "dislike" and "I hate it" are treated straightforwardly as negative factors or are completely ignored in traditional approaches. Applying a local diffusion algorithm on three benchmark data sets, *MovieLens*, *Netflix* and *Amazon*, our study arrives at a very surprising result, namely the negative ratings may play a positive role especially for very sparse data sets. In-depth analysis at the microscopic level indicates that the negative ratings from less active users to less popular objects could probably have positive impacts on the recommendations, while the ones connecting active users and popular objects mostly should be treated negatively. We finally outline the significant relevance of our results to the two long-term challenges in information filtering: the sparsity problem and the cold-start problem.

1. Introduction

Thanks to advanced computer techniques, the amount of available information grows very fast, as indicated by the exponential expansion of the Internet [1] and the World Wide Web [2]. Now we have to make a choice from billions of web pages, millions of books and tens of thousands of movies. However, to evaluate all the alternatives is not feasible for normal users, and if the alternative objects are not well organized, the increasing number makes the users feel even worse than before. In a word, people are facing too much data and sources to be able to find out what they like most. Automatic information filtering techniques are required to solve this so-called *information overload problem* in modern information science.

The search engine is a landmark of information filtering, by which users can find the relevant objects with the help of properly chosen keywords [3]. Essentially speaking, the search engine aims at finding *what you want*, namely before using the search engine, users already know something about the objects they are looking for, otherwise they could not determine which keywords they should input. Usually, users are looking for some exact information, like the email address of a certain person, the online submission system of a conference, the homepage of a journal, and so on. Without an informative purpose, Googling "give me something I like" will probably return nothing relevant. In contrast, a recommender system is designed for finding *what you like* [4], which may recommend to you some books or URLs according to your purchased books and

* Corresponding author at: Department of Modern Physics, University of Science and Technology, Hefei Anhui 230026, PR China.
E-mail address: zhutou@ustc.edu (T. Zhou).

collected bookmarks. Generally speaking, these recommendations are drawn automatically and the system does not ask for keywords to activate the recommending process. Users are expected to be interested in the recommended objects yet they probably did not know the recommendations before and may never have know them without the recommender system. Actually, a "perfect" recommender system should be not only able to recommend objects that users like, but also able to recommend objects that users do not know [5].

Recommender systems have already found significant applications in e-commerce [6]. For example, *Amazon.com* uses one's purchase record to recommend books [7], *AdaptiveInfo.com* uses one's reading history to recommend news [8], and the *TiVo* digital video system recommends TV shows and movies on the basis of users' viewing patterns and ratings [9]. The design of recommendation algorithms thus has not only academic interests but also practical benefits, which has attracted increasing attention of many branches of science and engineering, ranging from computer science and management science to mathematics and physics [10]. Representative recommending techniques developed by the computer science community include collaborative filtering [11,12], singular value decomposition [13,14], content-based analysis [15], latent semantic models [16], latent Dirichlet allocation [17], principle component analysis [18], and so on. Recently, physical perspectives and approaches have also found applications in designing recommendation algorithms, including iterative refinements [19–21], random-walk-based algorithms [22–26] and heat conduction algorithms [5,27]. Generally speaking, the performance of the above-mentioned algorithms can be further improved by using a hybrid method [28,29] or ensemble learning [30], or by exploiting additional information, like time [31] and tags [32].

In many real recommender systems, users vote objects by discrete ratings. For example, in Yahoo Music, users vote each song with one to five stars representing "Never play again" ($\star$), "It is OK" ($\star\star$), "Like it" ($\star\star\star$), "Love it" ($\star\star\star\star$) and "Can't get enough" ($\star\star\star\star\star$). The similarity between two users is usually defined as the Pearson correlation on all their commonly voted objects and thus ratings lower than the average play a negative role. Some algorithms only care about whether an object will be selected by a user, and the ratings are coarse-grained. For example, in Ref. [33], an object is considered to be collected by a user if and only if the given rating is at least 3. A similar coarse-graining method has been used in many other algorithms. In a word, up to our knowledge, in most of the known algorithms, the negative ratings[1] either play a negative role or are neglected.

Notice that, even if a user dislikes a book after reading it, she/he might be attracted by its title, abstract, author, type or others, otherwise she/he will never read it. Users may be more captious to the objects they are familiar with or interested in. In fact, the negative ratings contain rich information, and should not be neglected or be considered all negative. Especially, when the ratings are very sparse, negative ratings may also be important to extract the tastes of users, at least which type of objects a user may like the most. Therefore, we should reconsider the usage of negative ratings in personalized recommendation. In this paper, we apply a modified version of the so-called *network-based inference* (NBI) [23], which can distinguish the contributions from positive and negative ratings. According to the extensive numerical analysis on three benchmark data sets, *MovieLens*, *Netflix* and *Amazon*, our study arrives at a very surprising result: negative ratings may play a positive role. Further analyses show that in the macroscopic level negative ratings are more valuable for sparser data sets, and in the microscopic level, negative ratings from inactive users onto less popular objects are more valuable. We finally outline the significant relevance of our results to the two long-term challenges in information filtering: the sparsity problem [34] and the cold-start problem [35].

2. Problem and metrics

A recommender system consists of a set of users $\{u_1, u_2, \ldots, u_m\}$ and a set of objects $\{o_1, o_2, \ldots, o_n\}$, and can be represented by a bipartite network [36] where users are connected to objects they have read, purchased, collected, etc. In the simplest case, every edge has the same meaning—representing a kind of interaction between a user and an object, while in this paper, we consider a more complicated case, say the edges are classified into two types: *positive edges* represent favor and *negative edges* represent disfavor. Notice that, a non-edge has a much different meaning from a negative edge since the latter means both disfavor and relevance [37] which are not indicated by the former. Fig. 1 illustrates an example with 4 users and 4 movies where red and blue lines represent positive and negative edges, respectively.

Given a target user u_i, a recommender system will sort all u_i'*s* uncollected objects and recommend the top-L objects to u_i. To test the recommendation algorithms, the data set is randomly divided into two parts: The training set and the testing set. The training set is treated as known information, while no information in the testing set is allowed to be used for recommendation.

The *precision* of recommendations to u_i, $P_i(L)$, is defined as the ratio of the number of u_i's removed positive edges (i.e. the objects collected by u_i with positive ratings in the testing set), $H_i(L)$, contained in the top-L recommendations to L, say

$$P_i(L) = H_i(L)/L. \tag{1}$$

The precision of the whole system is the average of individual precisions over all users, given as

$$P(L) = \frac{1}{m}\sum_{i=1}^{m} P_i(L). \tag{2}$$

[1] Negative ratings here refer to the ratings lower than the average. In a system with five discrete ratings, the ratings 1 and 2 are usually considered as negative ratings.

Fig. 1. (Color online) Illustration of a recommender system consisting of four users and four movies. The basic information contained in every recommender system is the relation between users and objects that can be represented by a bipartite graph.

The *recall* of recommendations to u_i, $R_i(L)$, is defined as the ratio of the number of u_i's removed positive edges contained in the top-L recommendations to the number of u_i's positive edges in the testing set, say

$$R_i(L) = H_i(L)/N_i, \tag{3}$$

where N_i denotes the number of $u_i's$ positive edges in the testing set. Similarly, the recall of the whole system is defined as

$$R(L) = \frac{1}{m}\sum_{i=1}^{m} R_i(L). \tag{4}$$

Higher precision and recall indicate higher accuracy of recommendations. Generally speaking, as the length of the recommendation list increases, the precision will decrease and the recall will increase. We therefore use an integrated index to quantify the algorithm's performance, called the $F1$ measure, defined as [38]:

$$F1(L) = \frac{2P(L)R(L)}{P(L) + R(L)}. \tag{5}$$

3. Algorithm

3.1. NBI on weighted networks

The standard NBI works on unweighted bipartite networks. Here, we introduce a weighted NBI algorithm. Considering a weighted bipartite network, denoted by a weighted adjacency matrix $W = \{w_{i\alpha}\}$ where $w_{i\alpha} > 0$ if there is an edge between u_i and o_α, $w_{i\alpha} = 0$ otherwise, and assuming that a unit resource is assigned to each object, who distributes its resource to all neighboring users, and then each user redistributes the received resource to all her/his collected objects. Accordingly, the resource that object o_α has received from object o_β is

$$s_{\alpha\beta} = \frac{1}{k(o_\beta)}\sum_{i=1}^{m}\frac{w_{i\alpha}w_{i\beta}}{k(u_i)}, \tag{6}$$

where $k(o_\beta)$ is the degree of o_β, namely the number of users who have collected o_β, $k(u_i)$ is the degree of u_i, namely the number of objects u_i has collected. This process can be expressed by the matrix form $\overrightarrow{f'} = S\overrightarrow{f}$, where $S = \{s_{\alpha\beta}\}$, $\overrightarrow{f}$ is the initial resource vector on objects, and $\overrightarrow{f'}$ is the final resource vector. Given the target user u_i, the corresponding initial resource vector is defined as

$$f_\alpha^{(i)} = a_{i\alpha}, \tag{7}$$

where $a_{i\alpha}$ is the element of the adjacency matrix, namely $a_{i\alpha} = 1$ if u_i has collected o_α and $a_{i\alpha} = 0$ otherwise. According to the resource-allocation process discussed before, the final resource vector $\overrightarrow{f'}$ is

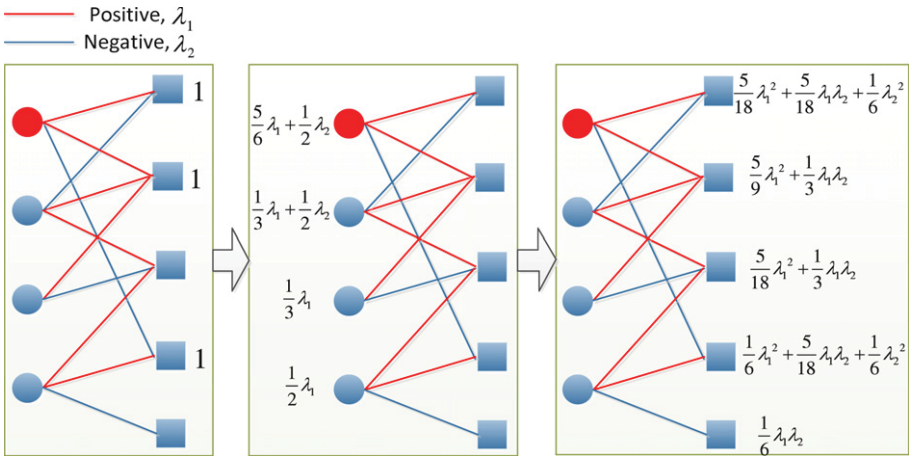


Fig. 2. (Color online) How NBI works on a signed network. The red lines denote positive edges and the blue lines denote negative edges. The weight of each positive edge is equal to λ_1 and of each negative edge is equal to λ_2. Circles and squares denote users and objects, respectively. The target user is indicated by the red circle. The initial resource distribution is shown in the left plot.

$$f'^{(i)}_{\alpha} = \sum_{\beta=1}^{n} s_{\alpha\beta} f^{(i)}_{\beta} = \sum_{\beta=1}^{n} s_{\alpha\beta} a_{i\beta}. \tag{8}$$

Then all u_i's uncollected objects are sorted in the descending order according to the final resource and those objects with the highest values are recommended.

3.2. NBI on a signed network

In many real recommender systems, users are allowed to vote objects by discrete ratings. Taking MovieLens (a movie recommender system) for example, users can evaluate movies by ratings on a scale of 1–5. As mentioned above, the negative ratings contain rich information. Two reasons may lead to negative ratings: (1) the user really dislikes the object, (2) the user is very strict to the objects she/he is interested in. Anyway, a rating, either positive or negative, indicates the attention on the object. We next extend the NBI on a signed network, where positive edges indicate favor and negative edges indicate disfavor.

Signed networks are known to the scientific communities in recent years [39,40]. For example, users can declare others to be either "friends" or "foes", or express trust or distrust of others. Since here we consider systems with 1–5 discrete ratings, we simply define an edge as positive (negative) if the corresponding rating is no less than 3 (less than 3). Accordingly, we can get a signed network, for example, in Fig. 2, the red lines denote positive edges while the blue lines denote negative edges. We respectively use the adjacency matrices A^P and A^N to represent positive and negative edges.

To see the contributions of negative edges in recommendation, each positive edge is assigned a weight λ_1 and each negative edge is assigned a weight λ_2. Fig. 2 illustrates the resource diffusion process of NBI on the signed network.

4. Results and analysis

4.1. Data description

Three real data sets are used to test the algorithms:

1. MovieLens (http://www.movielens.org) is a movie recommendation web site, which uses users' ratings to generate personalized recommendations.
2. Netflix (http://www.netflix.com) is an online DVD and Blu-ray Disc rental service in the US. The data we used is a random sample that consists of 3000 users who have voted at least 45 movies and 2779 movies having been voted by at least 23 users.
3. Amazon (http://www.amazon.com) is a multinational electronic commerce company. The original data were collected from 28 July 2005 to 27 September 2005, and what we used here is a random sample, consisted of 3604 users and 4000 books.

The basic statistics of these three data sets are presented in Table 1, where one can see that users prefer to give positive ratings to objects.

Table 1
Basic statistics of the tested data sets. The sparsity is defined by $\frac{E}{m \cdot n}$, where E is the number of edges in the bipartite networks, and m and n respectively denote the number of users and objects.

Data set	#Users	#Objects	#Edges	Sparsity	(+ratings) / (−ratings)
MovieLens	943	1682	100000	6.3×10^{-2}	4.72/1
Amazon	3604	4000	134680	9.3×10^{-3}	8.46/1
Netflix	3000	2779	197248	2.4×10^{-2}	5.77/1

Table 2
The optimum values of λ corresponding to different L and p.

	Amazon		MovieLens		Netflix	
	$L = 10$	$L = 20$	$L = 10$	$L = 20$	$L = 10$	$L = 20$
$p = 0.2$	0.13	0.28	−0.18	−0.18	−1.10	−1.10
$P = 0.4$	0.50	0.54	0	0	−0.94	−0.83
$P = 0.6$	0.73	0.74	0.14	0.14	−0.57	−0.32
$P = 0.8$	0.76	0.82	0.60	0.52	0.23	0.27

Table 3
The optimum values of λ corresponding to different subsets.

		R_1^L	R_2^L	R_3^L	R_4^L	R_5^L
MovieLens	$L = 10$	−3.14	−3.28	−2.12	−0.48	1.32
	$L = 20$	−3.39	−3.11	−2.19	0	1.90
Netflix	$L = 10$	−3.23	−3.36	0	0	2.12
	$L = 20$	−3.24	−3.17	−1.94	0	1.48
Amazon	$L = 10$	−2.30	−1.58	0.40	0.85	1.05
	$L = 20$	−2.20	−1.60	0.82	1.11	0.91

4.2. Macroscopic analysis

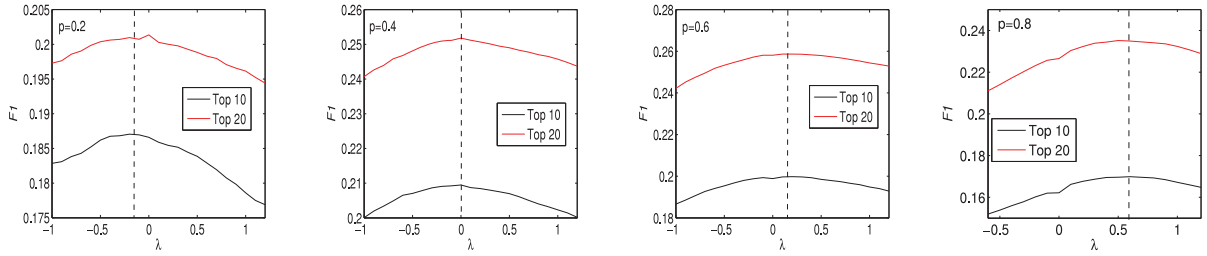
For the sake of seeing the contribution of negative edges (corresponding to negative ratings), each of the aforementioned data sets, namely MovieLens, Netflix and Amazon, was divided into two parts: a training set and a testing set. In the testing set, only the ratings no less than 3 are conserved for testing the algorithm's accuracy. The ratio of the edges in the testing set to the total edges is p and thus a larger p corresponds to a sparser training data. We use the $F1$ measure to quantify the algorithm's accuracy.

In the training set, positive and negative edges were assigned weights 1 and λ, respectively. By tuning the value of λ, one can adjust the amount of contribution of negative edges. Results are presented in Fig. 3. The optimum values of λ corresponding to different L and p are shown in Table 2. For MovieLens and Netflix, as the training data gets sparser, the optimal value of λ goes from negative to positive. For the very sparse Amazon data, the optimal value of λ is always positive. This surprising result displays a completely different scenario from the traditional understanding, suggesting that negative ratings can play positive role in information filtering. The reason for the pimples in Fig. 3 is that when $\lambda = 0$, all the negative links are removed, so the resulting bipartite network is much sparser compared with the ones with nonzero λ, and thus a slight fluctuation of the $F1$ value appears. Specifically, if the negative ratings play a negative role, a peak will appear at $\lambda = 0$ (e.g., the Netflix dataset when $p = 0.2$). On the contrary, if the negative ratings play a positive role, there will be a downward hollow at $\lambda = 0$ (e.g., the cases for the Amazon dataset). As mentioned above, negative ratings indicate not only disfavor but also relevance. When the training set is dense, the relevance information is less important and thus the optimal value of λ is negative, while for the sparse training set, the significance of relevance information results in positive optimal λ, suggesting that much valuable information will be lost if we thoughtlessly remove these negative edges.

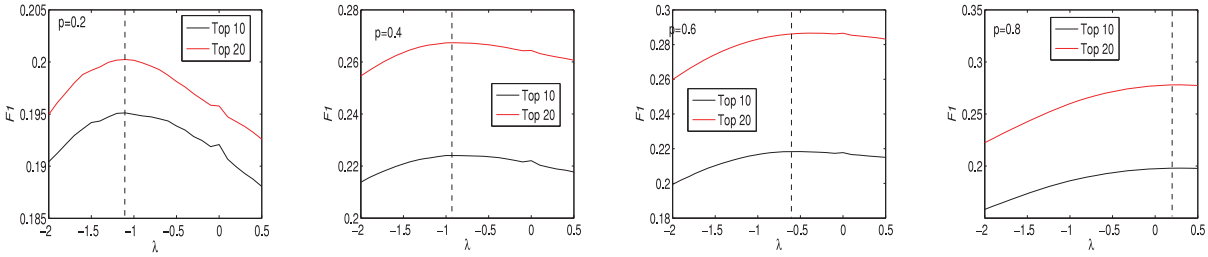
4.3. Microscopic analysis

In this subsection, we investigate the contribution of negative edges microscopically. By the aforementioned way, each data set is divided into a training set and a testing set. The ratio of edges in the testing set to the total edges is fixed to 0.2 and only the ratings no less than 3 are conserved in the testing set for testing the algorithm's accuracy.

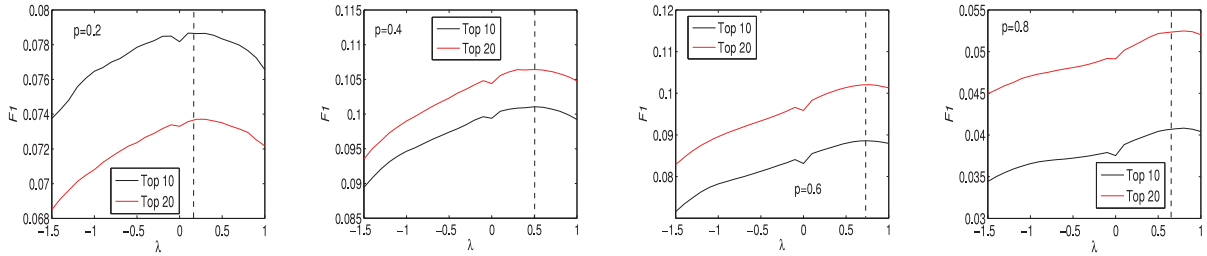
Let R^L be the set of negative edges in the training set. All the negative edges in R^L are sorted in the descending order of $k(u_i) \times k(o_\alpha)$, where $k(u_i)$ and $k(o_\alpha)$ denote the degree of user u_i and object o_α, respectively. An edge connecting an active user and a popular object will have a higher degree product, and thus we call it the *popularity* of an edge, which is known to be able to quantify an edge's functional significance in percolation [41], synchronization [42] and transportation [43]. The set R^L is further divided into five subsets with almost equal sizes, namely R_1^L, R_2^L, R_3^L, R_4^L and R_5^L. That is to say, if $1 \leq i < j \leq 5$, every edge in R_i^L has higher popularity than any edge in R_j^L. For a certain $i \in \{1, 2, 3, 4, 5\}$, the weight of edges in R_i^L is set



(a) MovieLens.

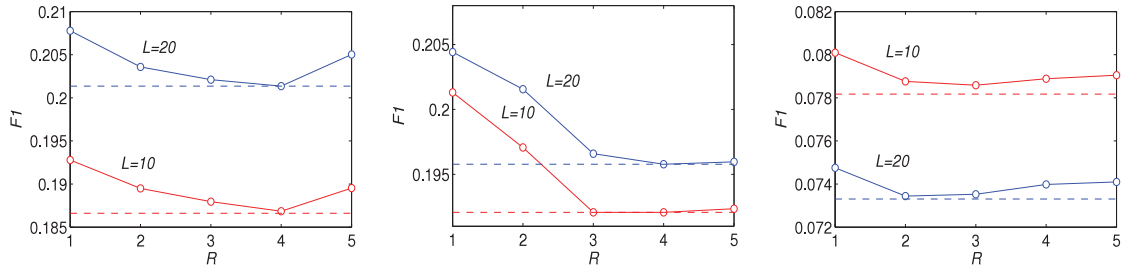


(b) Netfix.



(c) Amazon.

Fig. 3. (Color online) The $F1$ measure as a function of p, where p denotes the ratio of the edges in the testing set to the total edges. A larger p corresponds to a sparser training set. The lengths of recommendation lists are specified as 10 and 20. The vertical dashed line in each graph corresponds to the optimal value of λ for the top 10.



(a) MovieLens. (b) Netfix. (c) Amazon.

Fig. 4. (Color online) The optimal value of the $F1$ measure corresponding to different subsets R_i^L.

as λ, the weight of edges in $R_j^L (j \neq i)$ is set as 0, and the weight of positive edges is set as 1. In this way, one can see the contribution of negative edges only in the subset R_i^L.

The results are presented in Fig. 4 and Table 3. In Fig. 4, the dashed lines denote the $F1$ value when $\lambda = 0$, namely all negative edges are neglected. The circles denote the optimal values of $F1$ taking into account the negative edges only in the subset $R_i^L (i = 1, 2, 3, 4, 5)$, with corresponding optimal values of λ presented in Table 3. As shown in Table 3, one can see that the negative edges between active users and popular objects (i.e., edges of high popularity) tend to play a negative role while those between inactive users and unpopular objects (i.e., edges of low popularity) are very likely to play a positive role in recommendation. This again supports our analysis about the information carried by negative edges, because the relevance information is more valuable for the inactive users and unpopular objects where the known information is deficient.

5. Conclusion and discussion

The design of recommender systems has attracted much attention from computer scientists for decades. Driven by the increasing requirement of algorithmic performance, many advanced machine learning methods, as well as the hybrid and ensemble learning frameworks have been developed recently. An ensemble learning framework may consist of thousands of individual algorithms, each of which may contain thousands of parameters. The overwhelming emphasis on the algorithmic performance continuously improves the accuracy of recommender systems, yet adds little insight to the understanding of these systems. In contrast, we are interested in not only improving the accuracy of recommendation algorithms, but also providing insight of recommender systems. For physicists, the latter may be even more significant.

In the traditional algorithms, especially collaborative filtering methods based on the Pearson correlation, the negative ratings play a negative role in recommending. That is to say, if similar users tend to give negative ratings to an object, this object will never be recommended. However, this paper reveals a completely different scenario where negative ratings can play a positive role in recommending. By tuning the sparsity of the training set, we show that when the data are dense, the negative edges have negative effects while in the sparse case, negative edges should be assigned positive weight. This result suggests that the negative ratings play a mixing role, simultaneously indicating the disfavor and relevance. The relevance information gets more significant when the data set is sparser, and thus speaking overall they should be assigned a positive weight when the training set is very sparse. Further analysis at the individual level shows that the negative edges connecting inactive users and unpopular objects tend to play a positive role while those connecting active users and popular objects tend to play a negative role. This again supports our idea about the mixing role embedded in the negative edges, since the relevance information is obviously more significant for inactive users and unpopular objects.

In a word, the negative ratings simultaneously indicate disfavor and relevance. For recommendation, the former is negative while the latter is positive, and thus negative ratings may play either a negative or positive role, depending on the sparsity of training set and the popularity of the corresponding edges. Therefore, thoughtlessly removing negative edges or assigning a negative weight to them may waste valuable information and lead to less accurate recommendations. The negative ratings are more significant for sparse data sets and for small-degree users/objects. The former is related to the so-called sparsity problem while the latter to the cold-start problem. These are two long-term challenges in the study of recommender systems [10]. We hope our analysis could contribute to the in-depth understanding of recommender systems, as well as the final solution of the sparsity and cold-start problems.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China under Grant Numbers 60973069 and 11075031. L.L. acknowledges the Swiss National Science Foundation 200020-121848.

References

[1] G.Q. Zhang, G.Q. Zhang, Q.F. Yang, S.Q. Cheng, T. Zhou, New J. Phys. 10 (2008) 123027.
[2] L.A. Adamic, B.A. Huberman, Commun. ACM 44 (9) (2001) 55.
[3] A.N. Langville, C.D. Meyer, Google's PageRank and Beyond: The Science of Search Engine Rankings, Princeton University Press, Princeton, 2008.
[4] N.J. Belkin, Commun. ACM 43 (2000) 58.
[5] T. Zhou, Z. Kuscsik, J.G. Liu, M. Medo, J.R. Wakeling, Y.C. Zhang, Proc. Natl. Acad. Sci. USA 107 (2010) 4511.
[6] J. Schafer, J. Konstan, J. Riedl, Data Min. Knowl. Discov. 5 (2001) 115.
[7] G. Linden, B. Smith, J. York, IEEE Internet Computing 7 (2003) 76.
[8] D. Billsus, C.A. Brunk, C. Evans, B. Gladish, M. Pazzani, Commun. ACM 45 (2002) 34.
[9] K. Ali, Stam W. van, Proc. 10th ACM SIGKDD Conference, ACM Press, New York, 2004, p. 394.
[10] G. Adomavicius, A. Tuzhilin, IEEE Trans. Knowl. Data Eng. 17 (2005) 734.
[11] J. Schafer, D. Frankowski, J.L. Herlocker, S. Sen, Lecture Notes in Comput. Sci. 4321 (2007) 291.
[12] X. Su, T.M. Khoshgoftaar, Adv. Artif. Intell. (2009) 421425.
[13] B. Sarwar, G. Karypis, J. Konston, J. Riedl, Proc. ACM WebKDD Workshop, ACM Press, New York, 2000.
[14] A. Paterek, Proc. KDD Cup Workshop, ACM Press, New York, 2007, p. 39.
[15] M. Pazzani, D. Billsus, Lecture Notes in Comput. Sci. 4321 (2007) 325.
[16] T. Hofmann, ACM Trans. Inf. Syst. 22 (2004) 89.
[17] D.M. Blei, A.Y. Ng, M.T. Jordan, J. Machine Learn. Res. 3 (2003) 993.
[18] K. Goldberg, T. Roeder, D. Gupta, C. Perkins, Inf. Retr. 4 (2001) 133.
[19] S. Maslov, Y.C. Zhang, Phys. Rev. Lett. 87 (2001) 248701.
[20] J. Barre, J. Stat. Mech. (2007) P08015.
[21] J. Ren, T. Zhou, Y.C. Zhang, EPL 82 (2008) 58007.
[22] Y.C. Zhang, M. Medo, J. Ren, T. Zhou, T. Li, F. Yang, EPL 80 (2007) 68003.
[23] T. Zhou, J. Ren, M. Medo, Y.C. Zhang, Phys. Rev. E 76 (2007) 046115.
[24] T. Zhou, L.L. Jiang, R.Q. Su, Y.C. Zhang, EPL 81 (2008) 58004.
[25] T. Zhou, R.Q. Su, R.R. Liu, L.L. Jiang, B.H. Wang, Y.C. Zhang, New J. Phys. 11 (2009) 123008.
[26] L. Lü, W. Liu, Phys. Rev. E 83 (2011) 066119.
[27] Y.C. Zhang, M. Blattner, Y.K. Yu, Phys. Rev. Lett. 99 (2007) 154301.
[28] R. Burke, User Model. User-Adap. Interact. 12 (2002) 331.
[29] W. Zeng, M.S. Shang, Q.M. Zhang, L. Lü, T. Zhou, Int. J. Mod. Phys. C 21 (2010) 1217.
[30] P. Polikar, IEEE Circ. Syst. Magazine 6 (3) (2006) 21.
[31] J. Liu, G.S. Deng, Physica A 388 (2009) 3643.
[32] Z.K. Zhang, T. Zhou, Y.C. Zhang, Physica A 389 (2010) 179.
[33] M. Blattner, Y.C. Zhang, S. Maslov, Physica A 373 (2007) 753.

[34] Z. Huang, H. Chen, D. Zeng, ACM Trans. Inf. Syst. 22 (2004) 89.
[35] Z.K. Zhang, C. Liu, Y.C. Zhang, T. Zhou, EPL 92 (2010) 28002.
[36] M.S. Shang, L. Lü, Y.C. Zhang, T. Zhou, EPL 90 (2010) 48006.
[37] M.S. Shang, L. Lü, W. Zeng, Y.C. Zhang, T. Zhou, EPL 88 (2009) 68008.
[38] J.L. Herlocker, J.A. Konstan, K. Terveen, J.T. Riedl, ACM Trans. Inf. Syst. 22 (2004) 5.
[39] R.V. Guha, R. Kumar, P. Raghavan, A. Tomkins, Proc. 13th WWW, New York, USA, 2004, p. 403.
[40] J. Leskovec, D. Huttenlocher, J. Kleinberg, Proc. 19th WWW, Raleigh, North Carolina, USA, 2010, p. 641.
[41] P. Holme, B.J. Kim, C.N. Yoon, S.K. Han, Phys. Rev. E 65 (2002) 056109.
[42] C.Y. Yin, W.X. Wang, G.R. Chen, B.H. Wang, Phys. Rev. E 74 (2006) 047102.
[43] G.Q. Zhang, D. Wang, G.J. Li, Phys. Rev. E 76 (2007) 017101.