Information Sciences 378 (2017) 444–461

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Regression-based three-way recommendation

Heng-Ru Zhang[a], Fan Min[a,*], Bing Shi[b]

[a] School of Computer Science, Southwest Petroleum University, Chengdu 610500, China
[b] College of Computer Science, Sichuan University, Chengdu 610065, China

ARTICLE INFO

Article history:
Received 19 August 2015
Revised 6 January 2016
Accepted 6 March 2016
Available online 25 March 2016

Keywords:
Granular computing
Numerical prediction
Promotion cost
Regression
Three-way decision

ABSTRACT

Recommender systems employ recommendation algorithms to predict users' preferences to items. These preferences are often represented as numerical ratings. However, existing recommender systems seldom suggest the appropriate behavior together with the numerical prediction, nor do they consider various types of costs in the recommendation process. In this paper, we propose a regression-based three-way recommender system that aims to minimize the average cost by adjusting the thresholds for different behaviors. This is undertaken using a step-by-step approach, starting with simple problems and progressing to more complex ones. First, we employ memory-based regression approaches for binary recommendation to minimize the loss. Next, we consider misclassification costs and adjust the approaches to minimize the average cost. Finally, we introduce coupon distribution action with promotion cost, and propose two optimal threshold-determination approaches based on the three-way decision model. From the viewpoint of granular computing, a three-way decision is a good tradeoff between the numerical rating and binary recommendation. Experimental results on the well-known MovieLens data set show that threshold settings are critical to the performance of the recommender, and that our approaches can compute unique optimal thresholds.

1. Introduction

Recommender systems have been studied extensively to manage items, such as movies [8,22,24] and music [1,49,70]. One of the most successful technologies for recommender systems is memory-based collaborative filtering (CF) [16], which uses a database of user preferences to predict additional topics or products that may appeal to a new user. These preferences are typically expressed as numerical ratings. Many CF approaches have been designed to minimize mean absolute error (MAE) [55]. However, as indicated in [16], minimizing MAE can produce a so-called "magic" barrier, where natural variability prevents obtaining good accuracy. In practice, the aim of recommender systems is to present to the user a reasonable suggestion rather than a numerical prediction.

Granular computing is a general computational theory for using granules such as classes, clusters, subsets, groups, and intervals to build an efficient computational model for complex applications [58]. Rough set is a leading special case of granular computing approach [30]. Three-way decision [31,32,64,68] is an extension of decision theoretical rough sets [57,62,69] for dealing with situations in which three different decisions can be made, namely, accept, reject, and wait-and-see. Within the trisecting-and-acting framework [67], three-way decision is described as two separated tasks of trisecting

* Corresponding author. Tel.: +86 135 4068 5200.
E-mail address: minfanphd@163.com (F. Min).

http://dx.doi.org/10.1016/j.ins.2016.03.019
0020-0255/© 2016 Elsevier Inc. All rights reserved.

and acting. With respect to trisecting [67], a universal set is divided into three regions as regions I, II, and III, respectively. With respect to acting [67], there are strategies I, II, and III, respectively. Recently, there is a trend to applying three-way decision to different applications, such as email spam filtering [77], risk decision making [27], face recognition [26], concept lattices [42] and recommender system [2,74].

In this paper, we propose a regression-based three-way recommender system, the aim of which is to minimize the average cost by adjusting the thresholds for different behaviors. We are essentially dealing with three problems, where the last problem is more general than the first. The first problem is regression-based binary recommendation. The regression subtask is fulfilled using the slope one [25] or k-nearest neighbors (kNN) [44] algorithm to predict the ratings. To convert the numerical prediction into a binary recommendation, a threshold is needed. An item with an above-threshold rating is recommended, while one with a below-threshold rating is not. We design a threshold learning approach to determine the threshold r_t^* minimizing classification loss.

The second problem involves misclassification costs [14,23] corresponding to incorrect recommendation behavior, including recommending items to users who dislike them, and non-recommending items to users who like them. In existing works, misclassification cost is the most widely considered cost since classification is one of the main tasks in data mining (see, e.g., [12,20,78]). Because misclassification costs are considered, the work essentially involves cost-sensitive learning [14,34–36,79]. A cost-sensitive learning approach is designed to determine the optimal threshold r_t^c according to the misclassification costs. Naturally, the objective is to minimize the average misclassification cost.

The last, but crucial problem introduces the coupon distribution action, including promotion cost, to enrich recommender behavior. Promotion cost derives from consultation with the user about the actual decision. We propose optimal threshold determination approaches based on the three-way decision model. This kind of decision often begins with a cost matrix including misclassification and delay costs. In our scenario, we consider promotion cost instead of delay cost. Consequently, we have three actions, namely, recommend, non-recommend, and promote. Determining the threshold pair (r_l^*, r_h^*) involves three steps. First, two parameters, α^* and β^*, are computed according to the cost matrix. Second, the probability PR that the user likes an item is predicted using the slope one or kNN algorithm. Third, the threshold pair is determined based on α^*, β^*, and PR. If the prediction for an item is greater than r_h^*, the item is recommended to the user, while a prediction less than r_l^* results in the item not being recommended. Otherwise, we consider user tendency, which incurs promotion cost.

In our scenario, numerical prediction is exceedingly fine for the recommendation, while binary recommendation is rather coarse, with three-way decision a good tradeoff between these. From the viewpoint of granular computing [29,53,56,59,72], three-way decision has good granularity.

Experimental results, obtained using the well-known MovieLens data set (http://www.movielens.org/), show that: 1) the loss of regression-based binary recommendation (where the minimum loss of the slope one algorithm is obviously lower than that of the kNN one) is a convex function with respect to threshold r_t, and has a unique minimum; 2) the misclassification cost settings directly influence the optimal setting of the recommendation threshold r_t^c, where the average cost considering unequal misclassification costs is obviously lower than that considering equal misclassification costs; and 3) the optimal threshold (r_l^*, r_h^*)-pair determined by three-way decision is optimal not only on the training set, but also on the testing set. With the introduction of promotion cost, the three-way approach often achieves a significantly lower average cost compared with the two-way approach.

The rest of the paper is organized as follows. Section 2 presents some preliminary knowledge including the rating system and memory-based recommendation. Sections 3–5 discuss regression-based binary recommendation, misclassification cost minimizing recommendation, and three-way-decision-based recommendation, respectively. Section 6 presents the experimental results on the MovieLens data set for the three models. Finally, our conclusions are given in Section 7.

2. Related works

Collaborative filtering recommender systems usually use the rating system as input, and recommender accuracy as a kind of evaluation metric. Our recommendation behavior considers both misclassification and promotion costs. Through cost-sensitive learning, we build proper classifiers to find the minimum average cost.

2.1. Rating system

First, we revisit the rating system proposed in [75]. Let $U = \{u_0, u_1, \ldots, u_{n-1}\}$ be the set of users of a recommender system and $V = \{t_0, t_1, \ldots, t_{m-1}\}$ be the set of all possible items that can be recommended to users. Then, the rating function is given by

$$R : U \times V \to V_k, \tag{1}$$

where V_k is the rating domain used by the users to evaluate items, and r_w and r_g are the lowest and highest ratings, respectively. For convenience, we represent the rating system with an $n \times m$ rating matrix $R = (r_{i,j})_{n\times m}$, where $r_{i,j} = R(u_i, t_j)$, $0 \le i \le n-1$, and $0 \le j \le m-1$.

Example 1. An example rating system is depicted in Table 1, where $V_k = \{1, 2, 3, 4, 5\}$. In Table 1, some elements are zero, indicating that the users do not watch the corresponding movies.

Table 1
Rating matrix(R).

UID ╲ TID	t_0	t_1	t_2	t_3	t_4
u_0	1	2	0	0	1
u_1	0	3	2	0	0
u_2	0	4	5	0	0
u_3	4	0	0	5	4
u_4	0	1	0	5	5
u_5	0	0	3	2	2

Given a rating system, we often test the performance of a regression approach using the leave-one-out scheme. For each $r_{i,j} \neq 0$, we predict its value using all the other data in R. The predicted value is denoted as $p_{i,j}$. In this way, we obtain the prediction matrix $P = (p_{i,j})_{n\times m}$. $p_{i,j} = 0$ if $r_{i,j} = 0$, indicating that the respective value is unknown/invalid. The MAE of the predictor is defined as

$$mae(P, R) = \frac{\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}|p_{i,j} - r_{i,j}|}{|\{(i, j) \in \{0..n-1\} \times \{0..m-1\} | r_{i,j} > 0\}|}. \tag{2}$$

Many regression approaches essentially deal with the following problem.

Problem 2. MAE minimizing regression problem.
Input: R
Output: P
Optimization objective: $mae(P, R)$

Naturally, the minimal possible value of $mae(P, R)$ is zero. However, because there is a "magic" barrier [16] in the data set, we have no way of knowing the true minimal MAE of an application.

2.2. Slope one predictor

Slope one is an item-based collaborative filtering recommendation algorithm based on linear regression [28] that determines the extent by which users prefer one item to another. It uses a simple formula that merely subtracts the average rating of the two items to determine the deviation. Then, given a user's ratings of certain items, the deviation can be used to predict the user's ratings of other items.

The underlying principle of the slope one algorithm is the use of linear regression for prediction [25]. The predictor takes the form $f(x) = x + b$, where the free parameter b is the average deviation of the ratings of two users or items. For many examples, the predictor is more accurate and faster than linear regression of $f(x) = ax + b$, while the algorithm requires at most half the amount of storage.

The prediction process involves two steps: (1) calculate the average deviation $dev_{j,i}$ of the target item t_j from another item t_i; and (2) predict the rating $p_{k,j}$ of the target item t_j by the currently active user u_k.

$S_{i,j}$ is the set of users rating both items t_i and t_j. Given two items t_i and t_j with ratings $r_{k,i}$ and $r_{k,j}$ respectively, the average deviation $dev_{j,i}$ is calculated as

$$dev_{j,i} = \sum_{k\in S_{i,j}} \frac{r_{k,j} - r_{k,i}}{|S_{i,j}|}. \tag{3}$$

Q_j is the item set rated by the target user. Item rating $p_{k,j}$ can be predicted as

$$p_{k,j} = \frac{1}{|Q_j|}\sum_{i\in Q_j}(dev_{j,i} + r_{k,i}) \tag{4}$$

Slope one is adaptive to data sparsity and can generate effective recommendations in real-time [25]. Moreover, it is extensible and easy to realize. It is used in many online recommendation systems [28], such as DVD and MP3 recommendation systems.

2.3. Neighbor-based collaborative filtering recommendation

Neighbor-based collaborative filtering recommendation [50] predicts the preferences of active users to items using only the rating matrix. The fundamental assumption of neighbor-based collaborative filtering is that if users u_x and u_y have similar behaviors (e.g., buying, watching, listening) or rate n items similarly, they are likely to rate other items similarly or act the same in other situations [50]. Neighbor-based collaborative filtering recommendation usually uses the measurement of similarity to obtain the distance between two users or two items. Similarity measures such as Pearson's correlation coefficient [9], vector space similarity [4], and cosine-based similarity [11,19,44] are widely used.

There are two kinds of neighbor-based collaborative filtering methods [33,45,54], namely, user- and item-based collaborative filtering. The basic idea of user-based collaborative filtering is to recommend an item to a user based on other like-minded users' opinions of that item. The user-based collaborative filtering approach first finds a set of nearest "neighbors" (similar users), sharing similar favorite items or interests. Then, a user's rating of an unrated item is predicted based on the ratings of that item given by the user's "neighbors". Contrarily, the item-based collaborative filtering approach recommends an item to a user based on other items with high correlations. Unlike user-based collaborative filtering, the item-based collaborative filtering approach first finds a set of nearest "neighbors" (similar items) for each item. Item-based collaborative filtering recommender systems attempt to predict a user's rating of an item based on his/her ratings of neighboring items to the target item.

Collaborative filtering recommender systems frequently use the *k*NN [40,44,73] algorithm to identify candidate items. In the *k*NN method, the top-*k* recommendations are generated only from items "liked" by a subset of users who are "closest" (less than a certain distance) to the target user and not from all items. The *k*NN algorithm is a non-parametric method used for classification and regression. In *k*NN regression, the output is the property value of the object; that is, the average of the values of its *k* nearest neighbors. In the classification phase, *k* is a user-defined constant, where the best choice for *k* depends on the data. Generally, larger values of *k* reduce the effect of noise on the classification. A shortcoming of the *k*NN algorithm is that it is sensitive to the local structure of the data.

2.4. Granular computing and three-way decision

Granular computing [39,41,66] focuses on a set of philosophy, methodology and paradigm for structured thinking, structured problem solving and structured information processing at multiple levels of granularity [63]. Granular structures consist of many hierarchies for multiview descriptions of a problem, with each hierarchy being composed of multiple levels of abstraction [63]. Zadeh [71,72] considers granular computing as a basis for computing with words. Yao [60,61] views granular computing as a complementary and dependant triangle: structured thinking within the philosophical perspective, structured problem solving within the methodological perspective, and structured information processing within the computational perspective. Skowron and Stepaniuk [18,48] view granular computing from rough set point of view. Rough approximations are used to model syntax, semantics, and operations of information granules. Recently, granular computing and rough sets have been widely applied to data mining [6,15,17,37,43].

Three-way decision may be related to a basic principle of granular computing [66]. In rough set theory, three-way decision focuses on a more general class of problems where a set of objects are divided into three pair-wise disjoint I, II, and III regions [7,65,67]. Decisions of acceptance, non-commitment or rejection are strategies for the three regions. When the available information is insufficient or the evidence is not strong enough to support an acceptance or a rejection at a particular level of granularity, a third option of non-commitment allows us to defer a decision to the next level of granularity [66].

The starting point of three-way decision is often a cost matrix with misclassification and delay costs. In this work, we consider promotion cost instead of delay cost. Promotion cost is incurred through buying decisional data from the user. The recommendation action involved in paying promotion cost forms an active learning scenario. Active learning guides the acquisition of new knowledge suitable for updating rated or browsed information [21,47,51] in time. Turney [52] used inductive concept learning to create a taxonomy of the different types of cost, and discussed the expected cost of classifying the new instance itself versus the cost of asking a teacher to classify the new instance. Both promotion and test costs are incurred when buying data; they differ in that promotion cost is incurred to obtain a decision or result, whereas test cost relates to obtaining an attribute value.

3. Regression-based binary recommendation

We present a two-step approach for regression-based binary recommendation. In the first step, the predication matrix P is computed using a regression approach. Because existing approaches, such as the slope one and *k*NN algorithms, are discussed in Section 2, we do not discuss this in detail here. In the second step, we transfer P to binary recommendations. This is undertaken by computing a threshold for converting numerical prediction to binary recommendation. For simplicity, throughout this section we assume that $0 \le i \le n-1$ and $0 \le j \le m-1$.

3.1. Problem statement

Because we present binary recommendations instead of numerical ones, we need to evaluate whether the recommendation is appropriate. Some researchers assume that a rating of 4 or 5 indicates that the user likes the item, whereas others assume that only a rating of 5 indicates like. Let l_t denote the rating threshold for like and Y and N indicate that the user likes and dislikes the item, respectively. The actual rating matrix can be transferred to the interest matrix $UM = (um_{i,j})_{n\times m}$, where

$$um_{i,j} = \begin{cases} Y, & \text{if } r_{i,j} \ge l_t; \\ N, & \text{if } 0 < r_{i,j} < l_t; \\ unknown, & \text{otherwise.} \end{cases} \quad (5)$$

Our algorithm is based on regression approaches providing numerical prediction of rating. Here we discuss how such predictions are converted to binary suggestions. Let the prediction matrix be $P = (p_{i,j})_{n\times m}$, where $p_{i,j}$ is the predicted rating by u_i of t_j, and $p_{i,j} = 0$ indicates an invalid prediction. Invalid prediction corresponds to zero values in the actual rating matrix. Unlike actual rating, which has only five discrete scores in our example, the domain of $p_{i,j}$ is continuous. Assuming that ratings of 4 and 5 indicate like, if $p_{i,j} = 3.8$, we cannot determine whether to recommend the item to the user. Thus, we need to find another appropriate threshold.

Let r_t denote the recommendation threshold, and e_R and e_N be the recommend and non-recommend behaviors, respectively. The prediction matrix P is transferred to the recommendation matrix $RM = (rm_{i,j})_{n\times m}$, where

$$rm_{i,j} = \begin{cases} e_R, & \text{if } p_{i,j} \ge r_t; \\ e_N, & \text{if } 0 < p_{i,j} < r_t; \\ invalid, & \text{otherwise.} \end{cases} \tag{6}$$

Based on l_t, r_t, and R, objects in P are classified into four regions, namely, true recommendation (RY), false recommendation (RN), false non-recommendation (NY), and true non-recommendation (NN). Given that $l_t = 4$ and $r_t = 3.5$, with $r_{i,j} = 5$ and $p_{i,j} = 3.8$, object $\langle i, j\rangle$ is classified into RY. The numbers of objects in the four regions are given by

$$\begin{aligned} RY(R,P,l_t,r_t) &= |\{\langle i,j\rangle | r_{i,j} \ge l_t, p_{i,j} \ge r_t\}|; \\ RN(R,P,l_t,r_t) &= |\{\langle i,j\rangle | 0 < r_{i,j} < l_t, p_{i,j} \ge r_t\}|; \\ NY(R,P,l_t,r_t) &= |\{\langle i,j\rangle | r_{i,j} \ge l_t, 0 < p_{i,j} < r_t\}|; \\ NN(R,P,l_t,r_t) &= |\{\langle i,j\rangle | 0 < r_{i,j} < l_t, 0 < p_{i,j} < r_t\}|. \end{aligned} \tag{7}$$

The total number of nonzero objects in the actual matrix R is given by

$$AN(R) = |\{\langle i,j\rangle | r_{i,j} > 0\}|. \tag{8}$$

Our regression approaches aim to minimize the loss, also called the misclassification rate in some literature (see, e.g., [5,38]). The loss is computed based on the numbers of objects in the four regions.

$$ls(R,P,l_t,r_t) = \frac{RN(R,P,l_t,r_t) + NY(R,P,l_t,r_t)}{AN(R)}. \tag{9}$$

Once the prediction matrix P has been computed, we are faced with the problem of determining the recommendation threshold such that the loss is minimized.

Problem 3. Optimal recommendation threshold problem.
Input: R, P, and l_t
Output: r_t
Optimization objective: min $ls(R, P, l_t, r_t)$

As P is already given, the minimal value of $ls(R, P, l_t, r_t)$ is deterministic in practice. This is different from Problem 2 where the minimal possible value of MAE is non-deterministic. We denote

$$r_t^* = \arg\min_{r_w \le r_t \le r_g} ls(R,P,l_t,r_t). \tag{10}$$

As can be seen in Fig. 2, r_t^* is usually unique.

3.2. Threshold determination for loss minimization

Algorithm 1 illustrates the entire process from the training set to determining r_t^*. To analyze the time complexity of Algorithm 1, let z be the number of non-zero ratings in the actual matrix R.

In Step 1, P is predicted by the leave-one-out scheme using the slope one or kNN algorithm (lines 1–2). This step is the most time-consuming one in the entire algorithm. When using the slope one algorithm, the worst case time complexity of this step is $O(z \times n \times m)$ [3]. Alternatively, when using the kNN algorithm, the corresponding time complexity is also $O(z \times n \times m)$ [76].

In Step 2 (lines 3–20), the trichotomy method [13] is employed to find r_t^* because ls is a convex function w.r.t. r_t as shown in Fig. 2. There are three aspects of the trichotomy method: (1) the rating domain V_k is divided into three parts: $[r_w, md]$, (md, mmd), and $[mmd, r_g]$; (2) $ls(R, P, l_t, md)$ and $ls(R, P, l_t, mmd)$ are computed based on Eq. (9); and (3) the optima is obtained when the difference between md and mmd is less than ε.

In Step 2.1, the numbers of objects in the four regions are counted (line 9). It should be noted that zero ratings are invalid and are therefore never counted. For each md or mmd, the time cost of classification is z, and therefore the time complexity of this step is $O(z)$.

In Step 2.2, we compute the loss for two thresholds md and mmd based on Eq. (9) (lines 11–12). If md is close to the optimal recommendation threshold r_t^*, then $right = mmd$; otherwise, $left = md$ (lines 13–17). We assume that the number of step lengths between r_w and r_g is q. The time cost for solving the extremum problem based on the trichotomy method is $O(\log q)$.

Algorithm 1 Regression-based recommendation.

Input: R, l_t
Output: r_t^*
Method: regressionBasedRecommendation

```
1: //Step 1. Prediction
2: Obtain the prediction matrix $P$ according to the actual matrix $R$;
3: //Step 2. Compute the optimal recommendation threshold $r_t^*$
4: $left = r_w, right = r_g$;
5: while $|right - left| > \varepsilon$ do
6:     $md = \frac{left+right}{2}$;
7:     $mmd = \frac{md+right}{2}$;
8:     //Step 2.1. Classification and count
9:     count the number of objects in the four regions using Equation (7);
10:    //Step 2.2. Compute the loss
11:    $ls(R, P, l_t, md) = \frac{RN(R,P,l_t,md)+NY(R,P,l_t,md)}{AN(R)}$;
12:    $ls(R, P, l_t, mmd) = \frac{RN(R,P,l_t,mmd)+NY(R,P,l_t,mmd)}{AN(R)}$;
13:    if $ls(R, P, l_t, md) < ls(R, P, l_t, mmd)$ then
14:        $right = mmd$;
15:    else
16:        $left = md$;
17:    end if
18: end while
19: //Step 2.3. Output $r_t^*$
20: $r_t^* = left$;
21: return $r_t^*$;
```

Table 2
Prediction matrix (P).

UID ╲ TID	t_0	t_1	t_2	t_3	t_4
u_0	1.5	1	0	0	3
u_1	0	1	4	0	0
u_2	0	5	3	0	0
u_3	4.5	0	0	4	4.5
u_4	0	5	0	3	2.75
u_5	0	0	2	2.75	2.25

In Step 2.3, when the difference between *left* and *right* reaches a preset error range (e.g., $\varepsilon = 0.01$), we can output *left* or *right* as r_t^* (line 20). The time complexity of this step is $O(1)$. Therefore, the time complexity of Step 2 is $O(z \times \log q)$.

To summarize, if the prediction algorithm uses slope one, the time complexity of Algorithm 1 is

$$O(z \times n \times m) + O(z \times \log q) = O(z \times n \times m) \tag{11}$$

since $\log q < n \times m$. If the prediction algorithm uses kNN, the time complexity of Algorithm 1 is

$$O(z \times n \times m) + O(z \times m) + O(z \times \log q) = O(z \times n \times m). \tag{12}$$

In other words, the time complexity depends only on Step 1.

3.3. Working example #1

In this subsection, we explain regression-based binary recommendation with the aid of a running example.

In Step 1, we employ the slope one algorithm to predict the test object based on the actual matrix R as shown in Table 1 with leave-one-out cross validation. The prediction matrix P is given in Table 2.

In Step 2, let the like threshold l_t be 4. Because the example is a small data set, ε is set to one. For ease of explanation, we list the regression-based classification in Table 3 and the loss in Table 4 for the different thresholds. For example, the loss is given by $\frac{4}{16} = 0.25$ when $r_t = 2.3 \sim 3.0$. It should be noted that zero ratings are invalid and therefore are never counted. We employ the trichotomy method to compute the optimal recommendation threshold r_t^*. The computed process is shown in Table 5. Finally, we determine the optimal recommendation threshold r_t^* as 2.6875.

Table 3
Regression-based classification.

r_t	Behavior	Y	N
1.0	e_R	7	9
	e_N	0	0
$1.1 \sim 1.5$	e_R	7	7
	e_N	0	2
$1.6 \sim 2.0$	e_R	7	6
	e_N	0	3
$2.1 \sim 2.2$	e_R	7	5
	e_N	0	4
$2.3 \sim 2.7$	e_R	7	4
	e_N	0	5
$2.8 \sim 3.0$	e_R	6	3
	e_N	1	6
$3.1 \sim 4.0$	e_R	4	2
	e_N	3	7
$4.1 \sim 4.5$	e_R	3	1
	e_N	4	8
$4.6 \sim 5.0$	e_R	1	1
	e_N	6	8
5.1	e_R	0	0
	e_N	7	9

Table 4
Losses for different thresholds.

r_t	$ls(R, P, l_t, r_t)$
1.0	0.56
$1.1 \sim 1.5$	0.44
$1.6 \sim 2.0$	0.38
$2.1 \sim 2.2$	0.31
$2.3 \sim 3.0$	0.25
$3.1 \sim 4.5$	0.31
$4.6 \sim 5.0$	0.44
5.1	0.44

Table 5
Threshold determination process.

$left$	$right$	md	mmd	$ls(R, P, l_t, md)$	$ls(R, P, l_t, mmd)$
1	5	3	4	0.25	0.31
1	4	2.5	3.25	0.25	0.31
1	3.25	2.125	2.6875	0.31	0.25
2.125	3.25	2.6875	2.96875	0.25	0.25
2.6875	3.25	–	–	–	–

4. Misclassification cost minimizing recommendation

In many applications different misclassifications lead to different costs. For recommender systems, incorrect recommendations correspond to misclassifications. Our purpose is to realize a recommender system with low average misclassification cost. We still follow the two-step approach presented in Section 3, with the same focus of determining the recommendation threshold. However, the objective is to minimize the average cost instead of the loss.

4.1. Problem statement

There are two recommendation behaviors, namely, recommend and not recommend. Let r_P be recommendation behavior and r_N be non-recommendation one, respectively. There are also two user preferences, namely, like and dislike. The misclassification cost is represented as a 2×2 matrix as shown in Table 6. In this paper we assume that $\lambda_{RY} = \lambda_{NN} = 0$; that is, correct recommendation incurs no cost.

Our misclassification cost minimization approach aims to minimize the average cost, which is computed using Eq. (7) and Table 6 as

$$am(R, P, l_t, C, r_c) = \frac{RN(R, P, l_t, r_c) \times \lambda_{RN} + NY(R, P, l_t, r_c) \times \lambda_{NY}}{AN(R)}. \tag{13}$$

Table 6
Misclassification cost (C).

Behavior \ Preference	Y	N
e_R	λ_{RY}	λ_{RN}
e_N	λ_{NY}	λ_{NN}

Table 7
Average misclassification costs for different thresholds.

$\frac{\lambda_{NY}}{\lambda_{RN}}$	1.0	1.1 ~ 1.5	1.6 ~ 2.0	2.1 ~ 2.2	2.3 ~ 2.7	2.8 ~ 3.0	3.1 ~ 4.0	4.1 ~ 4.5	4.6 ~ 5.0	5.1
0.25	60.8	47.3	40.5	33.8	27.0	21.9	18.6	13.5	16.9	11.8
0.5	50.6	39.4	33.8	28.1	22.5	19.7	19.7	16.9	22.5	19.7
1	38.0	29.5	25.3	21.1	16.9	16.9	21.1	21.1	29.5	29.5
2	25.3	19.7	16.9	14.1	11.3	14.1	22.5	25.3	36.6	39.4
4	15.2	11.8	10.1	8.4	6.8	11.8	23.6	28.7	42.2	47.3

When the misclassification cost C is assigned in a real application, we are faced with the problem of determining the recommendation threshold, such that the average cost is minimized.

Problem 4. Optimal recommendation threshold problem.
Input: R, P, l_t, and C
Output: r_c
Optimization objective: min $am(R, P, l_t, C, r_c)$

As P and C are already given, the minimal value of $am(R, P, l_t, C, r_c)$ is deterministic in practice. Considering the misclassification cost, this is more general than Problem 3. We denote

$$r_c^* = \arg\min_{r_w \le r_t \le r_g} am(R, P, l_t, C, r_c). \tag{14}$$

As can be seen in Fig. 2, r_c^* is usually unique.

4.2. Misclassification cost minimizing algorithm

Although Problem 4 is more general than Problem 3, the basic characteristics of both algorithms are the same. Therefore, we can employ the same search strategy to find the optimal threshold. We revise Algorithm 1 in the following way to deal with Problem 4. Line 11 is replaced by the computation of $am(R, P, l_t, C, md)$, while line 12 is replaced by the computation of $am(R, P, l_t, C, mmd)$. The computation is given by Eq. (13).

For the sake of brevity, we do not list the algorithm. Naturally, while $\lambda_{RN} = \lambda_{NY}$, the new algorithm coincides with Algorithm 1.

4.3. Working example #2

In this subsection, we explain the misclassification cost minimization recommendation using a running example.

The first step is the same as that in running example #1. The regression-based classification listed in Table 3, is the same as that in running example #1.

In Step 2, the misclassification costs are assigned as $\lambda_{RN} = 90$ and $\lambda_{NY} = 45$. We compute the average cost for different ratios as shown in Table 7 according to Eq. (13). For example, the average cost of $r_c = 4.1$ and $\frac{\lambda_{RN}}{\lambda_{NY}} = 0.5$ is given by

$$am(R, P, 4, C, 4.1) = \frac{1 * 90 + 4 * 45}{3 + 1 + 4 + 8} \approx 16.9.$$

Because determination of the optimal recommendation threshold r_c^* is similar to determination of r_t^* by means of the trichotomy method, we do not discuss this in detail.

5. Three-way-decision-based recommendation

As discussed in Section 2.4, in an active learning scenario, promotion cost is incurred when buying a decision or result from the user. With both misclassification and promotion costs, a three-way-decision model is built. Our purpose is to obtain a recommender system with low average three-way cost. We propose two approaches for determining the optimal recommendation threshold pair: a proportion-based approach and a minimizing search approach. However, the objective is to minimize the average three-way cost instead of the average misclassification cost. As before, we assume $0 \le i \le n - 1$ and $0 \le j \le m - 1$ in this section.

Table 8
Three-way cost (W).

Behavior ╲ Preference	Y	N
e_R	λ_{RY}	λ_{RN}
e_P	λ_{PY}	λ_{PN}
e_N	λ_{NY}	λ_{NN}

5.1. Problem statement

We apply the trisecting-and-acting framework [67] to divide the candidate recommended items V into three pair-wise disjoint regions: recommendation, promotion, non-recommendation. Corresponding to the three regions, we have three recommendation behaviors, namely recommend, promote, and non-recommend, respectively. Let e_P be the promotion behavior. Meanwhile, we define two user preferences, namely like and dislike.

The three-way cost is represented as a 3×2 matrix shown in Table 8. If user dislikes the recommended items or RS does not recommend user's liked items, we pay the misclassification costs λ_{RN} or λ_{NY}, respectively. If the the candidate recommended items fall in the promotion region, we need to provide promotion/educational cost λ_{PY} or λ_{PN} (usually, $\lambda_{PY} = \lambda_{PN}$). In this paper, we assume that $\lambda_{RY} = \lambda_{NN} = 0$; that is, correct recommendation incurs no cost.

Similar to the problems proposed in Sections 3.1 and 4.1, once again, we need to compute thresholds. The difference lies in that we need a three-way threshold (r_l, r_h)-pair instead of only a single threshold. r_l and r_h denote, respectively, the lower and upper thresholds for three-way decision. Given a threshold pair, the computation of the average cost is non-trivial.

In Step 1, we determine whether each object belongs to the recommendation, non-recommendation, or promotive regions by means of a three-way decision. Based on r_l and r_h, we obtain the following decision rules:

(N1) If $p \in (0, r_l]$, the object belongs to the non-recommendation region and the recommendation action is e_N.
(B1) If $p \in (r_l, r_h)$, the object belongs to the promotive region and the recommendation action is e_P.
(P1) If $p \geq r_h$, the object belongs to the recommendation region and the recommendation action is e_R.

In Step 2, we count the numbers of objects in the different regions. The numbers of objects in the six regions are given by

$$\begin{aligned} RY(R, P, l_t, W, r_l, r_h) &= |\{\langle i, j\rangle | r_{i,j} \geq l_t, p_{i,j} \geq r_h\}|; \\ RN(R, P, l_t, W, r_l, r_h) &= |\{\langle i, j\rangle | 0 < r_{i,j} < l_t, p_{i,j} \geq r_h\}|; \\ PY(R, P, l_t, W, r_l, r_h) &= |\{\langle i, j\rangle | r_{i,j} \geq l_t, r_l < p_{i,j} < r_h\}|; \\ PN(R, P, l_t, W, r_l, r_h) &= |\{\langle i, j\rangle | 0 < r_{i,j} < l_t, r_l < p_{i,j} < r_h\}|; \\ NY(R, P, l_t, W, r_l, r_h) &= |\{\langle i, j\rangle | r_{i,j} \geq l_t, 0 < p_{i,j} \leq r_l\}|; \\ NN(R, P, l_t, W, r_l, r_h) &= |\{\langle i, j\rangle | 0 < r_{i,j} < l_t, 0 < p_{i,j} \leq r_l\}|. \end{aligned} \tag{15}$$

In Step 3, the total cost of all regions is computed according to the three-way cost and the number of objects in the respective region. It is given by

$$\begin{aligned} tt(R, P, l_t, W, r_l, r_h) = {} & \lambda_{RY} RY(R, P, l_t, W, r_l, r_h) + \lambda_{RN} RN(R, P, l_t, W, r_l, r_h) \\ & + \lambda_{PY} PY(R, P, l_t, W, r_l, r_h) + \lambda_{PN} PN(R, P, l_t, W, r_l, r_h) \\ & + \lambda_{NY} NY(R, P, l_t, W, r_l, r_h) + \lambda_{NN} NN(R, P, l_t, W, r_l, r_h). \end{aligned} \tag{16}$$

In Step 4, we compute the average three-way cost as

$$at(R, P, l_t, W, r_l, r_h) = \frac{tt(R, P, l_t, W, r_l, r_h)}{AN(R)}. \tag{17}$$

When the three-way cost W is assigned in a real application, we are faced with the problem of determining the three-way recommendation threshold pair that minimizes the average three-way cost.

Problem 5. Three-way recommendation threshold pair problem.
Input: R, P, l_t, W
Output: r_l, r_h
Optimization objective: min $at(R, P, l_t, W, r_l, r_h)$

Because P and W are already given, the minimal value of $at(R, P, l_t, W, r_l, r_h)$ is deterministic in practice. With the introduction of promotion cost, this is more general than Problem 4. We denote

$$(r_l^*, r_h^*) = \arg \min_{r_w \leq r_l \leq r_g, r_w \leq r_h \leq r_g} at(R, P, l_t, W, r_l, r_h). \tag{18}$$

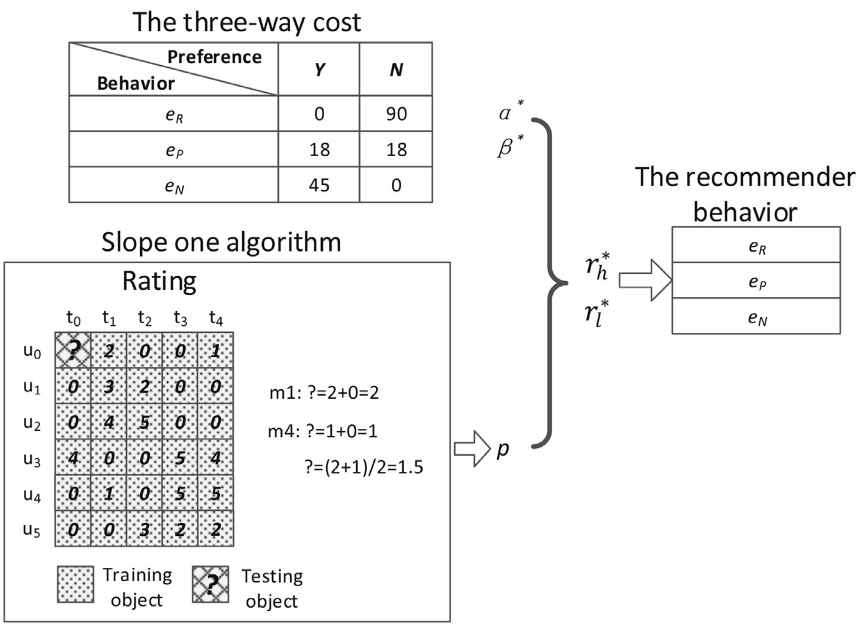

Fig. 1. Framework integrating the three-way decision and slope one algorithm.

5.2. Three-way recommendation framework

To determine the three-way threshold (r_l^*, r_h^*)-pair, we construct the framework of three-way decision-based recommendation depicted in Fig. 1. This is divided into four stages: (1) three-way cost matrix for a real application, including misclassification and promotion costs; (2) prediction for the test set using the slope one or *k*NN algorithm; (3) computation of the threshold (r_l^*, r_h^*)-pair, which is the most important stage in our framework; and (4) recommendation of the prediction using the (r_l^*, r_h^*)-pair.

The three-way cost in our example includes: (1) the cost of correct classification, that is, $\lambda_{RY} = \lambda_{NN} = 0$; (2) two types of promotion costs, that is, $\lambda_{PY} = \lambda_{PN} = 18$; and (3) the cost of recommending an item that a user does not like, $\lambda_{RN} = 90$, as well as that of non-recommending an item that a user likes, $\lambda_{NY} = 45$.

The prediction *p* of each test object is calculated using the slope one or *k*NN algorithm as discussed in Section 2. However, this is not discussed in detail here.

We adopt two approaches for determining the three-way recommendation threshold pair, called the proportion-based approach and minimizing search approach, respectively. In the former approach, if the probability that users like an item is greater than α^*, the corresponding prediction rating is assigned as the recommendation threshold r_h^*. If it is less than β^*, the corresponding prediction is assigned as the non-recommendation threshold r_l^*. In the latter approach, we split the rating system into two parts based on r_t^*, which is obtained in Section 4.2. Each part uses three-way decision to determine the recommendation threshold.

Three-way decision-based recommendation selects a proper recommender action based on the prediction *p*, and threshold (r_l^*, r_h^*)-pair. With the three-way threshold pair, the prediction matrix *P* is transferred to the three-way recommendation matrix $TM = (tm_{i,j})_{n\times m}$, where

$$tm_{i,j} = \begin{cases} e_R, & \text{if } p_{i,j} \geq r_h^*; \\ e_P, & \text{if } r_l^* < p_{i,j} < r_h^*; \\ e_N, & \text{if } 0 < p_{i,j} \leq r_l^*. \end{cases} \tag{19}$$

5.3. Three-way recommendation algorithm

In this subsection, we first compute the three-way parameters (β^*, α^*) based on the three-way cost, and then we use the two approaches to determine the recommendation threshold (r_l^*, r_h^*)-pair.

5.3.1. Computation of three-way parameters

We consider a special kind of cost functions with $\lambda_{RY} \leq \lambda_{PY} < \lambda_{NY}$ and $\lambda_{NN} \leq \lambda_{PN} < \lambda_{RN}$ [69]. That is, the cost of classifying movie *m* in *Y* into the recommendation region is no greater than that of classifying *m* into the promotive region, and both costs are strictly less than the cost of classifying *m* into the non-recommendation region. The reverse cost order is used for classifying a movie in N. We further assume that the cost functions satisfy the condition:

$$\frac{\lambda_{NY}-\lambda_{PY}}{\lambda_{PN}-\lambda_{NN}} > \frac{\lambda_{PN}-\lambda_{NN}}{\lambda_{RN}-\lambda_{PN}}\,. \tag{20}$$

Given the above two assumptions, we obtain the following equations:

$$\begin{aligned}&\lambda_{RY}\times\alpha^* + \lambda_{RN}\times(1-\alpha^*) = \lambda_{PY}\times\alpha^* + \lambda_{PN}\times(1-\alpha^*)\,,\\&\lambda_{RY}\times\beta^* + \lambda_{NN}\times(1-\beta^*) = \lambda_{PY}\times\beta^* + \lambda_{PN}\times(1-\beta^*)\,.\end{aligned} \tag{21}$$

After mathematical conversion, the optimal threshold (β^*, α^*)-pair is computed as [69]

$$\begin{aligned}\alpha^* &= \frac{\lambda_{RN}-\lambda_{PN}}{(\lambda_{RN}-\lambda_{PN})+(\lambda_{PY}-\lambda_{RY})}\,,\\ \beta^* &= \frac{\lambda_{PN}-\lambda_{NN}}{(\lambda_{PN}-\lambda_{NN})+(\lambda_{NY}-\lambda_{PY})}\,,\end{aligned} \tag{22}$$

with $0 \le \beta^* < \alpha^* \le 1$.

5.3.2. Proportion-based approach

In this subsection, we discuss how to determine the recommendation threshold (r_l^*, r_h^*)-pair by constructing the proportion model.

Under the condition of sufficient classification accuracy, we find that the higher the prediction is, the higher is the actual rating. That is, the like proportion increases as the prediction increases.

There are three steps in computing the like proportion. First, we obtain the objects whose predictions are between r_e and $r_e + s$ from the test set. Here s denotes the step length. The collection $ES(r_e, s)$ composed of these objects is given by

$$ES(r_e, s) = \{\langle i, j\rangle \,|\, 0 \le i < n, 0 \le j < m, r_e \le p_{i,j} < r_e + s\}, \tag{23}$$

where $p_{i,j} \in P$. Second, we obtain the objects with two conditions from the training set: 1) their ratings are greater than and equal to the like threshold l_t; and 2) the index pair of each object is also in the collection $P(r_e, s)$. The collection $TS(r_e, s, l_t)$, composed of these objects, is given by

$$TS(r_e, s, l_t) = \{\langle i, j\rangle \in P(r_e, s) \,|\, r_{i,j} \ge l_t\}. \tag{24}$$

Third, having counted the number of objects in the two collections, we compute the proportion as

$$PR(r_e, s, l_t) = \frac{|TS(r_e, s, l_t)|}{|ES(r_e, s)|}. \tag{25}$$

We can determine the recommendation threshold (r_l^*, r_h^*)-pair based on PR, β^*, and α^*. The lower recommendation threshold r_l^* is given by

$$r_l^* = \arg\min_{r_w \le r_e \le r_g} |PR(r_e, s, l_t) - \beta^*|. \tag{26}$$

The upper recommendation threshold r_h^* is given by

$$r_h^* = \arg\min_{r_w \le r_e \le r_g} |PR(r_e, s, l_t) - \alpha^*|. \tag{27}$$

5.3.3. Minimizing search approach

In this subsection, we discuss how to determine the recommendation threshold (r_l^*, r_h^*)-pair using the minimizing search approach.

We can obtain the two-way threshold r_t^* through misclassification cost minimizing recommendation. The r_t^* splits the domain of rating V_k into two intervals of $[r_w, r_t^*]$ and $[r_t^*, r_g]$. The lower threshold r_l^* is calculated based on the interval $[r_w, r_t^*]$, while the upper threshold r_h^* is determined based on the interval $[r_t^*, r_g]$.

The determination of r_l^* includes four steps. First, we determine whether each object belongs to the recommendation, non-recommendation, or promotive region based on a three-way decision. For $\forall r_l \in [r_w, r_t^*]$, we obtain the following decision rules:

(N2) If $p \in [r_w, r_l]$, the object belongs to the non-recommendation region.
(B2) If $p \in (r_l, r_t^*)$, the object belongs to the promotive region.
(P2) If $p \in [r_t^*, r_g]$, the object belongs to the recommendation region.

Second, we count the numbers of objects in the different regions, that is, $RY(R, P, l_t, W, r_l, r_t^*)$, $RN(R, P, l_t, W, r_l, r_t^*)$, $PY(R, P, l_t, W, r_l, r_t^*)$, $PN(R, P, l_t, W, r_l, r_t^*)$, $NY(R, P, l_t, W, r_l, r_t^*)$, and $NN(R, P, l_t, W, r_l, r_t^*)$ using Eq. (15).

Third, the average three-way cost $at(R, P, l_t, W, r_l, r_t^*)$ is computed using Eq. (17).

Finally, we determine the lower threshold r_l^* as

$$r_l^* = \arg\min_{r_w \le r_l \le r_t^*} at(R, P, l_t, W, r_l, r_t^*). \tag{28}$$

Determination of the upper threshold r_h^* follows the same steps as for r_l^*. For $\forall r_h \in [r_t^*, r_g]$, we obtain the following decision rules:

Table 9
Proportion for different thresholds.

$[r_l, r_h)$	$TS(r_e, s, l_t)$	$ES(r_e, s, l_t)$	$PR(r_e, s, l_t)$
[1.0, 1.5)	0	2	0
[1.5, 2.0)	0	1	0
[2.0, 2.5)	1	3	0.33
[2.5, 3.0)	1	2	0.5
[3.0, 3.5)	2	3	0.67
[3.5, 4.0)	0	0	0
[4.0, 4.5)	1	2	0.5
[4.5, 5.0)	1	1	1
[5.0, 5.1)	1	2	0.5

(N3) If $p \in [r_w, r_t^*]$, the object belongs to the non-recommendation region.
(B3) If $p \in (r_t^*, r_h)$, the object belongs to the promotive region.
(P3) If $p \in [r_h, r_g]$, the object belongs to the recommendation region.

We count the numbers of objects in the different regions, that is, $RY(R, P, l_t, W, r_t^*, r_h)$, $RN(R, P, l_t, W, r_t^*, r_h)$, $PY(R, P, l_t, W, r_t^*, r_h)$, $PN(R, P, l_t, W, r_t^*, r_h)$, $NY(R, P, l_t, W, r_t^*, r_h)$, and $NN(R, P, l_t, W, r_t^*, r_h)$ using Eq. (15).
The average three-way cost $at(R, P, l_t, W, r_t^*, r_h)$ is computed using Eq. (17).
Finally, we determine the upper threshold r_h^* as

$$r_h^* = \arg \min_{r_t^* \le r_h \le r_g} at(R, P, l_t, W, r_t^*, r_h). \tag{29}$$

5.4. Working example #3

In this subsection, we explain how to obtain the recommendation threshold (r_l^*, r_h^*)-pair using the proportion-based approach. First, we compute the three-way threshold (β^*, α^*)-pair based on three-way cost. Next, we compute the proportion based on the actual rating matrix R and prediction matrix P. Finally, we compute the three-way recommendation (r_l^*, r_h^*)-pair based on the proportion and the (β^*, α^*)-pair.
According to the cost matrix in Fig. 1 and the three-way decision model, we calculate α^* and β^* as

$$\alpha^* = \frac{90 - 18}{(90 - 18) + (18 - 0)} = 0.8\,,$$
$$\beta^* = \frac{18 - 0}{(18 - 0) + (45 - 18)} = 0.4\,.$$

Based on the actual matrix given in Table 1 and the prediction matrix presented in Table 2, we compute the proportion by changing r_e. This involves three steps: (1) we count the number $ES(r_e, s)$ of prediction p between r_e and $r_e + s$ in the prediction matrix P; (2) we count the number $TS(r_e, s, l_t)$ of the actual rating r greater than l_t from the corresponding positions of the actual matrix obtained in the preceding step; (3) proportion $PR(r_e, s, l_t)$ is equal to $TS(r_e, s, l_t)$ divided by $ES(r_e, s)$. The calculation results are given in Table 9.
Owing to the example's small data set, we cannot obtain a three-way threshold pair.

6. Experiments

In this section, we report on the extensive computational tests carried out mainly to address the following questions.

1. Does the loss of the regression-based binary recommendation change smoothly?
2. Is the change in the recommendation threshold r_t^c obvious when considering misclassification costs?
3. Do the two three-way recommendation approaches obtain the same threshold (r_l^*, r_h^*)-pair?

Question 1 aims to find the optimum point on the loss curve. That is, the loss decreases continuously on the left-hand side of the point, and increases continuously on the right-hand side. Question 2 is aimed at the effect of the cost setting on the recommendation threshold r_t^c while considering the different misclassification costs. Finally, Question 3 focuses on whether the threshold pairs of the two three-way recommendation approaches are reliable.

6.1. Data set

In the experiments we used the well-known MovieLens data set, which is widely used in recommender systems (see, e.g., [10,46]). The database schema is as follows.

- User (userID, age, gender, occupation)

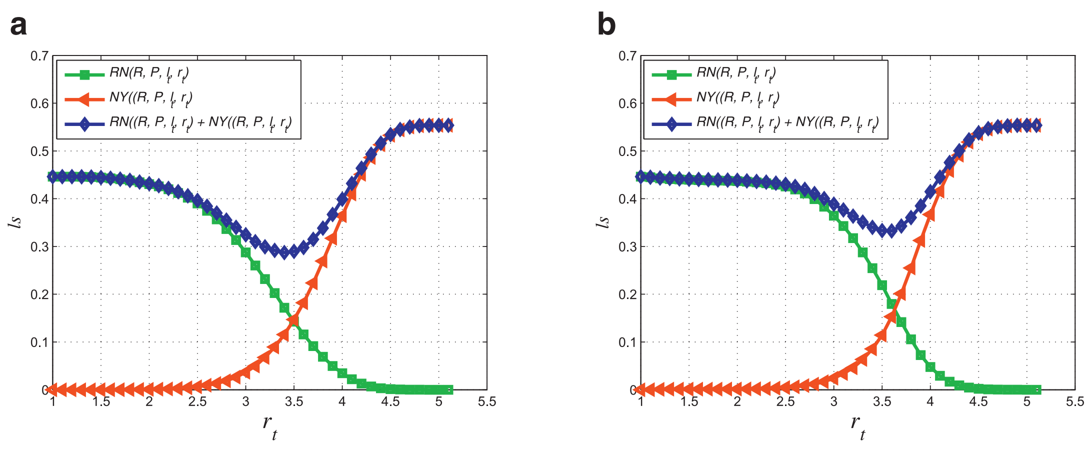

Fig. 2. Optimal recommender threshold: (a) slope one ($r_t^* = 3.4$), and (b) kNN ($r_t^* = 3.6$).

Table 10
Proportion of user-item ratings.

Rating	1	2	3	4	5
Proportion	0.06	0.11	0.27	0.34	0.21

- Movie (movieID, release-year, genre)
- Rating (userID, movieID)

We used the version with 943 users and 1,682 movies. The data set consists of 100,000 movie ratings. The original rating relation contains the rating of movies on a five-point scale. Each rating as a proportion of all ratings is given in Table 10.

Ratings greater than three account for 55% of all the ratings, while the rest account for 45%. There is a small proportion of low ratings because ratings less than three account for 17% of all ratings. Despite all users having watched at least one movie, the rating matrix is still sparse because no one has watched more than 45 percent of all the movies, and only 20 percent of the users have watched more than 10 percent of all the movies.

6.2. Experimental design

We designed three sets of experiments to answer the questions raised at the beginning of this section.

Exp1. We determined the two-way recommendation threshold r_t^* based on the loss.

Exp2. We determined the two-way recommendation threshold r_t^c while considering the different misclassification costs.

Exp3. We determined the three-way recommendation threshold (r_l^*, r_h^*)-pair while considering both misclassification and promotion costs.

6.3. Results

The following experimental results are presented to answer each of the questions raised at the beginning of the section. The original set iteratively leaves one rating out for the test set, while training on the rest.

6.3.1. Regression-based binary recommendation

Fig. 2 illustrates the determination of the optimal recommendation threshold for loss minimization. There are three curves: $RN(R,\ P,\ l_t,\ r_t)$ denotes the number of objects in the false recommendation region (RN) vs. the recommendation threshold (r_t). $NY(R,\ P,\ l_t,\ r_t)$ denotes the number of objects in the false non-recommendation region vs. the recommendation threshold r_t. $RN(R, P, l_t, r_t) + NY(R, P, l_t, r_t)$ denotes the number of objects in two regions vs. the recommendation threshold r_t. $RN(R,\ P,\ l_t,\ r_t)$ and $NY(R,\ P,\ l_t,\ r_t)$ are monotonic, whereas $RN(R,\ P,\ l_t,\ r_t)$ is entirely decreasing and $NY(R,\ P,\ l_t,\ r_t)$ is entirely increasing. The losses of the three curves change continuously without "corners"; that is, the curves change smoothly. Because the $RN(R, P, l_t, r_t) + NY(R, P, l_t, r_t)$ curve is a convex function, there exists a unique minimum loss. We can determine the recommendation threshold r_t^* based on the unique minimum loss.

Fig. 2(a) illustrates the determination of the optimal recommendation threshold using the slope one algorithm. The unique minimum loss of $RN(R, P, l_t, r_t) + NY(R, P, l_t, r_t)$ is 0.287, and thus, we obtain r_t^* as 3.4.

Fig. 2(b) illustrates the determination of the optimal recommendation threshold using the kNN algorithm. The unique minimum loss of $RN(R, P, l_t, r_t) + NY(R, P, l_t, r_t)$ is 0.333, and thus, we obtain r_t^* as 3.6.

Table 11
Accuracy comparison of different approaches.

Algorithm	MAE	Minimum loss
Slope one	0.74	0.287
kNN	0.8	0.333
Difference	$\frac{0.8-0.74}{0.74}=8.1\%$	$\frac{0.333-0.287}{0.287}=16.0\%$

Table 12
Average misclassification cost and r_t^c for different ratios of $\frac{\lambda_{NY}}{\lambda_{RN}}$.

$\frac{\lambda_{NY}}{\lambda_{RN}}$	Average misclassification cost			r_t^c	
	Slope one	kNN	Difference	Slope one	kNN
0.125	8.1	8.3	$\frac{8.3-8.1}{8.1}=2.5\%$	4.4	4.6
0.25	13.4	14.2	$\frac{14.2-13.4}{13.4}=5.6\%$	4.1	4.2
0.5	18.3	20.6	$\frac{20.6-18.3}{18.3}=12.6\%$	3.7	3.9
1	19.4	22.5	$\frac{22.5-19.4}{19.4}=16.0\%$	3.4	3.6
2	16.2	18.4	$\frac{18.4-16.2}{16.2}=13.6\%$	3.1	3.2
4	11.1	11.9	$\frac{11.9-11.1}{11.1}=7.2\%$	2.7	2.2
8	6.5	6.7	$\frac{6.7-6.5}{6.5}=3.1\%$	2.3	1.0

Table 13
Two- vs. three-way average-cost comparison for different promotion cost settings.

Promotion cost	(β^*, α^*)	Algorithm	(r_l^*, r_h^*)	Two-way	Three-way	Decrease
18	(0.4, 0.8)	Slope one	(3.2, 4.1)	18.3	14.9	$\frac{18.3-14.9}{18.3}=18.6\%$
		kNN	(3.4, 4.2)	20.6	16.1	$\frac{20.6-16.1}{20.6}=21.8\%$
22.5	(0.5, 0.75)	Slope one	(3.4, 3.9)	18.3	16.8	$\frac{18.3-16.8}{18.3}=8.2\%$
		kNN	(3.6, 4.1)	20.6	18.5	$\frac{20.6-18.5}{20.6}=10.2\%$

Table 11 compares the performance of the slope one and kNN-based approaches, in terms of MAE and minimum loss. The MAE difference between the two approaches is 8.1%, whereas the difference in minimum loss is 16.0%. The misclassification is obviously greater than the MAE.

Based on the above analysis, the following conclusions are presented: (1) the loss curve is a convex function, and therefore it has a unique minimum value; (2) the optimal recommendation threshold is associated with the algorithms; (3) there is an obvious difference between the loss due to misclassification and the MAE (16% vs. 8.1%).

6.3.2. *Misclassification cost minimization recommendation*

The total of the two non-zero misclassification costs is set to 135. For different situations, we change the ratio of the two non-zero misclassification costs from 0.125 to 8. Assuming a ratio of 0.5 for example, the misclassification cost for recommending uninteresting items is 90, while the misclassification cost for non-recommending interesting items is 45.

Table 12 shows the optimal recommendation thresholds under the different settings of the two non-zero misclassification costs. We compute the minimal average misclassification cost for each setting using the trichotomy method. Assuming a ratio of 0.5 for example, the minimal average misclassification cost is 18.3 using the slope one algorithm with $r_t^c = 3.7$.

We present the following conclusions based on Table 12: (1) the change in the optimal recommendation threshold r_t^c is gradual and linear with an increase in the ratio, using both the slope one and kNN algorithms; and (2) the average of the asymmetric misclassification costs is lower than that of the symmetric ones.

6.3.3. *Three-way-decision-based recommendation*

In practice, different values can be assigned to misclassification and promotion costs. In Fig. 1, the two nonzero misclassification costs are set to 90 and 45, respectively. Promotion cost is set to 18 or 22.5 in the comparative experiments as shown in Table 13. If promotion cost is set to 18, we obtain the optimal threshold pair of three-way decision as $(\beta^* = 0.4, \alpha^* = 0.8)$, whereas a setting of 22.5 yields $(\beta^* = 0.5, \alpha^* = 0.75)$.

We designed four sets of experiments to consider different promotion costs and different regression-based algorithms. Because similar results were obtained for different promotion costs, we only present the results with the promotion cost set to 18 in Figs. 3–5.

Fig. 3 illustrates the determination of the three-way threshold pair using the proportion-based approach. Here p is predicted using the slope one or kNN algorithm. The proportion PR is computed by Eq. (25). The curve of PR with respect to p

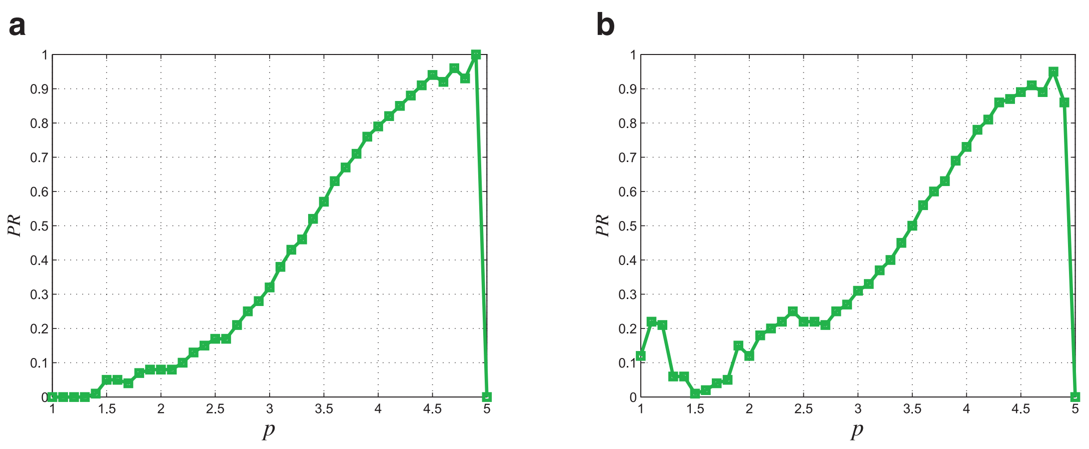

Fig. 3. Three-way threshold pair: (a) slope one ($r_l^* = 3.2$, $r_h^* = 4.1$), (b) kNN ($r_l^* = 3.4$, $r_h^* = 4.2$).

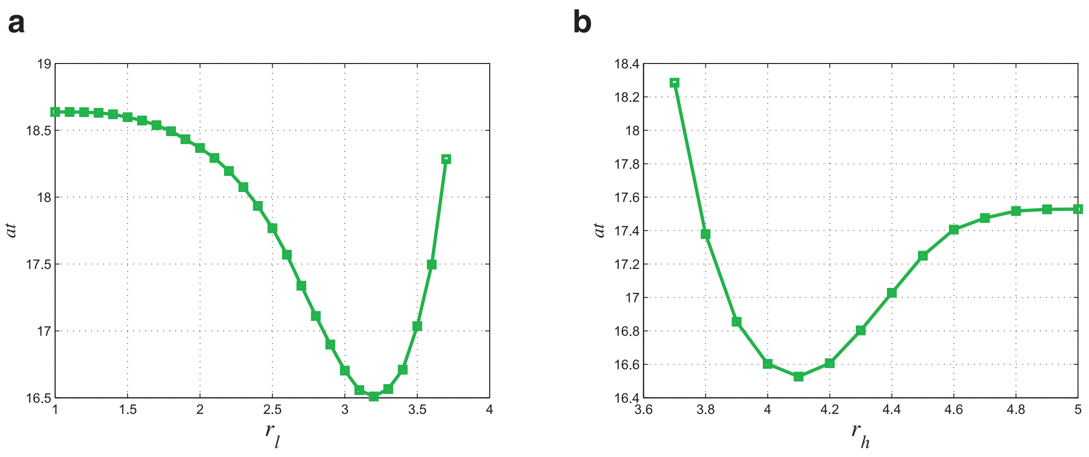

Fig. 4. Threshold pair of three-way growth algorithm: (a) $r_l^* = 3.2$, (b) $r_h^* = 4.1$.

is basically gradual and linear. We can determine the three-way recommendation (r_l^*, r_h^*)-pair based on the values of α^* and β^*.

Fig. 3(a) shows the optimal three-way threshold pair obtained by the slope one algorithm. Given that (β^*, α^*) is (0.4, 0.8), the three-way recommendation pair is calculated as $(r_l^* = 3.2, r_h^* = 4.1)$.

Fig. 3(b) shows the optimal three-way threshold pair obtained by the kNN algorithm. Given that (β^*, α^*) is (0.4, 0.8), the three-way recommendation pair is calculated as $(r_l^* = 3.4, r_h^* = 4.2)$.

Fig. 4 depicts the minimizing search approach using the slope one algorithm. According to Table 12, the optimal recommendation threshold r_t^c is 3.7 when $\frac{c_{NY}}{c_{RN}} = 0.5$. Fig. 4(a) depicts the lowest recommendation threshold r_l^* in the interval [1, 3.7]. We obtain $r_l^* = 3.2$ based on the minimum average cost. Fig. 4(b) depicts the highest recommendation threshold r_h^* in the interval [3.7, 5]. We obtain $r_h^* = 4.1$ based on the minimum average cost.

Fig. 5 depicts the minimizing search approach using the kNN algorithm. According to Table 12, the optimal recommendation threshold r_t^c is 3.9 when $\frac{c_{NY}}{c_{RN}} = 0.5$. Fig. 4(a) depicts the lowest recommendation threshold r_l^* in the interval [1, 3.9]. We obtain $r_l^* = 3.4$ based on the minimum average cost. Fig. 4(b) shows the highest recommendation threshold r_h^* in the interval [3.9, 5]. We obtain $r_h^* = 4.2$ based on the minimum average cost.

Table 13 compares the two- vs. three-way average costs for different promotion cost settings. We compute the three-way average cost based on the recommendation threshold (r_l^*, r_h^*)-pair. Using the slope one algorithm, the three-way average cost is 14.9 with a promotion cost of 18, and 16.8 with a promotion cost of 22.5. Using the kNN algorithm, the three-way average cost is 16.1 with a promotion cost of 18, and 18.5 with a promotion cost of 22.5. The decrease in three-way average cost compared with two-way average cost is between 8.2% and 21.8%.

We present the following conclusions based on Figs. 3–5 and Table 13: (1) the three-way recommendation threshold pairs obtained by the proportion-based and minimizing search approaches are the same; (2) the (r_l^*, r_h^*)-pair is relevant to not only the cost matrix, but also the prediction algorithm; and (3) the average cost in the three-way decision model is obviously less than the two-way model.

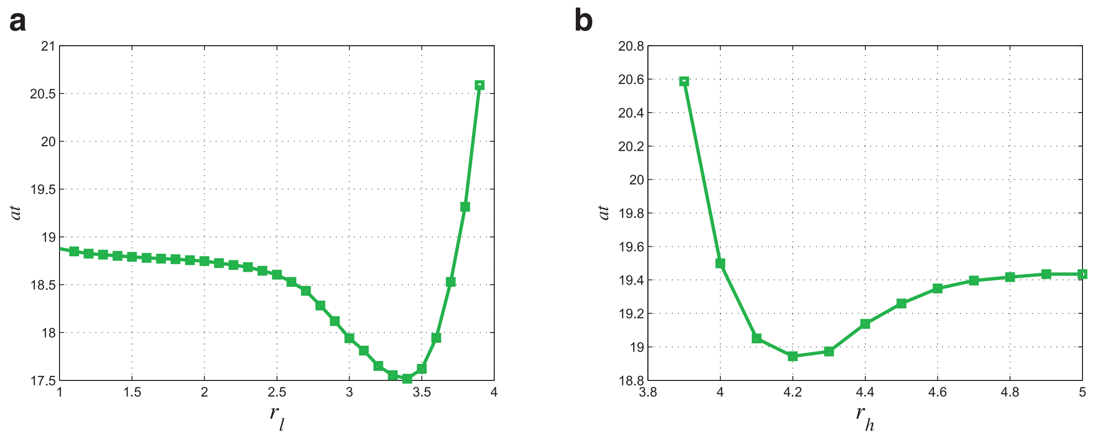

Fig. 5. Threshold pair of three-way growth algorithm: (a) $r_l^* = 3.4$, (b) $r_h^* = 4.2$.

6.4. Discussion

We are now in a position to answer the questions raised at the beginning of this section:

1. The loss of regression-based binary recommendation changes smoothly.
2. The change in recommendation threshold r_t^c is obvious when considering misclassification costs.
3. The two three-way recommendation approaches obtain the same threshold (r_l^*, r_h^*)-pair with the same prediction algorithm.

7. Conclusion

In this paper, we proposed a framework integrating three-way decision and a regression-based approach to suggest appropriate recommender behavior with predicted numerical ratings. Regression-based binary recommendation is used to obtain the optimal recommendation threshold. Considering misclassification and promotion costs, a binary recommendation can be generated from the three-way recommendation. According to our experiments on the MovieLens data set, the loss of regression-based binary recommendation changes smoothly and the two three-way recommendation approaches obtain the same threshold (r_l^*, r_h^*)-pair with the same prediction algorithm.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China (Grant 61379089) and the Natural Science Foundation of the Department of Education of Sichuan Province (Grant 16ZA0060).

References

[1] M. Athani, N. Pathak, A.U. Khan, B. Gour, Real time recommender system for music data, Int. J. Comput. Sci. Inf. Secur. 12 (8) (2014) 113.
[2] N. Azam, J.-T. Yao, Game-theoretic rough sets for recommender systems, Knowl. Based Syst. 72 (2014) 96–107.
[3] A. Basu, J. Vaidya, H. Kikuchi, Efficient privacy-preserving collaborative filtering based on the weighted slope one predictor, J. Internet Serv. Inf. Secur. 1 (4) (2011) 26–46.
[4] K.B. Ghauth, N.A. Abdullah, Building an e-learning recommender system using vector space model and good learners average rating, in: Proceedings of the 9th IEEE International Conference on Advanced Learning Technologies, IEEE, 2009.
[5] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[6] D.-G. Chen, C.-Z. Wang, Q.-H. Hu, A new approach to attribute reduction of consistent and inconsistent covering decision systems with covering rough sets, Inf. Sci. 177 (17) (2007) 3500–3518.
[7] D. Ciucci, D. Dubois, H. Prade, Oppositions in rough set theory, Rough Sets and Knowledge Technology, Springer, 2012, pp. 504–513.
[8] L.O. Colombo-Mendoza, R. Valencia-García, A. Rodríguez-González, G. Alor-Hernández, J.J. Samper-Zapater, Recommetz: a context-aware knowledge-based mobile recommender system for movie showtimes, Expert Syst. Appl. 42 (3) (2015) 1202–1222.
[9] P. Cremonesi, Y. Koren, R. Turrin, Performance of recommender algorithms on top-n recommendation tasks, in: Proceedings of the 4th ACM Conference on Recommender Systems, ACM, 2010.
[10] P. Cremonesi, R. Turrin, F. Airoldi, Hybrid algorithms for recommending new items, in: Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems, ACM, 2011.
[11] M. Deshpande, G. Karypis, Item-based top-n recommendation algorithms, ACM Trans. Inf. Syst. 22 (1) (2004) 143–177.
[12] P. Domingos, Metacost: A general method for making classifiers cost-sensitive, in: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 1999.
[13] S. Elaydi, K. Janglajew, Dichotomy and trichotomy of difference equations, J. Dif. Equ. Appl. 3 (5-6) (1998) 98–103.
[14] G. Fumera, F. Roli, Cost-sensitive learning in support vector machines, in: Proceedings of VIII Convegno Associazione Italiana per L' Intelligenza Artificiale, 2002.
[15] X. He, F. Min, W. Zhu, Parametric rough sets with application to granular association rule mining, Math. Probl. Eng. 2013 (2013) 1–13.

[16] J.L. Herlocker, J.A. Konstan, L.G. Terveen, J.T. Riedl, Evaluating collaborative filtering recommender systems, ACM Trans. Inf. Syst. 22 (1) (2004) 5–53.
[17] Q.-H. Hu, D.-R. Yu, J.-F. Liu, C.-X. Wu, Neighborhood rough set based heterogeneous feature subset selection, Inf. Sci. 178 (18) (2008) 3577–3594.
[18] A. Jankowski, A. Skowron, Toward rough-granular computing, Rough Sets, Fuzzy Sets, Data Mining and Granular Computing, Springer, 2007, pp. 1–12.
[19] G. Karypis, Evaluation of item-based top-n recommendation algorithms, Proceedings of the 10th International Conference on Information and Knowledge Management, ACM, 2001.
[20] R. Kohavi, J.R. Quinlan, Data Mining Tasks and Methods: Classification: Decision-Tree Discovery, Handbook of Data Mining and Knowledge Discovery, Oxford University Press, 2002.
[21] A. Krogh, J. Vedelsby, Neural network ensembles, cross validation, and active learning, in: Advances in Neural Information Processing Systems, 1995, pp. 231–238.
[22] M. Kumar, D. Yadav, A. Singh, V.K. Gupta, A movie recommender system: Movrec, Int. J. Comput. Appl. 124 (3) (2015).
[23] Y.-H. Lee, P.J.-H. Hu, T.-H. Cheng, Y.-F. Hsieh, A cost-sensitive technique for positive-example learning supporting content-based product recommendations in b-to-c e-commerce, Decis. Support Syst. 53 (1) (2012) 245–256.
[24] G. Lekakos, P. Caravelas, A hybrid approach for movie recommendation, Multimed. Tools Appl. 36 (1–2) (2008) 55–70.
[25] D. Lemire, A. Maclachlan, Slope one predictors for online rating-based collaborative filtering, SIAM Data Mining (2005) 21–23.
[26] H.-X. Li, L.-B. Zhang, B. Huang, X.-Z. Zhou, Sequential three-way decision and granulation for cost-sensitive face recognition, Knowl. Based Syst. 91 (2016) 241–251.
[27] H.-X. Li, X.-Z. Zhou, Risk decision making based on decision-theoretic rough set: a three-way view decision model, Int. J. Comput. Intell. Syst. 4 (1) (2011) 1–11.
[28] J.-J. Li, L.-M. Sun, J. Wang, A slope one collaborative filtering recommendation algorithm using uncertain neighbors optimizing, Web-Age Information Management, Springer, 2012, pp. 160–166.
[29] T.Y. Lin, Granular computing on binary relations i: data mining and neighborhood systems, Rough Sets Knowl. Discov. 1 (1998) 107–121.
[30] T.Y. Lin, A roadmap from rough set theory to granular computing, Rough Sets and Knowledge Technology, Springer, 2006, pp. 33–41.
[31] D. Liu, T.-R. Li, D.-C. Liang, Three-way government decision analysis with decision-theoretic Rough sets, Int. J. Uncertain. Fuzziness Knowl. Based Syst. 20 (2012) 119–132.
[32] D. Liu, Y.Y. Yao, T.-R. Li, Three-way investment decisions with decision-theoretic Rough sets, Int. J. Comput. Intell. Syst. 4 (2011) 66–74.
[33] H. Ma, I. King, M.R. Lyu, Effective missing data prediction for collaborative filtering, in: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2007.
[34] F. Min, H.-P. He, Y.-H. Qian, W. Zhu, Test-cost-sensitive attribute reduction, Inf. Sci. 181 (2011) 4928–4942.
[35] F. Min, Q.-H. Hu, W. Zhu, Feature selection with test cost constraint, Int. J. Approx. Reason. 55 (1) (2014) 167–179.
[36] F. Min, W. Zhu, Optimal sub-reducts with test cost constraint, in: Rough Sets and Knowledge Technology, 2011, pp. 57–62.
[37] F. Min, W. Zhu, Mining significant granular association rules for diverse recommendation, Rough Sets and Current Trends in Computing, Springer, 2014.
[38] M.P. O'Mahony, N.J. Hurley, G.C. Silvestre, Recommender systems: Attack types and strategies, in: AAAI, 2005.
[39] S.K. Pal, B.U. Shankar, P. Mitra, Granular computing, rough entropy and object extraction, Pattern Recognit. Lett. 26 (16) (2005) 2509–2517.
[40] Y. Park, S. Park, W. Jung, S.-G. Lee, Reversed CF: A fast collaborative filtering algorithm using a k-nearest neighbor graph, Expert Syst. Appl. 42 (8) (2015) 4022–4028.
[41] W. Pedrycz, A. Skowron, V. Kreinovich, Handbook of Granular Computing, John Wiley & Sons, 2008.
[42] J.-J. Qi, T. Qian, L. Wei, The connections between three-way and classical concept lattices, Knowl. Based Syst. 91 (2016) 143–151.
[43] Y.-H. Qian, J.-Y. Liang, W. Pedrycz, C.-Y. Dang, Positive approximation: an accelerator for attribute reduction in rough set theory, Artif. Intell. 174 (9) (2010) 597–618.
[44] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, ACM, 2001.
[45] J.B. Schafer, D. Frankowski, J. Herlocker, S. Sen, Collaborative filtering recommender systems, The Adaptive Web, Springer, 2007, pp. 291–324.
[46] A.I. Schein, A. Popescul, L.H. Ungar, D.M. Pennock, Methods and metrics for cold-start recommendations, in: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2002.
[47] B. Settles, in: Active Learning Literature Survey, University of Wisconsin, Madison 52, 2010, pp. 55–66.
[48] A. Skowron, J. Stepaniuk, Information granules: towards foundations of granular computing, Int. J. Intell. Syst. 16 (1) (2001) 57–85.
[49] M. Soleymani, A. Aljanaki, F. Wiering, R.C. Veltkamp, Content-based music recommendation using underlying music preference structure, in: Proceedings of the 2015 IEEE International Conference on Multimedia and Expo (ICME), IEEE, 2015.
[50] X.-Y. Su, T.M. Khoshgoftaar, A survey of collaborative filtering techniques, in: Proceedings of the Advances in Artificial Intelligence, 2009.
[51] S. Tong, D. Koller, Support vector machine active learning with applications to text classification, J. Mach. Learn. Res. 2 (2002) 45–66.
[52] P.D. Turney, Types of Cost in Inductive Concept Learning, In Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning, 2010, pp. 15–21.
[53] B.-L. Wang, J.-Y. Liang, Y.-H. Qian, Determining decision makers weights in group ranking: a granular computing method, Int. J. Mach. Learn. Cybern. 6 (3) (2014) 511–521.
[54] J. Wang, A.P. De Vries, M.J. Reinders, Unifying user-based and item-based collaborative filtering approaches by similarity fusion, in: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2006.
[55] C.J. Willmott, K. Matsuura, Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance, Climate Res. 30 (1) (2005) 79–82.
[56] X.-B. Yang, J.-Y. Yang, Multigranulation rough sets in incomplete information system, Incomplete Information System and Rough Set Theory, Springer, 2012, pp. 195–222.
[57] X.-P. Yang, J.-T. Yao, Modelling multi-agent three-way decisions with decision-theoretic rough sets, Fundam. Inf. 115 (2-3) (2012) 157–171.
[58] J.-T. Yao, Information granulation and granular relationships, in: IEEE, 2005.
[59] Y.Y. Yao, Granular computing: basic issues and possible solutions, in: Proceedings of the 5th Joint Conference on Information Sciences, vol. 1, 2000.
[60] Y.Y. Yao, Perspectives of granular computing, in: IEEE, 2005.
[61] Y.Y. Yao, The art of granular computing, Rough Sets and Intelligent Systems Paradigms, Springer, 2007, pp. 101–112.
[62] Y.Y. Yao, Decision-theoretic rough set models, Rough Sets and Knowledge Technology, Springer, 2007, pp. 1–12.
[63] Y.Y. Yao, Granular computing: past, present, and future, Lect. Notes Comput. Sci. 5009 (2008) 27–28.
[64] Y.Y. Yao, Three-way decisions with probabilistic rough sets, Inf. Sci. 180 (3) (2010) 341–353.
[65] Y.Y. Yao, An outline of a theory of three-way decisions, Rough Sets and Current Trends in Computing, Springer, 2012.
[66] Y.Y. Yao, Granular computing and sequential three-way decisions, Rough Sets and Knowledge Technology, Springer, 2013, pp. 16–27.
[67] Y.Y. Yao, Rough sets and three-way decisions, Rough Sets and Knowledge Technology, Springer, 2015, pp. 62–73.
[68] Y.Y. Yao, C. Gao, Statistical interpretations of three-way decisions, Rough Sets and Knowledge Technology, Springer, 2015, pp. 309–320.
[69] Y.Y. Yao, S.K.M. Wong, A decision theoretic framework for approximating concepts, Int. J. Man-Mach. Stud. 37 (6) (1992) 793–809.
[70] K. Yoshii, M. Goto, K. Komatani, T. Ogata, H.G. Okuno, An efficient hybrid music recommender system using an incrementally trainable probabilistic generative model, IEEE Trans. Audio Speech Lang. Process. 16 (2) (2008) 435–447.
[71] L.A. Zadeh, Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic, Fuzzy Sets Syst. 90 (2) (1997) 111–127.
[72] L.A. Zadeh, Some reflections on soft computing, granular computing and their roles in the conception, design and utilization of information/intelligent systems, Soft Comput. A Fus. Found. Methodol. Appl. 2 (1) (1998) 23–25.

[73] C. Zeng, C.-X. Xing, L.-Z. Zhou, Similarity measure and instance selection for collaborative filtering, in: Proceedings of the 12th International Conference on World Wide Web, ACM, 2003.
[74] H.-R. Zhang, F. Min, Three-way recommender systems based on random forests, Knowl. Based Syst. 91 (2016) 275–286.
[75] H.-R. Zhang, F. Min, X. He, Aggregated recommendation through random forests, Sci. World J. 2014 (2014) 1–11.
[76] W.-D. Zhao, S.-L. Tang, W.-H. Dai, An improved *k*NN algorithm based on essential vector, Elektronika ir Elektrotechnika 123 (7) (2012) 119–122.
[77] B. Zhou, Y.Y. Yao, J.-G. Luo, A three-way decision approach to email spam filtering, Advances in Artificial Intelligence, Springer, 2010, pp. 28–39.
[78] Z.-H. Zhou, X.-Y. Liu, Training cost-sensitive neural networks with methods addressing the class imbalance problem, IEEE Trans. Knowl. Data Eng. 18 (1) (2006) 63–77.
[79] W. Zhu, F.-Y. Wang, Reduction and axiomization of covering generalized rough sets, Inf. Sci. 152 (1) (2003) 217–230.