A New Adaptive–Support Algorithm for Association Rule Mining

Weiyang Lin, Carolina Ruiz, and Sergio A. Alvarez

Department of Computer Science
Worcester Polytechnic Institute,
Worcester, MA 01609, USA,
{wy_lin,ruiz,alvarez}@cs.wpi.edu,
http://www.cs.wpi.edu/~{wy_lin,ruiz,alvarez}

Abstract. In this paper we propose a new approach for mining association rules of classification type particularly suited for use in collaborative recommender systems. Such systems rely on information about relationships between different users' preferences in order to recommend items of potential interest to the target user. Despite their successful application to other domains, existing association rule mining techniques are not suitable for the recommendation domain because they mine many rules that are not relevant to a given user. Also, they require that the minimum support (also known as the significance) of the mined rules be specified in advance, often leading to too many or too few rules. In contrast, our approach adjusts the minimum support so that the number of rules obtained is within a specified range, thus avoiding excessive computation time while guaranteeing that enough rules are provided to allow good classification performance. This paper describes our approach. The results of an experimental evaluation of our approach are also described. These results show that the rules mined by our approach allow excellent recommendation performance.

1 Introduction

Collaborative recommender systems are of great current interest, largely because of their impact on electronic commerce. Such systems rely on identifying similarities and dissimilarities between the preferences of different users in order to suggest items of potential interest to such users (see e.g. [13], [12], [3], [4]). Despite this interest, both the number of available published techniques and information about their performance are quite limited. It is of great importance to explore more techniques for this domain. In the present paper we describe an association rule mining technique specially designed for use in collaborative recommender systems. We include experimental results that show that very good results may be obtained using this technique.

1.1 Association Rules

Association rules were independently introduced by P. Hájek et al. [6, 8] and by Agrawal et al. [1]. While [6] and [8] introduce association rule mining as a machine learning approach to the logic of discovery, [1] concentrates on the mining of associations over sales data.

Given a set of transactions, where each transaction is a set of items, an association rule is a rule of the form $X \Rightarrow Y$, where X and Y are sets of items. An example of an association rule in the basket market analysis domain is: "90% of transactions that contain bread and butter also contain milk; 30% of all transactions contain the three of them". Here, $X = \{\text{bread, butter}\}$, $Y = \{\text{milk}\}$, 90% is called the confidence of the rule, and 30% the support of the rule. X and Y are respectively called the *body* and the *head* of the rule.

Given a set of transactions, where each transaction is a set of items, and user-specified minimum support and minimum confidence, the standard problem of mining association rules is to find all association rules that are above the user-specified minimum support and minimum confidence. We propose a variant of this problem in section 2 of the present paper.

1.2 Scope and advantages of our approach

We have designed and implemented an algorithm to mine association rules which is adapted from the Apriori algorithm [2] and CBA-RG [9] and is particularly tailored to collaborative recommender systems. Our motivation to mine association rules for recommender systems comes from the following observation: Rules like "90% of users who like article A and article B also like article C, 30% of all users like all of them" and "90% of articles liked by user A and user B are also liked by user C, 30% of all articles are liked by all of them" are very useful for recommendation purposes. We refer to rules of the first kind as article associations and rules of the second kind as user associations. Article associations represent relationships among articles and user associations represent relationships among users that are useful for recommendation. We explore article associations and user associations on two levels (*like* and *dislike*) by using extensions of the basic

association rules. One example of two level user associations is "90% of articles liked by user A and disliked by user B are liked by user C, 30% of all articles are liked by user A and C and disliked by user B".

Our new mining algorithm focuses on mining rules for only one target user/article at a time. This has the following advantages:

1. Since we are only interested in predicting articles that a target user would like, in user associations, we only need rules with that user in the rule head. Such rules could be mined more efficiently than the rules with arbitrary heads. Since we need to mine user associations online, the efficiency of the process is of great importance.
2. By mining article associations for one article at a time we are able to obtain rules for articles that have only received a limited number of ratings, for example a new movie. This would not be possible if we mined article associations for all articles at once, because rules for new articles would fail to have the necessary support.
3. A significant amount of runtime is saved by mining rules only over the subset of the transaction data that is related to the target user/article instead of over the whole data.

A main feature and advantage of our algorithm in comparison with existing mining methods is that it automatically selects the minimum support so that the mining process produces an appropriate number of rules for each target item. This property of our algorithm allows recommender systems that rely on the rules mined by it to achieve very good performance both in terms of response time and accuracy of the predictions. Instead of the minimum support, our algorithm is given a range for the desired number of rules before the mining process.

1.3 Relation to other work

A large variety of association rule frameworks and algorithms have been published in the literature, including GUHA [6, 8, 7, 11], Apriori [2], and DIS [5]. One extension of the basic binary association rules, called quantitative association rules [14], finds associations between attributes with categorical values. Quantitative association rules have the potential to extend association rules to general classification domains. Some results of adapting those rules to classification tasks are shown in [9, 8]. [9] presents the CBA-RG algorithm (which is based on the Apriori algorithm) and a good framework to perform the so-called associative classification.

However, previously proposed association rule mining algorithms are not suitable for collaborative recommender systems. Two significant reasons for this are:

- Previous algorithms do not provide a mechanism to choose a proper minimum support for the given minimum confidence and the desired range for the number of rules. This often leads to either too many or too few rules, and thus to either excessive computation time or else poor recommendation performance.

– Most existing algorithms do not allow the heads of the rules to be specified in advance. Although CBA-RG has addressed the problem of mining rules for a single target class, the recommendation problem is even more focussed since we need to mine rules for only one target class value.

2 Our Association Rule Mining Algorithm

In this section we describe our algorithm to mine association rules (AR-CRS). This algorithm adjusts the minimum support of the rules during mining in order to obtain an appropriate number of significant rules for the target predicate.

Problem Definition Given a transaction dataset, a target item, a specified minimum confidence and a desired range [minRulenum,maxRulenum] for the number of rules, find association rules with the target item in the heads of the rules such that the number of rules is in the given range, the rules have the highest possible support, and the rules satisfy the minimum confidence constraint.

Note. Since we use the same algorithm to mine user associations as well as article associations, we use the term *target item* to denote "target user" in the case of user associations and "target article" in the case of article associations.

2.1 Algorithm Description

Our AR-CRS algorithm solves the above problem. AR-CRS consists of two parts: AR-CRS-1 and AR-CRS-2.

AR-CRS-1 In order to mine only a given number of most promising rules for each target item, we use AR-CRS-1 to control the minimum support count and find the rules with the highest supports. The minimum support count is the minimum number of transactions that satisfy a rule in order to make that rule frequent, i.e., it is the multiplication of the minimum support and the whole number of transactions. The overall process is shown in detail in Figure 1. It consists of three parts:

1. AR-CRS-1 initializes the minimum support count according to the frequency of the target predicate and calls AR-CRS-2 to mine rules.
2. When AR-CRS-2's output is returned, AR-CRS-1 will check first if the number of rules returned is equal to maxRulenum (as we describe below, AR-CRS-2 terminates the mining process when the number of rules generated is equal to maxRulenum). If it is, that means the minimum support count is low which results in more than maxRulenum rules, so the AR-CRS-1 will keep increasing the minimum support count and calling AR-CRS-2 until the number of rules is less than maxRulenum.

```
Input: Transactions, targetItem, minConfidence, minRulenum, maxRulenum,
Output: mined association rules

1)    Set initial minsupportCount based on targetItem's like ratio;
2)    R = AR-CRS-2();
3)    while (R.rulenum = maxRulenum) do
4)     minsupportCount++;
5)     R1 = AR-CRS-2();
6)     if R1.rulenum > minRulenum then R = R1;
7)     else return R;
8)    end
9)    while (R.rulenum < minRulenum) do
10)    minsupportCount--;
11)    R = AR-CRS-2();
12)   end
13)   return R;
```

Fig. 1. The AR-CRS-1 Algorithm

3. Finally, AR-CRS-1 will check if the number of rules is less than minRulenum; if it is, it will keep decreasing the minimum support count until the rule number is greater than or equal to minRulenum.

Within a given support, rules with shorter bodies are mined first. Hence, if with minimum support count say 15 there is no rule available, but with minimum support count 16 there are at least maxRulenum rules, then AR-CRS-1 will return the shortest maxRulenum rules with support count of at least 16.

AR-CRS-2 AR-CRS-2 is a variant of CBA-RG [9] and therefore of the Apriori algorithm [2] as well. AR-CRS-2 is a variant of CBA-RG in the sense that instead of mining rules for all target classes, it only mines rules for one target item. It differs from CBA-RG in that it will only mine a number of rules within a certain range. When it tries to generate a new rule after having obtained maxRulenum rules already then it simply terminates its execution and returns the rules it has mined so far.

Here we use *k-condset* to denote a set of items (or *itemset*) of size k which could form a rule: *k-condset* $\Rightarrow$ *target-item*. The support count of the *k-condset* (called $condsupCount$) is the number of transactions that contain the *k-condset*. The support count of the corresponding rule (also called $rulesupCount$ of this *k-condset*) is the number of transactions that contain the *condset* as well as the target item.

AR-CRS-2 is very similar to CBA-RG as mentioned above. We describe the process here in order to make our paper self-contained. Association rules are generated by making multiple passes over the transaction data. The first pass counts the $rulesupCounts$ and the $condsupCounts$ of all the single items and finds the frequent 1-*condsets*. For pass $k > 1$, it generates the candidate frequent k-*condsets* by using the frequent $(k-1)-condsets$; then it scans all transactions

to count the *rulesupCounts* and the *condsupCounts* of all the candidate *k-condsets*; finally, it will go over all candidate *k-condsets*, selecting those whose *rulesup* is above the minimum support as frequent *k-condsets* and at the same time generating rules *k-condset* $\Rightarrow$ *target-item*, if the confidence of the rule is above the minimum confidence. The algorithm is presented in Figure 2.

```
Input: Transactions, targetItem, minConfidence, maxRulenum, minsupportCount
Output: mined association rules

1)   F_1 = {frequent 1-condsets};
2)   R = genRules(F_1);
3)   if R.rulenum = maxRulenum then return R;
4)   for (k = 2; F_{k-1} ≠ ∅; k++) do Begin
5)     C_k = candidateGen(F_{k-1});
6)     for each transaction t ∈ Transactions do Begin
7)       C_t = all candidate condsets of C_k contained in t;
8)       for each candidate c ∈ C_t do Begin
9)         c.condsupCount++;
10)        if t contains targetItem then c.rulesupCount++;
11)      end
12)    end
13)    F_k = {c ∈ C_k | c.rulesupCount ≥ minsupportCount};
14)    R = R ∪ genRules(F_k);
15)    if R.rulenum = maxRulenum then return R;
16)  end
17)  return R;
```

Fig. 2. The AR-CRS-2 Algorithm

2.2 Real-time recommendation

A requirement of many recommender systems is the realtime response. Our algorithm can satisfy the real time constraint for the following reasons:

- We mine rules offline for article associations;
- The training data to mine rules for one target user is only a small subset of all the ratings, i.e., the ratings from training users and the target user for the articles that the target user has rated. So the training data size is small;
- The mining process AR-CRS-2 will stop after it mines *maxRulenum* rules. If the *maxRulenum* is small, it is very fast;
- In the main process, we choose an initial minimum support count according to users' like ratios. For most users, the main process only need to call the mining process two or three times. We can switch to article associations for users who need more iterations.

2.3 Algorithm Implementation

We have implemented our algorithm in C++. In order to speed up the mining process, we have chosen data structures that efficiently support the key opera-

tions of our algorithm: (1) subset test: how to find all candidate *condsets* that are contained in one transaction; (2) candidate generation - join step: how to find frequent *condsets* that could be joined together; (3) candidate generation - prune step: how to test if any (k-1)-subset of a candidate k-*condset* is a frequent (k-1)-*condset*. As described in [2], using a hash-tree to store candidate itemsets and a bitmap to store a transaction could speed up the support counting process. In addition to using bitmaps and hash–trees, we use a new data structure that we call *set–tree*, which we have designed and implemented to facilitate the join and prune operations on candidate itemsets.

3 Experimental Evaluation

In this section, we describe an experimental evaluation of our algorithm in the context of collaborative recommendation based on associations between users.

3.1 Training and Test Data

We use the EachMovie Dataset as the test-bed of our approaches. The EachMovie data set is an online data source provided by DEC's Systems Research Center [10]. It contains ratings from 72,916 users for 1,628 movies. User ratings were recorded on a numeric six–point scale (0.0, 0.2, 0.4, 0.6, 0.8, 1.0). We use 1,000 training users and 100 test users from the EachMovie dataset, all of who have rated more than 100 movies each.

3.2 Recommendation Using the Mined Association Rules

The same mining process may be used to mine a certain number of rules for each user/article for both user associations and article associations. The main difference between the implementation of these two association types is that we use different training data to mine the association rules. In the current paper we restrict our attention to user associations. Further information about item associations and a combination of the two types of associations is provided in an upcoming paper.

Mapping Ratings to Transactions The conversion from item ratings available for recommendation tasks to "transactions" as required for association rule mining is determined by what kind of associations and how many levels of associations we want to discover. We map the numeric ratings for an item into two categories: *like* and *dislike* according to whether the rating for the item is greater than or less than some chosen threshold value. Then we convert the chosen *like* and *dislike* ratings into transactions. In order to mine *like* and *dislike* associations among users, we extend each user, say $user_k$, to two "items", one item corresponding to [$user_k$: like] and another item corresponding to [$user_k$: dislike]. If $user_k$ likes an article, then the corresponding transaction contains the item

[$user_k$: like]; If $user_k$ dislikes the article, the corresponding transaction contains the item [$user_k$: dislike]; If $user_k$ did not rate the article, the corresponding transaction does not contain either of these two items.

Recommendation Strategy The rules mined are akin to [$training\_user_1$: $like$] AND [$training\_user_2 : dislike$] $\Rightarrow$ [$target\_user : like$]. For a test article of the target user, if the $training\_user_1$ likes this article and the $training\_user_2$ dislikes this article, then we say this rule *fires* for this article. We associate each rule with a score, which is the product of the support and the confidence of the rule. We also assign a score to each article, which is the sum of the scores of all the rules that fire for that article. If $score_{article_i}$ is greater than a threshold, then we recommend $article_i$ to the target user.

3.3 Performance Measurement

We use the *accuracy*, a commonly used performance measure in machine learning, together with two standard information retrieval measures, *precision* and *recall*. Accuracy is the percentage of correctly classified articles among all those classified by the system; Precision is the percentage of articles recommended to a user that the user likes; Recall is the percentage of articles liked by a user that are recommended to him/her. For recommendation tasks, precision is perhaps most significant because we are more concerned about making high quality recommendations than about recommending a large number of items.

For all the experiments, we employed a *4-fold* cross-validation approach when evaluating the performance for each test user, and we report the averaged performance over the 100 test users.

3.4 Parameters

The main parameters for our system are listed below:

- the *like* and *dislike* threshold for the ratings;
- the maximum length of rules.
- the minimum confidence of the association rules;
- minRulenum and maxRulenum.

Results for different parameter values are described in the next section. Currently, we set 0.7 as the *like* threshold, i.e., if a user's rating for an article is greater than 0.7, then we assume that user likes the article. With this *like* threshold, the ratio of the number of movies liked over the total number of movies rated among all the test users is 0.45.

3.5 Performance Results

Maximum Rule Length We use rule length to refer to the number of the items present in the body of a rule. Table 1 lists the performance for different

Rule Length	2	4	6	8	10
Accuracy	0.693712	0.696117	0.695676	0.69759	0.694547
Precision	0.704357	0.724006	0.733482	0.737896	0.736086
Recall	0.572606	0.545425	0.528625	0.528411	0.520813

Table 1. Performance for Different Maximum Rule Length

maximum rule lengths. Here we choose minimum confidence as 100%, and the number of rules in the range [5,100].

From Table 1 we could see that when the maximum rule length is around 8, we get the best performance. Given these results and the fact that longer rules are in more danger of overfitting the data, we have chosen the maximum rule length to be 8. We use this maximum rule length for all the following experiments.

Minimum Confidence We tested the performance when varying the minimum confidence. The results are shown in Figure 3 and summarized below.

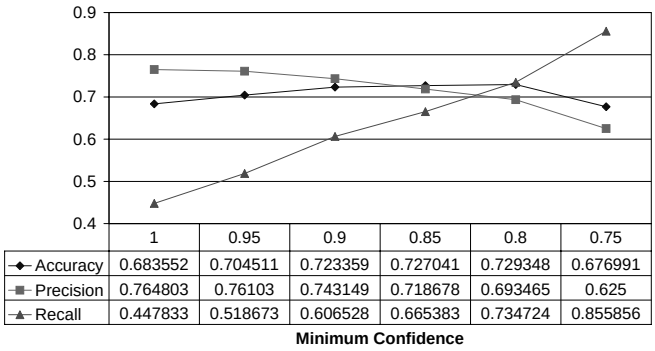

Fig. 3. Performance for Different Minimum Confidence Values

- The minimum confidence has a significant impact on the performance: the higher the minimum confidence, the higher the precision but the lower the recall. We achieve the highest precision of 0.76 with a recall of 0.45 for the minimum confidence of 100%.
- When the minimum confidence is varied, a tradeoff between the precision and the recall becomes evident. If we take accuracy as a measure that reflects both the precision and the recall, we achieve the best combination of the precision and the recall not with minimum confidence 100% but with minimum confidence from 80% to 90%. This is because more liked articles are identified as like, i.e., we have higher recalls.

Although we think the precision is the most important measure for a recommender system, it is important to obtain a good combination of very high

precision with a reasonably high recall. Taking this into account, we use a minimum confidence of 0.9 for the remaining experiments.

Range for the Number of Rules In order to decide what is the appropriate range of number of rules, we ran experiments with different ranges. As shown in Figure 4, we achieve quite similar performance for ranges [10,100], [20,200], and [50,500]. Our experiments verify that a limited number of rules is desirable for making recommendations to a user. Too many rules do not improve recommendation performance and lead to an unnecessarily large run time.

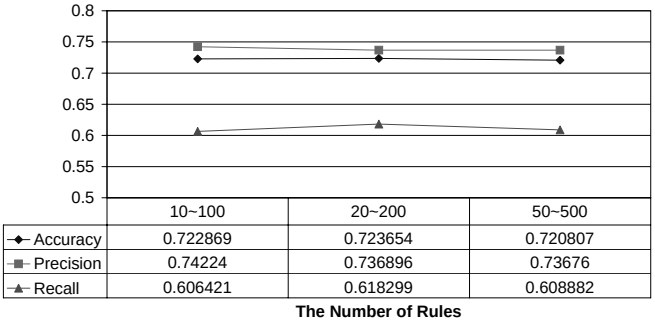

Fig. 4. Performance for Different Rule Set Sizes

***Like* and *Dislike* Associations** In order to obtain both *like* and *dislike* associations, we map ratings into *like* and *dislike* by using two thresholds: the *like* threshold and the *dislike* threshold. The *like* associations we discussed before correspond to choosing the *dislike* threshold as 0. Figure 2 gives the comparison of the performance for different *dislike* thresholds.

Dislike Threshold	0	0.3	0.7
Accuracy	0.721936	0.724341	0.718549
Precision	0.725214	0.726137	0.717452
Recall	0.634029	0.640663	0.637453

Table 2. Performance for Like and Dislike Associations. like_threshold = 0.7

There is no significant difference between the performance for different *dislike* thresholds. So employing *like* and *dislike* associations does not outperform employing *like* associations only.

4 Conclusions

We have presented a new algorithm for mining association rules with a specific target predicate in the heads of the rules. Such rules are needed in applications such as recommender systems, which are important for electronic commerce. Unlike most existing association rule mining algorithms, which require that the minimum support of the rules to be mined be specified in advance, our algorithm adjusts the minimum support during the mining process so that the number of rules generated lies within a specified range. This keeps the running time under control, and ensures that enough rules are available for the target predicate.

References

1. Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 207–216, Washington, D.C., May 1993. ACM.
2. Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th VLDB Conference*, pages 487–499, Santiago, Chile, 1994.
3. Marko Balabanovic and Yoav Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, March 1997.
4. Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *Proc. of the Fifteenth International Conference on Machine Learning*, Madison, Wisconsin, 1998. Morgan Kaufmann Publishers.
5. Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 255–264, New York, May 1997. ACM, ACM Press.
6. P. Hájek, I. Havel, and M. Chytil. The guha method of automatic hypotheses determination. *Computing*, 1:293–308, 1966.
7. P. Hájek, S. Sochorová, and J. Zvárová. GUHA for personal computers. *Computational Statistics & Data Analysis*, 19:149–153, 1995.
8. Petr Hájek and Tomas Havranek. On generation of inductive hypotheses. *Int. J. Man-Machine Studies*, 9:415–438, 1977.
9. Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 80–86, New York, August 1998.
10. P. McJones. Eachmovie collaborative filtering data set. http://www.research.-digital.com/SRC/eachmovie, 1997. DEC Systems Research Center.
11. J. Rauch. GUHA as a data mining tool. In M. Wolf and U. Reimer, editors, *Proc. of the 1st Intl. Conf. on Practical Aspects of Knowledge Management*, 1996.
12. P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW94)*, pages 175–186, 1994.
13. U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of the Conference on Human Factors in Computing Systems (CHI95)*, pages 210–217, 1995.
14. Ramakrishnan Srikant and Rakesh Agrawal. Mining quantitative association rules in large relational tables. In *Proc. of the 1996 ACM SIGMOD Conference on Management of Data*, Montreal, Canada, June 1996. ACM.