An Efficient Association Rule Mining Algorithm for Classification

A. Zemirline, L. Lecornu, B. Solaiman, and A. Ech-cherif*

ITI Department, ENST Bretagne
29285 Brest, France
{abdelhamid.zemirline,laurent.lecornu,basel.solaiman}@enst-bretagne.fr
Laboratoire Lamosi-USTO-Oran*
Algeria
a.echerif@alum.rpi.edu

Abstract. In this paper, we propose a new Association Rule Mining algorithm for Classification (ARMC). Our algorithm extracts the set of rules, specific to each class, using a fuzzy approach to select the items and does not require the user to provide thresholds. ARMC is experimentaly evaluated and compared to state of the art classification algorithms, namely CBA, PART and RIPPER. Results of experiments on standard UCI benchmarks show that our algorithm outperforms the above mentionned approaches in terms of mean accuracy.

1 Introduction

Association Rule Mining [1] is the most popular data mining task due to its numerous applications and the efficiency of the APRIORI algorithm which produces interesting relationships among items of large databases. Associative classification [8], [7] attempts to solve classical classification problems using association rule mining. Successful algorithms in this category including the well known $C4.5$ predict the class label for novel (unseen) examples .

In this paper, we propose a new Association Rule Mining algorithm for Classification (ARMC) based on the extraction of both common and exception rules, specific to each class. ARMC uses a fuzzy method to automatically select the items that generate the rules and does not require the user to provide thresholds.

ARMC proceeds by combining on one hand, the weighted voting and the decision list algorithms. On the other hand, a fuzzy method is used in order to distinguish the important rules from the less important ones for each class.

The remainder of this paper is structured as follows. In the next section the basic terminology used throughout the paper is introduced. Related research on associative classification is surveyed in section 3. The ARMC algorithm is presented in section 4. Experimental results are given in section 5. We conclude the study in section 6.

2 Preliminaries

2.1 Association Rule Mining

Let B be a database consisting of one table over n items $I = \{a_1, a_2 \cdots, a_n\}$ whose values are nominal and containing k instances.

A database instance d is said to satisfy an item set $X \subseteq \{a_1, a_2, \cdots a_n\}$ if $X \subseteq d$.

An association rule is an implication $X \Rightarrow Y$ where $X, Y \subseteq I$, $Y \neq \emptyset$ and $X \cap Y = \emptyset$.

The support of an item set X is the number of database instances d which satisfy X: $sup(X) = \frac{|\{t \in B |\ X \subseteq t\}|}{|B|}$.

The confidence of an association rule is a percentage value that shows how frequently the consequent part occurs among all the groups containing the rule antecedent part : $conf(X \rightarrow Y) = \frac{|\{t \in B | X \cup Y \subseteq t\}|}{|\{t \in B | X \subseteq t\}|}$.

2.2 Association Rules for Classification

Given a relational table B containing N cases (training examples) belonging to C classes, each one is described by an itemset and let I be the set of all items of B. Classical classification methods seek a rule or a hypothesis that predicts the label of an unseen example.

Table 1. An example of training dataset

Id	Set of items	Class	Id	Set of items	Class
1	x_1, x_4, x_9	A	8	x_1, x_4, x_9	B
2	x_1, x_4, x_{10}	A	9	x_1, x_4, x_9	B
3	x_1, x_4, x_{11}	A	10	x_1, x_5, x_{11}	B
4	x_1, x_5, x_{11}	A	11	x_2, x_6, x_{12}	B
5	x_2, x_6, x_{12}	A	12	x_2, x_6, x_{10}	B
6	x_2, x_7, x_{13}	A	13	x_2, x_6, x_{12}	B
7	x_3, x_8, x_{14}	A	14	x_2, x_7, x_{12}	B
			15	x_3, x_8, x_{13}	B

The class association rule is an implementation of the form $X \rightarrow c$, where $X \subseteq I$, and $c \in C$.

The objectives are to generate the complete set of class association rules that satisfy the minimum support as well as the minimum confidence ($minConf$) constraints and to build a classifier from the class association rule set.

The classifier is applied to classify examples. To this aim, one combines the prediction of all rules which satify the exemple: if there is only one rule, the consequent of this rule is taken to be the predicted class for the example; if there is no rule satisfying the example, then a default class is taken to be the predicted class; and if there are multiple rules satisfying the example, then their predictions must be combined. Various strategies for realizing this are discussed in section §3.3.

3 Methodology

The associative classification algorithm can be divided into two fundamental parts: association rule mining and classification. The mining of association rules is a typical data mining task that works in an unsupervised manner. A major advantage of association rules is that they are theoretically capable of revealing all interesting relationships in a database.

3.1 Current Approaches for Discovering Frequent Items and Rule Generation

Many approaches for frequent items discovery and rule generation have been proposed. In this section, we present some paradigms used in the most important association rules and associative classification methods: type algorithms *Apriori* Agrawal et *al.* [1] discover large frequent itemsets, by making multiple passes over the dataset. In the first pass, the support of individual are counted and then the frequent ones are selected. In each subsequent pass, the algorithm starts with a seed set of frequent items found in the previous pass. This process continues until no new frequent items are found. This algorithm needs to scan several times the dataset in order to produce the frequent itemsets.

The CMAR algorithm [7] adopts some properties of the FP-growth method [5]. FP-growth is a frequent pattern mining algorithm which is faster than conventional *Apriori* method. Whose advantages are: it constructs a highly compact Frequent-Pattern-Tree, which is smaller than dataset and that avoids to scan the dataset in the subsequent mining process. Also, in contrary to FOIL and CPAR methods, it avoids costly candidate generation.

The MMAC [11], [12] algorithms use the technique based on intersection methods [15]. They scan the dataset once to count the support of individual items and determine those having minimum support. They store items along with their location (rowIds) inside arrays. Then, by intersecting the rowIds of the frequent items discovered so far, can easily obtained the remaining frequent items that involve more than one attribute.

3.2 Current Approaches for Extracting Exception Rules

After performing data mining tasks as using association rule methods [6], which usually terminate with a large set of rules, there is a need to find some interesting rules that can be used by the decision maker. In [6], [9], exception rules are defined as the rules that contradict the common belief. They play an important role in making critical decision. The exception rules usually have a small cardinality as a set, and they are not known or omitted.

There are some intuitive ways [6], [9] such as multi-support or generate-and-remove. For discovering exception rules which generate the weak patterns using the traditional data mining techniques. For example, in order to find exception rules, we can introduce two support thresholds $[s_1, s_2]$ as delimiter and we search the itemsets whose support values fall into the range $[s_1, s_2]$.

Generate-and-remove is straightforward method where the rules are generated from the training data T by an induction algorithm as follows: remove T' from T that is covered by R; then generate rules R' on $T - T'$; repeat the process until no data are left. This procedure can find many weak patterns.

3.3 Current Approaches to Classification

In most associative classification methods such as CBA, MMAC, ... we use the classification approach which is called decision list algorithm. This approach takes into account just one rule in order to classify an instance. Therefore the set of class association rules is stored in a list data structure whose first rule covering the instance to be classified is used for prediction.

In CBA [8], the class association rules are stored as follows: given the rules, r_a and r_b, r_a precedes r_b if the confidence of r_a is greater than that of r_b, or if their confidences are identical, but the support of r_a is greater than that of r_b, or both the confidences and support of r_a and r_b are the same but r_a is generated earlier than r_b.

Any associative classification using this approach, has approximately these same definition rules. The resulting classifier has the form: $< r_1, r_2, ..., r_n$, $default\_class >$ where r_i precedes r_{i+1}. For classifying a new case, the first rule that satisfies the case classifies it. If no rule applies to the case, it takes the default class.

Another approach called the weighted vote algorithm. Uses a set of the best rules of each class for prediction, according to the following procedure: (1) select all the rules whose bodies are satisfied by the instance; (2) from the rules selected in step (1), select the best rules for each class; and (3) compare the measure of the combined effect of selected rules of each class and choose the class with the highest measure.

There are many possible ways to measure the combined effect of a group of rules [13],[7]. For example, one can compute the average of the confidence values of a group of rules or expected accuracy [13] as the predicted class. Another alternative is to use the strongest rule as a representative of the predicted class [7]. In other words, the rule with highest χ^2 value is selected. However, this method is based on a single rule for making predictions.

4 ARMC Algorithm

We present a new algorithm for associative classification, based on multi-rule classification. Our algorithm extracts rules from data subset instead of the whole dataset. Each class of the dataset has its instance subset which consists of the cases of the same class. In this approach, exception rules are discovered by all.

4.1 Item Discovery and Rule Generation

Our approach generates local rules (i.e., the rules are generated from a subset of dataset that is composed of the instances of the same class label for freqent items discovery and rule generation tasks, and employs a technique).

Based on the intersection method [15] in order to scan the dataset once and to compute confidence value of rules from confidence value of frequent single items.

It clusters the training data set T into several subsets. Each subset holds the instances of the same class. We scan each subset once in order to count the occurrence of single items. Each single item has a number of occurences for each class. For a given class, the whole number of items is used in order to build a memberships function to the given class. These memberships sort the single items in two subsets: one rare items subset and another frequent items one. We use fuzzy sets [14] subsets in order to extract the frequent items inherent in a given class.

The support threshold value for some methods is pre-fixed, in order to select the items iteration numbers exceeding this support threshold. Although this procedure is widely used by most methods it has some disadvantages as:

– If the whole number of item iterations are higher or equal than the threshold, then no frequent items are selected.
– Some interesting items are not selected as frequent items though their iteration numbers are close to the threshold.

In order to build membership functions for determination of the frequent items, we use the method introduced in [14]. This method enables the interpretation of the linguistic variable frequency for the term set $\{rare, frequent\}$. Each term is characterized by a fuzzy sets in a universe of frequency values of items for a specific class label.

Before presenting how to build these membership functions for class c_i, some notations have to be specified: use of subsets of a database instead of the entire dataset requires the adaptation of certain functions such as the *support* function which calculates the frequency of appearances of an itemset in the whole data base. The *support* function is adapted to calculate the support of an itemset in a subset of data and named *supportLocal*:

$$suppLocal_{c_i}(X \rightarrow c_i) = \frac{|\{t \in B_{c_i} |\ X \subseteq t\}|}{|B_{c_i}|}$$

where : B_{c_i} is a set of instances of the class c_i.

To define these membership functions, we denote the linguistic variable which is characterized by [14]: $(x, T(x), U)$ where

– x is the linguistic variable. In our application x is equal to the linguistic variable frequency.
– $T(x)$ is the set of terms associated with the linguistic value, in which the frequency is represented according to the following set $\{$*Never, Rare, Frequent, Always*$\}$.
– U is the universe of discourse and $U = \{s \in I |\ suppLocal_{c_i}(s \rightarrow c_i)\}$.

The terms of $T(x)$ are characterized by fuzzy subsets defined by the following membership functions [14]:

Table 2. SupportLocals calculated from training dataset of Table 2

Item	Support	$suppLocal_A$	$suppLocal_B$
x_1	7/15	4/7	3/8
x_2	6/15	2/7	4/8
x_3	2/15	1/7	1/8
x_4	5/15	3/7	2/8
x_5	3/15	1/7	2/8
x_6	3/15	1/7	2/8
x_7	2/15	1/7	1/8
x_8	2/15	1/7	1/8
x_9	2/15	1/7	1/8
x_{10}	3/15	1/7	2/8
x_{11}	3/15	2/7	1/8
x_{12}	4/15	1/7	3/8
x_{13}	2/15	1/7	1/8
x_{14}	1/15	1/7	0

– K: is the set of centroids obtained by the K-means algorithm which is applied to U for $K = \{0, s_{rare}, s_{freq}, 1\}$ (freq = frequent).
– $\mu_{c_i,rare}$: corresponds to the membership function in the linguistic term *rare*. It is built from a set of instance frequencies which belong to the class c_i.

$$\mu_{c_i,rare}(s) = \begin{cases} 1 & \text{if } s \leq s_{rare} \\ 1 - \frac{s - s_{rare}}{s_{freq} - s_{rare}} & \text{if } s_{rare} < s \leq s_{freq} \\ 0 & \text{otherwise.} \end{cases}$$

– $\mu_{c_i,freq}$: corresponds to the membership function in the linguistic term $frequent$. It is built from a set of frequency values of instances which belong to label class c_i.

$$\mu_{c_i,freq}(s) = \begin{cases} 0 & \text{if } s \leq s_{rare} \\ 1 - \frac{s - s_{freq}}{s_{freq} - s_{rare}} & \text{if } s_{rare} < s \leq s_{freq} \\ 1 & \text{otherwise.} \end{cases}$$

where : $s = suppLocal_{c_i}(X \rightarrow c_i)$ such as $X \in I$

These membership functions are able to define fuzzy subset of single items. The frequent single items are all the items in which the frequency value of membership degree of the term *frequent* is higher than that of term rare, i.e. $\{s \in I|\ \mu_{c_i,freq}(suppLocal_{c_i}(s \rightarrow ci)) > \mu_{c_i,rare}(suppLocal_{c_i}(s \rightarrow c_i))\}$. Therefore two subjective thresholds (first support $t_1^{c_i}$ and second support $t_2^{c_i}$) can be computed (Figre 1). The first support $t_1^{c_i}$ is the value that corresponds to the intersection of $\mu_{c_i,freq}$ and $\mu_{c_i,rare}$. The second support $t_2^{c_i}$ is the centroid s_{rare} ($t_2^{c_i} = s_{rare}$).

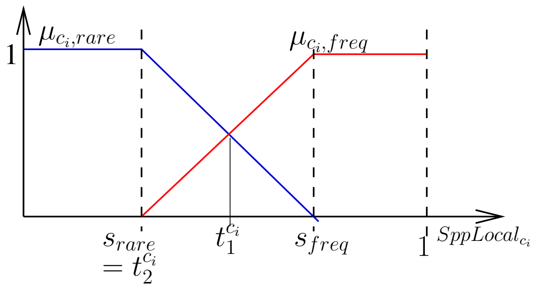

Fig. 1. Membership functions ($\mu_{c_i,freq}$, $\mu_{c_i,rare}$) and the thresholds (t_1, t_2)

According to this method, we define subjective thresholds for a specific class by selecting. All the single items whose local support value exceeds the first threshold. We call these items frequent single items. Then, to produce the rules from these items, the local support value is calculated, and the rules whose local support is greater than the first threshold $t_1^{c_i}$ are considered as common rules. The others are considered as exception rules if their local support exceeds the second threshold $t_2^{c_i}$ and they are stored respectively in common rule set and exception set of this class.

In order to illustrate the advantages of the combination of these approaches to find the exception rules, we present an example: the training dataset of Table 2, represents a set of the instances whose label class is A or B. From this table, we apply our method to produce the rules. In table 3 and 3, there are the rules extracted from the instances whose class label is A and B respectively. In these tables, the rules are grouped by the way either by the first or by the second threshold or by the generate-and-remove.

Double local support thresholds are computed by the method defined above. For the instances whose class label is A, the results obtained are $t_1^A = 0.29$ and to $t_2^A = 0.17$. Using the first threshold t_1^A, we select the set of the single frequent item $\{x_1, x_4\}$, with which, our method produces three rules: $x_1 \rightarrow A$, $x_1x_4 \rightarrow A$ and $x_4 \rightarrow A$. The last rule is pruned because its included in the second rule and the both of them cover the same instances. Using the second threshold t_2^A, the items x_2 and x_{11} are added to the set of the frequent single item, which are combined to generate many rules, but only rules $x_1x_{11} \rightarrow A$ and $x_2 \rightarrow A$ are saved. The other rules are pruned as: $x_2x_4 \rightarrow A$, $x_2x_{11} \rightarrow A$ because they do not cover any instances. The rule $x_3x_8x_{13} \rightarrow A$ is found by the generate-and-remove approach.

For class B, results of threshold computing gives $t_1^B = 0.307$ and $t_2^B = 0.23$ respectively. The items x_1, x_2, x_6, $and x_{12}$ are the frequent single items. The rules generated from these items are those having the Id 1, 2, 3, and 4 of Table 4. Other rules are generated but they are pruned because, either they do not cover any instances or they are included in other rules. the second set of frequent single items are $\{x_{10}, x_4\}$ and the rules generated by adding these new frequent

Table 3. Rules found in training subset instances of class A

Approach	Id	Rule	Sup	SupL
1st threshold t_1^A	1	$x_1 \rightarrow A$	2/7	4/7
	2	$x_1x_4 \rightarrow A$	3/14	3/7
2nd threshold t_2^B	3	$x_2 \rightarrow A$	1/7	2/7
	4	$x_1x_{11} \rightarrow A$	1/7	2/7
gen-and-rem	5	$x_3x_8x_{14} \rightarrow A$	1/14	1/7

Table 4. Rules found in training subset instances of class B

Approach	Id	Rule	Sup	SupL
1st threshold t_1^B	1	$x_2 \rightarrow B$	4/15	1/2
	2	$x_1 \rightarrow B$	3/15	3/8
	3	$x_2x_6 \rightarrow B$	3/15	3/8
	4	$x_2x_7 \rightarrow B$	3/15	3/8
2nd threshold t_2^B	5	$x_1x_4 \rightarrow B$	1/3	1/4
	6	$x_2x_6x_{12} \rightarrow B$	2/15	1/4
	7	$x_{10} \rightarrow B$	2/15	1/4
gen-and-rem	8	$x_3x_8x_{13} \rightarrow B$	1/15	1/8

items are the rules with Id 5, 7, 8 of Table 4. The rule $x_3x_8x_{13} \rightarrow B$ is found by the generate-and-remove approach.

This approach is quite effective in terms of runtime and storage because it does not rely on the traditional Apriori approach [2] of discovering frequent items requiring multiple scans. Moreover, it generates more rules than the traditional method and therefore avoids missing important ones.

4.2 Classification

Our approache builds a global model for classification by combining the two approaches presented above (3.3): the weighted vote algorithm and the decision list algorithm. Firstly, for each class c_i, we store a set of rules R_{c_i} in two sets: the set of the *common* rules to class c_i and the set of the *exception* rules to the class c_i.

Given R a rule set containing the rules for each class, R is composed of common and exception rules, and given also the instance, we use the best rules of each class for prediction, with the following procedure:
(1) Select all the rules from R whose bodies are satisfied by the instance;
(2) Compare the average local support value of the *common* rules of each class and choose the class with the highest average as the predicted class. If there are no the common rules for a specific class, the local suport of the best rule from the specific class is selected.

The following function is used for selecting and combining rules as discussed above:

$$Average_{c_i}(e) = \begin{cases} \frac{\sum_{r \in R_{c_i,com}(e)} SuppLocal_{c_i}(r)}{|R_{c_i,com}(e)|} \\ if\ R_{c_i,com}(e) \neq \emptyset. \\ \\ Max_{r \in R_{c_i,\neg com}(e)}(SuppLocal_{c_i}(r)) \\ if\ R_{c_i,com}(e) = \emptyset\ and\ R_{c_i,\neg com}(e) \neq \emptyset. \\ \\ 0\ \ else. \end{cases}$$

where:

- $R_{c_i,com}(e)$: set of common rules which cover the instance e from class c_i.
- $R_{c_i,\neg com}(e)$: set of exception rules which cover the instance e from class c_i.

We use multiple rules in prediction for the following reasons: the accurate rules cannot be precisely estimated and we cannot expect that any single rule can perfectly predict the class label of every example satisfying its body. Moreover, we use the best rule instead of using all of them because there are different number of rules for different classes and we do not want to use low rank rules in prediction when there are already enough rules to make a prediction.

4.3 ARMC Algorithm

Our algorithm proceeds in two phases: The first phase generates the common and exception rules. The last one builds the classifier.

Before presenting this algorithm, some notations have to be specified:

- B_{c_i} is the set of instances of the class c_i.
- R_{c_i} is the set of rules extracted from B_{c_i}.
- S_f is the *frequent* item set.
- S_r is the *rare* item set.
- $B_{c_i}(r)$ is set of istances covered by the rule r.
- $B_{c_i}(R_{c_i})$: is the set of instances covered by the set of rules R_{c_i}.

ARMC Algorithm
Input: Database B.
Output: The rule set for each class (R_{c_i}).

```
1. Scan the database $B$ to group the instances according to class labels.
   For each set $B_{c_i}$ :
   (a) Generation of $S_f$ and $S_r$ from $B_{c_i}$
   (b) Generation of common rules :
       For each $X_1 \subset S_f$ and $X_2 \subset S_f$
       $R_{c_i} \leftarrow < X_1 \cup X_2 \rightarrow c_i >$ if $suppLocal_{c_i}(X_1 \cup X_2 \rightarrow c_i) \geq t_1^{c_i}$
   (c) Generation of exception rules :
       For each $X_1 \subset S_f$ and $X_2 \subset S_r$
       $R_{c_i} \leftarrow < X_1 \cup X_2 \rightarrow c_i >$ if $suppLocal_{c_i}(X_1 \cup X_2 \rightarrow c_i) \geq t_2^{c_i}$
       For each $X_1 \subset S_r$ and $X_2 \subset S_r$
       $R_{c_i} \leftarrow < X_1 \cup X_2 \rightarrow c_i >$ and $suppLocal_{c_i}(X_1 \cup X_2 \rightarrow c_i) \geq t_2^{c_i}$
   (d) Purning $R_{c_i}$ :
       Remove $X' \rightarrow c_i$ if $B_{c_i}(X' \rightarrow c_i) \subseteq B_{c_i}(X \rightarrow c_i)$ and $X \subseteq X'$
   (e) $B_{c_i} \leftarrow \ B_{c_i} - B_{c_i}(R_{c_i})$
   (f) Repeat (a), (c), (d) and (e) until $B_{c_i} = \emptyset$
2. Build the classifier.
```

In phase 1 of our algorithm, double local support threshold is applied, then the generate-and-remove is applied to discover the remaining rules in order to cover all the training instance by rules. This combination enables us to find more rules than the traditional methods and also to find some rules that can be to be generated by either generate-and-remove approach alone or by double support threshold approach alone.

5 Evaluation

We have evaluated the accuracy, efficiency and scalability of our algorithm. In this section, we report on our experimental results by comparing our algorithm with three popular classification techniques namely RIPPER [3], PART [4] and CBA [8] in order to evaluate the predictive power of the proposed method. As in [11], datasets from UCI Machine Learning Repository [10] are used. 10-fold cross validation is used for every dataset. In our experiments, confidence threshold was set to 3, the datasets selected from [10] were reduced by ignoring their integer and/or real attributes. The accuracy of our algorithm has been computed using the top label evaluation measure. We obtain better results with ARMC.

Table 5 shows the classification average accuracy of the classifiers extracted by RIPPER, PART, CBA, and ARMC on the 26 benchmark problems. Our algorithm outperforms the rule-learning methods in terms of accuracy rate, and the won_loss_tied records of ARMC against RIPPER, PART and CBA are 16_10_0, 15_10_1 and 18_8_0, respectively.

Table 6 compares the number of classification rules discovred by RIPPER, PART, CBA and ARMC algorithms. We can see that on average, ARMC finds many more rules than RIPPER, PART and CBA. The reason why ARMC finds more rules is that it mines rules which cover all the instances of a dataset by the generate_and_remove procedure.

Table 7 shows the classification average accuracy of the different classifiers approach on the 26 benchmark problems.

Table 5. Classification accuracy average of PART, RIPPER and ARMC

Algorithm	RIPPER	PART	CBA	ARMC
Average	83.519	83.923	83.242	85.367

Table 6. Rule number Average of PART, RIPPER and ARMC

Algorithm	RIPPER	PART	CBA	ARMC
Average	6.29	18.58	44.71	47.79

Table 7. Classification average accuracy of ARMC-1, ARMC-Multi, ARMC

Approach	ARMC-1	ARMC-Multi	ARMC
Average	84.91	82.1	85.36

The ARMC-1 denotes the result of the weighted vote algorithm when sed by our method. The ARMC-Multi denotes the result of the decision list algorithm when used by our method.

We conclude that the combination of two approaches (the weighted vote algorithm and the decision list algorithm) outperforms the weighted vote algorithm and the decision list algorithm.

6 Conclusion

A new approach for classification rules has been proposed having different features: (1) Covering all training instances and leaving no unclassified instances (2) requiring only one pass to discover rules (3) using a novel approach for building a classification model. All these features are not offered by the traditional associative classification methods. The first series of experiments on databases in UCI machine learning repository showed that ARMC is robust and highly effective for various classification tasks.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proc. of the 1993 ACM SIGMOD International Conference on Management of Data SIGMOD 1993, Washington, DC, pp. 207–216 (1993)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Bocca, J.B., Jarke, M., Zaniolo, C. (eds.) Proc. 20th Int. Conf. Very Large Data Bases, VLDB, pp. 487–499. Morgan Kaufmann, San Francisco (1994)
3. Cohen, W.: Fast effective rule induction. In: Proceedings of the 12th International Conference on Machine Learning, pp. 115–123. Morgan Kaufmann, San Francisco (1995)
4. Frank, E., Witten, I.: Generating accurate rule sets without global optimisation. In: Morgan Kaufmann Madison (ed.)Proceedings of the Fifteenth International Conference on Machine Learning, pp. 144–151 (1998)
5. Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., Hsu, M.: Freespan: frequent pattern-projected sequential pattern mining. In: KDD, pp. 355–359 (2000)
6. Hussain, F., Liu, H., Suzuki, E., Lu, H.: Exception rule mining with a relative interestingness measure. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 86–97 (2000)

7. Li, W., Han, J., Pei, J.: Cmar: Accurate and efficient classification based on multiple class-association rules. In: ICDM, pp. 369–376 (2001)
8. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: KDD, pp. 80–86 (1998)
9. Liu, H., Lu, H., Feng, L., Hussain, F.: Efficient search of reliable exceptions. In: Zhong, N., Zhou, L. (eds.) PAKDD 1999. LNCS (LNAI), vol. 1574, pp. 194–203. Springer, Heidelberg (1999)
10. Merz, C.J., Murphy, P.M.: UCI repository of machine learning databases. Department of Information and Computer Science, University of California, Irvine (1996)
11. Thabtah, F.A., Cowling, P.I., Peng, Y.: Mmac: A new multi-class, multi-label associative classification approach. In: ICDM, pp. 217–224 (2004)
12. Thabtah, F.A., Cowling, P.I., Peng, Y.: Multiple labels associative classification. Knowl. Inf. Syst. 9(1), 109–129 (2006)
13. Yin, X., Han, J.: CPAR: Classification based on predictive association rules. In: Proceedings of 2003 SIAM International Conference on Data Mining, San Fransisco, CA (2003)
14. Zadeh, L.: The concept of a linguistic variable and its application to approximate reasoning - ii. Information Sciences (Part 2) 8(4), 301–357 (1975)
15. Zaki, M.J., Parthasarathy, S., Ogihara, M., Li, W.: New algorithms for fast discovery of association rules. Technical report, Rochester, NY, USA (1997)