Journal of Computer Science 10 (4): 570-577, 2014
ISSN: 1549-3636
© 2014 Science Publications
doi:10.3844/jcssp.2014.570.577 Published Online 10 (4) 2014 (http://www.thescipub.com/jcs.toc)

PARTIAL RULE MATCH FOR FILTERING RULES IN ASSOCIATIVE CLASSIFICATION

Mohamed Hayel Refai and Yuhanis Yusof

School of Computing, Universiti Utara Malaysia, Sintok, Malaysia

Received 2013-08-21; Revised 2013-09-18; Accepted 2013-12-09

ABSTRACT

In this study, we propose a new method to enhance the accuracy of Modified Multi-class Classification based on Association Rule (MMCAR) classifier. We introduce a Partial Rule Match Filtering (PRMF) method that allows a minimal match of the items in the rule's body in order for the rule to be added into a classifier. Experiments on Reuters-21578 data sets are performed in order to evaluate the effectiveness of PRMF in MMCAR. Results show that the MMCAR classifier performs better as compared to the chosen competitors.

Keywords: Modified Multi-Class Classification Based on Association Rule, Associative Classification, Text Mining

1. INTRODUCTION

Associative Classification (AC) is a learning mechanism that mainly integrates association rule discovery and classification in data mining (Han, 2003). In classification task in data mining, given a set of labeled data, the objective is to build a classification model consisting of a set of rules in order to use it for predicting test cases. Hence, it is a typical supervised learning approach in which the target class is previously defined in the training data set and the aim is to guess the class for unseen data set. On the other hand, association rule discovers hidden relationships among products or items in a relational database such as sales promotion, shelving and planning (Niu *et al*., 2009). Many recent studies (Liu *et al*., 1999; Han, 2003; Thabtah *et al*., 2005; Li *et al*., 2008) give indications that AC is able to derive higher accurate models than classic classification approaches like KNN (Tan, 2005), rule induction (William, 1995) and decision trees (Quinlan, 1993). One reason of getting higher prediction accuracy by AC algorithms is the additional knowledge they discovered through association rule mining. Such an approach reveals possible joints between the items and the class values in the training data. Nevertheless, the massive number of rules generated by AC algorithms may be problematic in some cases as we will see later in this article.

There are many AC algorithms disseminated in literature in the last decade, such as CBA (Liu *et al*., 1999), CMAR (Li *et al*., 2001), CPAR (Han, 2003), MCAR (Thabtah *et al*., 2005), CACA (Tang and Liao, 2007), ACCF (Li *et al*., 2008), BCAR (Yoon and Lee, 2008), MAC (Abdelhamid *et al*., 2012) and CBAR (Han, 2003) PCBA (Chen *et al*., 2012). These techniques use different methods to discover the rules, sort the rules, store rules, filter out redundant rules and assigns the right class to test cases. AC algorithm such as CACA (Tang and Liao, 2007) or MCAR (Thabtah *et al*., 2005) discover rules according to two main parameters inherited from association rule named minimum support (*minsupp*) and minimum confidence (*minconf*). Meaning, when a ruleitem (item(s) along with its class) within the training dataset has the frequency larger than a predefined *minsupp* and *minconf* values, it becomes a rule. The algorithm then chooses one subset of the discovered rules to build a classifier that is able to predict the classes of new dataset (i.e., test data). One of the vital problems associated with AC approach is the very large number of rules inside its classifiers. This numerous number of rules exist as AC algorithms employ an association rule mining approach like Apriori

Corresponding Author: Mohamed Hayel Refai, School of Computing, Universiti Utara Malaysia, Sintok, Malaysia or FP-Growth in discovering the rules from the input data set. For example, CBA, CACA, L3G and LC are known AC algorithms that employ Apriori-like learning methodologies during the training phase in which each ruleitem is tested in an iterative manner. In other words, the AC algorithm is going to compute the support of each ruleitem starting from those of size one and ending with those of size N. The outcome will be very large frequent ruleitems where many of them are converted into rules if they exceed the *minconf* value. This large set of rules may contain rules that are redundant, contradictory and/or overlap in the training examples, hence creating the need for filtering methods that treat these shortcomings to end up with a reasonable size classifier.

In this study, we propose a Partial Rule Match Filtering method (PRMF) to be employed in MMCAR (Yusof and Refai, 2013). The PRMF considers different scenarios when evaluating rules on the training data set during the process of constructing the classifier. This study is structured as follows: AC problem and its solution scheme are given in Section 2. Current prediction and rule filtering methods used in AC are reviewed in Section 3. Section 4 is devoted to present details on Partial Rule Match Filtering Method and the comparison results between PRMF and other classification methods are discussed in Section 5. Finally, the conclusion is given in Section 6.

2. LITERATURE REVIEW

2.1. The AC Problem and Related Definitions

According to (Thabtah *et al*., 2005), AC is a particular case of the association rule problem in which only the class attribute is considered in the rule's right hand side so for a rule like A→C, C is the class. Hereunder is the main related to definition to AC (Thabtah, 2007), where the input training data T has n attributes A_1, A_2,…, A_m and C is a set of classes. The size of T is denoted |T|.

Definition 1:

A training example in T can is a joint of attributes A_i and values a_{ij}, plus a class c_j.

Definition 2:

An item is as a term name A_i and a value a_i, denoted <(A_i, a_i)>.

Definition 3:

An itemset is as a set of disjoint items contained in a training example, denoted < (A_{i1}, a_{i1}), …, (A_{ik}, a_{ik})>.

Definition 4:

A *ruleitem r* is of the form <*itemset*, *c*>, where cεC is the class.

Definition 5:

The occurrence (*occr*) of a *ruleitem r* in T is the number of example in T that match the *items* of r.

Definition 6:

The support count (*suppcount*) of *ruleitem r* is the number of example in T that match *r's itemset* and belong to the class *c* of *r*.

Definition 7:

The occurrence of an *itemset i* (*occatt*) in T is the number of rows in T that match i.

Definition 8:

A *ruleitem r* passes the *minsupp* threshold if $(suppcount(r)/|T|) \geq minsupp$.

Definition 9:

A *ruleitem r* passes the *minconf* threshold if $(suppcount(r)/occr(r)) \geq minconf$.

Definition 10:

Any *ruleitem r* that passes the *minsupp* threshold is said to be a *frequent ruleitem*.

Definition 11:

A rule is in the form: $(A_{i1},a_{i1}) \wedge \ldots \wedge (A_{i1},a_{i1}) \rightarrow c$, where the left had side is an *itemset* and the right hand side is a class.

A classifier is a mapping form H: A→Y, where A is the set of *ruleitems* and Y is the set of class labels. The main task of AC is to construct a set of rules (model) that is able to predict the classes of previously unseen data, known as the test data set, as accurately as possible. In other words, the goal is to find a classifier hεH that maximizes the probability that h (a) = y for each test case.

Generally, the AC works as follow. First, all frequent ruleitems (1,2,…, N) that have frequencies in the training data set above the *minsupp* parameter are extracted. So an AC algorithm firstly finds frequent 1-ruleitems (ruleitems having a single item) in the first pass and then in each next pass, they start with ruleitems found to be frequent in the previous pass in order to generate possible frequent ruleitems involving more items. This means, frequent 1-ruleitems is the input for the process of discovering frequent 2-ruleitems and frequent 2-ruleitems is the input for discovering frequent 3-ruleitems and so forth.

When the complete set of frequent ruleitems are found, the AC algorithm checks their confidences and transform any of which pass the *minconf* parameter. Meaning a ruleitem that have confidence bigger than or equal the *minconf* is produced as a candidate rule. However, the number of rules extracted could be large and many of which are redundant. Therefore, rule filtering processes discard these redundant rules and a considerable number of rules might be removed at phase of building the classification model. Finally, rules that cover training cases and pass the filtering phase becomes part of the model that is laterally employed to assign class values to test data.

2.2. Rule Filtering Methods in Associative Classification

A number of rule filtering methods have been used in AC mining to reduce the number of rules in the produced classifiers. Some of these methods were taken from decision trees like Information Gain (Quinlan, 1993) and others from statistics such as weighted Chi-Square (Li *et al*., 2001). These methods are used during the construction of the model and sometimes in early phases especially during rule discovery. Throughout this section, we discuss filtering methods that are developed within AC mining and not adopted from statistics and machine learning. This is simply because we focus on removing rules during the step of building the model and not during frequent ruleitems discovery step.

2.2.1. Database Coverage Filtering

The database coverage is a filtering method used in CBA (Liu *et al*., 1999) and operates once the candidate rules have been created and sorted. This is when all frequent ruleitems that exceed the *minconf* value is transformed into rules and got sorted based on parameters such as confidence, support and rule length. The discovered rules are then evaluated on a training data set to check their effectiveness in covering the dataset. The database coverage evaluates rules in a top down manner starting from the first ordered rule and checks if the particular rule is able to cover at least one training example in the training data set. If so, the selected rule is inserted into the model and all examples covered by the rule are deleted from the training data set. The same process is repeated on the remaining candidate rules until all of the candidate rules are tested or no more uncovered example are left in the training data set. The database coverage method was used first by CBA (Liu *et al*., 1999) and then latterly by other AC algorithms, including CMAR (Li *et al*., 2001), ARC-BC (Antonie and Zaiane, 2002), CAN (Kundu *et al*., 2008), Multi-label Classification based on Association Rules and ACCF (Li *et al*., 2008).

2.2.2. Specific Rules Filtering

A rule filtering method that removes large and rules with less confidence value than general rules was developed in (Li *et al*., 2001) and laterally used in (Antonie and Zaiane, 2002). Once the rule generation process is finished and rules are sorted, the proposed rule filtering works as follows: For the candidate rules such as $I' \rightarrow c$ from the set of produced rules, if there is some general rule $I \rightarrow c$ of a higher rank and $I \subseteq I'$ then $I' \rightarrow c$ is removed. The specific rule deletion took place immediately after a rule is inserted into the compact data structure, the CR-tree within the CMAR (Li *et al*., 2001). When a rule is added to the CR-tree, a query is issued to check if the inserted rule can be filtered or some other existing rules in the tree can be removed.

2.2.3. General Rules Filtering

There are certain AC algorithms called lazy such as in (Baralis *et al*., 2004; 2008) that prefer keeping massive rules for the sake of improving the classification accuracy of the classifiers. Such work believe that filtering should happen only on rules that lead to wrong classification during building the model. The main difference between lazy and database coverage is that rules which don’t cover a training example are stored by the lazy AC algorithms in the memory whereas algorithms that employ database coverage completely remove such rules.

2.2.4. No Class Identically Filtering Method

Recently, an AC algorithm called MAC (Abdelhamid *et al*., 2012) proposed a new rule filtering method based database coverage. This method works as follows: For each sorted candidate rule and during building the classifier, starting with the first training example, look for the highest sorted rule that may cover it and without checking the class similarity of the candidate rule and the training example. If there are candidate rules that may cover the training example, mark the candidate rule and remove the training example. Repeat the process on the remaining examples in the training data set until it becomes empty. At that point, insert all marked candidate rules into the classification system and delete any unmarked rules. This method surely increases the rule coverage on training examples and reduces the size of the model since no class similarity is required as the database coverage and other filtering methods.

3. PARTIAL RULE FILTERING METHOD

Most of the current rule filtering methods in AC mining are based on database coverage in which they consider a candidate rule is part of the classifier when this rule correctly covers at least one training example during the classification development. Hence, there exist two conditions that must be met before a rule can be inserted into the classifier:

- The candidate rule items (left hand side) must be within the items of training examples
- The class of the candidate rule (right hand side) must be similar to the training example class

Based on such approach, we argue on two main issues

- Situations when there isn't any candidate rule(s) that covers the training case (no identical similarity). Currently, existing methods use the remaining unclassified training example to be converted as a default rule. Such an approach may raise higher error rate
- The condition of having similar classes is unnecessary and can cause overlearning the training data by keeping greedily rules that maximizes the accuracy rate only on the training data without taken into account that the rules are not yet generalized for testing on unseen data which is the main goal of classification in data mining. We believe that by relaxing this constraint and merge it with the partly matching we can end up with a much smaller size of classification model. This can be achieved by allowing the candidate rule to cover more training examples and therefore many redundant lower sorted rules will be unmarked and thus deleted after the building the model step is finished

Hereunder we present a rule filtering methods proposed by us in AC context. We assume that all candidate rules are extracted and sorted from highest to lowest using confidence, support and rule length criteria.

For each training data, PRMF finds the first rule that satisfies the training example by having all of the rule's items inside the training example. When the rule is found, the algorithm marks it and deletes the training example. However, when there isn't any rule that fully match the training example (don't have a body that could be inside the training example) PRMF takes on the first rule that partly covers the training example rather than leaving this example to be covered later by the default class rule. By doing this, PRMF rule filtering method minimizes the number of training examples that will be used to make the default rule. The main difference between the PRMF and of database like is that the proposed PRMF method includes not only full covered rules but also the partly covered rules into the model. In addition, existing filtering method consider class similarity between the training example and the candidate rule as an important condition to cover the training example.

Whereas the PRMF ignores the class similarity as it aims to reduce overlearning. In the remainder of this section we distinguish between our rule filtering method and those of MCAR and CBA using example. Consider **Table 1** that shows candidate rules and **Table 2** that lists six (6) training examples. Please note that the last two columns of **Table 2** denotes the rule that have been used by our method and those of (MCAR, CBA) respectively. For the first training data (1), the proposed classifier and those of MCAR and CBA used rule number (RuleID 2). This is because there is a fully match between RuleID 2 and training data (#1). The same thing occurs for training data (#2) in which RuleID 1 has been used to cover the data. Though, for the third training data, the CBA and MCAR methods leave it for the default rule because there is no candidate rule to cover it. This is because none of the existing candidate rules matches with training data (#3). On the other hand, our classifier that employs PRMF uses the first partly match rule, which is RuleID 2 to cover training data (#3). The same scenario happens again for training data (#4) in which our classifier uses the partly match rule (#6) whereas other AC algorithms employs default rule. The above example shows the demonstration of the proposed rule filtering method that indeed reduces error by allowing partly matching rule to be part of the classifier instead on taking the default rule. All rules that have been applied during the classifier builder are inserted into the classifier whereas the remaining rules get deleted since they have no training data coverage. In summary, the proposed PRMF is as shown in **Fig. 1**. The input of the PRMF method is the training data (TranD) and discovered Rules Rank is (RuleR) and the output is classifier (C).

Table 1. Candidate rules

RuleID	Rule detail	Rule support	Rule confidence	Rule rank
1	Overcast→yes	0.285715	1.0000	1
2	high^sunny→no	0.214286	1.0000	2
3	normal^sunny→yes	0.142858	1.0000	3
4	hot^sunny→no	0.142858	1.0000	4
5	false^hot→yes	0.142858	0.6667	5
6	Cool^normal→yes	0.214286	0.7500	8

Table 2. Examples of training data

	Outlook	Temperature	Humidity	Windy	Actual Class	Rule applied using our method	Rule applied using MCAR and CBA methods
1	sunny	hot	high	true	no	2	2
2	overcast	hot	high	false	yes	1	1
3	rainy	mild	high	false	yes	2	Default rule
4	rainy	cool	high	true	no	6	Default rule
5	sunny	mild	normal	true	yes	3	3
6	sunny	mild	high	false	no	2	2

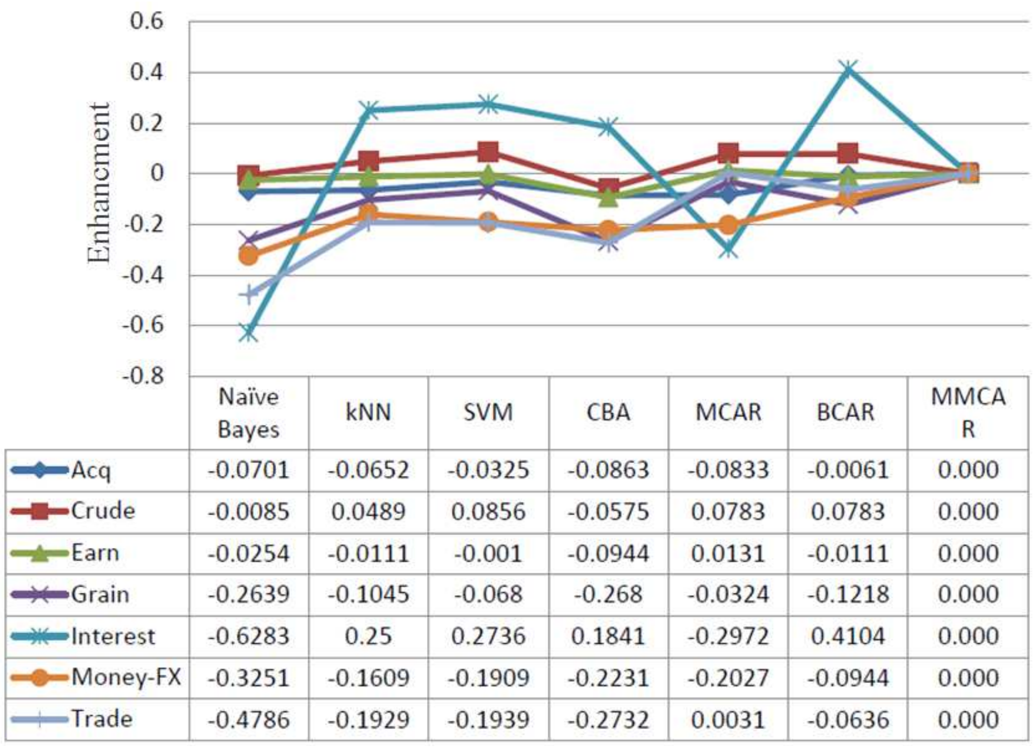
```
$Rule_R' = \text{rank}(Rule_R)$;
$\forall$ $Tran_D$ in $Rule_R'$

find the first $Rule_R'$ in $R_n$ that can cover the current training example ($Ti$)

        if $R_n$'s body inside $Ti$
        input the rule into $C$
        else discard $Ti$ from $Tran_D$.

        if $Rule_R'$ in $R_n$ that can partly cover the current training example ($Ti$)
        input the rule into $C$
        else discard $Ti$ from $Tran_D$.
else
delete $R_n$
end
```

Fig. 1. PRMF method

4. EXPERIMENT AND RESULTS

In this section, different traditional classification algorithms as well as rule-based classification algorithms are compared with MMCAR based on the prediction accuracy. The data used in the experiments is the Reuters-21578 (Lewis). The Reuters-21578 is the most commonly used data set in the text categorisation research. We used the ModApte version of Reuters-21578 that leads to a corpus of 9,174 documents consisting of 6,603 training and 2,571 of testing documents. The algorithms used in the comparison are CBA (Liu *et al*., 1999), BCAR (Baralis *et al*., 2004) and MCAR (Thabtah *et al*., 2005) from the associative classification approaches while Naïve Bayes (Lewis, 1998), K-NN (Tan, 2005) and SVM (Japkowicz and Stephen, 2002) represent the traditional approaches. We tested the proposed algorithm using the minsupp and minconf values of 2 and 40%, respectively. **Table 3** shows the number of documents in training and testing sets per category (REUTERS-21578).

Table 4 depicts a comparison results between the proposed algorithm against other well-known classifiers. It should be noted that the results of the BCAR algorithm is reported in (Yoon and Lee, 2008) while for MCAR the results were obtained via experiment.



	Naïve Bayes	kNN	SVM	CBA	MCAR	BCAR	MMCAR
Acq	-0.0701	-0.0652	-0.0325	-0.0863	-0.0833	-0.0061	0.000
Crude	-0.0085	0.0489	0.0856	-0.0575	0.0783	0.0783	0.000
Earn	-0.0254	-0.0111	-0.001	-0.0944	0.0131	-0.0111	0.000
Grain	-0.2639	-0.1045	-0.068	-0.268	-0.0324	-0.1218	0.000
Interest	-0.6283	0.25	0.2736	0.1841	-0.2972	0.4104	0.000
Money-FX	-0.3251	-0.1609	-0.1909	-0.2231	-0.2027	-0.0944	0.000
Trade	-0.4786	-0.1929	-0.1939	-0.2732	0.0031	-0.0636	0.000

Fig. 2. Results on relative BEF

Table 3. Number of documents (REUTERS-21578)

Category	Training	Testing
Acq	1650	719
Crude	389	189
Earn	2877	1078
Grain	433	149
Interest	347	130
Money-FX	538	197
Trade	396	117

Table 4. Results on precision/recall-BEP

Category/Algorithm	Naïve bayes	*k*NN	SVM	CBA	MCAR	BCAR	MMCAR
Acq	91.50	92.00	95.20	89.9	90.20	97.80	98.40
Crude	81.00	85.70	88.70	77.0	88.10	88.10	81.70
Earn	95.90	97.30	98.40	89.2	99.80	97.40	98.40
Grain	72.50	88.20	91.80	72.1	95.30	86.50	98.50
Interest	58.00	74.00	75.40	70.1	41.60	83.50	59.20
Money-FX	62.90	78.20	75.40	72.4	74.30	84.40	93.20
Trade	50.00	77.40	77.30	69.7	96.20	89.80	95.90
AVG	73.11	84.69	86.03	77.2	83.64	89.64	89.33

Table 5. Results on win/lose/tie records

	Naïve Bayes	KNN	SVM	CBA	MCAR	BCAR
MMCAR	7-0-0	5-0-2	4-1-2	6-0-1	4-0-3	5-0-2

Table 4 reveals that the proposed method has the highest accuracy for three of the dataset-Acq, Grain and Money-FX. With an average of 89.33%, it can be considered that the MMCAR employing PRMF shown an acceptable result as compared to other algorithms. This is due to the fact that the MMCAR obtains more number of winning as compared to other competitors. The results of a win-tid-lose record is depicted in **Table 5**. The three values (Win/lose/tie) record are respectively the number of datasets for which a method obtains higher, lower or equal classification accuracy, compared with an alternative method. For example, it is learned that MMCAR overcomes (win) the BCAR for 5 dataset and lost two, hence generating a values of 5-0-2.

Figure 2 shows the "relative BEF rate" that denotes the variation in the accuracy rates of the proposed algorithm with reference to those resulting by the chosen competitors. In other words, it indicates, how good or bad MMCAR performs with reference to the competitors on the utilized datasets. The relative accuracy rate is obtained using the following relations Equation (1):

$$\text{RR} = \frac{\left(\text{Accuracy}_{\text{others methods}}\right) - \left(\text{Accuracy}_{\text{MMCAR}}\right)}{\left(\text{Accuracy}_{\text{MMCAR}}\right)} \qquad (1)$$

where, $\text{Accuracy}_{\text{MMCAR}}$ is the accuracy rate of the proposed method and the $\text{Accuracy}_{\text{Other methods}}$ is the accuracy of other algorithms such as KNN, Naïve Bayes, SVM, MCAR, CBA and BCAR.

5. CONCLUSION

In this article, the problem of rule filtering in associative classification data mining has been investigated. The outcome is a Partial Rule Filtering Method which is employed in MMCAR Experimental results on Reuters-21578 indicated that our proposed method is highly competitive when compared with traditional classification algorithms such as SVM, KNN and Bayes in terms of prediction accuracy, win/lose/tie and relative BEF. Furthermore, our method scales well if compared with popular AC approaches like CBA, MCAR and BCAR with regards to breakeven point.

6. REFERENCES

Abdelhamid, N., A. Ayesh, F. Thabtah, S. Ahmadi and W. Hadi, 2012. MAC: A multiclass associative classification algorithm. J. Inform. Knowl. Manage., 11: 1-10. DOI: 10.1142/S0219649212500116

Antonie, M.L. and O.R. Zaiane, 2002. Text document categorization by term association. Proceedings of the IEEE International Conference on Data Mining, Dec. 9-12, IEEE Xplore Press, pp: 19-26. DOI: 10.1109/ICDM.2002.1183881

Baralis, E., S. Chiusano and P. Garza, 2004. On support thresholds in associative classification. Proceedings of the 2004 ACM Symposium on Applied Computing, Mar. 14-17, ACM Press, New York, USA., pp: 553-558. DOI: 10.1145/967900.968016

Baralis, E., S. Chiusano and P. Garza, 2008. A lazy approach to associative classification. IEEE Trans. Knowl. Data Eng., 20: 156-171. DOI: 10.1109/TKDE.2007.190677

Chen, W.C., C.C. Hsu and Y.C. Chu, 2012. Increasing the effectiveness of associative classification in terms of class imbalance by using a novel pruning algorithm. Expert Syst. Applic., 39: 12841-12850. DOI: 10.1016/j.eswa.2012.05.009

Han, J., 2003. CPAR: Classification based on predictive association rules. Proceedings of the 3rd SIAM International Conference on Data Mining, (CDM' 03), pp: 331-335.

Japkowicz, N. and S. Stephen, 2002. The class imbalance problem: A systematic study. Intell. Data Anal., 6: 429-449.

Kundu, G., M.M. Islam, S. Munir and M.F. Bari, 2008. ACN: An associative classifier with negative rules. Proceedings of the 11th IEEE International Conference on Computational Science and Engineering, Jul. 16-18, IEEE Xplore Press, Sao Paulo, pp: 369-375. DOI: 10.1109/CSE.2008.48

Lewis, D.D., 1998. Naive (Bayes) at forty: The independence assumption in information retrieval. Proceedings of the 10th European Conference on Machine Learning, Apr. 21-23, Springer Berlin Heidelberg, Germany, pp: 4-15. DOI: 10.1007/BFb0026666

Li, W., J. Han and J. Pei, 2001. CMAR: Accurate and efficient classification based on multiple class-association rules. Proceedings IEEE International Conference on Data Mining, Nov. 29-Dec. 02, IEEE Xplore Press, San Jose, CA., pp: 369-376. DOI: 10.1109/ICDM.2001.989541

Li, X., D. Qin and C. Yu, 2008. ACCF: Associative classification based on closed frequent itemsets. Proceedings of the 5th International Conference on Fuzzy Systems and Knowledge Discovery, Oct. 18-20, IEEE Xplore Press, Shandong, pp: 380-384. DOI: 10.1109/FSKD.2008.396

Liu, B., Y. Ma and W. Hsu, 1999. Integrating Classification and Association Rule Mining. In: New Directions in Rough Sets, Data Mining and Granular-Soft Computing, Zhong, N., A. Skowron and S. Ohsuga (Eds.), ISBN-10: 3540666451, pp: 443-447.

Niu, Q., S.X. Xia and L. Zhang, 2009. Association classification based on compactness of rules. Proceedings of the 2nd International Workshop on Knowledge Discovery and Data Mining, Jan. 23-25, IEEE Xplore Press, Moscow, pp: 245-247. DOI: 10.1109/WKDD.2009.160

Quinlan, J.R., 1993. C4.5: Programs for Machine Learning. 5th Edn., Morgan Kaufman Publ. Incorporated, San Mateo, ISBN-10: 1558602380, pp: 302.

Tan, S., 2005. Neighbor-weighted k-nearest neighbor for unbalanced text corpus. Expert Syst. Applic., 28: 667-671. DOI: 10.1016/j.eswa.2004.12.023

Tang, Z. and Q. Liao, 2007. A new class based associative classification algorithm. Int. J. Applied Mathemat., 36: 685-689.

Thabtah, F., 2007. A review of associative classification mining. Knowl. Eng. Rev., 22: 37-65. DOI: 10.1017/S0269888907001026

Thabtah, F., P. Cowling and Y. Peng, 2005. MCAR: Multi-class classification based on association rule. Proceedings of the 3rd ACS/IEEE International Conference on Computer Systems and Applications, Jan. 3-6, IEEE Xplore Press. DOI: 10.1109/AICCSA.2005.1387030

William, W.C., 1995. Fast effective rule induction. Proceedings of the 12th International Conference on Machine Learning, (CML' 95), pp: 115-123.

Yoon, Y. and G.G. Lee, 2008. Text categorization based on boosting association rules. Proceedings of the IEEE International Conference on Semantic Computing, Aug. 4-7, IEEE Xplore Press, Santa Clara, CA., pp: 136-143. DOI: 10.1109/ICSC.2008.70

Yusof, Y. and M.H. Refai, 2013. Modified multi-class classification using association rule mining. Pertanika J. Sci. Technol., 21: 205-216.