ACN: An Associative Classifier with Negative Rules

Gourab Kundu, Md. Monirul Islam, Sirajum Munir, Md. Faizul Bari
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka-1000
{gourabkundu, mdmonirulislam}@cse.buet.ac.bd, {sirajum.munir,faizulbari}@gmail.com

Abstract

Classification using association rules has added a new dimension to the ongoing research for accurate classifiers. Over the years, a number of associative classifiers based on positive rules have been proposed in literature. The target of this paper is to improve classification accuracy by using both negative and positive class association rules without sacrificing performance. The generation of negative associations from datasets has been attacked from different perspectives by various authors and this has proved to be a very computationally expensive task. This paper approaches the problem of generating negative rules from a classification perspective, how to generate a sufficient number of high quality negative rules efficiently so that classification accuracy is enhanced. We adopt a simple variant of Apriori algorithm for this and show that our proposed classifier "Associative Classifier with negative rules"(ACN) is not only time-efficient but also achieves significantly better accuracy than four other state-of-the-art classification methods by experimenting on benchmark UCI datasets.

Keywords: association rule, classification, data mining, negative rule

1. Introduction

Classification is a very important problem that has been studied for years now. The goal of the classification algorithms is to construct a model from a set of training data whose target class labels are known and then this model is used to classify unseen instances. Many different types of classification techniques have been proposed in literature that includes decision trees [13], naive-Bayesian methods [6], statistical approaches[11] etc.

Recently a new classification technique has come into existence. In this technique, at first classification rules are mined from training data using an association rule mining algorithm and then a subset of these rules are used to build a classifier. This technique which uses association rules for classification is called "Associative Classification(AC)". Experiments have shown that these classifiers are significantly more accurate than decision tree classifiers. A number of associative classification algorithms have been proposed in literature but most of them use only positive association rules and differ mainly in rule mining and classifier construction from the set of mined rules.

One of the first algorithms to use association rules for classification was CBA(Classification based on Association)[12]. CBA uses the Apriori algorithm[1] in order to discover all frequent ruleitems. Then it converts any frequent ruleitem that passes the minimum confidence threshold into a rule. After that it sorts the rules based on a rule ranking criteria and selects a subset of the rules that are needed to cover the dataset. These ordered rules are later used for classification.

An AC approach that uses multiple rules for making a single prediction is CMAR (Classification based on Multiple Association Rules)[10] . Instead of using Apriori, this method adapts the FP-tree algorithm[7] to mine the class association rules and makes use of a CR-tree structure to store and retrieve the mined rules. But the major distinguishing feature from CBA is that here the classification is performed based on a weighted χ^2 analysis using multiple association rules. Some other associative classifiers based on positive association rules are discussed in [9, 22, 19, 4, 17]. A survey on these works can be found in [16].

A positive association rule is of the form X $\rightarrow$ Y where X , Y both are a set of items and X $\bigcap$ Y is ϕ. A negative association rule is of the form X $\rightarrow$ Y where in addition to being a set of items, X or Y will contain at least one negated item. As discussed before,the idea behind most of the existing approaches has been the mining of positive class association rules from the training set and then selecting a subset of the mined rules for future predictions. However, in most cases, it is found that the final classifier contains some weak and inaccurate rules that were selected for covering some training instances for which no better rules were available. These rules make poor predictions of unseen test instances and only for these rules, the overall classification accuracy is drastically reduced. The idea of this paper is to eliminate these weak and inaccurate positive rules as far as possible by accurate negative rules.

An interesting approach that uses both positive and negative rules for classification is ARC-PAN[2]. It examines the correlation of each frequent itemset with the class label. If the correlation is positive, a positive rule is discovered. If the correlation is negative, two negative rules are discovered. The negative rules produced are of the form $X \rightarrow \neg Y$ or $\neg X \rightarrow Y$ which the authors term as "confined negative association rules". Here the entire antecedent or consequent is either a conjunction of negated attributes or a conjunction of non-negated attributes. The problem with this approach is that it results in a very small set of rules which may not be adequate to provide accurate classification for all training and test instances.

In this paper, we introduce a new method for associative classification named "Associative Classifier with Negative Rules"(ACN) that extends the Apriori algorithm to mine a relatively large set of negative association rules and then uses both positive and negative rules to build a classifier. The set of mined negative rules and the way ACN generates them are totally different from ARC-PAN. The benefit of our approach is that the number of negative rules generated is much larger and so in general, a lot of good (high confidence and high support) negative rules are found that can be used in place of some weak positive rules. As a result, the number of inaccurate positive rules in the final classifier is greatly reduced and classification accuracy is increased. Moreover, we only consider those negative rules that arise naturally during the Apriori rule mining process so that no extra overhead is needed. The major bottleneck in associative classification is the computational cost for the discovery of association rules and if a classifier uses negative rules together with positive rules, this cost can increase further. ACN tries to address this issue by mining a relatively large set of negative rules but with as low overhead as possible. So it adopts a variant of Apriori algorithm and generates a set of negative rules from the available frequent positive itemsets. These negative rules come almost free-of-cost since their support and confidence can be readily calculated from available positive rules and no extra database scan is required.

The rest of the paper is organized as follows. In Section 2, we discuss the basic concepts of negative association mining. Section 3 presents ACN in details. Section 4 presents our experimental results and finally in section 5, we conclude the paper with some remarks.

2. Negative Association Mining

Efficient mining of negative association rules has recently received much attention from a large group of researchers. Negative associations can inform us of facts like "If a customer purchases Coke, he is not likely to purchase Juice." etc. Such associations are quite natural and occur frequently in practice. However, mining transactional databases for negative associations is really challenging because typically a super-store contains thousands of items and each transaction contains only a few of them. As a result, each transaction has a large number of negated or absent items and the number of possible negative associations under the support-confidence framework turns out to be overwhelmingly large. Moreover, a large portion of the discovered rules may be uninteresting from the user's perspective. These difficulties have forced researchers to incorporate additional rule interestingness measures[15, 20] over the support-confidence framework or incorporate domain knowledge[14, 23] to reduce the search space.

However, it is evident that important differences exist between the mining of negative associations in transactional databases and the mining of them in classification datasets. Firstly, in classification datasets, typically the number of attributes is not large and each attribute has a small number of possible values. So the value for an attribute in a record indicates the absence of other possible values for that attribute. This is much smaller compared to the number of absent items in a transaction in a super-store database. Secondly, the purpose of mining is on increasing classification accuracy rather than presenting the set of mined rules to the user and satisfying his interest. Thirdly, the process of negative rule generation must be as cheap as possible to avoid further increasing the complexity of the mining phase of an associative classifier.

Keeping the above challenges in mind, we designed an efficient algorithm for generation of both positive and negative rules. Using both sets of rules, we were able to obtain classification accuracy higher than other methods. A number of negative rule mining algorithms have already been proposed in the literature[23, 20, 3, 21, 14, 18, 15]. However, to the best of our knowledge, the algorithm for negative rule mining in ACN best meets the challenges described above. This approach is novel and has not been introduced in literature before.

3 ACN

This section describes ACN in more details. First, we discuss how ACN generates rules, then we present the classifier builder for ACN and finally we discuss different pruning strategies to reduce the number of generated rules.

3.1 ACN Rule Generator

Let a relational schema contains n data attributes (A_1,A_2,..., A_n) and one class attribute Z. Each attribute has a set of possible values. Each record is of the form (a_1,a_2,...,a_n, z) where a_1,a_2,...,a_n are the values of the corresponding data attributes and z is the class label. Given a set of training instances, the task of a classifier is to learn the relation between set of attribute values to class label and later use this relation to predict class labels for test instances from the values of their data attributes only.

For mining class association rules using Apriori, each itemset is considered to be of the form (conditionset, z) which represents a rule: conditionset $\rightarrow$z where conditionset is a set of (attribute, value) pairs and z is a class label. The terms rule and itemset will be used interchangeably afterwards and should be understood from context. The rule has confidence equal to (ruleSupportCount / conditionSupportCount) * 100% where, conditionSupportCount is the number of cases in the dataset D that contain conditionset and ruleSupportCount is the number of cases in the dataset D that contain conditionSet and are labeled with class z. The rule has support equal to (ruleSupportCount/|D|) *100%, where |D| is the size of the dataset.

The idea behind the Apriori algorithm is the property that all subsets of any frequent itemset must also be frequent . This allows Apriori to generate the frequent itemsets in a level wise manner. In the following discussion, we assume that the notation C_k represents the candidate itemsets of length k and L_k represents the frequent itemsets of length k generated by Apriori. We also assume that the notation l_i[j] refers to the j'th item in l_i where l_i represents an itemset(l_i ϵ L_k and $1 \leq j \leq k$). Obviously for all k, L_k is a subset of C_k and each member of L_k has support higher than the user specified support threshold. In Apriori, there are two steps: the **join step** and the **prune step**. In the join step; a set of candidate itemsets C_k is generated by joining L_{k-1} with itself. If we assume that the items in the itemset are sorted in lexicographic order, then members $l_1 \rightarrow$ z and $l_2 \rightarrow$z of L_{k-1} are joinable if (l_1[1]=l_2[1]) $\wedge$ (l_1[2]=l_2[2]) $\wedge$ (l_1[3]=l_2[3]) $\wedge$ ...(l_1[k-2]=l_2[k-2])$\wedge$(l_1[k-1] <l_2[k-1]). The resulting rule generated by joining l_1 and l_2 is l=l_1[1] $\wedge$ l_1[2] $\wedge$ ... $\wedge$ l_1[k-1] $\wedge$ l_2[k-1] $\rightarrow$ z. Now comes the prune step of Apriori. It checks to see whether the all subsets of the generated candidate itemset are frequent. If not, that itemset cannot be frequent and can immediately be discarded. Otherwise, a full database scan must be made to count the support for the new itemset to decide whether it is actually frequent or not.

We call each candidate itemset for which all subset of itemsets are frequent, a "legal candidate". For each literal of this legal candidate, ACN replaces this literal with the corresponding negated literal, creates a new negative rule and adds it to the negative ruleSet. The generation of positive rules continues without disruption and the abundant but valuable negative rules are produced as by-products of the Apriori process.

For the trivial cases, the set of frequent 1 positive rule items are generated without any join step. Frequent 1 negative rules are generated by simply considering the negation of the single antecedent of each positive rule item.

Example :
We will explain the rule generation of ACN using an example. Let us consider a case with 5 data attributes A,B,C,D,E and one class attribute Z. The domains for the four data attributes are respectively {a1,a2,a3},{b1,b2,b3,b4},{c1,c2},{d1,d2,d3},{e1,e2,e3,e4}. Z can take on the possible values of y and n.

Suppose, in the Apriori rule mining process, the set of frequent ruleitems of length 3 are found to be a1 $\wedge$ b1 $\wedge$ c1 $\rightarrow$ y, a1 $\wedge$ b1 $\wedge$ d1$\rightarrow$y, b1 $\wedge$ c1 $\wedge$ d1$\rightarrow$y, a2 $\wedge$ b2 $\wedge$ c2$\rightarrow$n, a2 $\wedge$ b2 $\wedge$ d4$\rightarrow$n, b1 $\wedge$ c1 $\wedge$ e3 $\rightarrow$ n.

Now the set of candidate ruleitems of length 4 after join step is a1 $\wedge$ b1 $\wedge$ c1 $\wedge$ d1 $\rightarrow$ y, a2 $\wedge$ b2 $\wedge$ c2 $\wedge$ d4 $\rightarrow$ n) Out of these two candidates, only the first one (a1 $\wedge$ b1 $\wedge$ c1 $\wedge$ d1 $\rightarrow$ y) possibly can be frequent since all its subsets are frequent as found from the set of frequent ruleitems of length 3. So according to our definition, it will be a legal candidate and from this ruleitem, four rules of the form $\neg$ a1 $\wedge$ b1 $\wedge$ c1 $\wedge$d1 $\rightarrow$ y, a1 $\wedge$ $\neg$ b1 $\wedge$c1 $\wedge$d1 $\rightarrow$ y, a1 $\wedge$b1 $\wedge$ $\neg$ c1$\wedge$d1 $\rightarrow$ y and a1$\wedge$b1$\wedge$c1$\wedge$ $\neg$ d1 $\rightarrow$ y will be generated. In this way, negative rules will be generated in all phases of the Apriori algorithm. These negative rules will not take part in generation of any new rule but they will compete for a place in the final classifier with the positive rules. Please note that each legal candidate ruleitem with n number of literals in the antecedent will generate n new negative rules, even if the candidate ruleitem turns out to be infrequent.

The algorithm for ACN rule generator is presented in Figure1.positive and negative rules of length 1 are mined in lines 1 and 2 respectively . Lines 3-15 are similar to Apriori rule generation process except in lines 5-10, where in each phase of Apriori, negative rules are produced from legal candidates and in line 12 where supports for these rules are calculated from supports of legal candidates of that phase and supports of frequent positive rules of previous phase. In line 14, negative rules are pruned based on a support threshold. Finally, in line 16, all positive and negative rules are taken together to form the class association ruleset for ACN.

```
1   $L_1$=frequent-1-Positive-itemsets(D)
2   $N_1$=frequent-1-Negative-itemsets(D)
3   for(k=2;$L_{k-1}$!=empty;k++)
4       $PC_k$= legal candidates generated for
          level k.
5       for each legal candidate generated
6           for each literal on the candidate
7               create a new negative rule by
                  negating that literal.
8               add this rule to $NC_k$.
9           end
10      end
11      calculate support for each rule of
        $PC_k$ by scanning the database.
12      calculate support for each rule of
        $NC_k$ from supports of members of $PC_k$
        and $L_{k-1}$.
13      $L_k$=candidates in $PC_k$ that pass
        support threshold.
14      $N_k$=candidates in $NC_k$ that pass
        support threshold.
15  end
16  Return L=$L_1 \cup L_2 \cup \ldots \cup L_{k-2} \cup N_1 \cup N_2$
    $\cup \ldots \cup N_{k-2}$
```

Figure 1. Algorithm for ACN Rule Generator

3.2 Classifier Builder

The classifier builder for ACN is presented in Figure2. (Line 1)ACN sorts the set of positive and negative rules on the following rule ranking criteria, a rule Ri will have higher rank than rule Rj if and only if 1) conf(Ri) $>$ conf(Rj) or 2)conf(Ri) = conf(Rj) but correlation(Ri) $>$ correlation(Rj) or 3) conf(Ri) = conf(Rj) and correlation(Ri) = correlation(Rj) but sup(Ri) $>$ sup(Rj) or 4) conf(Ri) = conf(Rj) and correlation(Ri) = correlation(Rj) and sup(Ri) = sup(Rj) but size of conditionset of Ri $<$ size of conditionset of Rj or 5) conf(Ri) = conf(Rj) and correlation(Ri) = correlation(Rj) and sup(Ri) = sup(Rj) and size of conditionset of Ri = size of conditionset of Rj but Ri is a positive rule and Rj is a negative rule. After sorting, ACN builds a classifier based on database coverage similar to CBA. In lines 2-11,ACN takes each rule according to the sorted order and tests if it can provide correct classification for at least one remaining training example. If it can, ACN checks to see if it is a positive or negative rule. If it is a positive rule,it is immediately taken in the final classifier(lines 7-9). On the other hand, if it is a negative rule, ACN calculates the accuracy of the rule on the examples remaining. This rule is taken in the final classifier only if the accuracy on the remaining examples is beyond a user-defined threshold(lines 4-6). In this way, ACN proceeds until all rules have been examined or all examples have been covered. In case database is uncovered, the default rule is the majority class from uncovered examples. Otherwise it is simply the majority class from the entire training set.

```
1   Sort rules based on rule ranking criteria
2   For each rule taken in order
3       If the rule classifies at least one remaining training
          example correctly
4           If the rule is a negative rule and accuracy on
              remaining data>threshold
5               Include that rule in classifier and delete those
                  examples
6           end
7           If the rule is a positive rule
8               Include the rule in classifier and delete those
                  examples
9           end
10      end
11  end
12  If database is uncovered
13      select majority class from remaining examples
14  else select majority class from entire training set.
15  end
```

Figure 2. Algorithm for ACN Classifier Builder

3.3 Pruning Strategies

ACN adopts several pruning strategies to cut down the number of generated rules. In the first strategy, only negative rules are pruned. Consider two rules l and m from which a negative rule n is produced. Let, l = l[1] $\wedge$ l[2] $\wedge \ldots \wedge$ l[k] $\Rightarrow$ z and m = l[1] $\wedge$ l[2] $\wedge \ldots \wedge$ l[i-1] $\wedge$ l[i+1] $\wedge \ldots \wedge$ l[k] $\Rightarrow$ z and n = l[1] $\wedge$ l[2] $\wedge \ldots \wedge \neg$ l[i] $\wedge \ldots \wedge$ l[k] $\Rightarrow$ z. If confidence of l $>$ confidence of m , it can be proved that confidence of n $<$ confidence of m. So according to our rule ranking criteria, m will precede n and n can be pruned because the coverage of m is a superset of coverage of n.

Secondly, ACN prunes all rules that have confidence less than the minimum confidence.

Thirdly, ACN calculates pearson's correlation coefficient for each rule(both positive and negative) and prunes a rule if its correlation measure is below a user-specified threshold.

3.4 Time-Efficiency of ACN:

In this section, we theoretically prove that ACN does not perform any extra dataset scan to count the support or confidence of the generated negative rules. So there is no big I/O overhead for generating the negative rules.

Theorem: ACN performs no extra dataset scan than normal Apriori Process.

Proof:

The proof is by contradiction. Suppose at some pass of Apriori, ACN generates a negative rule $l_1 \wedge l_2 \wedge \ldots \wedge l_{i-1} \wedge \neg\, l_i \wedge \ldots \wedge l_p \Rightarrow z$ for which the support and confidences cannot be calculated without a database scan. This rule can only be generated from a candidate ruleitem $l_1 \wedge l_2 \wedge \ldots \wedge l_{i-1} \wedge l_i \wedge \ldots \wedge l_p \Rightarrow z$. According to the rule generation method of ACN, this ruleitem must be a legal ruleitem, i.e., all subsets of this ruleitem of length p-1 must be frequent. As a result, the ruleitem of the form $l_1 \wedge l_2 \wedge \ldots \wedge l_{i-1} \wedge l_{i+1} \ldots \wedge l_p \Rightarrow z$ must be frequent and its support and confidences must have been calculated in the previous pass of the Apriori algorithm. For the legal candidate ruleitem $l_1 \wedge l_2 \wedge \ldots \wedge l_{i-1} \wedge l_i \wedge \ldots \wedge l_p \Rightarrow$ z, support and confidence will be calculated via database scan at current pass. Now for the rule $l_1 \wedge l_2 \wedge \ldots l_{i-1} \wedge \neg\, l_i \wedge \ldots \wedge l_p \Rightarrow z$ we can obtain,

supp $(l_1 \wedge l_2 \wedge \ldots l_{i-1} \wedge \neg\, l_i \wedge \ldots \wedge l_p \Rightarrow z)$= *supp*$(l_1 \wedge l_2 \wedge \ldots \wedge l_{i-1} \wedge l_{i+1} \ldots \wedge l_p \Rightarrow z)$ - *supp*$(l_1 \wedge l_2 \wedge \ldots \wedge l_{i-1} \wedge l_i \wedge \ldots \wedge l_p \Rightarrow z)$

supp$(l_1 \wedge l_2 \wedge \ldots \wedge l_{i-1} \wedge \neg\, l_i \ldots \wedge l_p)$= *supp* $(l_1 \wedge l_2 \wedge \ldots \wedge l_{i-1} \wedge l_{i+1} \wedge \ldots \wedge l_p)$-*supp*$(l_1 \wedge l_2 \wedge \ldots \wedge l_{i-1} \wedge l_i \wedge l_{i+1} \wedge \ldots \wedge l_p)$

conf $(l_1 \wedge l_2 \wedge \ldots l_{i-1} \wedge \neg\, l_i \wedge \ldots \wedge l_p \Rightarrow z)$= *supp* $(l_1 \wedge l_2 \wedge \ldots l_{i-1} \wedge \neg\, l_i \wedge \ldots \wedge l_p \Rightarrow z)$ / *supp*$(l_1 \wedge l_2 \wedge \ldots \wedge l_{i-1} \wedge \neg\, l_i \ldots \wedge l_p)$

So no such negative rule will be generated by ACN for which a database scan will be needed to count the support and confidence. So ACN performs exactly the same number of dataset scans as Apriori. □

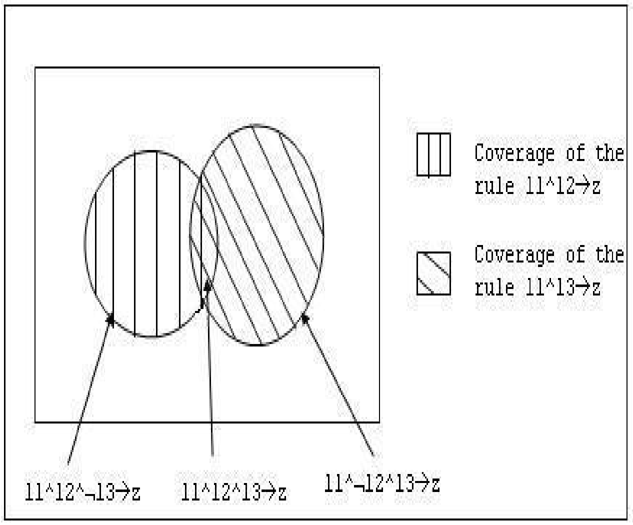

Figure 3. Support and Confidence calculation of negative rule items from other positive rule items

So ACN does not need to perform any extra database scan to calculate support and confidences of the generated negative rules. For any rule, to gather the support and confidence values, we need to consult O(r) records where r is the number of records in the database. But here the support and confidence of a negative rule can be calculated in O(1) time and no I/O operation is needed. Since I/O overhead is very large compared to the processor time required for calculating the support and confidence values for negative rules from their positive counterparts, the mining phase of ACN remains time-efficient.

Figure 3 graphically shows an example how ACN calculates the support and confidence for negative rules. Suppose two rules $l_1 \wedge l_2 \Rightarrow$z and $l_1 \wedge l_3 \Rightarrow$z are frequent. Now Apriori will generate a candidate $l_1 \wedge l_2 \wedge l_3 \Rightarrow$z and count its support. Since the coverage region for $l_1 \wedge l_2 \wedge l_3 \Rightarrow$z is the region where the coverage regions for both rules ($l_1 \wedge l_2 \Rightarrow$z) and ($l_1 \wedge l_3 \Rightarrow$z) overlap, ACN will be able to calculate the support and confidences of two new negative rules

$l_1 \wedge l_2 \wedge \neg l_3 \Rightarrow z$ and $l_1 \wedge \neg l_2 \wedge l_3 \Rightarrow z$ using the equations described above.

4 Experimental Tests

In this section, we present some experimental facts regarding ACN and also compare it with other state-of-the-art classification algorithms in terms of accuracy.

Table 1. Number of positive and negative rules generated for two data sets

Confidence	Diabetes	Diabetes	Heart	Heart
	+ rules	- rules	+ rules	- rules
50-60%	223	571	3438	10496
60-70%	200	546	1474	4823
70-80%	163	520	1770	5201
80-90%	133	448	952	2917
90-100%	142	474	999	2900

Table 1 gives the number of positive and negative rules generated in experiments on two data sets Diabetes and Heart. From the table, it is evident that the number of generated negative rules is several times in number than the number of generated positive rules. In general, a rule is good if it has high confidence. So taking the negative rules into account, ACN generates more good rules than other classification methods that generate only positive rules. So the class association rule set for ACN is much larger and richer.

Table 2 gives the comparison of accuracy among ACN, C4.5, CBA and CMAR. The accuracy of ACN was obtained by 10 fold cross validation over 16 datasets from UCI ML repository[5]. We have used C4.5's shuffle utility to shuffle the datasets. Discretization of continuous attributes is done using the same method in CBA.

For ACN, the minimum confidence was set to 50%, the correlation coefficient threshold was set to 0.2 and the remaining accuracy threshold for negative rules was set to 55%. It is possible to use two different support thresholds for pruning positive and negative rules. However, in the experimental tests, support threshold was set to 1% for both types of rules. We conducted several experiments by varying these parameter values and finally decided on the values that resulted in best classification accuracy.

For C4.5, since the rule method has better accuracy than decision tree method, we only present the results for the rule method.

Table 2. Comparison of C4.5, CBA, CMAR and ACN on accuracy

Dataset	ACN	CMAR	CBA	C4.5
diabetes	**76.3**	75.8	74.5	74.2
pima	75.1	75.1	72.9	**75.5**
tic-tac	**99.7**	99.2	99.6	99.4
iris	**95.3**	94	94.7	**95.3**
heart	**82.2**	**82.2**	81.9	80.8
lymph	**83.1**	**83.1**	77.8	73.5
glass	73.8	70.1	**73.9**	68.7
austra	85.5	**86.1**	84.9	84.7
led7	71.9	72.5	71.9	**73.5**
horse	**83.7**	82.6	82.1	82.6
sonar	**79.8**	79.4	77.5	70.2
hepati	**83.2**	80.5	81.8	80.6
crx	**85.2**	84.9	84.7	84.9
cleve	81.5	82.2	**82.8**	78.2
hypo	98.9	98.4	98.9	**99.2**
sick	97.3	97.5	97	**98.5**
Average	**85.4**	84.6	84.3	83.0

For CBA and CMAR, we conducted the experiments using the same parameter settings originally proposed by their authors. For CBA, the minimum support is set to 1% and minimum confidence to 50%. Other parameters remain default. For CMAR, the support and confidence thresholds are set as it is in CBA. The database coverage threshold is set to 4 and the confidence difference threshold is set to 20%.

From Table2, we see that the average accuracy of ACN is better than CBA, CMAR and C4.5. Moreover, out of the 16 datasets, ACN achieves the best accuracy on more than half (9) datasets.

Table 3 gives the comparison of accuracy among ACN and ARC-PAN. We have used only 6 datasets for comparison here since the accuracy of ARC-PAN was available for these 6 datasets only.

For ARC-PAN, we consider the results obtained when all rules (both positive and negative) are used for classification.

From table 3, we see that the win-loss-tie record of ACN against ARC-PAN is 4-2-0.

Table 3. Comparison of ACN and ARC-PAN on accuracy

Dataset	ACN	ARC-PAN
diabetes	**76.3**	74.9
pima	**75.1**	73.1
iris	**95.3**	94.0
heart	82.2	**83.8**
led7	**71.9**	71.1
breast	95.3	**96.2**
Average	**82.68**	82.18

5 Conclusions

In this paper we proposed a novel classification algorithm ACN that mines both positive and negative class association rules and uses both sets for classification. We showed that the number of generated negative rules is large and so using them in place of some weak positive rules can enhance classification accuracy. Our experiments on UCI datasets show that ACN is consistent, highly effective at classification of various kinds of databases and has better average classification accuracy compared to C4.5,CBA,CMAR and ARC-PAN.

6 Acknowledgement

This work is performed as a part of undergraduate thesis in the Department of Computer Science and Engineering of Bangladesh University of Engineering and Technology (BUET)[8]. Our special thanks to BUET for providing such a good environment of research activities in the field of Computer Science and Engineering.

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB*, Chile, September 1994.

[2] M. Antonie and O. R. Zaïne. An associative classifier based on positive and negative rules. In *DMKD*, Paris, France, June 2004.

[3] M. Antonie and O. R. Zaïne. Mining positive and negative association rules: an approach for confined rules. In *Principles and Practice of Knowledge Discovery in Databases*, 2004.

[4] F. Berzal, J.-C. Cubero, D. Sänchez, and J. M. Serrano. ART: A hybrid classification model. *Machine Learning*, 54(1), January 2004.

[5] C. Blake and C. Merz. UCI repository of machine learning databases.

[6] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

[7] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *SIGMOD*, dallas, Texas, 2000.

[8] G. Kundu, M. M. Islam, S. Munir, and M. F. Bari. New algorithms for associative classifications. In *Undergraduate Thesis*, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, 2007.

[9] G. Kundu, S. Munir, M. F. Bari, M. M. Islam, and K. Murase. A novel algorithm for associative classification. In *International Conference on Neural Information Processing*, Japan, 2007.

[10] W. Li, J. Han, and J. Pei. Accurate and efficient classification based on multiple class association rules. In *ICDM*, San Jose, CA, November 2001.

[11] T. Lim, W. Loh, and Y. Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 39, 2000.

[12] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *KDD*, New York, August 1998.

[13] J. Quinlan. *C4.5:Programs for Machine Learning*. Morgan Kaufmann, 1993.

[14] A. Savasere, E. Omiecinski, and S. Navathe. Mining for strong negative associations in a large database of customer transactions. In *ICDE*, Florida, USA, 1998.

[15] S.Brin, R. Motwani, and C.Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *ACM SIGMOD*, Tucson, Arizona, 1997.

[16] F. Thabtah. A review of associative classification mining. *The Knowledge Engineering Review*, 22(1), March 2007.

[17] F. Thabtah, P. Cowling, and Y. Peng. MCAR: Multi-class classification based on association rules. In *3rd ACS/IEEE International Conference on Computer Systems and Applications*, Cairo, Egypt, 2005.

[18] D. Thiruvady and G. Webb. Mining negative association rules using grd. In *PAKDD*, Sydney, Australia, 2004.

[19] J. Wang and G. Karypis. HARMONY: Efficiently mining the best rules for classification. In *SDM*, California, USA, 2005.

[20] X. Wu, C. Zhang, and S. Zhang. Efficient mining of both positive and negative association rules. *ACM Trans. on Information Systems*, 22(3), 2004.

[21] P. Yan, G. Chen, C. Cornelis, M. D. Cock, and E. Kerre. Mining positive and negative fuzzy association rules. In *LNCS 3213*, 2004.

[22] X. Yin and J. Han. CPAR: Classification based on predictive association rules. In *SDM*, California, USA, 2003.

[23] X. Yuan, B. P. Buckles, Z. Yuan, and J. Zhang. Mining negative association rules. In *Seventh International Symposium on Computers and Communications*, Italy, June 2002.