An Efficient Algorithm That Optimizes The Classification Association Rule Set

Nguyen Quoc Huy, Tran Anh Tuan
Saigon University
nqhuy@sgu.edu.vn, dasanbob22122002@gmail.com

Nguyen Thi Ngoc Thanh
Ho Chi Minh City Open University
thanh.ntn@ou.edu.vn

***Abstract*—Associative classification is one interesting approach of supervised learning that are easily explainable. This approach is often constructed based on both classification and association rule mining techniques, to find out a set of rules called the classification association rules (CAR), to classify target attribute. The algorithm ACAC belongs to the family of CAR problems, but the classification accuracy is still low and not working in large data. This paper proposes a great FSAC algorithm improved significantly the classification accuracy as well as running time of ACAC algorithm. Experimental results show that the accuracy is increment after each step discarding the redundancy attributes of training data, resulting in the CAR set is reduced, and the processing time is faster also.**

***Index Terms*—CAR; feature selection; entropy; hill-climbing**

I. Introduction

Transaction databases account for a large proportion of data today, so the need to mine and process this data is equally important as the others such as image, audio, geographic data, etc. The classification problem of transaction data has many methods, the classification association rule (CAR) method is relatively easy to understand and easy to implement, which has been studied so much in the past decade. Most of the association rule-based classification algorithms are based on "support - confidence", and this way is not efficient due to the large number of rules generated when the threshold of support is small, and correspondingly, the cost of running time is also too high[4,5,6]. In 2003, Omiecinski proposed a very effective All-confidence calculation in mining association patterns. Based on this measure, Zaixiang Huang et al proposes the ACAC[1,4] algorithm that uses both the All-confidence and support measures to exploit itemsets that are not only frequent but also mutual, this property is essential for CAR problems. However, this algorithm still has many limitations running time and classification accuracy when the number of attributes is large. This paper proposes a method to remove unimportant attributes to improve the ACAC method. This improvement helps to increase running speed while increasing accuracy significantly.

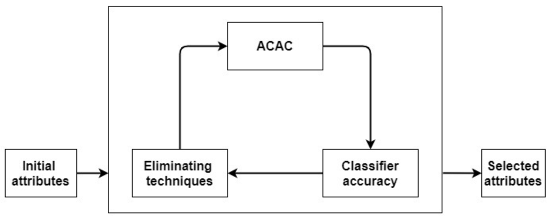

Fig. 1: Process of redundancy attributes removal

Figure 1 proposes a process to remove unimportant attributes according to the measure of Entropy (Entropy value between an attribute and the target), the accuracy of the model after removing is validated by cross-validation method. The set of selected attributes are the optimal attributes for classification.

Entropy values are in the range [0,1], attributes with high entropy values that will not contribute much information to the classification should be discarded. In datasets with few attributes, we can calculate all the entropy of the attributes then order them from highest to lowest. After that, sequentially check the removal of each attribute according to the priority of Entropy from high to low, and evaluate the accuracy of the remaining set of attributes. If the accuracy of the remaining attribute set is equal to or greater than the old value (when the attribute has not been removed), then the removed attribute should really be removed. However, for a large dataset with too many attributes, the evaluation cost for removing each attribute will be very large and the above improvement idea will not be feasible. The paper proposes the FSAC algorithm that helps not exhaustively evaluate the ability to remove each attribute, but only evaluates some attributes based on the initial threshold chosen by the user.

The main contribution of the paper focuses on section 2.2 and section 3. The content of the paper is arranged as follows: section 2A will refer to classification association rules and properties of Entropy measure, section 2B presents FSAC algorithm, experimental results and proof of effectiveness are described in section 3, and research conclusions are presented in section 4.

II. Associative Classification

According to this problem, the training data T has m attributes a_1, a_2, …, a_m, and a target attribute. Attribute values can be continuous or discrete. For the discrete values, we can use consecutive positive integers to represent them. For the continuous values, we discretize these values first and use consecutive positive integers to represent.

An itemset X=a_{i_1},...,a_{i_j},...,a_{i_k} is a set of values of different attributes. Thus, the 1-itemset is defined by a value of any attribute, and is called a_{i_j}. The number of transactions in T matching itemset X is called the support of X, represented as sup(X). The all-confidence of the itemset is defined as follows:

$$\text{Allconf}(X) = \frac{\text{sup}(X)}{\max(\text{sup}(a_{i_1}), \ldots, \text{sup}(a_{i_k}))} \quad (1)$$

This formula (1) determines the minimum confidence of all rules generated from an itemset X[1,7,8].

Given the training dataset T, let c be the class label. A ruleitem is of the form condset, c, where condset is an itemset. Each ruleitem represents basically one rule: $condset$ →c Ruleitem that its condset has k items is called k-ruleitem. The support of condset (called consup) is the number of occurrences of condset in T. The support of a rule (called rulesup) is the number of occurrences of both condset and c in T.

A. ACAC method

Most classification algorithms have three phases:

1. Generate rules from the training set, association rules have the form $condset$ →c.
2. Remove rules that may cause overfitting or redundancy.
3. Classify the new dataset

To imagine easily, let us consider an example on a given dataset T as shown in Table 1. In this example, the threshold of support is 2, the threshold of all-confidence is 50%, and the threshold of confidence is 100%.

In this table, we select ruleitems that satisfy condsup and all-confidence and put them in the candidate set F1 containing 1-ruleitems. From the set F1, we choose the ruleitems that satisfy the confidence threshold and put them into the set R1, then delete the set F1.

Table I: A trainning dataset

A	B	C	D	class
32	55	80	83	90
33	52	80	85	89
33	52	80	85	89
33	55	79	82	90
34	55	79	82	89
34	55	77	82	89
32	55	80	88	90
33	55	79	82	90

Table II: The set of candidate 1-ruleitems F1

item	cond-sup	rule-sup	allconf	conf	class
33	4	2	100	50	90
33	4	2	100	50	89
55	6	4	100	33	90
55	6	2	100	67	89
79	3	2	100	67	90
79	3	1	100	33	89
80	4	2	100	50	90
80	4	2	100	50	89
82	4	2	100	50	90
82	4	2	100	50	89

Table III: The set of rule R1

item	cond-sup	rule-sup	allconf	conf	class
32	2	2	100	100	90
34	2	2	100	100	89
52	2	2	100	100	89
85	2	2	100	100	89

The candidate 2-ruleitems are combined from the set F1, assume that the set F1 has two 2 rules:

Ruleitem(55), 90 with rulesup is 4

Ruleitem(79), 90 with rulesup is 2

The generated rule will be (55, 79), 90 with rulesup is 2, all-conf = 2/max(4,2) = 50%. Output of 2-ruleitems are put on set F2, and the set R2 contains 2-ruleitems which satisfy the confidence threshold.

This process is repeated until the candidate set of k-ruleitem is empty. Algorithm 1 is named ACAC-RG[1], where the candidate set C_k containing k-itemsets, the frequent set F_k containing k-itemset. R is a set of rules. Lines 1-2 compute the condsup and rulesup of each item simultaneously, then the

Table IV: The set of candidate 1-ruleitems F2

item	cond-sup	rule-sup	allconf	conf	class
55 79	3	2	50	67	90
55 82	4	2	50	50	89
55 82	4	2	100	50	90
79 82	3	2	100	67	90

Table V: The set of rule R2

item	cond-sup	rule-sup	allconf	conf	class
33 55	2	2	100	100	90
33 79	2	2	100	100	90
33 80	2	2	100	100	89
33 82	2	2	100	100	90
33 80	2	2	100	100	90

ruleSelection function is called at each iteration (line 6). In each iteration, the algorithm performs 3 main operations. First, the frequent itemsets F_{k-1} found in the iteration (k-1) are used to generate the candidate itemsets C_k according to the candidateGen function (line 4), which is similar to the Apriori-Gen function. Next, the supportCount function scans the dataset and computes the condsup and rulesup of the candidates in C_k (line 5). Finally, the ruleSelection function is executed.

Algorithm 1 ACAC-RG(T)[1]

Require: Training data set; Minimum Support (minSup); Minimum confidence (minConf); Minimum all-confidence (minAllConf)
Ensure: R is set of rules for predicting class labels for objects

```
1: $C_1 \leftarrow init-pass(T)$;
2: $ruleSelection(C_1, F_1, R)$
3: for $k \leftarrow 2$; $F_{k-1} \neq 0; k++$ do
4:     $C_k \leftarrow candidateGen(F_{k-1})$;
5:     $supportCount(C_k)$;
6:     $ruleSelection(C_k, F_k, R)$;
7: end for
8: return $R$
```

The ruleSelection function calculates the all-confidence of each candidate itemset in each class and the confidence of each candidate rule. If the candidate rules satisfy the support threshold, all-confidence threshold, and confidence threshold, then the rules will be put in the set R. If the candidate rule only satisfies the support threshold, all-confidence threshold, then put it into the set F_k. Once a candidate rule is selected, its condset will not be expanded in the next inner-loop.

Algorithm 2 ACPredict(R,D)[1]

Input: R is the rule set obtains from ACAC-RG, Dnew (the new dataset),
Output: D' (the new dataset with label)

```
1: $D' = empty$
2: for each $t_i$ in $D$ do
3:     $v_1$ = S(R, $t_i$,Yes),
4:     $v_2$ = S(R, $t_i$,No),
5:     if $v_1 > v_2$ then
6:         $t_i$ [Label] ← Yes //assigned Yes
7:     else
8:         $t_i$ [Label] ← No //assigned No
9:     end if
10:    $D' \leftarrow D' + t_i$
11: end for
12: return $D'$
```

After running the ACAC-RG[1] algorithm, we obtain a set of rules R. Based on the set of rules R, the classification steps are as the ACPredict[1] algorithm: Calculate the Info(entropy) value for each attribute according to the following formula:

$$\text{Info}(X) = \frac{1}{\log_2 k} \sum_{i=k}^{n} P(C_i|X) \log_2 P(C_i|X) \quad (2)$$

where k is the number of classes, $P(C_i|X)$ is probability of X fitting with C_i. These entropy formula is used to calculate formula (3), another formula to calculate how well the rules will be fitted:

$$S(r_i) = 0.9 \left(1 - \frac{\sum \text{sup}(X_i)\text{Info}(X_i)}{\sum \text{sup}(X_i)}\right) + 0.1 \frac{n}{n_{\text{tot}}} \quad (3)$$

where X_i is subrule set of r_i ($r_i \in R$), n_{tot} is total of rules which are fitted with new object, n is the number of rules in R set. After calculating the value of $S(r_i)$, ACAC uses the group with the highest $S(r_i)$ to assign label for the new data. The classification process according to the formula (3) is described in the ACClassifier algorithm.

B. The improved method

FSAC algorithm optimizes ACAC by discarding the redundant attributes, line 13 is the original ACAC algorithm, line 14 calculates the accuracy of the model obtained from line 13. The code from line 8 to line 17 repeats the improvement work until the accuracy of model is at optimal point. In each iteration, based on the current entropy, we select a neighborhood (lines 11, 12) entropy e', the algorithm determines the feature set De that includes the attributes whose entropies relative to the target attribute are less than e'. In other words, the algorithm tries to remove the redundant attribute from the training set in each iteration. Line 15 checks whether the discard is correct, if it is, then updates the better value for the current entropy. The algorithm will find the best model at out of loop (the optimal set of CAR) as line 18. In line 2, D is the data set divided into two groups Dtrain and Dtest according to the ratio 80/20. Based on Dtrain, we find CAR set, and Dtest is used to test the accuracy of CAR set.

Algorithm 3 FSAC

```
1: minSup, minConf, minAllConf;
2: D: dataset ($D_{train}$, $D_{test}$);
3: e := entropy threshold ;
4: $D_e$ := attrribute $a_i$ from $D_{train}$ where entropy($a_i$, label) $\leq$ e;
5: neighbor=[e,e $\pm \triangle$];
6: e' := random(neighbor);
7: $D_{e'}$ := attrribute $a_i$ from $D_{train}$ where entropy($a_i$, label) $\leq$ e;
8: repeat
9:     $model_e$ := ACAC-RG($D_e$);
10:    $accuracy_e$:= $model_e$.predict($D_{test}$);
11:    neighbor=[e,e $\pm \triangle$];
12:    e' := random(neighbor);
13:    $model_{e'}$ := ACAC-RG($D_e$);
14:    $accuracy_{e'}$:= $model_{e'}$.predict($D_{test}$);
15:    if ($accuracy_e \leq accuracy_{e'}$) then
16:        e:=e'
17:    end if
18: until $accuracy_e \leq accuracy_{e'}$
19: return $model_e$
```

III. Experiments

The experimental environment is processed centrally on computers with configuration Intel(R), Core(TM) i7-6820HQ CPU @ 2.70GHz (8 CPUs), 2.7GHz and the Windows 10 operating system. The experiment is compared on two different data sets.

A. Mushroom Dataset

The first experiment is implemented on Mushroom data [2]. Because this data is used by the original ACAC algorithm, it has 8125 rows and 23 columns. In the data, the first attribute is a target attribute named 'class' to classify mushrooms as poisonous or non-poisonous (edible=e, poisonous=p).

In the remaining 22 attributes, there are many attributes that are not necessary in classification, and they need to be removed. The paper uses the entropy to determine the ability to contribute in the classification of each attribute. The attribute of high entropy value will be removed from the dataset, the smaller dataset is retried by ACAC method to find the CAR set. This CAR set will be tested by cross-validation method (80% - 6500 data rows for training, 20% - 1625 data rows for testing). If the precision increases, then the attribute removal is correct, and this is done until the accuracy reaches the highest value. In other words, the process will stop when the accuracy is decreased.

To compare fairly with ACAC, this paper also uses the same thresholds (minsup = 0.1, minconf = 0.8, minallconf = 0.5) used on the original ACAC. The FSAC improved ACAC by using entropy to remove unnecessary attributes on the mushroom dataset. The table VI shows that the FSAC algorithm removed 17 redundant attributes (unnecessary for classification), reduced the original rule set from 55 rules to 9 rules, and the processing time is significantly faster (131 ms compared to 2384 ms). Especially, the classification accuracy is increased significantly (54.28% vs 99.51%). Readers can duplicate simply step by step the experiment with decreasing entropy as shown in Table VII.

Table VI: Efficiency of FSAC vs ACAC

	Attr-ibutes	run-times(ms)	rules	accu-raccy (%)
FSAC	5	131	9	99.51
ACAC	22	2384	55	54.28

From the results in Table VII, we have a graph as Figure 2. In this graph, the vertical axis represents the accuracy, the horizontal axis represents the entropy of each attribute with target attribute. The graph has the maximum point of classification accuracy of 99.51% from the entropy score of 0.757, corresponding to 8 attributes. However, the algorithm is run continuously with entropy values (0.745, 0.727, 0.714) respectively to remove 3 more attributes, and the classification value still does not decrease. The classification accuracy only decreases by removing one more attribute. So the algorithm stops at the entropy value of 0.714, and gets the highest classification accuracy of 99.51%.

In this experiment as Table VII, we have a graph as Figure 3. In this graph, the vertical axis describes the number of rules used for classification, the horizontal axis represents the entropy of each attribute relative to the target attribute. The graph shows that the more redundant attributes in the training set are removed (unnecessary attribute for classification), the number of rules in the CAR set is more reduced. The graph in Figure 4 has a vertical axis describing the processing time, and the horizontal axis representing the entropy of each attribute relative to the target attribute. Figure 4 shows that the processing time is also significantly reduced. With the graphs of Figure 2, Figure 3, Figure 4, we see that it is reasonable to use the entropy to evaluate the importance of classification for each attribute in the training set. The results show that the more redundant attributes are removed, the higher the accuracy is, the CAR set is also reduced, and the processing time is reduced as well.

Table VII: Entropy between attributes and target attribute

Num	Attributes	entropy	accuraccy(%)
1	odor	0.093	52.98
2	spore-print	0.52	52.98
3	gill-color	0.582	97.35
4	ring-type	0.681	97.35
5	stalk-surf-abo	0.714	99.51
6	stalk-surf-bel	0.727	99.51
7	stalk-color-abo	0.745	99.51
8	stalk-color-bel	0.757	99.51
9	gill-size	0.768	87.08
10	population	0.797	87.08
11	bruises	0.806	87.08
12	habitat	0.842	87.08
13	stalk-root	0.864	76.06
14	gill-spacing	0.898	76.06
15	cap-shape	0.95	76.06
16	ring-number	0.961	75.57
17	cap-color	0.963	75.57
18	cap-surface	0.97	75.57
19	veil-color	0.971	75.57
20	*gill*	0.984	75.57
21	stalk-shape	0.991	54.28
22	veil-type	0.999	54.28

B. Mail Spam Dataset

We have encouraging results on the Mushroom dataset. However, the Mushroom is still a small data compared to many types of data that need to be classified in practice. To be more robust, we implement the experiment on the more complex Spam Emails Dataset [3], which has 4602 rows and 58 columns. In which, the last attribute is a label attribute named 'spam' to classify emails as spam or ham (spam=1,

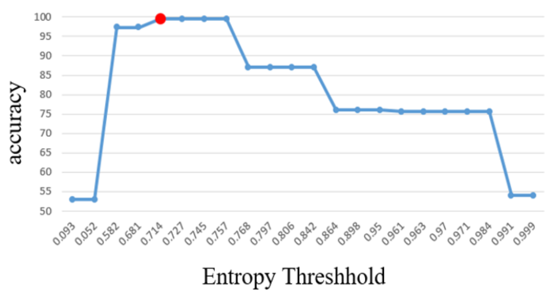

Fig. 2: The optimal CAR set

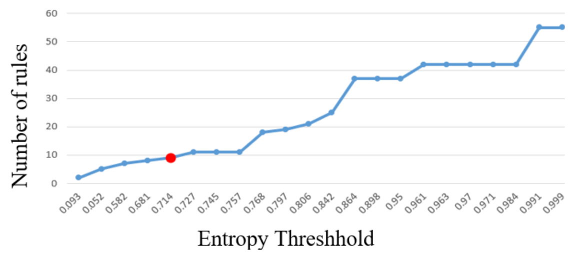

Fig. 3: The optimal CAR set

ham=0). There are many attributes that are not necessary for classification, and need to be removed by FSAC algorithm. The CAR set will be tested by cross-validation method (80% - 5000 data rows for training, 20% - 1001 data rows for testing). This experiment also uses the same thresholds (minsup = 0.1, minconf = 0.9, minallconf = 0.5).

Table VIII: The experimental result of FSAC

Entropy	Attr	rules	times(ms)	Acc(%)
0.88	31	2239	3.5E+11	94.4
0.875	29	1612	11E+9	94.9
0.87	28	1254	261000000	94.06
0.865	27	989	51000000	93.04
0.86	26	692	10000000	92.4
0.85	24	338	2200000	90.99
0.84	22	109	35170	88.93
0.83	17	24	343	87.84
0.81	14	17	151	81.11
0.79	13	13	145	80.61

Table VIII describes that the FSAC algorithm is implemented 12 times from an initial entropy value of 0.79 gradually increasing, and the experimental result in Figure 5 illustrates the classification accuracy at entropy of 0.875 has accuracy of 94.9%, the number of attributes is 29 compared to the original number of 57. The accuracy decreases if the entropy value is increased. Figure 6 illustrates that the number of

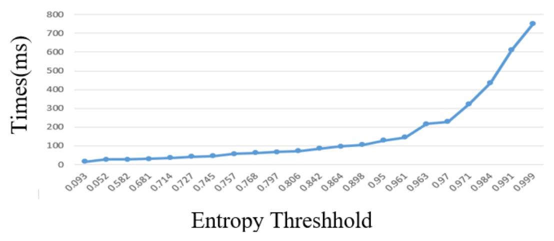

Fig. 4: Running time of processing

rules at the optimal entropy of 0.875 is 1612, when the data is complex (Spam Mail), the number of rules increases significantly compared to simple data (Mushroom) .Figure 7 decribes the processing time,

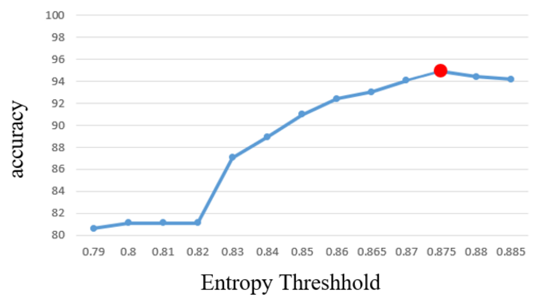

Fig. 5: The accuracy of FSAC in a large dataset

it consumes significantly at the number of attributes from 29 to 31.

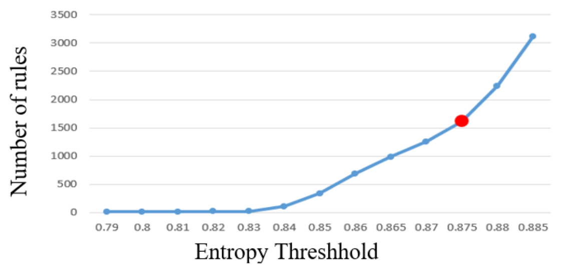

Fig. 6: The optimal CAR set in a large dataset

Experimenting the FSAC algorithm on complex data (Spam Mail) also gives encouraging results when using the entropy to remove attributes that are unnecessary for classification. Figures 5, 6, and 7 show that when redundant attributes are removed, the classification accuracy increases, the CAR set as well as processing time is reduced also. For the FSAC algorithm, we do not need to calculate the entropy for all the attributes exhaustively, we only need to choose the first entropy threshold and its neighborhood. Then apply the idea of hill climbing to optimize the CAR set.

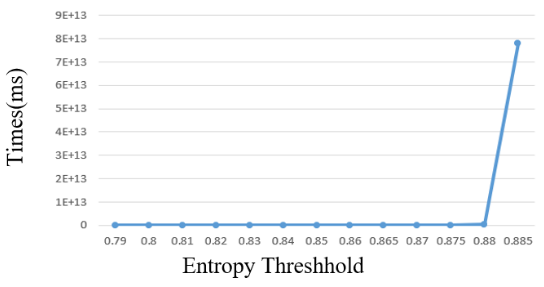

Fig. 7: Running time of processing in Spam Mail

IV. Conclusion

The FSAC algorithm significantly improves the ACAC algorithm based on the idea of removing redundant attributes in the training dataset. To determine whether the attribute has a role in classification, the paper used the measure of entropy between the data attribute and the target attribute. Through the experimental process on the Mushroom and Spam Mail dataset, it has been shown that the attribute with high entropy is prioritized to be eliminated, then the classification accuracy will increase, the classification rule set is reduced, and the processing time is also reduced.

For large data like Spam Mail, the processing time is still quite large. In the next direction of improvement, we will find the relationship between the original CAR set and a smaller CAR set after removing an attribute. From there, it is possible to find a method to reduce the cost of recalculating the smaller CAR set.

Reference

[1] Z. Huang, Z. Zhou, T. He, and X. Wang, “ACAC: Associative Classification Based on All-Confidence," *IEEE International Conference on Granular Computing*, pp. 289–293, 2011.

[2] *Mushroom Classification Dataset*, [Online]. Available: `https://www.kaggle.com/datasets/uciml/mushroom-classification`

[3] *Spam Emails Dataset*, [Online]. Available: `https://www.kaggle.com/datasets/yasserh/spamemailsdataset`

[4] H. F. Ong, C. Y. M. Neoh, V. K. Vijayaraj, Y. X. Low, “Information-Based Rule Ranking for Associative Classification," *ISPACS*, 2022.

[5] M. Abrar, A. Tze and S. Abbas, “Associative Classification using Automata with Structure based Merging," *IJACSAA*, vol. 10, 2019.

[6] D. L. Olson and G. Lauhoff, “Market Basket Analysis" in Descriptive Data Mining," *Springer Singapore*, 2019.

[7] K. D. Rajab, “New Associative Classification Method Based on Rule Pruning for Classification of Datasets," *IEEE Access*, vol. 7, pp. 157783-157795, 2019.

[8] H. F. Ong, N. Mustapha, H. Hamdan, R. Rosli and A. Mustapha, “Informative top-k class associative rule for cancer biomarker discovery on microarray data," *Expert Systems with Applications*, vol. 146, 2020.