How overall coverage of class association rules affects the accuracy of the classifier?

Jamolbek Mattiev
Department of Information Sciences and Technologies
University of Primorska
Koper, Slovenia
jamolbek.mattiev@famnit.upr.si

Branko Kavšek[1,2]
[1]Artificial Intelligence Laboratory
Jožef Stefan Institute
Ljubljana, Slovenia
branko.kavsek@ijs.si
[2]Department of Information Sciences and Technologies
University of Primorska
Koper, Slovenia
branko.kavsek@upr.si

ABSTRACT

Associative classification (AC) is a data mining approach that combines classification and association rule mining to build classification models (classifiers). Experimental results show that in average the CBA-based approaches could achieve higher accuracy than some of the traditional classification methods.

In this paper, we focus on associative classification, where class association rules are generated and analyzed to build a simple, compact, understandable and relatively accurate classifier. Furthermore, we discuss how overall coverage and average rule coverage of such classifiers affect their classification accuracy. We compare our method that uses constrained exhaustive search with some "classical" classification rule learning algorithm that uses greedy heuristic search on accuracy in some "real-life" datasets. We have performed experiments on 11 datasets from UCI Machine Learning Database Repository.

Experimental evaluation shows that with decreasing overall coverage our proposed method tends to get slightly worse classification accuracy than the "classical" classification rule learning algorithms. Otherwise, the accuracy is similar or on some datasets even better than Naive Bayes and C4.5. On the other hand, the average rule coverage of our proposed method seems to have no effect on classification accuracy.

CCS CONCEPTS

• **Computing methodologies → Machine learning → Machine learning approaches → Rule learning**
• **Computing methodologies → Machine learning → Cross-validation**

• **Computing methodologies → Machine learning → Learning paradigms → Supervised learning → Supervised learning by classification**

KEYWORDS

Attribute, frequent Itemset, Minimum Support, Minimum Confidence, Class Association Rules (CAR), Associative Classification.

1 INTRODUCTION

Frequent patterns and their corresponding association rules characterize interesting relationships between attribute conditions and class labels, and thus have been recently used for effective classification. Association rules show strong associations between attribute-value pairs (or items) that occur frequently in a given dataset. Association rules are commonly used to analyze the purchasing patterns of customers in stores. Such analysis is useful in many decision-making processes, such as product placement, catalog design, and cross-marketing. The discovery of association rules is based on frequent itemset mining.
Associative classification mining is a promising approach in data mining that utilizes the association rule discovery techniques to construct classification systems, also known as associative classifiers. In the last few years, a number of associative classification algorithms have been proposed such as CBA: Classification based Association [11], CMAR: Classification based on Multiple Association Rules [10], CPAR: Classification based on Predicted Association Rule [13]. These algorithms employ several different methods, such as rule discovery, rule ranking, rule pruning, rule prediction and rule evaluation. Machine learning is one of the main phases in knowledge discovery from databases, which extracts useful patterns from data. Associative classification (AC) is lately among the focus areas in machine learning. AC integrates two known data mining tasks, association rule discovery and classification. The main aim is to build a model (classifier) for the purpose of prediction. Classification and association rule discovery are similar tasks in data mining, with the exception that the main aim of classification is the prediction of class labels, while association rule discovery describes correlations between items in a transactional database. In the last few years, association rule discovery methods have been successfully used to build accurate classifiers, which have resulted in a branch of AC mining. Several studies [4,9,10,11] have proved that AC algorithms are able to extract classifiers competitive with those produced by decision trees [3,12], rule induction [5,6,8] and probabilistic approaches [2].
In comparison with some traditional rule-based classification approaches, associative classification has two main characteristics. Firstly, it generates a large number of association classification rules. Secondly, support and confidence thresholds are applied to evaluate the significance of classification association rules. However, associative classification has some weaknesses. First, it often generates a very large number of

classification association rules in association rule mining, especially when the training dataset is large. It takes great efforts to select a set of high quality classification rules among them. Second, the accuracy of associative classification depends on the setting of the minimum support and the minimum confidence. Unbalanced datasets may heavily affect the accuracy of the classifiers. Third, the efficiency of associative classification may be also low, when the minimum support is set to be low and the training dataset is large. Although associative classification has some drawbacks, it can achieve higher accuracy than rule and tree-based classification algorithms on certain real life datasets.

In this paper, we propose a simple classification method that selects a reasonable number of rules for classification, then, we find the overall coverage and average rule coverage of the classifier. We perform experiments on 11 datasets from the UCI Machine Learning Database Repository [7] and compare the results with some of the well-known classification algorithms (Naïve Bayes [2], PART [8], Ripper [6], C4.5 [12]).

2 PRELIMINARY CONCEPTS

Association rules consist of two parts, an antecedent (if) and a consequent (then). An antecedent is an item found in the data. A consequent is an item that is found in combination with the antecedent. Association rules are generated from frequent itemset by analyzing the dataset and *support* and *confidence* thresholds are used to identify the most important relationships. *Support* is an indication of how frequently the items appear in the dataset. *Confidence* indicates the number of times the if/then statements have been found to be true.

Associative classification is a special case of association rule discovery in which only the class attribute is considered in the rule's right-hand side (consequent), for example, in a rule such as X→Y, Y must be a class attribute. One of the main advantages of using a classification based on association rules over classic classification approaches is that the output of an AC algorithm is represented in simple if–then rules, which makes it easy for the end-user to understand and interpret it

Let D be a dataset with n attributes $\{A_1, A_2, ..., A_n\}$ that are classified into M known classes and $|D|$ objects. Let $Y = \{y_1, y_2 \dots . y_m\}$ be a list of class labels. A specific value of an attribute A_i and class Y is denoted by lower-case letters a_{im} and y_j respectively.

Definition 1. An itemset is a set of some pairs of attributes and a specific value, denoted $\{(A_{i1}, a_{i1}), (A_{i2}, a_{i2}), \dots, (A_{im}, a_{im})\}$.

Definition 2. A class association rule R has the form $\{(A_{i1}, a_{i1}), ., (A_{im}, a_{im})\} \to y_j$ where $\{(A_{i1}, a_{i1}), ., (A_{im}, a_{im})\}$ is an itemset and $y_j \in Y$ is a class label.

Definition 3. The support count *SuppCnt(R)* of a rule *R* in *D* is the number of records of *D* that match *R*'s antecedent (left-hand side).

Definition 4. The support of rule *R*, denoted by *Supp(R)*, is the number of records of *D* that match *R*'s antecedent and are labeled with *R*'s class.

Definition 5. The confidence of rule R, denoted by *Conf(R)*, is defined as follows: $Conf(R) = Supp(R)/SuppCnt(R)$.

3 PROBLEM DEFINITION

Our proposed research assumes that the dataset is a normal relational table which has *N* examples described by *L* distinct attributes. These *N* examples are classified into *M* known classes. An attribute can be categorical (or nominal) or continuous (or numeric). In this paper, we treat all the attributes uniformly. Categorical attribute's values are mapped to a set of consecutive positive integers. Numeric attributes are discretized into intervals (bins), and the intervals are also mapped to consecutive positive integers. Discretization methods will not be discussed in this paper as there are many existing algorithms in the machine learning literature that can be used.

Our first goal is to generate the complete set of strong class association rules that satisfy the user-specified minimum support and minimum confidence constraints, and the second goal is to extract a reasonable number of strong CARs by pruning to build a simple and accurate classifier, the third and main goal is to find the overall coverage, average rule coverage and accuracy of the intended classifier.

4 OUR PROPOSED METHOD

Our proposed method consists of three steps. Firstly, a complete set of strong class association rules is generated from the given dataset. We then select a reasonable number of strong rules to build our simple and accurate classifier in the second step. Finally, we find the overall coverage, average rule coverage and accuracy of the classifier.

4.1 Generating class association rules

Association rule generation is usually split up into two separate steps:

1. First, we find all the frequent itemsets in a dataset by applying minimum support threshold. This step is the most important one, because, if minimum support is set to low, then we may have huge number of rules that lead to combinatorial complexity. If minimum support is set to high, then we may lose some interesting or strong rules, therefore, appropriate minimum support must be applied by analyzing the dataset.
2. Second, minimum confidence constraint is applied to generate strong class association rules from these frequent itemsets generated in the first step.

The second step is straightforward, that is why, we pay more attention to the first step. Apriori is a seminal algorithm described in [1] and it is mostly suggested for mining frequent itemsets.

Once the frequent itemsets from the dataset have been found, it is straightforward to generate strong class association rules from them (where strong CARs satisfy both minimum support and minimum confidence constraints). This can be done using following equation for confidence:

$$confidence(A \to B) = \frac{support\_count(A \cup B)}{support\_count(A)}. \qquad (1)$$

The equation (1) is expressed in terms of itemsets support count, where *A* is premises (itemsets that is left-hand side of the rule), *B* is consequence (class label that is right-hand side of the rule), *support_count*$(A \cup B)$ is the number of transactions containing the itemsets $A \cup B$, and *support_count*(*A*) is the number of transactions containing the itemsets *A*. Based on this equation, CARs can be generated as follows:

● For each frequent itemset in *L* and class label *C*, generate all nonempty subsets of *L*.

● For every nonempty subset *S* of *L*, output the rule "*S*→*C*" if $\frac{support\_count(L)}{support\_count(S)} \geq \min\_conf$, where $\min\_conf$ is the minimum confidence threshold.

4.2 Building our proposed classifier

We build our intended simple classifier by extracting the reasonable number of strong class association rules (already satisfied the minimum support and confidence requirements) that are generated in 4.1. Our proposed method is outlined in Algorithm 1.

Algorithm 1: Simple and accurate classification algorithm

Input: a set of CARs with their *support* and *confidence* constraints
Output: a subset of rules for classification

```
1:    D= Dataset();
2:    F= frequent_itemsets(D);
3:    R= genCARs(F);
4:    R= sort(R, minconf, minsup);
5:    G=Group(R);
6:    for (k=1; k≤ numClass; k++) do begin
7:        X= extract(class[k], numrules);
8:        Classifier= Classifier.add(X);
9:    end
10:   for each rule $y \in Classifier$ do begin
11:       if y classify new_example then
12:       class_count[y.class]++;
13:   end
14:   if max(class_count)==0 then
15:       predicted_class=majority_class(D);
16:   else predicted_class= index_of_max(class_count);
17:   return predicted_class
```

In lines 1-2 find all frequent itemsets in the dataset by using the Apriori algorithm. Line 3 generates the strong class association rules that satisfiy the minimum support and confidence constrains from frequent itemsets. In line 4, CARs are sorted by *confidence* and *support* in descending order as follow:

Given two rules R_1 and R_2, *R1* is said having higher rank than *R2*, denoted as $R_1 > R_2$,

- If and only if, $conf(R_1) > conf(R_2)$; or
- If $conf(R_1) = conf(R_2)$ but, $supp(R_1) > supp(R_2)$: or
- If $conf(R_1) = conf(R_2)$ and $supp(R_1) = supp(R_2)$, R_1 has fewer attribute values in its left-hand side than R_2 does;

Line 5 defines how to group the class association rules by their class labels (for example, if the class has three values, then, rules are grouped into three groups). In lines 6-9, we extract the reasonable number of rules per class that are equal to *numrules* to form a simple and accurate classifier. These set of rules become our final classifier. In lines 10-13, classification is performed by extracted CARs in line 6-9, if the rule can classify the example correctly, then, we increase the corresponding class count by one and store it. In lines 14-17, if none of the rules can classify the example correctly, then, algorithm returns the majority class value for the training dataset. Otherwise, it returns the majority class value of correctly classified rules.

4.3 Overall coverage and average rule coverage

After our classifier is built in 4.2, it is straightforward to compute the overall coverage and average rule coverage of the classifier. To compute the overall coverage, we count the transactions that are covered by the classifier and divide it to total number of transactions in dataset. For the rule coverage, we count all the transactions that are covered by each rule in classifier and we take the average of them divided by total transactions.

Algorithm 2: Overall and average rule coverage of the classifier

Input: dataset and classifier
Output: overall coverage and average rule coverage

```
1:    n=D.length();
2:    C= Classifier;
3:    fill(classified_example)=false;
4:    for (i=1; i≤ C.length(); i++) do begin
5:        for (j=1; j≤ n; j++) do begin
6:            if C[i].premise classifies D[j].premise then
7:                rulecover[i]++;
8:                classified_example[j]=true;
9:        end
10:       avg_rulecover=avg_rulecover+ rulecover[i]/n;
11:   end
12:   for (i=1; i≤ n; i++) do begin
13:       if classified_example[i] then
14:           count++;
15:   end
16:       Overallcover_dataset=count/n;
17:   return Overallcover_dataset, avg_rulecover
```

First line finds the length of the dataset. We form our classifier introduced in 4.2 (method is already created in lines 6-9 of algorithm 1) from the intended dataset in line 2. In the third line, we fill all initial values of *classified_example* array as false. Lines 4-11generally find the average rule coverage of the classifier. More precisely, we try to classify all the examples in the dataset by our classifier in lines 5-9. If rule's premise (left hand-side of the rule) classifies the example's premise (left hand-side of the example) in the dataset, then we increase the count for that rule's coverage and we mark that example as classified (this helps to compute the overall coverage of the dataset). Line 10 calculates the average rule coverage. We count all correctly classified examples in the dataset in lines 12-15 and overall coverage of the dataset is found in line 16. Line 17 returns the overall coverage and average rule coverage.

5 EXPERIMENTAL RESULTS

To find out the overall coverage, average rule coverage and to compare our results with some existing well-known classification methods on accuracy, we performed experiments on 11 real-life datasets from the UCI Machine Learning Database Repository. We used the WEKA software to explore the classification methods and 10 times random-split method (average result is taken over 10 experiments) is used to perform experiments for both our method and other classification methods. In order to get enough rules for each class value and achieve a reasonable overall coverage, the parameter "*#Rules per class*" was set to 50 for all experiments. For other classification algorithms, Naive Bayes (NB), C4.5, PART (PT) and JRip (JR), we set up the default parameters.

Table 1. Overall coverage and average rule coverage

Dataset	# attr	# Cls	# recs	Min sup (%)	Min conf (%)	#Rules per class	Overall coverage (%)	Avg. rule coverage (%)	Accuracy (standard deviation) (%)				
									SA	C4.5	PT	JR	NB
Breast.Cancr	10	2	286	5	70	50	77.2	6.83	74.8(3.1)	72.0(3.5)	69.9(2.7)	68.9(4.4)	72.7(2.9)
Vote	17	2	435	1	80	50	93.1	28.86	95.4(2.4)	95.1(1.8)	95.5(1.4)	95.5(1.1)	89.1(1.9)
Balance.Sc	5	3	625	1	80	50	87.6	3.04	80.2(2.5)	67.2(2.4)	77.3(3.2)	77.4(2.0)	91.9(2.2)
Car.Evn	7	4	1728	0.8	70	50	76.2	7.14	81.4(2.8)	89.5(1.5)	95.0(1.5)	83.4(2.5)	84.8(0.9)
Tic-tac-toe	10	2	958	3	80	50	71.9	2.67	84.4(2.4)	84.7(3.2)	89.3(2.8)	97.5(0.6)	69.9(1.9)
Nursary	9	5	12960	2	60	50	98.0	3.78	88.6(2.6)	96.2(0.4)	98.7(0.4)	95.9(0.3)	90.4(0.4)
Hayes	6	3	160	1	50	50	100.0	5.56	80.1(7.1)	76.0(4.2)	73.3(7.7)	79.3(5.5)	79.7(7.9)
Mushroom	23	2	8124	20	80	50	84.4	4.76	68.2(1.6)	68.1(0.8)	64.3(0.7)	68.8(2.9)	69.7(0.5)
Lymp	19	4	148	3	70	50	81.0	18.76	75.3(6.4)	80.0(3.6)	79.0(6.9)	81.0(6.7)	85.1(4.1)
Monks	7	2	554	1	70	50	93.0	2.93	94.3(2.2)	98.4(2.7)	98.4(2.4)	98.4(2.2)	96.2(2.0)
Spect	23	2	267	0.5	60	50	81.4	27.21	78.6(3.1)	70.6(2.3)	67.1(5.3)	70.2(3.3)	69.9(4.1)
Average							85.8	10.14	81.9(3.3)	81.6(2.0)	82.5(3.2)	83.3(2.9)	81.8(2.6)

By analyzing the table of results (Table 1) we can observe that our classifier achieved better average accuracy than C4.5 and Naïve Bayes (81.9, 81.6 and 81.8 respectively). Standard deviations were higher for all methods on "Hayes" and "Lymp" datasets, that is, the differences between accuracies fluctuated and were reasonable high in 10 times random-split experiments. The overall coverages were lower than 80% on "Breast cancer", "Car evaluation" and "Tic-tac-toe" and in those cases also the accuracy is slightly worse than that of the "classical" classifiers. On almost all other datasets our method achieves similar or slightly better accuracy. On the other hand, average rule coverage is surprisingly high on "Vote", "Lymp" and "Spect" datasets, but seems to have no effect on classification accuracy.

6 CONCLUSION AND FUTURE WORK

Our comparison on selected 11 UCI ML datasets shows that with decreasing overall coverage our proposed method tends to get slightly worse classification accuracy than the "classical" classification rule learning algorithms. This fact is not surprising, since uncovered examples get classified by the majority classifier. When the overall coverage is above 85%, the accuracies of our classifier is similar or (on some datasets) ever better then Naive Bayes and C4.5. On the other hand, the average rule coverage of our proposed method seems to have no effect on classification accuracy.

This research shows that overall rule coverage should be considered when selecting (pruning) the "appropriate" class association rules which we plan to implement in future research.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the European Commission for funding the InnoRenew CoE project (Grant Agreement #739574) under the Horizon2020 Widespread-Teaming program and the Republic of Slovenia (Investment funding of the Republic of Slovenia and the European Union of the European Regional Development Fund).

REFERENCES

[1] Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: VLDB '94 Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487-499. Chile (1994).
[2] Baralis, E., Cagliero, L., Garza, P.: A novel pattern-based Bayesian classifier. IEEE Transactions on Knowledge and Data Engineering 25(12), 2780–2795 (2013).
[3] Breiman, L.: Random Forests. Machine Learning 45(1), pp. 5-32 (2001).
[4] Chen, G., Liu, H., Yu, L., Wei, Q., Zhang, X.: A new approach to classification based on association rule mining. Decision Support Systems 42(2), 674–689 (2006).
[5] Clark, P., Niblett, T.: The CN2 induction algorithm. Machine Learning, 3(4), 261–283 (1989).
[6] Cohen, W., W.: Fast Effective Rule Induction. In: ICML'95 Proceedings of the Twelfth International Conference on Machine Learning, pp. 115-123, Tahoe City, California (1995).
[7] Dua, D., Graff, C.: UCI Machine Learning Repository, Irvine, CA: University of California (2019).
[8] Frank, E., Witten, I.: Generating Accurate Rule Sets Without Global Optimization. In: Fifteenth International Conference on Machine Learning, pp. 144-151. USA (1998).
[9] Hu, L, Y., Hu, Y, Han., Tsai, C, F., Wang, J, S., Huang, M, W.: Building an associative classifier with multiple minimum supports, SpringerPlus, 5:528, (2016).
[10] Li, W., Han, J., Pei, J.: CMAR: accurate and efficient classification based on multiple class-association rules. in *Proceedings of the 1st IEEE International Conference on Data Mining (ICDM '01)*, pp. 369–376, San Jose, California, USA (2001).
[11] Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. in *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD '98)*, pp. 80–86, New York, USA (1998).
[12] Quinlan, J.: C4.5: Programs for Machine Learning, Machine Learning 16(3), 235-240 (1993).
[13] Xiaoxin, Y., Jiawei, H. CPAR: Classification based on Predictive Association Rules. Proceedings of the SIAM International Conference on Data Mining, pp. 331-335, San Francisco, U.S.A (2003).