CLASSIFICATION RULE EXTRACTION APPROACH based on HOMOGENEOUS TRAINING SAMPLES

Prabhat Mahanti  
Department of CSAS  
University of New Brunswick  
Saint John, Canada  
pmahanti@unbsj.ca

D. K. Swami  
Department of CA, Samrat Ashok Technological Institute,  
Vidisha, India  
dhirendrakswami@yahoo.com

Mohammed Al-Fayoumi  
Department of Computer Science  
University of Graduate Studies  
Amman, Jordan  
mfayoumi99@yahoo.com

Abstract

This paper presents an efficient and accurate classifier construction method based on extracting class wise rules from homogeneous training data samples. Finally, rule ranking mechanism employs measure of the Intensity of Implication over traditional confidence measure.

Key words: ARM, Classification, Homogenous Training Data, Confidence, Intensity of Implication.

1 INTRODUCTION

Association Rule Mining (ARM) and classification are two important data mining tasks which have been studied extensively and a myriad of algorithm have been proposed [6]. The recent studies in data mining community proposed approaches based on integration of ARM and classification and approach initially introduced in CBA is called Associative classification (AC) [7]. The other approaches of AC such as CMAR, ADT, CPAR, CorClass, and MCAR were discussed by various authors [5, 12, 13, 15, and 11] and a survey of major AC algorithms can be found as well in [10]. It has been shown experimentally in many of the above approaches that AC in the terms of accuracy outperforms traditional classification approaches such as decision trees [8]. In AC approaches, arbitrary values of support and confidence are chosen and the issue of obtaining a suitable value of support and confidence was highlighted and suggests the strategy for tuning these parameters [2]. Associative classifiers are not robust enough to handle missing values in future objects, [3] proposes an approach to make associative classification robust.

In general the process of classification task is to build a classifier model based on a training dataset, to test the classifier on a collection of objects with known class labels and used to predict the future objects classes. Here the training dataset is a set of objects described by a set of attributes with known class labels whereas for future objects class label is unknown.

AC aims at finding a limited subset of association rules known as Class Association Rules (CARs) in which precedent of rule takes the form of attribute value pairs and consequent is restricted to take only class attribute value. From this set of CARs a subset of strongest CARs is selected to form the classifier. CBA [7] adapts the popular Apriori algorithm [1] to extract CARs from training dataset. These CARs are then sorted by descending order of their confidence and pruned to get minimal number of rules that are necessary to cover training data and achieve satisfying accuracy. Since CBA uses apriori based method it suffers efficiency due to multiple database scans involved in generating CARs. CBA uses rule confidence as rule interestingness measure. The limitations of confidence measure and improvements of CBA using ‘intensity of implication’ as a measure of rule interestingness was described in [4]. MCAR [11] approach is highly effective in run time since this employs a technique based on an intersection method [14] that needs only one database scan to discover CARs unlike multiple database scans required by most of other AC algorithms.

This paper proposes an efficient and accurate classifier construction method that mines CARs class wise from homogeneous datasets without physically partitioning heterogeneous datasets. Here we adapt MCAR’s [11] frequent item discovery method to apply it on homogeneous training data samples, that needs only one database scan to discover CARs. In order to reduce storage and execution time in CAR generation, we apply intersection method on homogeneous data samples separately to remove the cause of concern in AC. It is shown experimentally that the rule of ranking in ‘intensity of implication’ over ‘confidence’ leads to better accuracy of classifier. In some domains a penalty is caused due to misclassification of objects. We have developed an ordering of class labels to minimize this penalty. The remainder of the paper is arranged as follows: the section 2 discusses background concepts and proposed algorithm CWC (Class Wise Classification) is presented in section 3 with explanation in two parts, rule generation and rule selection. The Section 4 contains experimental results and finally section 5 presents conclusion.
2 BACK GROUND CONCEPTS

This section presents the required concepts and definitions.

2.1 Training Dataset D

Training dataset D is collection of n objects. Object in the training dataset are represented by features/attributes \( A_1, A_2, A_3, \ldots, A_m \) and each object belongs to some class in finite set of class labels. Each object is uniquely represented by a rowId in D. It is assumed that all the attributes \( A_i \) contain categorical data. Continuous valued attributes if any need to be discretized in a preprocessing step. An attribute \( A_i \) has finite domain of \( v \) values. The values are replaced by numbers as 1,2,..\( v \). The number of values in \( A_i \) and \( A_j \) (\( i \neq j \)) may be different i.e. \( v \) depends on attribute \( A_i \).

DEFINITION 2.1.1. (item). An item \( I_{ij} \) represents an attribute value pair for an attribute \( A_i \) and its \( j^{th} \) value in order in the domain of \( A_i \).

DEFINITION 2.1.2. (k-itemSet). A k-itemSet is set of k items, which always is a subset of \(<I_{1j}, I_{2j}, \ldots I_{mj}>\).

2.2 Class Association Rule (CAR)

Class Association Rules are special subsets of Association Rules. In a CAR the consequent part essentially takes a value from the set of class labels.

DEFINITION 2.2.1. (CAR). Let I be the set of all items in D and C be set of class labels. A class association rule (CAR) is an implication of the form \( X \rightarrow Y \) where \( X \subseteq I \) and \( Y \in C \). A rule \( X \rightarrow Y \) holds in D with confidence \( c \) if \( e \% \) of objects in D that contain \( X \) are labeled with class \( Y \). The rule has support \( s \) in D if \( s \% \) of the objects in D contain \( X \) and are labeled in the class \( Y \).

DEFINITION 2.2.2. (k-ruleItem). A k-ruleItem is of the form \(<k-itemSet, Y>\) where \( Y \in C \).

DEFINITION 2.2.3. (frequent k-ruleItem). A k-ruleItem is frequent if it has support above user specified minimum support.

2.3 Rule Generation in MCAR

MCAR scans the training dataset once to count the occurrences of single items to determine 1-itemSets that pass minimum support threshold constraint. It stores items along with their locations (rowIds) inside arrays. Thereby intersecting the rowIds of the frequent items discovered, remaining frequent items are obtained that involve more than one items. The rowIds of frequent single items are used to obtain support and confidence values for rule involving more than one item. To generate rules only one database scan is needed. Thus once an item has been identified as a frequent item, MCAR algorithm finds all rules with that item as condition that passes the minimum confidence threshold. For a frequent item only the rule with highest confidence value is considered.

Since the training data has been scanned only once to discover and generate the rules, this approach is highly effective in run time and storage as compare to multiple data scan approaches. However in cases where there is large number of candidate items held in main memory, the possible intersections required to generate frequent items may be tremendous. This is one drawback of MCAR algorithm, which may consume more resources like storage etc.[11]

2.4 Intensity of Implication (IoI) versus confidence measure

Confidence has been used to rank the rules in many AC algorithms including CBA and MCAR. Authors in [4] argue some of weaknesses of confidence measure for ranking classification rules in AC.

1. Given a CAR \( X \rightarrow Y \) that holds is dataset D. Its confidence defined as
\[
\frac{\text{support (}X \cup Y\text{)} / \text{support (}X\text{)}}
\]
remains invariable when total number of tuples or samples with class label \( Y \) in D, varies. Whereas the fact is that the rule will be more effective when number of objects in D with class label \( Y \) increases or D decreases.

2. Some rules may come up due to noise in data that have very high confidence value with low support. In case the minimum support is lowered to accommodate more CARs to form classifier, such rule with high confidence due to noise data may get place in classifier and lead to incorrect prediction.

The mentioned drawbacks associated with confidence measure indicate that taking confidence as a measure of quality of CARs is not the best choice.

Intensity of implication measures the statistical surprise of having so few negative examples on a rule as compared with a random draw. Suzuki et. al [9] came up with an approximate formula to calculate intensity of implication for very large datasets.

Given a CAR \( X \rightarrow Y \) that hold in D, formula for Intensity of implication (IoI) is given by

\[109\]
\[ \text{IoI} = 1 - \sum_{k=0}^{n_{ab}} \frac{\lambda^k e^{-\lambda}}{k!}, \quad \text{where} \quad \lambda = \frac{n_a(n - n_b)}{n} \]

\[ n = |D|, \quad n_a = \text{number of examples in } D \text{ that support } X, \quad n_b = \text{number of examples in } D \text{ that have class label } Y, \]

\[ n_{ab} = \text{number of examples not labeled with class } Y \text{ in } D \text{ that support } X. \]

Computation of IoI considers database size as well as examples with label Y in D. Experiments in [4] show performance & accuracy improvements in some databases in adapted CBA for IoI in place of confidence measure.

2.5 Misclassification Penalty

From the point of view of the users of the classifier (end user) misclassification can cause a penalty. This fact can be understood with the help of following examples.

EXAMPLE 2.5.1. In some scenario it is required that customers are to be classified as "potential customer" or "not a potential customer" for sale of a new product. A customer if classified as "not a potential customer" whereas this customer was a potential customer. In this type of misclassification the end user is penalized to loose the profit linked with the sale of product. In this situation not using the classifier would have been beneficial.

EXAMPLE 2.5.2. In spam detection filters, where classification algorithms play important role, following misclassification may occur

i) An email message which is actually a spam is not classified as spam and classified as a legitimate message.

ii) A legitimate message is classified as spam

In the above two classifications scenarios a recipient of message will be annoyed with second type of misclassification because a spam declared as legitimate message can be found later to be a spam. But declaring a legitimate message to be a spam may cause recipient of the message to miss out an important message and we say for the second type of misclassification penalty is more.

3 CLASSIFIER DEVELOPMENT BASED ON HOMOGENEOUS TRAINING SAMPLES

The proposed algorithm CWC (Class Wise Classification) is applied on homogeneous training dataset which has two parts, namely, CWC_rule_generation and CWC_rule_selection. We have explained it in detail in section-3 and algorithm is presented in Figure-1. Homogeneous training dataset concept and rule ranking procedures are also explained as well.

3.1 Homogeneous training samples

Object with same class label in training dataset are called homogeneous training samples. Homogeneous samples are expected to have higher affinity among them as compare to their affinity grouping with samples belonging to other classes. Due to this fact we except fast convergence of CAR generation process when applied on homogeneous training samples.

3.2 Rule Generation Procedure

The training dataset is scanned only once to count for single items. Thus for each single item \( I_j \), we take \( l \) (number of class labels in D) arrays so that array\(_{l_1}\), array\(_{l_2}\),...,array\(_{l_l}\) store rowIds of samples in D that contain single item labeled with class label 1, class label 2,...class label \( l \) respectively. Apparently, for an item \( I_j \) count of rowIds in array\(_{I_j}\) gives the support to the rule \( I_j \rightarrow l \), count of rowIds in array\(_{l_1}\) gives the support to the rule \( I_j \rightarrow 1 \) and so on, similarly count of rowIds in array\(_{l_l}\) gives the support to the rule \( I_j \rightarrow l \). Confidence and IoI measure of rules can be computed with the help of \( l \) arrays associated with item \( I_j \). After rowIds have been stored in corresponding class wise arrays for each single item, we check for each item, count of rowIds in corresponding arrays. If count in all class wise arrays for some item is below the minimum specified support then this item along with associated class wise arrays is deleted. If count in at least one class wise array is found to fulfill minimum specified support then confidence is computed for the array in which count is maximum. Hence if confidence is found maximum for class wise array\(_{l_k}\), then the rule \( I_j \rightarrow r \) is chosen and IoI for the rule is computed. Rule \( I \rightarrow r \) is inserted along with support, confidence and IoI values in the set of rules R . At the end of this step all 1-rule item are stored in R and we are left with a collection of 1-frequent item set (we call \( L_1 \)) with rowIds of their occurrence stored in class wise arrays. By intersecting the class wise stored rowIds of 1-frequent-item discovered so far we get all 2-item sets (we call \( L_2 \)) with their rowIds of occurrences stored in class wise arrays. Further \( L_2 \) to \( L_L \) and so on are computed until some \( L_k \) is found empty. In order to optimize \( L_1 \) to \( L_{L+1} \) procedure we find intermediate candidate item set Cand\(_{L_{L+1}}\) by adapting apriori's candidate generation procedure. Once \( L_i \) is computed, \( L_{i+1} \) is deleted from memory.

At the end of this process all rules will be stored in R to become the input to next phase of rule selection.

The algorithm is presented in Figure 1.
Algorithm CWC_Rule_Generation
input:D, minSupport, 1 // number of class labels
output: R // set of rules
begin
n_i = 0, i = 1,..l // number of samples with class label i
// scan D once compute Cand
for each sample d in D do
if d contains class label i
n_i++;
for each 1-itemSet in d do
insert rowId in itemSet.array;
end for
end if
end for
Cand_i = \cup all 1-itemset
Label one:// assigning statement label for goto
k = 1
for each k-itemset in Cand_k do
// compute count in each array
count = (count(rowIds in k-itemSet.array_i), i = 1,..l)
// compute maximum rowId count
maxCount = max_{rowIds(count), i = 1,..l}
// m is class label for which count is maximum
if maxCount < minSupport
delete k-itemSet from Cand_k
else
// generate rule
rule.itemset = k-itemset
rule.class = m
rule.support = maxCount
n_a = \sum count_i
n_b = n_i // samples in d for class with maxCount
n_{a*b} = n_a - maxCount;
compute rule.IoI // given n_a, n_b, n_{a*b}, n
/ k-itemset and class label m pair forms the rule
insert rule in R
end if
end for
L_k = Cand_k
// L_k to C_{k+1} procedure
C_{k+1} = \phi
for each tl k-itemSet in L_k do
for each 2 k-itemSet in L_k do
// tl(x), t2(x) represent x^{th} item in k-itemSet
if (t1(1), i = t2(1), i \wedge t1(1), j = t2(1), j \wedge t1(2), i = t2(2), i
\wedge t1(2), j = t2(2), j \wedge \ldots \wedge t1(k-1), i = t2(k-1), i
\wedge t1(k-1), j = t2(k-1), j \wedge (t1(k), i < t2(k), i
\wedge (t1(k), i = t2(k), i \wedge t1(k), j = t2(k), j))
t = t1(1), t1(2), \ldots t1(k-1), t1(k)
for i = 1,..l do
end for
Cand_{k+1} = Cand_{k+1} \cup t
end if
end for
end for
...continued

If C_{k+1} \neq \phi
goto Label one;
end if
end.

Figure 1. Algorithm CWC_rule_generation

Consider an example data D in Figure 2 with three attributes A_1, A_2 with two values each and A_3 with two values and class attribute C with two class labels. A trace of algorithm produces 1, 2 and 3 frequent items Sets L_1, L_2 and L_3 given in Figure 3. Parentheses contain i, j indices of I_{i,j} itemset and braces contain rowIds of itemsets appearing in D with class label 1 and class label 2 respectively. Item I_{11}, i.e. attribute value pair (A_1, 1) occurs in row ids \{1,2,3,4,5,6,8,11\}, in our algorithm these occurrences are stored in class wise arrays \{4,5,6\} and \{1,2,3,7,11\} for I_{11} to have class label 1 and class label 2 respectively. This representation reduces the size of array containing rowIds and hence it is expected this will lead to an early completion of frequent set mining process in view of minimum support constraint. If we intersect row ids two 1-itemSets say I_{11}, I_{32} with array \{4,5,6\} and array \{7,11\}. Similarly other computations can be traced. Taking class wise arrays of rowIds, freeing memory of L_k once L_{k+1} is found and using candidate generation procedure based on Apriori in computing L_{k+1} from L_k reduces execution time and storage requirements as compare to similar approach of MCAR.

<table>
<thead>
<tr>
<th>RowID</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 2. Example dataset
3.3 Rule Ranking

Given two rules \( r_a \) and \( r_b \), \( r_a \) is ranked higher than \( r_b \) if:

1. \( \text{IoI of } r_a > \text{IoI of } r_b \)
2. \( \text{IoI of both rules are same but support of } r_a > \text{support of } r_b \)
3. \( \text{IoI and support of rules are same but class label in } r_a \) has higher misclassification penalty.
4. If all other criteria are identical for \( r_a \) & \( r_b \) but \( r_a \) was generated before \( r_b \).

In rule ranking procedure we make our algorithm suitable for domains where misclassification can cause a penalty. If misclassification penalty is not known, all class labels are assumed to have same misclassification penalty.

3.4 Rule Selection

Rule selection procedure is based on coverage analysis. Firstly, set of rules \( R \) returned by rule generation procedure, is sorted in the order of rank as defined in section 3.3. The rules higher in the rank which classify at least one correct example in \( D \) are retained in classifier. Once classifier is ready, first it is tested for accuracy on objects with known class labels. If tests produce enough accuracy the classifier model is used for predicting class labels of new objects. Algorithm is presented in Figure 4.

```plaintext
//build classifier by coverage analysis
sort (R) // on rank order
begin
for each rule r in R in sequence do
  scan each sample in D
  if r classifies at least one example correctly
    r.selected=true
    delete all examples form D covered by r
  end if
end for
end
```

4 EXPERIMENTAL RESULTS

To investigate the performance of our algorithm CWC, we carried out experiments using tic-tac dataset taken from the UCI Machine Learning Repository. Dataset tic-tac contains Number of Instances: 958, Number of Attributes: 9 (categorical with three possible values in each) and two Class labels. The implementation of CWC was done using MATLAB. Table presented in Figure 5 shows comparison of our approach CWC on tic-tac with CBA and MCAR. Results of CBA and MCAR on tic-tac we have taken from MCAR [11], results of CWC are due to our experiments. We applied our algorithm randomly choosing 90% records from tic-tac data set for classifier building and 10% for testing classifier. Results show that with 5% minimum support we obtained 100% accuracy of classification thus outperforming other approaches. Number of rules in the classifier using our approach is also minimum (with minimum support of 3%) with 100% accuracy.

<table>
<thead>
<tr>
<th></th>
<th>Accuracy %</th>
<th>Number of rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBA</td>
<td>98.85</td>
<td>25</td>
</tr>
<tr>
<td>MCAR</td>
<td>99.76</td>
<td>26</td>
</tr>
<tr>
<td>CWC</td>
<td>100</td>
<td>29</td>
</tr>
</tbody>
</table>

Figure 5. Accuracy comparison

The above CWC results are based on intensity of implication measure (IoI) for rule ranking in place of confidence measure used in most of associative classification algorithm. For a further comparison of confidence and IoI measure we constructed classifiers with different values of supports taking IoI measure and alternatively confidence measure for rule ranking. A comparative analysis is presented in table given in figure 6. Our observation shows that IoI measure is more...
consistent for classifier construction as compared to confidence measure. We also make a new observation for different classifiers on different support values, the average IoI of rules in the classifier is an indicative of classification accuracy on test data whereas average confidence of rules in a classifier is not so well correlated with classification accuracy. This observation can be precisely observed in the graph presented in Figure 7. We have plotted the (x,y) chart from table given in Figure 6, on average IoI, accuracy pairs and on average confidence, accuracy pairs for different classifiers produced taking different support values.

<table>
<thead>
<tr>
<th>Min Sup. No. of rules</th>
<th>Avg IoI</th>
<th>Acc. %</th>
<th>No. of rules</th>
<th>Avg Conf.</th>
<th>Acc. %</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>25</td>
<td>1.00</td>
<td>100</td>
<td>29</td>
<td>0.96</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td>1.00</td>
<td>100</td>
<td>28</td>
<td>0.89</td>
</tr>
<tr>
<td>4</td>
<td>31</td>
<td>1.00</td>
<td>100</td>
<td>40</td>
<td>0.75</td>
</tr>
<tr>
<td>5</td>
<td>29</td>
<td>1.00</td>
<td>100</td>
<td>28</td>
<td>0.75</td>
</tr>
<tr>
<td>6</td>
<td>24</td>
<td>0.96</td>
<td>90.52</td>
<td>26</td>
<td>0.71</td>
</tr>
<tr>
<td>7</td>
<td>19</td>
<td>0.94</td>
<td>85.26</td>
<td>20</td>
<td>0.36</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
<td>0.92</td>
<td>82.10</td>
<td>13</td>
<td>0.84</td>
</tr>
<tr>
<td>9</td>
<td>25</td>
<td>0.72</td>
<td>67.36</td>
<td>25</td>
<td>0.81</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>0.61</td>
<td>71.57</td>
<td>24</td>
<td>0.78</td>
</tr>
</tbody>
</table>

Figure 6. Comparison of IoI and confidence

Figure 7. Accuracy versus classifier rules
average IoI/confidence chart

5 CONCLUSION

This paper proposes an efficient approach of discovering frequent itemsets generation for CAR discovery based on homogeneous training data samples, which requires only one database scan. It is shown experimentally that a measure of Intensity of Implication is more consistent to be used for rule ranking in classifier construction than commonly used measure of confidence to produce higher accuracy. This approach is also able to handle misclassification penalties, if any.

References