AC-Stream: Associative Classification over data Streams using multiple class association rules

Bordin Saengthongloun, Thanapat Kangkachit, Thanawin Rakthanmanon and Kitsana Waiyamai
Data Analysis and Knowledge Discovery Lab
Department of Computer Engineering, Faculty of Engineering, Kasetsart University.
Bangkok, Thailand
g5314550113@ku.ac.th, g5185041@ku.ac.th, thanawin.r@ku.ac.th and fengknw@ku.ac.th

Abstract— Data stream classification is one of the most interesting problems in the data mining community. Recently, the idea of associative classification was introduced to handle data streams. However, single rule classification over data streams like AC-DS implicitly has two flaws. Firstly, it tends to produce a large bias on simple rules. Secondly, it is not appropriate for data streams that are slowly changed from time to time. To overcome this problem, we propose an algorithm, namely AC-Stream, for classifying a data stream using multiple rules. AC-Stream is able to find k-rules for predicting unseen data. An interval estimated Hoeffding-bound is used as a gain to approximate the best number of rules, k. Compared to AC-DS and other traditional associative classifiers on large number of UCI datasets, AC-Stream is more effective in terms of average accuracy and F1 measurement.

Keywords—data streams classification, associative - classification, multiple class-association rules.

I. INTRODUCTION

Unlike classification from static databases, classification from data streams poses many new challenges such as ability to produce results in real time, and process data in a single pass, while using limited memory. Associative classification is a classification technique that predicts class from association rules. Compared to other classification techniques, associative classification is very accurate and easily explainable. Many associative classification algorithms have been proposed to deal with static data such as CBA [1], CMAR [2], CPAR [3] and SIM [4]. Classifying a data stream with an associative classifier is a newly explored area of research [5].

Characteristics of data streams are as follows: 1) Data come continuously and unboundedly. 2) The extracted transactions are changing from time to time [6, 7]. From these characteristics, classifying a data stream with an associative classifier poses many challenges such as how to deal with the limited size of memory, data sampling, concept drift, or accuracy for online classification. In [5], AC-DS is proposed as a new associative classification algorithm for data streams. AC-DS is based on the estimation of support threshold and a landmark window model. In the classification step, AC-DS uses single rule for predicting a new data stream. This is biased on general rule, and not appropriate for streams that are slowly changing from time to time.

In this paper, we propose AC-Stream, an algorithm for associative classification over data streams using multiple rules. Like AC-DS, AC-Stream is based on the estimation of support threshold and a landmark window model. To avoid bias on single rule prediction, AC-Stream is able to determine best k-rules for predicting unseen data. An interval estimated Hoeffding-bound is used as a gain to separate the best class from other classes to determine k number of rules. In order to show benefits of multiple rules prediction, we compare single rule prediction methods with multiple rule prediction methods in different scenarios of data streams characteristics such as prediction heuristic orders, stream block sizes and maximum itemset sizes. Moreover, we compare our novel classification method to AC-DS and other traditional associative classifiers using large UCI datasets [8]. As we shall show, AC-Stream is more effective than other methods in terms of average accuracy and average-F1 measurement.

II. PRELIMINARIES AND RELATED WORK

A. Associative Classification

We denote Associative Classification as AC. AC is composed of two main steps Rule Generation and Classifier Building. C is set of classes. Let \( C = \{ c_1, c_2, \ldots, c_l \} \).

Let \( I = \{ i_1, i_2, \ldots, i_k \} \) is set of singleton itemsets. An itemset subset of \( I \) that contains \( k \) items is called a \( k \)-itemset.

1) Rule Generation

All Class Association Rules (CARs) are generated. CAR is defined as \( X \rightarrow c \) where \( X \subseteq I, c \in C \) and \( X \cap \{ c \} = \emptyset \). \( X \) is an itemset and \( c \) is a class.

2) Classifier Building

Building a classifier based the generated CARs from the previous step. To build a classifier, a subset of CARs is selected based on some heuristic orders such as confidence and coverage. Rules are ranked based on precedence order.

B. Associative Classification over Data Stream

Data streams can be defined as follows. A transaction data stream \( D_S = \{ B_0, B_1, \ldots, B_N \} \) is an infinite sequence of ordered blocks. Each block \( B_i = \{ T_1, T_2, \ldots, T_k \} \) is set of transactions. \( |B_i| \) is number of transaction in each block. Current total number of transactions in data streams is given as \( DL = |B_0| + |B_1| + |B_2| + \cdots + |B_N| \).
Compared to traditional associative classification, associative classification over data stream is an incremental mining task. For the rule generation steps, many strategies have been proposed [6] to generate all the frequent itemsets with limit amount of memory. Here, we explain a landmark window model which is based on the estimation of support count of frequent itemsets. Support of an itemset \( i \) can be estimated at different times and can be defined as follows.

\[
sup(i) + minSup \times (t1 - 1) + CL < minSup \times t + CL \tag{1}
\]

Where \( minSup \) is the minimum support threshold specified by end-user, \( t \) is current time, \( t1 \) is the first time timestamp, \( CL \) is total number of transactions of first block to current block. This support estimation constraint is also used as a decay gain for pruning frequent itemsets. That is, support of a frequent itemset is decreased by the value of minimum support each time there is an update.

Where \( minSup \) is the minimum support threshold specified by end-user, \( t \) is current time, \( t1 \) is the first time timestamp, \( CL \) is total number of transactions of first block to current block. This support estimation constraint is also used as a decay gain for pruning frequent itemsets. That is, support of a frequent itemset is decreased by the value of minimum support each time there is an update.

\[
sup(i) + minSup \times (t1 - 1) + CL < minSup \times t + CL \tag{1}
\]

Where \( minSup \) is the minimum support threshold specified by end-user, \( t \) is current time, \( t1 \) is the first time timestamp, \( CL \) is total number of transactions of first block to current block. This support estimation constraint is also used as a decay gain for pruning frequent itemsets. That is, support of a frequent itemset is decreased by the value of minimum support each time there is an update.

Where \( minSup \) is the minimum support threshold specified by end-user, \( t \) is current time, \( t1 \) is the first time timestamp, \( CL \) is total number of transactions of first block to current block. This support estimation constraint is also used as a decay gain for pruning frequent itemsets. That is, support of a frequent itemset is decreased by the value of minimum support each time there is an update.

Where \( minSup \) is the minimum support threshold specified by end-user, \( t \) is current time, \( t1 \) is the first time timestamp, \( CL \) is total number of transactions of first block to current block. This support estimation constraint is also used as a decay gain for pruning frequent itemsets. That is, support of a frequent itemset is decreased by the value of minimum support each time there is an update.

Where \( minSup \) is the minimum support threshold specified by end-user, \( t \) is current time, \( t1 \) is the first time timestamp, \( CL \) is total number of transactions of first block to current block. This support estimation constraint is also used as a decay gain for pruning frequent itemsets. That is, support of a frequent itemset is decreased by the value of minimum support each time there is an update.
Now, we explain step to approximate the value of K, which is the number of rules used to predict unseen data stream. By nature, data streams may evolve in several ways, and it is not possible to fix value K in advance. A possible solution is to separate best class and second class that significantly different and close to real bound of the sampling data. Hoeffding-bound equation is given below:

\[ \epsilon = \sqrt{\frac{\ln(\frac{1}{\delta})}{2n}} \]  

where R is range of possible values of a heuristic orders (for example, confidence’s range is between 0 and 1), \( \delta \) is a level of confidence, n is a number of observations. This method is used in (H1’s notation represents H1A; Heuristic orders Confident with Approximate K,) and (H5’s notation represents H5A; Heuristic orders Variants with Approximate K,) because these heuristic orders can find static range. The proposed algorithm is given in Fig. 3.

**AC-Stream 1.** Approximate Best K - multiple rules method

**Input:** \( U_t = \text{Unseen Data}. \)  
\( C = \text{set of predict class} \)  
\( \text{sumheuristic}[i] = \text{sum heuristic orders class } i \)  
\( \text{heuristicType} = \text{confidence or variants} \)  
\( \text{sumSupport}[i] = \text{sum support class } i \)  
\( \text{tempTopk}[i] = \text{temp top-k class } i \)  
\( \text{Observation} = \text{number of observation} \)  
\( \text{Range} = \text{Range of possible value} \)  
\( \delta = \text{level of confidence (default 0.000001)} \)

**Output:** the predict class of unseen data

1. Do
2. IF (no CARs satisfied unseen data)  
3. Return(bestClass)
4. For each Ci in C
5. tempRule = getTopKSatisfiedRule(Ci, k);
6. IF (tempRule is not null)
7. sumheuristic[Ci] = sumheuristic[Ci] + \( r_k \). heuristicType
8. sumSupport[Ci] = sumSupport[Ci] + \( r_k \). support
9. tempTopk[Ci] = k.next
10. Continue; // skip to another class
11. EndIF
12. EndFor
13. Range = \( k^{*} \text{(range of heuristicType)} \)  
14. bestClass = class Ci in C with highest heuristicType
15. 2ndCi = class Ci in C with second-highest heuristicType
17. Compute \( \epsilon \) using equation 3; //
18. IF (sumheuristic[bestClass] - sumheuristic[2ndCi] > \( \epsilon \))
19. Return bestClass // predict class
20. EndIF
21. k++;
22. Endwhile

**Table II.** Heuristic orders used for multiple rules prediction

<table>
<thead>
<tr>
<th>Name</th>
<th>Prediction function</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confidence</td>
<td>( \text{conf}(R_c) = \sum_{i} \text{conf}(r_i) \land \forall r \in R_c )</td>
<td>(H1)</td>
</tr>
<tr>
<td>C-Coefficient</td>
<td>( E(R_c) = \sum_{i} \log \left( \frac{\text{conf}(r_i)}{1-\text{conf}(r_i)} \right) \land \forall r \in R_c )</td>
<td>(H2)</td>
</tr>
<tr>
<td>( cE(R_c) = \text{conf}(R_c) \land E(R_c) )</td>
<td>(H3)</td>
<td></td>
</tr>
<tr>
<td>Weight of evidence</td>
<td>( E(R_c) = \sum_{i} \log \left( \frac{\text{conf}(r_i)}{1-\text{conf}(r_i)} \right) \land \forall r \in R_c )</td>
<td>(H4)</td>
</tr>
<tr>
<td>( cE(R_c) = \text{conf}(R_c) \land E(R_c) )</td>
<td>(H5)</td>
<td></td>
</tr>
<tr>
<td>Variants</td>
<td>( \text{conf}(R_c) \land vE(R_c) )</td>
<td>(H6)</td>
</tr>
<tr>
<td>( \text{conf}(R_c) \land cvE(R_c) )</td>
<td>(H7)</td>
<td></td>
</tr>
</tbody>
</table>

**Fig. 3.** Overview Approximate Best K-multiple rules Method

If there is a significantly difference between the classes with the highest and second-highest heuristic order values, then predict the class with highest one. Otherwise, find \( r_{i+1} \). If unseen data isn’t satisfied by any CARs, return majority class.

**Fig. 4.** Example of Approximate Best K Method
IV. EXPERIMENT RESULTS

We compare our proposed method AC-Streams with AC-DS [5] (confidence-based single rule prediction, noted by U1K1) and its variations (such as Chi-Square), and classifiers on static data: CBA [1] and CMAR [2]. Table III shows benchmark datasets from UCI machine learning repository as well as the size of data block for each dataset.

All the algorithms are implemented based on LUCS-KDD libraries [13]. CBA and CMAR are trained using all data with parameters: \( \text{Chi-Square threshold} = \chi^2_{0.05} \), and \( \text{cover} = 4 \). We set minimum support = 0.01 and minimum confidence = 0.5. Additional parameters are shown in Table IV.

### Table III. Major characteristic of UCI Datasets

<table>
<thead>
<tr>
<th>Datasets</th>
<th>#target classes</th>
<th>#examples</th>
<th># attr.</th>
<th>#block size</th>
<th>#max size item</th>
</tr>
</thead>
<tbody>
<tr>
<td>adult</td>
<td>2</td>
<td>48842</td>
<td>15</td>
<td>2000</td>
<td>3</td>
</tr>
<tr>
<td>chessKRvK</td>
<td>18</td>
<td>28056</td>
<td>7</td>
<td>1000</td>
<td>5</td>
</tr>
<tr>
<td>led7</td>
<td>10</td>
<td>3200</td>
<td>8</td>
<td>200</td>
<td>5</td>
</tr>
<tr>
<td>letRecorg</td>
<td>26</td>
<td>20000</td>
<td>17</td>
<td>1000</td>
<td>4</td>
</tr>
<tr>
<td>mushroom</td>
<td>2</td>
<td>8124</td>
<td>23</td>
<td>200</td>
<td>3</td>
</tr>
<tr>
<td>nursery</td>
<td>5</td>
<td>12960</td>
<td>9</td>
<td>200</td>
<td>5</td>
</tr>
<tr>
<td>pageBlock</td>
<td>5</td>
<td>5473</td>
<td>11</td>
<td>200</td>
<td>5</td>
</tr>
<tr>
<td>PenDigit</td>
<td>10</td>
<td>10992</td>
<td>17</td>
<td>200</td>
<td>4</td>
</tr>
<tr>
<td>waveform</td>
<td>3</td>
<td>5000</td>
<td>22</td>
<td>200</td>
<td>3</td>
</tr>
</tbody>
</table>

### Table IV. Experimental set up

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>For each data block</td>
<td></td>
</tr>
<tr>
<td>Training data</td>
<td>90%</td>
</tr>
<tr>
<td>Testing data</td>
<td>10%</td>
</tr>
<tr>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>Minimum support</td>
<td>0.01</td>
</tr>
<tr>
<td>Minimum confident</td>
<td>0.50</td>
</tr>
<tr>
<td>Max size of item set</td>
<td>&lt; 5</td>
</tr>
</tbody>
</table>

A. Comparison between multiple rule & single rule prediction on average accuracy and average F1 for all data sets and methods.

To avoid a bias heuristic orders, same heuristic orders are implemented for both multiple rules (M) and single rule (S) prediction. Fig. 5 shows performance comparison between AC-Stream (M) and AC-DS (H1S) in terms of average accuracy on all datasets and heuristic orders. The y-axis shows average accuracies and the x-axis shows different heuristic orders. Experimental results show that best average accuracy (72.74) is obtained with multiples rules method. Fig. 5

On data streams, best average accuracy from AC-DS is 71.76, which is obviously less the accuracies obtained from static classifiers such as CBA and CMAR (71.92 and 72.26 respectively). However, with our proposed AC-Stream method using multiple rules, we obtained the best accuracy as 72.74. This observation is much more evident if classification accuracy is evaluated using average F1-measure. Fig. 6 shows result comparisons of the results obtained by all the methods. Average F1-measure of AC-streams is 60.50, which is much higher than CBA and CMAR (55.05 and 55.82 respectively).

B. Accuracy of each individual Dataset

Datasets’ names on graph header are described as follows, e.g., waveformB200I3; waveform is dataset name, B200 is number of data per block. I3 is max length of antecedent. Training data split 90% and test data holdout 10% in each block.

![Fig. 5. Comparison between multiple rules (M) and single rule prediction (S) on average accuracy for all datasets & all heuristic orders](image)

![Fig. 6. Average average-F1 measure of all methods from all datasets](image)

![Fig. 7. average Accuracy all methods on waveform datasets](image)
Here, objective is to evaluate performance of all the methods for some specific characteristics of the datasets. Two major characteristics: incremental and imbalanced datasets are analyzed. AC-streams gives better accuracy than AC-DS and all of its variations (i.e. using other heuristic orders than just confidence), as shown in Fig 7, 8 and 9.

C. Accuracy and F1-measure of our method

It has been recognized that associative classification is one the most competitive classifiers because it can support large number of classes.

Single rule prediction methods such as CBA are best when one rule can perfectly explain an unseen data. However, these approach biases on general rules, especially when those rules have very high heuristic order score. Consequently, classification accuracy of the single rule classifiers is very poor on evolved data streams. To the best of our knowledge, this paper firstly introduces a multiple rules associative classification over data streams and shows that multiple rules are more suitable for prediction on evolved data streams.

From experimental results in Fig 5-9, multiple rules prediction exhibits better performance than single rule prediction for almost of the datasets with all the heuristic orders. However, the best performance depends on the quality of the set of rules in a classifier. It is very hard to set the number of rules since this depends on the characteristics of datasets. In this paper, we proposed one feasible method to approximate the good number of rules based on an interval estimated Hoeffding-bound.

Fig. 10 and 11 shows experimental results in terms of accuracy and average F1-measure when best k number of rules are approximated for each individual dataset. We notice that almost of the time multiple rule prediction is better than single rule prediction. Nevertheless, single rule prediction is better in some cases. Anyway, when approximated K is used, multiple rule prediction is almost always better.

V. CONCLUSTION AND FUTURE WORK

In this work we present AC-Stream, an algorithm for associative classification over data streams using multiple rules. AC-Stream is based on the estimation of support threshold and a landmark window model. To avoid bias on single rule prediction, AC-Stream is able to determine k-rules for predicting unseen data. An interval estimated Hoeffding-bound is used as a gain to separate the best class from other classes to approximate K number of rules. We compare AC-Stream with single-rule prediction method on different characteristics of 9 large datasets from UCI-Datasets. AC-Stream exhibits a better performance in terms of average accuracy and average-F1 on the 9 datasets. As a future work,
we plan to improve performance of AC-Stream on evolved data streams. Update class association rules for each new coming data stream is not necessary, a method to determine when to update class association rules is to be investigated.

REFERENCES


