Associative Classification using Automata with Structure based Merging

Mohammad Abrar[1]
Department of Computer Science
Bacha Khan University Charsadda
KPK, Pakistan

Alex Tze Hiang Sim[2]
Department of Information Systems
Faculty of Computing
Universiti Teknologi Malaysia
81310 Johor Bahru, Johor

Sohail Abbas[3]
Department of Computer science
College of Sciences
University of Sharjah, UAE

***Abstract*—Associative Classification, a combination of two important and different fields (classification and association rule mining), aims at building accurate and interpretable classifiers by means of association rules. The process used to generate association rules is exponential by nature; thus in AC, researchers focused on the reduction of redundant rules via rules pruning and rules ranking techniques. These techniques take an important part in improving the efficiency; however, pruning may negatively affect the accuracy by pruning interesting rules. Further, these techniques are time consuming in term of processing and also require domain specific knowledge to decide upon the selection of the best ranking and pruning strategy. In order to overcome these limitations, in this research, an automata based solution is proposed to improve the classifier's accuracy while replacing ranking and pruning. A new merging concept is introduced which used structure based similarity to merge the association rules. The merging not only help to reduce the classifier size but also minimize the loss of information by avoiding the pruning. The extensive experiments showed that the proposed algorithm is efficient than AC, Naive Bayesian, and Rule and Tree based classifiers in term of accuracy, space, and speed. The merging takes the advantages of the repetition in the rules set and keep the classifier as small as possible.**

***Keywords*—Associative classification; automata; ranking and pruning; rules merging; classification**

I. Introduction

Classification considers to be one of the main pillars in DM and ML [1, 2]. It is a data analysis technique, used to categorize data into different classes based on some common characteristics or associations in the data. Generally, classification consists of two basic steps i.e. a) preparation of classification model - Classifier, from available data (training dataset) and b) Classification - the prediction of unknown class label, based on the classifier model.

Classification is sometime called a predictive data mining technique due to its predominant applications in predictive domains. It is successfully applied in health care and biomedicine [3], geographical information systems, [4], marketing [5], agriculture [6], risk management [7], web traffic prediction [8], and the list continues. The broader applications of classification and its importance in some areas, particularly in health, finance, and agriculture became the motivating factors for this research.

DM contains a rich set of classification models; specifically, Support Vector Machine [9], Rule Based [10], Decision Tree [11, 12], Bayesian classification [2], k - Nearest Neighbor [13], and AC [14]. Among all, AC is relatively new and promising [15, 16, 17, 18, 19, 20, 21, 22, 23, 24] as it combines the best approaches of association rules mining (ARM) and classification. AC is based on ARM where, first, the strongest Class Association Rules (CAR) are discovered from dataset, followed by converting those rules into classifier model. Those stronger associations from the data, in the form of CAR, make the classifier more logical and improve accuracy.

After the introduction of AC in 1997, numbers of algorithms are developed in this family e.g. CBA [14, 25], CMAR [18], CPAR [21], MCAR [19], MAC [26], CMARAA [27], MRAC & MRAC+ [15], DAC [23], CBA-Spark and CPAR-Spark [24] and G3P-ACBD [28]. Almost all consist of three basic steps - a) Association rule generation, b) Classifier building - rule pruning and rule ranking c) Classification of unknown records using the classifier.

CAR, used by AC is a variation of ARM whose right hand side (consequent) is a class label instead of ordinary attribute. The ARM generates those CARs which passes minimum support and confidence threshold and classifier is then built on the basis of these CAR. AC further applies rule ranking and rule pruning to minimize the number of rules in the classifier. The reason for reduction is to make the classifier smaller which improves the efficiency of the classification. The small number of rules in the classifier is advantageous in the context of speed but is less appreciated for accuracy [17, 29, 30]. In addition to the negative effects on accuracy, these two additional steps also add their own computational overhead to the classifier.

Besides the computational overhead of rule pruning and rule ranking, AC has two other limitations. The first is the reduction of rules which generally eliminates the more significant rules and therefore may leads to reduce accuracy [17, 29, 30]. Secondly, pruning and ranking both require a detailed insight of data so as to be able to wisely decide upon the pruning and ranking criteria, e.g.: how to choose the bottom line, what should be included and what should be removed from the classifier? Further, the term “interestingness" is relative that means different things to different experts even in the same domain. All these facts make it more complicated to decide upon an appropriate pruning strategy.

Similarly, ARM generate exponential number of rules that does not make it suitable for very large dataset. Because the number of rules generation and then pruning and ranking strategies require more time which make the use of these techniques less effective [31]. Therefore, a new storage structure is necessary that can help reducing the size of dataset in order to make the processing less time consuming while improving the accuracy.

Keeping in mind these shortcomings, we propose to replace ranking and pruning with our automata based. On one side it will reduce the computational overhead of these steps while on the other side it will also minimize the loss of information by avoiding the unnecessary pruning. In order to achieve the goals of the eliminating redundancy, a new merging criteria based on structure similarity is introduced that would help to reduce the classifier size in order to improve the efficiency. Furthermore, automata based storage structure is designed to make the classification of large dataset more efficient and reduce the space allocation during classification.

The incorporation of automata observed a number of properties: including; a) its efficient structure to store data [32], b) The capability of loss-less absorbency of redundant rules, c) its sequential nature [33] to perform efficient string matching [34, 35], and the ability to deal with frequent and rare class at the same time. Therefore, the integration of automata and AC resulted in more efficient and robust for both AC as well as classification in general.

The rest of paper is organized as follows. The design of Associative Classification using Automata phase in Section II. Conflict Resolution is highlighted in Section III followed by classification of test instances in Section IV. The weighting criteria is discussed in Section V. The Section VI explains the environment and parameters used in the experiments, followed by the dataset and algorithm selected for experiments in Sections VII and VIII. Section IX deals with accuracy comparison and analysis. Finally, the complexity analysis were estimated in Section X, the paper was concluded in Section XI and future work is highlighted in Section XII.

II. Associative Classification using Automata

In this section the steps taken towards building ACA are discussed. The automata is used as a replacement of pruning and ranking phases of traditional AC algorithms. Automata also provides an efficient way for accessing and processing the test instances during classification. The following subsections highlight different aspects of the ACA.

A. Building Automata from data

This section deals with the algorithm designed to develop the automata from dataset. The task is divided into two sub task where first the CAR are generated using ARM and then automata is built on those CARs.

B. Class Association Rules Generation and Rules Pruning

Associative Classification requires CAR as a basis for classification, therefore, before building the Automata model, CAR needs to generated. In order to generate the CAR, any ARM algorithm can be used. In the current implementation of ACA, apriori is used. One of the inherited limitations of CAR is its exponential number of rules which are complex to handle efficiently, thus, after the generation of CAR, rule pruning is the mandatory phase of other techniques to eliminate the redundant or less interested rules. In ACA, the pruning and ranking phase is replaced by the use automata which has the capability to handle with the redundant rules.

C. Building Automata using CAR

A new algorithm, *Automata_Construction*, is developed and discussed in this section. The discussion and explanation of the algorithm uses Table I as an example that represents CARs for IRIS2D dataset which is available at UCI repository [36]. The CARs presented in this section are generated using Weka's Apriori algorithm with default parameters. In Table I there are a total of nine rules where each row represents one rule. Every row consists of, specifically: a rule number (Column R_No), attributes (e.g. columns PatelLength (PL) and PatelWidth (PW)) and class label (column Class). Each cell then represents the value for the attributes (the column in which the value appears in the table) for the rule in which row it is shown in the table: i.e. "4.75-max" is the value of PatelLength for R_No 5 and 9. The process of building automata automata from Table I is summarized in Algorithm 1.

The key part of the ACA is that of transition function (δ) that defines the rules of movement from one state to another. The general form of transition function is $\delta = Q \times \Sigma \to Q$ which means that one can move from any state (from Q) using any input symbol (from Σ) to any state (in Q). CARs in Table I represent δ for algorithm.

Algorithm 1 Creation of NFA from Class Association Rules - CARs

Name: Automata_Construction
Input: Set of association rules (ruleSet).
Output: Set of Automata

```
create Automata with rule1 from ruleSet and increment Level

while $ruleSet \neq \phi$ do
    read rule one by one

    if no_conflict(ruleSet, set_FA) then
        insert into Automata

    else
        Add rule to conflictRuleSet

    end if
end while
Update $ruleSet = conflictRuleSet$

Call Automata_Construction with updated ruleSet

```

In order to represent Table I as automata, it fed to the algorithm as δ for $A = \{Q, \Sigma, \delta, q_0, F\}$ where, specifically: Q is the union of set of all attributes and distinct class labels from datasets; Σ is the collection of distinct values of all attributes; q_0 is the start state and can be any attribute from set of Q. Finally F, set of final states, is the set of distinct class

TABLE I. CARs for IRIS2D dataset, generated using Weka 3.7.10

R_No	PatelLength (PL)	PatelWidth (PW)	Class
1	min-2.45		Iris-Setosa - C1
2		min-0.8	Iris-Setosa - C1
3	min-2.45	min-0.8	Iris-Setosa - C1
4		1.75-max	Iris-Virginica - C2
5	4.75-max	1.75-max	Iris-Virginica - C2
6	2.45-4.75		Iris-versicolor - C3
7	2.45-4.75	0.8-1.75	Iris-versicolor - C3
8		0.8-1.75	Iris-versicolor - C3
9	4.75-max		Iris-Virginica - C2

labels. Example 1: This example shows how Table I represents all input elements required by automata. All rules in Table I represent δ. These rules, actually, provide the mechanism of movement between different states of automata. Set of states are:
Q = {PatelLength, PatelWidth, C1, C2 and C3} (Collection of all attributes and distinct class label).
Input symbols consist of:
$\Sigma = \{”min-2.45”,\ ”2.45-4.75”,\ ”4.75-max”,\ ”min-0.8”,\ ”0.8-1.75”,\ ”1.75-max”\}$ (set of distinct values of all attributes.)
q_0 can be any attribute that starts a rule; e.g. for rule 1 $\&$ 3 q_0 is patelLength, while for rules 2 $\&$ 4, q_0 is patelWidth.
The set of final states are $F = \{C1,\ C2,\ C3\}$ (distinct class labels). The transition function for ACA has additional characteristics and some restrictions that are explained in detail below.

D. Properties of Automata in ACA

In the context of ACA the automata observed the following properties which impose some level of determinism and also helps to control the number of transition (movement from one state to other state) in δ. Further, if an attribute q_i comes before attribute q_j in the dataset, then the former is called the predecessor while the latter is the successor. It imposes sorting order of attributes and formally, this can be shown as follows:

$$q_i \prec q_j\ iff\ q_i\ preceeds\ q_j\ in\ the\ datset \tag{1}$$

Example 2: PatelLength comes before PatelWidth in the dataset, therefore PatelLength is predecessor and PatelWidth is successor. Keeping in view this assumption, Automata in ACA will also observe the following properties.

1) This property defines the start state conflict and its resolution.

$$EveryAutomata\ \ should\ have\ one\ and\ only\ one\ start\ state\ and\ it\ will\ be\ the\ first\ attribute\ of\ the\ rule. \tag{2}$$

Example 3: R_i = (PL) : $min-2.45 \rightarrow Iris-Setosa - C1$
R_j = (PW) : $min-0.8 \rightarrow Iris-Setosa - C1$
R_i and R_j have different first attributes; therefore they cannot be absorbed by a single automaton and accordingly creates a separate automaton for each rule.

2) From any state from Q, using any input symbol from Σ, a transition can be made to any successor state (except final state(s)) i.e.

$$\delta = q_i \times \Sigma \rightarrow q_j \mid q_i \in Q \,\Lambda\, q_i \prec q_j \,\Lambda\, q_j \in (Q-F) \tag{3}$$

This property allows ‘forward only’ moves and this restriction is used to control the increase in number of transitions.
Example 4: From state PatelLength the transition can be toward PatelWidth, but the reverse is a violation of this Property (3).
In case, if the transition is toward final states, then ACA maintains the following sub property.

a) There should be no two transitions in automata where two different final states can be reached by the same input from the same state. i.e.

$$\nexists\ \mathrm{q} \times \alpha \rightarrow f_i \,\Lambda\, \mathrm{q} \times \alpha \rightarrow f_j \mid q \in Q;\ \alpha \in \Sigma;\ f_i\ and\ f_j \in F \,\Lambda\, i \neq j \tag{4}$$

Example 5: If $R_i = PatelLength : min-2.45 \rightarrow C1$ then
$R_j = patelLength : min-2.45 \rightarrow C2$ is illegal (same state, same input label but different final states.) R_j is a conflicting rule and makes a separate Automata.

3) If there is a transition from state q_k on input symbol α to final state f_i and q_k is the immediate successor of f_i, then there should be no transition to non-final state on α. Formally

$$\begin{aligned} &If\ \ q_i \times \alpha \rightarrow f\, where\, f \in F \\ &then\, \nexists\, q_i \times \alpha \rightarrow F - f \end{aligned} \tag{5}$$

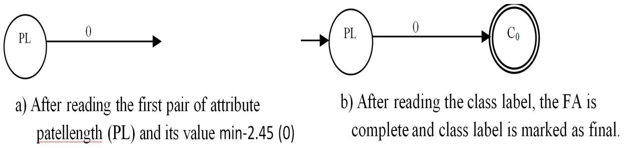
This property maintains determinism at the second last state of the automata. It reduces ambiguity which also minimizes the rate of miss-classification.

After the enlightenment of above properties and assumptions for automata, now the algorithms is explained with an example. In this example Table I is used but for the sake of simplicity, it is converted to “attribute-value" pairs as shown in Table II. In the converted form, every “even" column represents the name of an attribute and its corresponding “odd" column represents the value for that attribute. Formally, column $2i$ represents attribute i while $(2i + 1)$ represents the value for attribute i and $0 \leq i < k$ where k is the total number of attributes. Thus *col0* represents PatelLength and *col1* is its value. Similarly *col2* represents PatelWidth and *col3* is its value. The only exception is the last column which represents class label and therefore has no attribute name. The first column representing the rules ID is only for referencing purpose and it is not part of the actual table. Each cell has a short name for values, which is given inside parentheses, and will be used in figures during examples.

The *Automata_Construction* (Algorithm 1) starts from first rule by reading the first "attribute - value" pair and starts building automata. The first attribute of the rule becomes start state and its value becomes the label towards the next state. While reading next pair, the newly-read attribute becomes the next state and establishes a link with the previous label. Its value now becomes the label for the next state and the procedure continues until the class label is reached. The class label is marked as a final state which ends the first rule. The same procedure is repeated for all rules. Example 6 explains the procedure below.

Example 6: The algorithm starts from rule 1 and reads the first attribute and its value such as col0 and col1 and makes the first part of automata, as shown in Fig. 1(a).

Next, col2 and col3 are empty; therefore the algorithm reads the last column and makes it the final state as shown in Fig. 1(b).



a) After reading the first pair of attribute patellength (PL) and its value min-2.45 (0)

b) After reading the class label, the FA is complete and class label is marked as final.

Fig. 1. Automata construction for Rule 1 from Table II

Algorithm 1 is repeated for all rules and every rule is added to the existing automata one by one. During the construction, the algorithm checks for the fulfillment of the above mentioned properties. In the case of violation, the rule that results in violation is marked as a conflicting rule and it becomes a candidate for a new automaton. The process continues until there are no more rules in the rule set. The step wise automata is shown in Fig. 2 to 6.

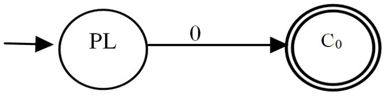


Fig. 2. Automata 0 after reading rule 1 from Table II

When the algorithm reads rule 2, it will starts from PW which is different from the start state of Automata 0 (violation of Property (1)). Therefore, Rule 2 will make new Automata as shown in Fig. 3.

After reading Rule 3, the algorithm will compare it with Automata 0. There is no conflict (i.e. the rule is not unique) and therefore Automata 0 will absorb rule 3 (Fig. 4).

Similarly when rule 4 is read, it violates Property (1) (i.e it is unique rule), therefore it will compare the rule with Automata 1, and provided there is no conflict, rule 4 will be added to Automata 1 (Fig. 5)

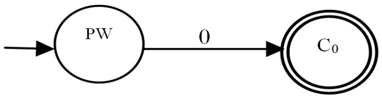


Fig. 3. Automata 1 after reading rule 2 from Table II

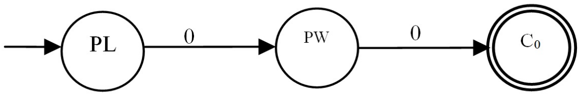


Fig. 4. Automata 0 after reading rule 3 from Table II

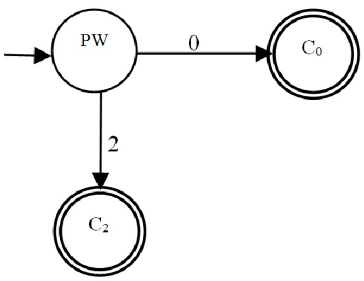


Fig. 5. Automata 1 after reading rule 4 from Table II

The same process will be repeated for all nine rules and, every time, the comparison will take place from Automata 0 onwards. The Automata with no conflicts will absorb the incoming rule; otherwise Algorithm will construct new Automata for the rule. The stepwise Automata are summarized in Fig. 6.

III. Conflict Resolution

In order to handle conflicts, every rule starts comparison with the first automata and try to absorb that rule. If the rule violates any property explained in Section II-D that specify conflicts, algorithm checks it with the next level automata and so on until either one of the existing automata absorbs the incoming rule or new automaton is built for that rule. Conflict resolution at this stage results in building new automata for the conflicting rule. At this stage the ACA ensures to shrink the redundant rules and absorb into similar automata.

IV. Classification of Test Instances

Finally, the test instances are used for classification using the new model. ACA_Classifier is used for prediction of unlabeled instance to label it with correct class. The classification algorithm is summarized in Algorithm 2.

During classification, there are two possibilities. If there is only one automaton in model, then the procedure is simple and all the test instances are tested against the single automaton. While in the case of multiple automata, the classification starts from the higher level of automaton. For example, the classification of IRIS2D automata of Fig. 6 starts from automata 1 and if it fails to classify the test instance, or the Weighted

TABLE II. CONVERTED FORM OF CARS (TABLE I) INTO "ATTRIBUTES-VALUE" PAIRS

Rule_No	col0 (att1)	col1(Val1)	col2(Att2)	col3(Val2)	col4(Class_label)
Rule 1	PatelLength (PL)	min-2.45 (0)			Iris-Setosa (C0)
Rule 2			PatelWidth (PW)	min-0.8 (0)	Iris-Setosa (C0)
Rule 3	PatelLength (PL)	min-2.45 (0)	PatelWidth (PW)	min-0.8 (0)	Iris-Setosa (C0)
Rule 4			PatelWidth (PW)	1.75-max (2)	Iris-Virginica (C2)
Rule 5	PatelLength (PL)	4.75-max (2)	PatelWidth (PW)	1.75-max (2)	Iris-Virginica (C2)
Rule 6	PatelLength (PL)	2.45-4.75 (1)			Iris-versicolor (C1)
Rule 7	PatelLength (PL)	2.45-4.75 (1)	PatelWidth (PW)	0.8-1.75 (1)	Iris-versicolor (C1)
Rule 8			PatelWidth (PW)	0.8-1.75 (1)	Iris-versicolor (C1)
Rule 9	PatelLength (PL)	4.75-max (2)			Iris-Virginica (C2)

Ratio (WR) (explained in section V) is less than 100 percent, it checks the test instance against Automata 0. Thus, if there are n automata, the comparison will start from $Automata_n$, in case of failure or smaller WR, the next comparison will be against $Automata_{n-1}$ and the process will continue till $Automata_1$. In case of less WR at the last automata, the class label with the highest WR will be selected as a class label. Weighted ratio calculation is explained in next section.

Algorithm 2 Classification of test dataset

Name: ACA_Classifier
Input: Set of Automata
testData
Output: Classified Dataset

```
while $testData \neq \phi$ do
    read rule one by one

    $acc = 0.0$

    $level = total\_autumata$

    while $Level\ >\ 0\ \&\ acc\ != \ 100\%$ do
        while $rule \neq \phi$ do
            if attribute-value pair matches then
                increment hit by 1

            else
                if Attribute or value does not match then
                    increment mis by 1
                end if
            end if
            Read next pair

        end while
        $ratio = (hit/totalStates) * (hit + totalStates) * 2$

        if $ratio > acc$ then
            $acc = ratio$

            $testClassLabel = autmataClassLabel$

        end if
        Decrement Level by 1

    end while
end while
```

V. WEIGHTED RATIO MEASUREMENT

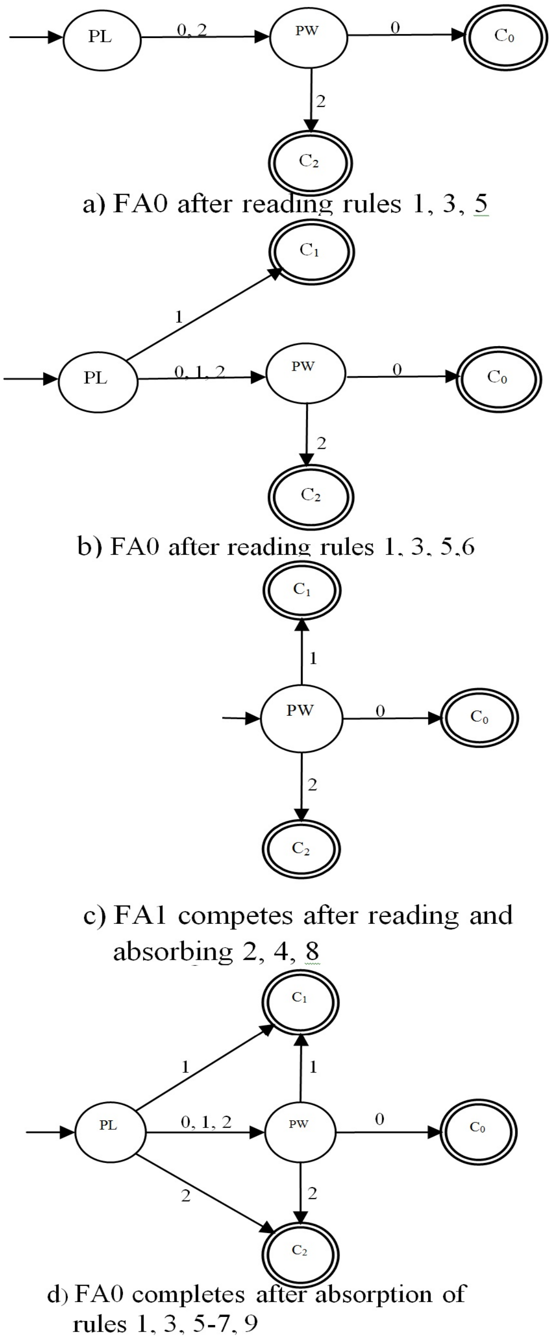
The classification of test instances is based on the "attribute-value" pairs, similar to the one used during automata construction. Algorithm 2 verifies both attribute and its value to a single state. If both match, the process then moves to the next state and reads the next pair from the test instance. Every match in automata increments "hit", while mismatches incur an increment of a "miss" variable. At the time of classification, algorithm check the ratio between "hit - miss" and the highest WR is considered the predicted class for the instance.

The WR is defined by Equation No (6). This equation gives a high weight to those automaton with maximum number of matching states. For example, for three hits out of six states, automata will give a high ration as compared to 2 hit out of 4 states; while the average for both is 0.5. Here the focus is on the maximum number of matches instead of the maximum average where 5 out of 10 and 10 out of 20 are equal using a simple average. However, Eq (6) will give preference to 10 out of 20 due to maximum numbers of correct matches. It results in a stronger classification as a consequence of high number of similarity with respect to the number of attributes. Further, if the numbers of hits are the same, then it gives preference to a lesser number of mismatches; therefore, in calculating weight, Eq (7) will increase the weight of low miss-count and decrease that of the high miss-count.

$$Ration\ =\ \omega\ +\ hit\_counts \qquad (6)$$

$$\omega\ =\ \left(\frac{1}{miss\_count}\right) * \left(\frac{1}{total\_attributes}\right) \qquad (7)$$

Example 7: Table III and Table IV correspond to contact lenses and dermatology datasets, respectively. In each table, row represents the outcome of comparison of test instance with one automaton. Column 1 (S No) is an identifier for each automaton while column 4 (Total) represents the total number of states in that automaton. In Table III, a test of test-instance is shown with 6 different Automata. The table also lists the hit and misses counts. According to Eq (6), the algorithm will select Q1 because it has the maximum number of hits and a lesser number of misses as compared to Q3. In this case, the simple average will also select Q1 because the average is highest among all; despite the issue that a single hit of Q5 is given preference over 2 hits of Q6. Now consider Table IV: the simple average will now select Q1 as a result of the highest average and will ignore Q2 where 14 attributes match out of 16. However, ACA will choose Q2 because it gives the highest ration.



a) FA0 after reading rules 1, 3, 5

b) FA0 after reading rules 1, 3, 5,6

c) FA1 competes after reading and absorbing 2, 4, 8

d) FA0 completes after absorption of rules 1, 3, 5-7, 9

Fig. 6. Automata 0 and Automata 1 after reading all rules from Table II. Figures c) and d) are final Automata for the classification Algorithm

VI. Results and Discussion

The ACA was implemented using Java 7, on Windows 8.1 running over 64bit, core i7 2.6GHz machine with 8GB of memory. The results of other techniques are generated using Weka 3.7.13 on the same system with the above specifications. In order to simplify the regeneration of results in Weka, all the experiments were conducted with default parameters or otherwise explained in the corresponding sections. The numeric datasets were discretized using using unsupervised discretizer of Weka, because most of the classification techniques did their experiments on the same grounds.

TABLE III. Partial Automata from Contact Lenses dataset for weighted class label comparison (Total No of Attributes: 4)

S No	Hit	Miss	Total	Eq (I)	Average
Q1	3	1	4	3.2	0.75
Q2	2	1	3	2.2	0.667
Q3	3	2	5	3.1	0.6
Q4	2	2	4	2.1	0.5
Q5	1	1	2	1.2	0.5
Q6	2	3	5	2.067	0.4

TABLE IV. Partial Automata from dermatology dataset for weighted class label comparison(Total No of Attributes: 16)

S No	Hit	Miss	Total	Eq(I)	Average
Q1	8	1	9	8.063	0.889
Q2	14	2	16	14.031	0.875
Q3	2	1	3	2.063	0.667
Q4	3	2	5	3.031	0.6
Q5	4	3	7	4.021	0.571
Q6	5	4	9	5.016	0.556

VII. Selection of datasets for Experiments

The most commonly cited datasets [18, 19, 21, 26, 37, 38] are selected for the experiments. These all datasets are available at UCI Machine Learning Repository [36]. In order to show the effect of new technique on different dataset, the datasets were divided into four categories, namely: a) Small Discrete Datasets, b) Large Discrete Datasets, c) Small Continuous Datasets, and d) Large Continuous Datasets. The discrete and continuous devision is based on the nature of data types i.e. Categorical or Numeric, while the distribution into small and large datasets is provided for the purpose to explain the effects of ACA on the smaller and larger datasets. The following sub-sections explain these categories in detail.

A. Small Discrete Dataset

Small discrete datasets include those datasets which produce less than are equal to 50,000 association rules. The reason behind setting the limit to 50,000 is that this number of association rules are taking much longer time on the system with small memory i.e. 2GB or 4GB and in some cases may lead to system halt. The experiments during association rule generation showed that in general when the number of attributes are 9 or more, then the numbers of association rules exceed 50,000 under minimum threshold for support and confidence for rules generation are set to 1% and 100% respectively. The datasets in this category are displayed in Table V. The table shows that there are datasets with 2 to 4 classes while the number of instances in datasets ranges from 12 to 1728. Similarly, the attributes ranges from 5 to 7.

B. Large Discrete Dataset

The large dataset in discrete category contain those datasets which either generate more than 50,000 rule under the threshold of 1% support and 100% confidence or the number of

TABLE V. DETAILED DESCRIPTION OF SMALL DISCRETE DATASETS

Dataset Name	Attributes	Instances	Class
weather.nominal	5	14	2
balloons	5	20	2
contact-lenses	5	24	3
shuttle-landing-control	7	15	2
car	7	1728	4

attributes are 9 or more. This category includes Vote, Zoo, Tic-Tac-Toe, Postoperative Patient Data, Breast Cancer, Nursery, and Mushroom. The detailed description is shown in Table VI. This category has the datasets where the attributes ranges from 9 to 23 while the number of instances are as minimum as 90 the maximum of 12960. Furthermore, these datasets also have more classes which ranges from bi-class problem to 21classes.

TABLE VI. DETAILED DESCRIPTION OF LARGE DISCRETE DATASET

Dataset Name	Attributes	Instances	Class
postoperative-patient-dat	9	90	3
nursery	9	12960	5
breast-cancer	10	286	2
tic-tac-toe	10	958	2
zoo	17	101	2
vote	17	435	2
primary-tumor	18	339	21
Mushroom	23	8124	2

C. Small Continuous Datasets

This category consists of those datasets that have any number of attributes with continuous/numeric data types. These datasets were discretized with the Weka built in feature i.e. unsupervised discretizer. Then the similar criteria of the number of rules were applied to distinguish between small and large datasets. Those datasets which are able to generate 50,000 or less rules are included in this category. The datasets in this category are IRIS2D, IRIS4D, Balance Scale, TEA, Hayes Roth, Weather Nominal, Data Banknote, and Liver Disorder. Table VII shows the detailed description of these datasets. The number of rows in this category in the similar range of small discrete dataset. bi-class and multi-class, both typoe of datasets are included in this category. Similarly, the number of instances also ranges from 14 to 1373.

TABLE VII. DETAILED DESCRIPTION OF CONTINUOUS SMALL DATASETS

Dataset Name	Attributes	Instances	Class
iris2D	3	150	3
weather-numeric	5	14	3
hayes-roth	5	28	4
iris	5	150	3
balance-scale	5	625	3
data_banknote	5	1373	2
tae	6	151	3
liver-disorders	7	345	2

D. Large Continuous Datasets

The final category consists of large dataset with continuous data types. The datasets which have any number of attributes with numeric data type and have the ability to generate more than 50,000 association rules under the threshold of 1% support and 100% confidence are included in this category. This includes Diabetes, Breast Cancer, CMC, Page Block, Labor, Heart-C, Heart-h, Anneal, Ionosphere, Dermatology, Glass, Hepatitis, and wine. Table VIII shows the detailed description of large continuous datasets. The table can show that this category is comparatively diversified in terms of attributes, number of classes and number of rows. The number of attributes in this category are as many as 39 which are considered quite high for AR algorithms. Similarly, both bi-class and multi-class problems are included in this category.

TABLE VIII. DETAILED DESCRIPTION OF LARGE CONTINUOUS DATASETS

Dataset Name	Attributes	Instances	Class
diabetes	9	768	2
Glass	10	214	6
w-breast	10	699	2
cmc	10	1473	3
page-blocks	11	5473	5
Wine	14	178	10
hungarian-14-heart-diseas	14	294	2
heart-c	14	303	2
labor	17	57	2
Hapititus	20	178	2
Dermatology	35	366	6
Ionosphere	35	366	2
Anneal	39	898	5

VIII. SELECTION OF ALGORITHMS FOR COMPARISON

ACA was compared with three classes of classification algorithms. The first is obviously based on association rules, so as to give comparison with its own family members. The algorithms in this class are CBA [25] and LAC[39]. The second class of algorithms is rule-based and Tree Based. This class of algorithms is similar in nature with AC and have J48 [12], CART [40], BFTree [40], Jrip (Ripper) [41], PART [42], DecisionTable, and ZeroR. Finally, Average One Dependence Estimator (AODE) [43], A2DE [44], Naive Bayes, and Bayes Net [45] is selected from Bayesian Family which considers a family of the most prominent classifiers with reasonably high accuracy.

IX. COMPARISON OF ACA WITH OTHER CLASSIFIERS BASED ON ACCURACY

This section deals with the comparison of ACA with different classifiers based on accuracy. The comparison is provided with Associative Classifiers, followed by Rules and Tree based classifier, and finally with Naive Bayesian family of classifiers.

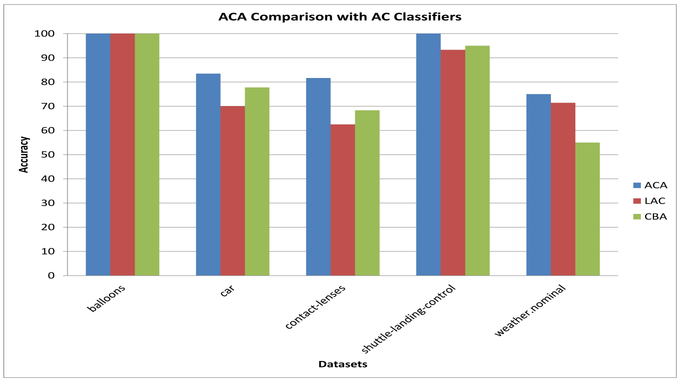


Fig. 7. ACA Comparison with AC Classifiers on small discrete datasets

A. Accuracy Comparison of ACA with Associative Classifiers

The accuracy comparisons with other associative classifiers are discussed in detail in this section. Two classifiers are selected i.e. CBA and LAC from AC classifiers. CBA is the most commonly referenced associative classifier and the latest implementation is used in these experiments. Lazy Associative Classifier is a based on maximum number of rules as compare to CBA. The ACA claims that additional rules might increase the accuracy therefore LAC was chose for comparison with ACA to give insight of rules' effects on accuracy. The discussion is divided into four section where each section discusses individual category of dataset for accuracy analysis.

1) Accuracy comparison with small discrete datasets: In this section, the analysis of ACA with CBA and LAC over small discrete datasets are presented. Table IX and Fig. 7 shows the results of the all three classifiers. The table shows that ACA is performing better than CBA and LAC in most of the datasets. The reason for high accuracy is that ACA is using large number of rules as compare to other classifiers which leads to more accurate results. Secondly, because the datasets are relatively small, the generated rules set is comparative small and complete.

TABLE IX. Accuracy Comparison of ACA with Associative Classifiers over small discrete datasets

Dataset Name	ACA	LAC	CBA
balloons	**100.00**	**100.00**	**100.00**
car	**83.49**	70.02	77.78
contact-lenses	**81.67**	62.50	68.33
shuttle-landing-control	**100.00**	93.33	95.00
weather.nominal	**75.00**	71.42	55.00

2) Comparison with small Continuous datasets: Next, The ACA is compared with small continuous datasets which were discretized using Weka's discretizer utility. Table X and Fig. 8 presents the results of the experiments. It shows that ACA outperform the AC classifiers in most cases while in balance scale and weather numeric datasets ACA did not perform well. Although in general ACA loses in two dataset but the win loss ratio is 7/8 with individual algorithms where ACA was surpassed by Balance Scale and Weather Numeric by LAC and CBA, respectively.

TABLE X. ACA Comparison with AC Classifiers over small continuous datasets

Dataset Name	ACA	LAC	CBA
balance-scale	80.11	**88.96**	45.76
data_banknote	**98.86**	97.88	68.37
hayes-roth	**89.57**	76.51	37.91
iris	**96.67**	**96.67**	66.00
iris2D	**98.67**	94.00	66.00
liver-disorders	**71.43**	66.08	57.98
tae	**69.67**	58.27	34.42
weather-numeric	60.00	35.71	**70.00**

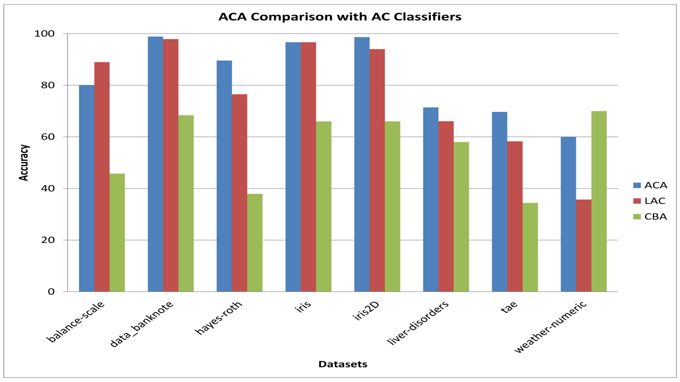


Fig. 8. ACA Comparison with AC Classifiers on small continuous datasets

3) Comparison with Large Discrete datasets: This section discuss the results of ACA with AC classifiers over large discrete datasets. Table XI and Fig. 9 shows the results. Experiments shows that ACA performs better in four out of seven datasets. The extensive experiments showed that those large datasets where the number of distinct values per attributes are small, the ACA performance is lower. The reason is that it results in smaller Automata where conflict of classes increases and even the weighted selection of classes, sometime, unable to produce the highest accuracy. This is the default sequential nature of automata. In individual comparison, the win loss ratio is 5/2 and 6/1 with LAC and CBA respectively which is reasonably high. Similarly on the average, the accuracy of ACA is higher than other AC classifiers.

TABLE XI. ACA Comparison with AC Classifiers over Large Discrete Datasets

Dataset Name	ACA	LAC	CBA
breast-cancer	**89.66**	73.77	70.30
nursery	86.79	**92.81**	66.25
postoperative-patient-dat	**73.50**	71.11	68.56
primary-tumor	**54.41**	36.87	38.91
tic-tac-toe	**88.01**	67.22	65.34
vote	68.47	91.72	**94.73**
zoo	86.10	**91.08**	60.39

4) Comparison with Large Continuous datasets: Finally, this section discusses the comparison of ACA with AC classifier over large continuous datasets. It show the strength and

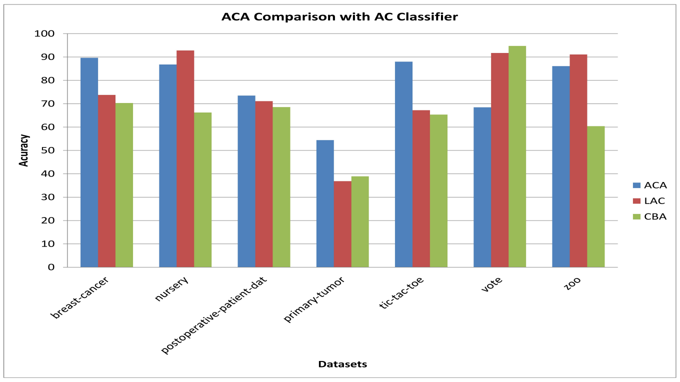


Fig. 9. Accuracy Comparison with AC Classifiers over large discrete datasets

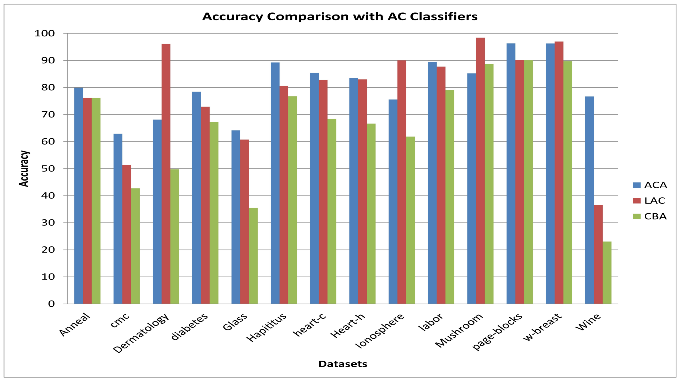


Fig. 10. Accuracy Comparison with AC Classifier over large Continuous datasets

scalability of ACA that as the datasets increase, the accuracy improves. The reason behind is that the large dataset generates more interesting rules and when the number of interesting rules increases, it positively effect the accuracy. Further, The number of distinct values per attributes is larger as compare to Large Discrete dataset and therefore it results in larger automata that produce higher accuracy. Table XII and Fig. 10 shows the results of classifiers. The win loss ration with individual classifier is 11/3 with LAC and 14/0 with CBA. The reason for CBA low performance is the size of dataset as well as less number of rules for their building. While LAC consider a reasonably large rule set from CBA and hence perform better than CBA. In overall ACA beat AC classifiers by 10 datasets out 14.

TABLE XII. Accuracy Comparison with AC Classifiers over large continuous datasets

Dataset Name	ACA	LAC	CBA
Anneal	**79.92**	76.16	76.17
cmc	**62.91**	51.39	42.70
Dermatology	68.12	**96.17**	49.73
diabetes	**78.43**	72.91	67.20
Glass	**64.15**	60.74	35.52
Hapititus	**89.25**	80.64	76.75
heart-c	**85.43**	82.83	68.42
Heart-h	**83.43**	82.99	66.63
Ionosphere	75.55	**90.02**	61.83
labor	**89.44**	87.71	79.00
Mushroom	85.23	**98.42**	88.68
page-blocks	**96.31**	90.13	90.08
w-breast	96.29	**96.99**	89.69
Wine	**76.68**	36.51	23.04

B. Accuracy Comparison of ACA with Rules and Tree based Classifiers

This section discusses the accuracy comparison of ACA with Tree and Rules based classifiers. There are total 7 classifiers from both categories i.e. Ripper, PART, J48, BFTree, CART, Decision Table, and ZeroR. The comparison is provided with individual groups of dataset as mentioned in Section VII. The following sub sections explains the results in details.

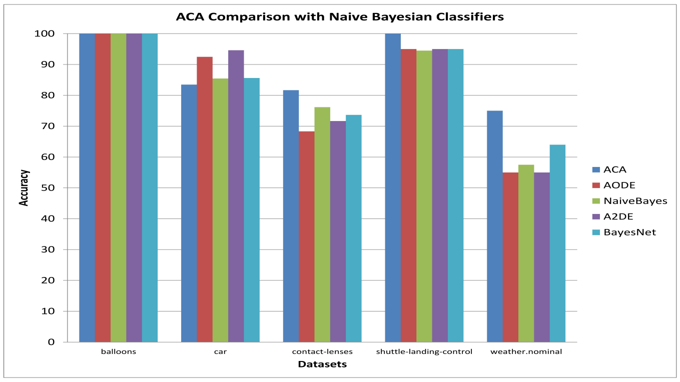


Fig. 11. Accuracy Comparison with Naive Bayesian Classifier over large discrete datasets

1) Comparison with Small Discrete datasets: Table XIII show the results of ACA and other tree based and rules based algorithms. The accuracy of ACA is higher in four datasets while in car, the accuracy goes down. The reason is the The Average Accuracy of ACA is again higher than all Tree and Rules based classifiers.

2) Comparison with Small Continuous datasets: Table XIV show the results of ACA with other tree based and rules based classifier for small continuous datasets. Except the weather-numeric dataset where ACA slightly performed low, in the rest of seven datasets, its accuracy is better. The reason for weather numeric dataset is the number of fewer rules during model construction.

3) Comparison with Large Discrete datasets: The comparison of ACA with tree based and rules based classifiers is shown in Table XV. In large discrete dataset the ACA out performs in three datasets, namely, Breast Cancer, Postoperative Patient Data, Primary Tumor, and Tic Tac Toe. The reason for low performance in vote and zoo is its excessive number of contradictory rules. This high number of contradictory rules results in deceived classifier models. This is the main reason that no single classifier could perform better for these datasets.

4) Comparison with Large Continuous datasets: Table XVI show the results of ACA with other classifiers for Large Continuous datasets. In this category the performance of ACA

TABLE XIII. ACA Comparison with Rules and Tree Based Classifiers over small discrete datasets

Dataset Name	ACA	Ripper	PART	J48	BFTree	CART	DecisionTable	ZeroR
balloons	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	60.00
car	83.49	86.46	95.77	92.36	97.05	**97.11**	91.03	70.02
contact-lenses	**81.67**	75.00	**81.67**	**81.67**	78.33	78.33	75.00	68.33
shuttle-landing-control	**100.00**	95.00	95.00	95.00	95.00	95.00	95.00	95.00
weather.nominal	**75.00**	70.00	60.00	55.00	55.00	40.00	40.00	70.00

TABLE XIV. ACA Comparison with Rules and Tree Based Classifiers over small Continuous datasets

Dataset Name	ACA	Ripper	PART	J48	BFTree	CART	DecisionTable	ZeroR
balance-scale	**80.11**	70.55	77.27	64.48	79.2	78.57	66.74	45.76
data_banknote	**98.86**	96.14	97.16	98.40	98.83	98.76	97.38	55.54
hayes-roth	**89.57**	82.53	74.12	72.69	79.51	83.3	47.03	37.91
iris	**96.67**	93.33	95.33	96.00	96.00	96.00	96.00	33.33
iris2D	**98.67**	93.33	92.67	96.00	95.33	96.00	96.00	33.33
liver-disorders	**71.43**	59.39	62.36	61.15	59.44	60.61	62.05	57.98
tae	**69.67**	56.29	47.04	49.71	52.96	54.96	52.21	34.42
weather-numeric	60.00	50.00	55.00	55.00	65.00	50.00	60.00	70.00

TABLE XV. ACA Comparison with Rules and Tree Based Classifiers over Large discrete datasets

Dataset Name	ACA	Ripper	PART	J48	BFTree	CART	DecisionTable	ZeroR
breast-cancer	**89.66**	70.95	71.33	75.54	67.86	69.26	73.47	70.30
nursery	86.79	96.84	99.21	97.05	99.49	**99.58**	94.7	33.33
postoperative-patient-dat	**73.50**	71.11	61.11	70.00	68.89	71.11	68.56	71.11
primary-tumor	**54.41**	39.24	40.70	39.80	39.80	40.96	38.91	24.78
tic-tac-toe	88.01	**97.81**	94.26	84.55	93.73	92.90	73.39	65.34
vote	68.47	95.41	94.71	**96.33**	95.40	95.42	94.73	61.38
zoo	86.10	89.00	**93.18**	93.06	91.08	90.09	86.00	40.55

is reasonably good where it out perform all other classifiers in ten datasets out of fourteen. The reason for higher accuracy is the larger number of automata that enable the classifier to accurately classify the class labels.

C. Accuracy Comparison of ACA with Naive Bayesian

The ACA Comparison with Naive Bayes Classifier is performed in this section. Naive Bayes classifier are the most prominent classifiers and work well with dataset with low or zero independence. Classifiers in this groups are NaiveBayes, BayesNet, AODE and A2DE. The following subsections elaborate the results in detail.

1) Comparison with Small Discrete datasets: Table XVII and Fig. 11 show the results of ACA and other Naive Bayesian classifiers. The accuracy of ACA is higher in four datasets while the average accuracy is above all classifiers in this category.

2) Comparison with Small Continuous datasets: Table XVIII and Fig. 12 represent the results of ACA and Naive Bayesian classifiers for small continuous datasets. The ACA outperformed all Naive Bayesian classifier in seven datasets while on balance scale dataset the NaiveBayes perform better. The average accuracy of ACA is 88.03 which highest among all classifiers.

TABLE XVI. ACA COMPARISON WITH RULES AND TREE BASED CLASSIFIERS OVER LARGE CONTINUOUS DATASETS

Dataset Name	ACA	Ripper	PART	J48	BFTree	CART	DecisionTable	ZeroR
Anneal	79.92	99.00	99.00	98.66	98.77	**98.88**	97.66	76.17
cmc	**62.91**	48.34	47.99	49.29	52.35	53.77	49.76	42.70
Dermatology	68.12	89.32	**94.80**	94.00	94.00	94.00	86.87	30.60
diabetes	**78.43**	72.79	73.44	73.83	71.23	73.97	72.40	65.11
Glass	**64.15**	48.14	60.80	57.92	62.16	60.30	48.53	35.52
Hapititus	**89.25**	80.00	86.46	81.25	78.00	79.33	81.13	79.37
heart-c	**85.43**	80.17	81.84	79.18	78.20	80.81	74.90	54.45
Heart-h	**83.43**	79.28	80.34	80.02	76.93	76.23	81.00	63.95
Ionosphere	75.55	89.18	87.19	86.62	89.19	**89.76**	84.90	64.10
labor	**89.44**	85.00	66.00	57.33	72.67	68.67	65.00	64.67
Mushroom	85.23	**100.00**	**100.00**	**100.00**	99.94	99.94	**100.00**	51.80
page-blocks	**96.31**	93.92	92.71	92.87	94.06	94.08	92.34	89.77
w-breast	**96.29**	94.13	94.28	94.42	93.56	93.42	92.85	65.52
Wine	**76.68**	30.85	32.55	32.65	32.03	29.84	30.92	23.04

TABLE XVII. ACA COMPARISON WITH NAIVE BAYESIAN CLASSIFIERS OVER SMALL DISCRETE DATASETS

Dataset Name	ACA	AODE	NaiveBayes	A2DE	BayesNet
balloons	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
car	83.49		85.46	**94.62**	85.61
contact-lenses	**81.67**	68.33	76.17	71.67	73.67
shuttle-landing-control	**100.00**	95.00	94.50	95.00	95.00
weather.nominal	**75.00**	55.00	57.50	55.00	64.00

TABLE XVIII. ACA COMPARISON WITH NAIVE BAYESIAN CLASSIFIERS OVER SMALL CONTINUOUS DATASETS

Dataset Name	ACA	AODE	NaiveBayes	A2DE	BayesNet
balance-scale	80.11	89.45	**91.44**	82.24	91.44
data_banknote	**98.86**	97.01	89.60	98.69	89.63
hayes-roth	**89.57**	76.48	82.59	68.96	82.66
iris	**96.67**	94.67	94.33	94.00	94.33
iris2D	**98.67**	96.67	96.87	96.67	96.87
liver-disorders	**71.43**	64.10	64.15	63.5	64.07
tae	**69.67**	56.21	54.25	58.88	54.52
weather-numeric	**60.00**	40.00	56.00	45.00	45.50

3) *Comparison with Large Discrete datasets:* Table XIX and Fig. 13 highlights the results of ACA with all Naive Bayesian classifier over large continuous datasets. Although ACA outperformed in only three datasets but on average accuracy, it is better than other.

4) *Comparison with Large Continuous datasets:* Table XX and Fig. 14 show the results of ACA with Naive Bayesian classifiers over large continuous datasets. In this category, ACA beaten all classifiers on eight datasets while AODE and BayesNet the accuracy on two datasets are higher while A2DE and NaiveBayes outperformed in one dataset each. The ACA is using a larger collection of rules,due to large datasets, that results in higher accuracy. The average accuracy of ACA is higher than all classifiers in this category.

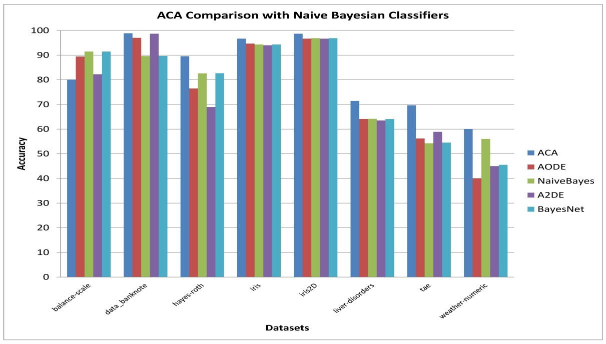


Fig. 12. Accuracy Comparison with Naive Bayesian Classifier over large discrete datasets

TABLE XIX. ACA COMPARISON WITH BAYESIAN CLASSIFIERS OVER LARGE DISCRETE DATASETS

Dataset Name	ACA	AODE	NaiveBayes	A2DE	BayesNet
breast-cancer	**89.66**	71.03	72.70	72.46	72.59
nursery	86.79	92.72	90.30	**94.83**	90.31
postoperative-patient-dat	**73.50**	64.78	68.11	64.78	65.89
primary-tumor	**54.41**	49.53	49.71	49.62	47.11
tic-tac-toe	88.01	72.96	69.64	90.71	69.59
vote	68.47	94.30	90.02	94.30	90.23
zoo	86.10	96.00	93.42	97.00	93.52

X. COMPLEXITY ANALYSIS OF ACA

This section provides the computational analysis regarding time and space requirement of the algorithm. Analysis shows that ACA is reasonably efficient. The subsections explain time and space complexities individually.

A. *Time Complexity Analysis*

ACA consists of two algorithms i.e. *FA Construction* and *Classifier*. The complexity of both algorithms are explained in the subsequent section.

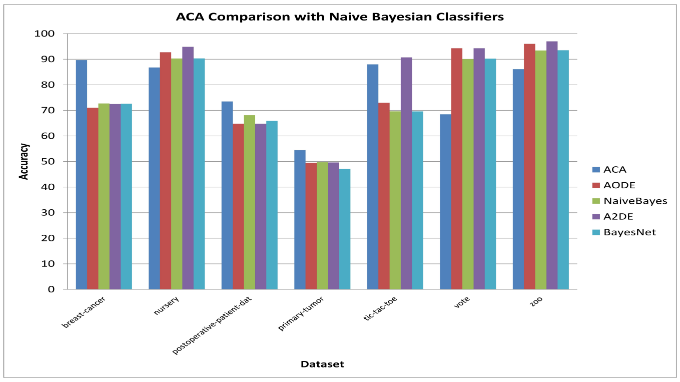


Fig. 13. Accuracy Comparison with Bayesian Classifier over large discrete datasets

TABLE XX. ACA Comparison with Bayesian Based Classifiers over Large Continuous datasets

Dataset Name	ACA	AODE	NaiveBayes	A2DE	BayesNet
Anneal	79.92	95.21	94.56	95.1	94.31
cmc	**62.91**	53.02	50.74	52.40	50.78
Dermatology	68.12	97.81	97.46	96.99	97.79
diabetes	**78.43**	76.96	75.68	74.75	75.57
Glass	**64.15**	59.42	57.69	60.78	58.48
Hapititus	**89.25**	85.13	84.31	83.87	83.91
heart-c	**85.43**	83.47	83.38	82.16	83.55
Heart-h	**83.43**	78.92	83.01	76.18	83.01
Ionosphere	75.55	92.88	90.86	92.60	90.86
labor	89.44	83.00	92.70	83.00	95.07
Mushroom	85.23	99.98	95.76	100.00	96.22
page-blocks	**96.31**	93.42	94.32	93.68	94.67
w-breast	96.29	96.71	97.30	96.99	97.30
Wine	**76.68**	32.68	38.56	33.66	36.92

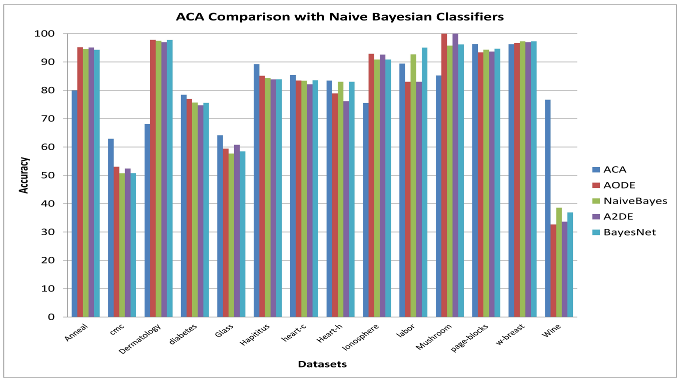


Fig. 14. Accuracy Comparison with Bayesian Classifier over large discrete datasets

1) *Analysis of FA Construction Algorithm::* The running time of FA_Construction is equal to the number of transitions from one state to another. If we have n attributes, and there is "forward only" transition to every state from the current state, then the possible numbers of required transitions are $m \to (m-1)$, $(m-1) \to (m-2)$, ... , $3 \to 2$, $2 \to 1$ (from attribute i, there are i - 1 possible transitions with forward only restriction). For example, if a rule set consists of three attributes A,B,C in a chronological order, then there are two possible transitions from state A, (one to B and one to C) and one move from B to C. Therefore, generally the total number of transitions are

$$\sum_{i=1}^{m}\sum_{j=i}^{m} j \tag{8}$$

which is $O\left(\frac{m^2}{2}\right)$in the worst case scenario and $\Omega(m)$ in the best case. If there are n rules in a rule set, the total time will become $O\left(n\left(\frac{m^2}{2}\right)\right)$. In ACA, single transition can have multiple labels from input symbols, therefore the number of labels is not considered. The hardest part of this module is conflict resolution. Theoretically, if every rule is causing conflict at final state, then the algorithm may be $O\big(nm^2\big)^2$. The reason is that, in each of the iterations, only one rule becomes a new automaton and all the remaining rules are checked against the newly-created automaton.

2) *Analysis of Classifier Algorithm::* It is assumed that there are j Automata with k states in every automaton. Following this, the classifier, in the best case scenario, will take only $\Omega(k)$ time when the first automaton classifies the test instance. In the worst case scenario, if the algorithm is unable to classify the instance at all, then the time will be $O(jk)$ for a single instance. For a whole test dataset with r number of rules, the running time will be $O(rjk)$and $\Omega(rk)$. Hence, the total running time of ACA is the summation of $O(nm^2)^2 + O(jk)$ in worst case while the best case is the sum of $\Omega(nm^2) + \Omega(jk)$ which is much smaller than most of the pruning and ranking algorithms.

B. Space Complexity Analysis

The space requirement depends on, specifically: a) the number of attributes, b) cardinality of attributes (i.e. distinct values of attributes which is called input symbols in ACA), and c) number of rules in CARs. Input symbols of the automata consist of the unique values of all attributes; so if there are three attributes A_1, A_2 and A_3 with 7, 8, 9 distinct values respectively, the size of the input symbol will be 24 $(7 + 8 + 9 = 24)$. If the attributes' values is transformed to consecutive integers such that every attribute's value starts from 0. Using this approach the language size will reduce to the cardinality of the largest attribute; i.e., in the above case the language size will reduce to 9 instead of 24.

Secondly, in the best case scenario, if all rules are absorbed in a single automaton then the space is $O(\alpha m)$ where α refers to the number of input symbols and m is the number of attributes. If every rule causes a conflict and creates separate automata, the space requirements will be bounded in $O(r\alpha m)$where r is the total number of rules. Therefore the space requirement is in $\Omega(\alpha m)$ and $O(r\alpha m)$.

XI. Conclusion

In the field of data mining, associative classification has been researched for more than two decades; however, the trade-off between pruning and accuracy is still under research, while the application of automata as a deterministic approach has received little attention. In this research, automata were utilized for two purposes: a) as a storage structure in classification; and b) to replace the rule pruning and rule ranking phases of associative classification. In the former case, the use of automata enabled the classification to deal with enormous datasets by reducing the space requirements for the purpose of classification. In the latter usage, the automata produced a loss-less associative classifier by eliminating the pruning steps that can result in a loss of useful information with unwise pruning. The automata also helped to reduce the size of the dataset by taking advantage of the repetition of data in the dataset. The sequential nature of the automata allowed the absorption of similar data items into a single automaton.

In the process of size reduction and similar data item absorption, automata can generate conflicts. These conflicts can arise from conflicting association rules, which in turn can lead to conflicting classification results. Consequently, these rules were put into separate automata to avoid the conflicts in the first place. Following the different strategies discussed in this paper, these automata can reduce conflicts by avoiding merger, but this results in a large number of automata and in some cases can lead to a larger classifier. Therefore, a new merging concept was developed to merge automata based on their structure-based similarity. This structure-based similarity helped to reduce the classifier size as well as to minimize the chance of a loss of information.

XII. Future Work

It is evident from this research that the use of automata has the potential to improve classification in general. The current approach is able to handle discrete value datasets only, but it can be extended to deal with the numeric data as well. This will increase the scalability of the ACA and will advance in application in additional areas of interest. The ACA can also extend to streaming data classification. One possible application might be for real-time virus and malware detection. Incoming data could be analyzed during its runtime to detect any malicious applications. Automata can also incorporate the probabilistic approach, where they can use the rough-set theory to decide from among overlapping class predictions. One possible application of this approach could be real-time fraud detection, where rough set theory can help to define the boundaries between legitimate and fraudulent transactions.

References

[1] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin, "The elements of statistical learning: data mining, inference and prediction," *The Mathematical Intelligencer*, vol. 27, no. 2, pp. 83–85, 2005.

[2] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. San Jose, California: John Wiley & Sons, 2012.

[3] I. Yoo, P. Alafaireet, M. Marinov, K. Pena-Hernandez, R. Gopidi, J.-F. Chang, and L. Hua, "Data mining in healthcare and biomedicine: A survey of the literature," *Journal of Medical Systems*, vol. 36, no. 4, pp. 2431–2448, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10916-011-9710-5

[4] B. Zhang, I. Valentine, P. Kemp, and G. Lambert, "Predictive modelling of hill-pasture productivity: integration of a decision tree and a geographical information system," *Agricultural Systems*, vol. 87, no. 1, pp. 1 – 17, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0308521X04002008

[5] S. F. Crone, S. Lessmann, and R. Stahlbock, "The impact of preprocessing on data mining: An evaluation of classifier sensitivity in direct marketing," *European Journal of Operational Research*, vol. 173, no. 3, pp. 781 – 800, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221705006739

[6] B. Zhang, I. Valentine, and P. D. Kemp, "A decision tree approach modelling functional group abundance in a pasture ecosystem," *Agriculture, Ecosystems & Environment*, vol. 110, no. 3 - 4, pp. 279 – 288, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167880905002069

[7] C. Mues, B. Baesens, C. M. Files, and J. Vanthienen, "Decision diagrams in machine learning: an empirical study on real-life credit-risk data," *Expert Systems with Applications*, vol. 27, no. 2, pp. 257 – 264, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417404000120

[8] S. Piramuthu, "On learning to predict web traffic," *Decision Support Systems*, vol. 35, no. 2, pp. 213 – 229, 2003, web Data Mining. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167923602001070

[9] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[10] J. Hong, I. Mozetic, and R. S. Michalski, "Aq15: Incremental learning of attribute-based descriptions from examples: The method and user's guide," *Reports of the Intelligent Systems Group*, vol. 86-05, pp. 1–59, 1986.

[11] J. R. Quinlan, "Simplifying decision trees," *International journal of man-machine studies*, vol. 27, no. 3, pp. 221–234, 1987.

[12] ——, *C4. 5: programs for machine learning*. Morgan kaufmann, 1993, vol. 1.

[13] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[14] W. H. Bing Liu and Y. Ma, "Integrating classification and association rule mining," in *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, 1998, pp. 80–86.

[15] A. Bechini, F. Marcelloni, and A. Segatori, "A mapreduce solution for associative classification of big data," *Information Sciences*, vol. 332, pp. 33 – 55, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025515007793

[16] F. Thabtah, S. Hammoud, and H. Abdel-Jaber, "Parallel associative classification data mining frameworks based mapreduce," *Parallel Processing Letters*, vol. 25, no. 02, p. 1550002, 2015. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S0129626415500024

[17] C.-H. Wu and J.-Y. Wang, "Associative classification with a new condenseness measure," *Journal of the Chinese Institute of Engineers*, vol. 38, no. 4, pp. 458–468, 2015. [Online]. Available: http://dx.doi.org/10.1080/02533839.2014.998287

[18] W. Li, J. Han, and J. Pei, "Cmar: Accurate and efficient classification based on multiple class-association rules," in *Proceedings IEEE International Conference on Data Mining*. IEEE, 2001, pp. 369–376.

[19] F. Thabtah, P. Cowling, and Y. Peng, "Mcar: multi-class classification based on association rule," in *Computer Systems and Applications, 2005. The 3rd ACS/IEEE International Conference on*. IEEE, 2005, p. 33.

[20] F. Thabtah, "Rules pruning in associative classification mining," in *Proceedings of the IBIMA Conference*. Citeseer, 2005, pp. 7–15.

[21] J. Han, "Cpar: Classification based on predictive association rules," in *Proceedings of the third SIAM international conference on data mining*, vol. 3. Siam, 2003, pp. 331–335.

[22] J. Li, H. Shen, and R. Topor, "Mining the smallest association rule set for predictions," in *Proceedings IEEE International Conference on Data Mining*. IEEE, 2001, pp. 361–368.

[23] L. Venturini, E. Baralis, and P. Garza, "Scaling associative classification for very large datasets," *Journal of Big Data*, vol. 4, no. 1, p. 44, 2017.

[24] F. Padillo, J. M. Luna, and S. Ventura, "Evaluating associative classification algorithms for big data," *Big Data Analytics*, vol. 4, no. 1, p. 2, 2019.

[25] K. Ali, S. Manganaris, and R. Srikant, "Partial classification using association rules," in *Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining - KDD97*, 1997, pp. 115–118.

[26] N. Abdelhamid, A. Ayesh, F. Thabtah, S. Ahmadi, and W. Hadi, "Mac: A multiclass associative classification algorithm," *Journal of Information & Knowledge Management*, vol. 11, no. 02, pp. 1–10, 2012.

[27] M. R. Schmid, F. Iqbal, and B. C. Fung, "E-mail authorship attribution using customized associative classification," *Digital Investigation*, vol. 14, Supplement 1, pp. S116 – S126, 2015, the Proceedings of the Fifteenth Annual {DFRWS} Conference. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1742287615000572

[28] F. Padillo, J. Luna, and S. Ventura, "A grammar-guided genetic programing algorithm for associative classification in big data," *Cognitive Computation*, pp. 1–16, 2019.

[29] E. Baralis and P. Garza, "A lazy approach to pruning classification rules," in *Proceedings IEEE International Conference on Data Mining*. IEEE, 2002, pp. 35–42.

[30] E. Baralis, S. Chiusano, and P. Garza, "A lazy approach to associative classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 2, pp. 156–171, 2008.

[31] P. Paranjape-Voditel and U. Deshpande, "A stock market portfolio recommender system based on association rule mining," *Applied Soft Computing*, vol. 13, no. 2, pp. 1055 – 1063, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494612004322

[32] T. Mielikäinen, "An automata approach to pattern collections," in *Knowledge Discovery in Inductive Databases*. Springer, 2005, pp. 130–149.

[33] J. E. Hopcroft, *Introduction to Automata Theory, Languages, and Computation*, 3rd ed. Pearson Addison Wesley, 2007.

[34] A. V. Aho and M. J. Corasick, "Efficient string matching: an aid to bibliographic search," *Communications of the ACM*, vol. 18, no. 6, pp. 333–340, 1975.

[35] Y. Fan, H. Zhang, J. Liu, and D. Xu, "An efficient parallel string matching algorithm based on dfa," in *Trustworthy Computing and Services*. Springer, 2013, pp. 349–356.

[36] K. Bache and M. Lichman, "UCI: machine learning repository," University of California, 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[37] F. Thabtah, Q. Mahmood, L. McCluskey, and H. Abdel Jaber, "A new classification based on association algorithm," *Journal of Information & Knowledge Management*, vol. 9, no. 01, pp. 55–64, 2010.

[38] B. Liu, Y. Ma, and C. K. Wong, "Improving an association rule based classifier," in *Principles of Data Mining and Knowledge Discovery*. Springer, 2000, pp. 504–509.

[39] A. Veloso, W. Meira, and M. J. Zaki, "Lazy associative classification," in *Data Mining, 2006. ICDM'06. Sixth International Conference on*. IEEE, 2006, pp. 645–654.

[40] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[41] W. W. Cohen, "Fast effective rule induction," in *Proceedings of the Twelfth International Conference on Machine Learning*, Lake Tahoe, California, 1995.

[42] E. Frank and I. H. Witten, "Generating accurate rule sets without global optimization," in *ICML '98 Proceedings of the Fifteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 144–151.

[43] G. I. Webb, J. R. Boughton, and Z. Wang, "Not so naive bayes: aggregating one-dependence estimators," *Machine learning*, vol. 58, no. 1, pp. 5–24, 2005.

[44] D. Ruta and B. Gabrys, "New measure of classifier dependency in multiple classifier systems," in *Multiple Classifier Systems*. Springer, 2002, pp. 127–136.

[45] T. D. Nielsen and F. V. Jensen, *Bayesian networks and decision graphs*. Springer Science & Business Media, 2009.