An Improved Associative Classification Algorithm based on Incremental Rules

Mohamed Salem Almnnaee *alsharji@wasl.ae*
WASL
Dubai, UAE

Fadi Thabtah *fadi.fayez@manukau.ac.nz*
Manukau Institute of Technology
Auckland, New Zealand

Joan Lu *j.lu@hud.ac.uk*
University of Huddersfield
Huddersfield, UK

Abstract

In Associative classification (AC), the step of rule generation is necessarily exhaustive because of the inherited search problems from the association rule. Besides which, the entire rules set must be induced prior constructing the classifier. This article proposes a new AC algorithm called Dynamic Covering Associative Classification (DCAC) that learns each rule from a training dataset, removes its classified instances, and then learns the next rule from the remaining unclassified data rather than the original training dataset. This ensures that the exhaustive steps of rule evaluation and candidate generation will no longer be needed, thereby maintaining a real time rule generation process. The proposed algorithm constantly amends the support and confidence for each rule rather restricting itself with the support and confidence computed from the original dataset. Experiments on 20 datasets from different domains showed that the proposed algorithm generates higher quality and more accurate classifiers than other AC rule induction approaches.

Keywords: Associative Classification, Data Mining, Machine Learning, Rules.

1. Introduction

Classification and association rules have been integrated to form a new research topic named associative classification (AC) [15]. AC primarily utilises association rule discovery [5] to train an input dataset in order to discover class association rules (CARs) and then adds on steps involving constructing the classifier, rule pruning and predicting test data. In the last decade, AC has been utilised in business applications, i.e. Website Phishing Detection [3], Fault Prediction [17], Recommendation Systems [16], and Text Mining [18] since AC approach derived competitive classifiers to other conventional approaches such as Greedy, Covering, Decision Trees and Probabilistic among others.

The majority of the existing AC algorithms induce the rules from the training dataset and then construct a classifier by evaluating the induced rules on the training dataset. Two main parameters named support and confidence (Definitions 6 and 8) are connected with each rule. The process of discovering and evaluating the rules is the concern of this article. In this context, the majority of current AC algorithms discover the rules using association rule mining methods in one step and then evaluate the extracted rules against the training dataset one by one in a separate step called classifier building. In evaluating each rule, starting from the highest ranked rule downwards, each training example will be covered only by a single rule. When a training example is covered by a rule it will be discarded and that rule will be inserted into the classifier. The rule evaluation process continues until all rules are tested or the training dataset becomes empty and only when this happens the classifier is formed. The classifier will contain rules that

had covered training examples and all other candidate rules are removed as they are either redundant or useless.

This article investigates major shortcomings associated with AC algorithms during the processes of rule discovery and building the classifier. Specifically, we look into building the classifier since currently, whenever a rule is inserted into the classifier, all its training data are discarded. However, these data are used to generate other possible rules hence these rules' confidence and support values must be updated based on the remaining data rather than the original dataset. All current AC algorithms maintain the support and confidence parameters computed initially for each rule despite the data removal. For instance, when training examples are discarded after any rule such as R_1 is inserted into the classifier, other non-evaluated rules, i.e. R_2 - R_n, that utilise the removed data examples should be re-ranked and possibly pruned earlier. In other words, some of the affected candidate rules will become higher ranked and others will become lower ranked and thus a different classifier can be built. The changes in the rule rank is due to the fact that the support and confidence values of the affected candidate rules have changed because of R_1's data removal. Hence, the next rule to be inserted into the classifier will be practically learnt from the remaining training data excluding R_1's data. This live update procedure can be embedded within the rule generation phase and the outcome is a more realistic classifier since

a) its rules are derived from a continuously updated training dataset, and

b) each rule is linked with its true frequency (support), strength (confidence) and class

We propose a new prospective learning within AC approach that integrates rule discovery and classifier building phases into a single incremental algorithm we named Dynamic Induction Associative Classification (DCAC). Our algorithm scans the training dataset and records 1-ruleitems (item plus class) of size one and their occurrences in a data structure. Then, DCAC seeks for the item plus class (1-ruleitem) with the highest confidence and appends it to the rule body and continues adding items until the rule fulfils the confidence requirement. Once this happens, the rule is derived, and all training data linked with it are deleted. This means, the remaining candidate 1-ruleitems support and confidence values are amended since they are now computed from the remaining training data (original training dataset –removed data examples). DCAC repeats the same steps until the original training dataset becomes empty or no more 1-ruleitem passes the minimum support threshold. The outcome is a classifier that ensures no data examples are overlapping among its rules and hence it usually contains fewer rules, solving a major problem in AC which is the exponential growth of rules (massive sized classifiers).

This article is structured as follows: Section 2 critically analyses the literature review. Section 3 discusses the proposed algorithm and its related phases along with highlighting the distinct differences between the proposed algorithm and other rule based classification ones (AC and rule induction). Section 4 is devoted to the data and the experimental results analysis and finally conclusions are provided in Section 5.

2. Literature Review

Usually, an AC algorithm must discover the complete rules set and then initiates the rule evaluation process, which can be problematic with reference to processing time and memory use. This is since the algorithm initially finds frequent ruleitems of size one (1-ruleitems) after the first data scan then merges the disjoint frequent 1-ruleitems to find candidate 2-ruleitems. The algorithm repeats the same process to find candidate 3-ruleitems from frequent 2-ruleitems and so forth. At each iteration the algorithm must go back and search the dataset to figure out whether a ruleitem is frequent by computing its support and compare it with the *minsupp* threshold. This exhaustive step has been inherited from association rule and studied extensively in AC. Hence, after the development of the first AC algorithm, i.e. CBA, almost all of the entire successors have focused on two primary issues to improve:

a) Enhancing the process of discovering frequent ruleitems

b) The exponential growth of rules

This problem occurs because the AC algorithm tests all possible correlations among the attribute value in the training dataset and the class value, hence very big numbers of correlations can be discovered. The problem becomes harder when the input dataset is highly dimensional [6,25]. This may lead to uncontrollable large classifiers that may limit the applicability of this learning approach in applications.

One of the first developed AC algorithms was CBA [15]. This algorithm utilised Apriori association rule mining method to discover and extract class association rules (CARs) from classification datasets. CBA was the algorithm that introduced the database coverage pruning method to choose high predictive rules for the classifier. This method in similar to the way greedy classification algorithms extract the rules. A number of successive AC algorithms adopt CBA rule learning, rule ranking and classification procedures including, i.e. CBA (2) (Liu et al., 2001), and ACCF [12]. One of the first major improvements of CBA algorithm in regard to training phase was proposed by [13] in an algorithm called CMAR. This AC algorithm employs the efficient method of FP-Growth in association rule mining to find the rules. CMAR constructs the rules in a data structure that takes the shape of a tree known as a Compact-tree. This data structure saves the rules in a ranked manner according to the rule's support. CMAR prunes the candidate rules by discarding any with large number of attributes values and keeps rules with smaller attribute values. Few AC algorithms have utilised CMAR approach in mining CARs including Lazy AC [6].

Unlike the abovementioned AC approaches (CBA, CMAR) that use the horizontal mining approach, [21] proposed a new vertical mining [28] based AC approach called MMAC. This approach depends on the information collected from the training dataset (items and their locations) that are saved in a data structure called the TID-List. By holding the TID-List of all items, one can locate the item's support, confidence without having to revisit the original training dataset. This significantly improves costs associated with the training phase. MMAC leans the rules by intersecting items' TID-Lists and improves upon the rules ranking process by adding additional tie breaking criteria. Recently, an improvement on the classification procedure of MMAC was proposed in [1] where multiple rules are used to decide on the class value of test data. This has enhanced accuracy of the classifiers.

An AC called MAC [4] was developed to enhance the rules pruning and classification steps of CBA. During evaluating the candidate rules on the training dataset, MAC tests each of them on the training dataset to decide the most significant ones. Unlike CBA which requires equality between the candidate rule's class and the training example class so that the rule can be inserted into the classifier, MAC considers only the similarity between the rule's body and the training example attributes values and omits the class similarity. This increases the data coverage per rule and reduces overfitting. The same evaluation process is repeated for each candidate rule until all training examples are completely covered. Lastly, MAC inserts all evaluated rules that had training data coverage into the classifier. MAC was applied successfully on generic classification datasets and domain specific datasets (website phishing classification).

Recently, a new parallel and distributed AC framework for big data was developed [20] called MapReduce Multiclass Classification based Association Rule (MR:MCAR) . This framework is the first distributed AC for big data with two distinct implementations (Hadoop and Weka). The novelty of MR: MCAR is the knowledge reasoning method which is based on MapReduce, where the algorithm keeps switching between horizontal data and vertical data formats until all knowledge is derived. The algorithm utilises an efficient search method for knowledge based on ColumnID and RowID (vertical mining) and embraces this method in all phases including knowledge reasoning, rules ranking, rules pruning and class prediction. Several experiments have been conducted to evaluate MR: MCAR effectiveness and efficiency. The results clearly indicated that MRMCAR is an efficient algorithm for big data and it generates high quality classifiers when compared with trees and rule induction algorithms.

3. Algorithm Development

The proposed algorithm (Figure 1) consists of two main phases: Inducing the rules, and classifying test data. The algorithm in phase (1) scans the training dataset to find the rules based on two main thresholds (*minsupp* and *minconf*). In phase (2), the discovered rules are utilised to allocate class labels to test data. Below are the relevant definitions of DCAC algorithm.

Given an input training dataset *T*, which has *n* distinct attributes $A_1, A_2, \ldots, A_n$ one of which is called the class, i.e. *l*, that contains a list of values. *T* size is denoted |*T*|.

Definition 1: An *attribute value* is an attribute plus its values name denoted (A_i, a_i).

Definition 2: A *training example* in *T* is a row combining a list of attribute values (A_{j1}, a_{j1}), …, (A_{jv}, a_{jv}), plus a class denoted by c_j.

Definition 3: A *ruleitem r* has the format <*body*, *c*>, where *body* is a set of disjoint attribute values and *c* is a class value.

Definition 4: The frequency threshold (*minSupp*) is a predefined threshold given by the end user.

Definition 5: The body frequency (*body_Freq*) of a *ruleitem r in T* is the number of examples in *T* that match *r's body*.

Definition 6: The frequency of a *ruleitem r in T* (*ruleitem_freq*) is the number of data examples in *T* that match *r*.

Definition 7: A *ruleitem r* passes the *minSupp* threshold if, *r*'s |*body_Freq*|/ |*T*| ≥ *freq*. Such a *ruleitem* is said to be a frequent *ruleitem*.

Definition 8: A *ruleitem r* confidence is defined as |*ruleitem_freq*|/ | *body_Freq*|.

Definition 9: A rule in our classifier is represented as: $body \rightarrow l$, where left hand side (body) is a set of disjoint attribute values and the right hand side (l)is a class value. The format of the rules is: $a_1 \wedge a_2 \wedge \ldots \wedge a_n \rightarrow l_1$

3.1 Inducing the Rules

In the process of discovering the rules, unlike the majority of the existing AC algorithms that require two steps, this newly developed algorithm implements a single step using vertical mining approach based on a special data structure named TidList to hold 1- ruleitems and their appearances in the training data (Line #'s). Hence after the initial training data scan, DCAC creates a TidList that contains ruleitems of size one in which each ruleitem is represented as <ColumnID, LineID> that denote the first column and row numbers that the ruleitem occurs in the training dataset. This data format has been recently employed in [20] due to its simplicity in mining the rules. The fact that the TidList is used to locate ruleitems frequency is a definite advantage especially in computing and updating the support and confidence that are the main criteria used to generate the rules.

According to Figure 1, the process of rule induction in DCAC involves generating the best rule using the confidence and support parameters. Once the highest confidence rule is identified, it will be inserted into the classifier, and its classified training data are discarded, and the frequency of all items appearing in the removed instances are decremented. This decrement process results in changing the support and confidence of several potential rules. This means some of potential rules will have higher rank and others have lower rank because the training dataset was lessened (Line 6). The process of discovering the rules continues on the same manner until the training dataset has no more data or no more rules with acceptable confidence and support. When this occurs, all generated rules become the classifier that is efficiently reduced in the size.

There are a few distinguishing features in DCAC when compared with most existing AC algorithms. Firstly, the proposed algorithm eliminates two major steps in AC which are classifier building (sometimes called rule evaluation) and frequent item discovery. The frequent item discovery step usually necessitates merging frequent items of size *k* to generate candidate items of size $k + 1$ repeatedly and this step is a burden since it requires massive computations as well as computing resources [2,19,22,26]. On the other hand, such classifier building steps require the complete rules being derived in advance before any of them can be evaluated. In addition, this step necessitates passing over all candidate rules and for each training case which is indeed a time consuming approach. We offer in DCAC either no rule evaluation separate step nor frequent item discovery step making our method efficient.

Input: Training data set *T*, *minsupp* and *minconf* thresholds
Output: A classifier that contains rules

Phase (1) Building the classifier procedure

1. For each attribute value (A_i, a) plus a class in T do
2. Calculate ruleitems support and confidence, i.e. p(item | class), and discard any 1-ruleitem that has not passed minsupp
3. Start building a rule r_i (Items, Class) by appending the item with the largest confidence to the body of r_i
4. Repeat steps 1-3 until r_j passed the minconf threshold
5. Insert r_j into the classifier
6. Remove r_j's data examples from Ti that are identical to r_j's body (set of items)
7. Amend the support and confidence values of all effected candidate ruleitems to reflect step 6
8. Repeat steps 1-7 until *T* becomes empty or no more ruleitems hold enough support
9. Generate the classifier.
10. end

Phase (2) predict the class of test data (Figure 2)

Fig. 1. DCAC algorithm steps

Theoretically, our learning mechanism produces, and in parallel, tests each rule. This makes building the classifier an implicit process within the rule induction step. Further, it guarantees that no rule can share training examples and thus minimises rule redundancy, eventually reducing the size of the classifier. The process of updating the candidate rules, confidence and support values whenever a rule is inserted into the classifier is novel, and indeed results in live rule induction. It also certifies a real time rule ranking based on the remaining data left in the training dataset rather the static support and confidence computed initially when the mining process starts. This can be seen as an implicit pruning in which weak candidate rules are identified without having to look them up in the training dataset that efficiently improves the mining process. We believe that a more realistic classifier is created since rule generation is dynamic rather static as in existing AC methods.

Table 1. Differences between DCAC and other AC and classic covering methods

Common rule induction and AC approaches	**DCAC**
Classic AC algorithms like CBA, CMAR, CPAR, MMAC, MCAR, etc, operates in four phases: frequent ruleitems discovery, rule generation, classifier building (pruning), and classification.	DCAC operates in only two steps: rule generation and classification.
In AC algorithms, all rules must be generated before each is evaluated. This means in order to form the classifier, the complete candidate rules must be induced first and then many of which are deleted after rules evaluation step	In DCAC, each rule is generated and evaluated in parallel manner so when rules are induced they represent the classifier. There is no rule evaluation step.
The support and confidence values which determine the rule's significance are static per rule and are computed from the original training dataset	The support and confidence values which determine the rule's significance are dynamic per rule and are computed from the different versions of the training dataset as the algorithm producing the rules.
There must be rule ranking in AC to distinguish among rules. Typically, AC algorithms use rule's confidence, support, length, class distribution as ranking parameter	No rule ranking since the rank is natural and based on the order of rule generation.
Classic AC algorithm employ candidate generation step so there are repetitive counting and joining of frequent ruleitems at iteration i to come up with candidate ruleitems at iteration i+1	In DCAC, no candidate generation at all. Only ruleitems of size 1 are needed throughout the algorithm lifecycle.
Classic covering methods like PRISM and its successors employ expected accuracy measure to generate the rules. Thus they only generate perfect yet low data coverage rules	DCAC employs minimum support and minimum confidence to differentiate among rules and allow the production of rules with small errors yet high data coverage
Other more advanced rule induction methods such as CN2, FOIL, AQ, etc produce the first rule in separate and conquer approach, removes data instances, then learn the second rule from the remaining data instances in repetitive manner.	In DCAC, once a rule is generated a special data structure is invoked on the fly to amend the frequency of the remaining potential rules without the need of a repetitive scan. Thus, the runtime performance is indeed improved.
Classic covering methods use extensive pruning such as backward and forward pruning. Further, no item or rules search space minimisation methods are employed.	A dynamic pruning during the training phase based on both support and confidence are employed in DCAC. Hence, both the items and rules search spaces are substantially minimised.

3.2 Test Data Classification Step

Normally, existing AC algorithms sort rules in the classifier using different criteria mainly rule's confidence, support, length, and information gain, etc. However, DCAC eliminated completely the rule sorting since the rules have now been favoured by the order in which they were generated. In other words, the best rule is the one that has been derived first, then the second one, then the third one and so forth. This approach offers a natural sorting mechanism without having to design a sorting method as in current AC methods. DCAC follows greedy algorithms such as RIPPER [8] and PRISM [7] in placing rules into the classifier yet it differs from these algorithms in the way the rule is found.

The last and most vital step in the life cycle of any classification algorithm is test data classification. In this step, the AC algorithm normally fires one or more rules to assign the class label to a test data. DCAC algorithm utilises the first rule that matches the test data in the classification step as shown in Figure 2. When a test data (ts_i) is about to be classified, our

```
Input: test data (Te), Classifier (C)
1 For each test data in T Do
2 For each rule r in C Do
3        If te = r
4            t's class = r's class
5        else
6     else t's class = default class
7    end if
8 end
9 end
10 compute the total number of errors of Ts
```

Fig. 2. Test data procedure of DCAC algorithm

algorithm goes over the classifier rules and identifies the first rule that its body (attribute values) is contained within the test data. Then it assigns the class of that rule to the test data. In cases where no rules in the classifier matches the test data then the default class is assigned. By applying this procedure, we eliminate any biased decision of favouring one rule over another since rules are sorted based on the order they are derived. Consequently, the class allocation decision of test data becomes more realistic and end-user will be confident of the outcome. The primary differences between the proposed algorithm and other rule based classification methods (rule induction and AC) are given in Table 1.

4. Analysis and Discussion

We have chosen datasets from University of California Irvine (UCI) repository [14] with different types and sizes for fair comparison (see Table 2). All numerical attributes of the chosen datasets have been discretised and missing values were replaced using ReplaceFilter in Weka [24]. Stratified ten folds cross validation method has been used for testing all the considered classification algorithms. This method is widely used in machine learning and data mining communities to produce fair average error rates of the classifiers. In this testing method and before mining, the dataset gets partitioned into ten parts and the algorithm is trained on nine parts and tested on the remaining part. This process is repeated ten runs in which each run generates an error rate and then the error rates derived from the ten runs are averaged to produce an overall error rate against the dataset.

A number of highly competitive classification algorithms that generate rule based classifiers implemented in Weka have been utilised to conduct the experiments. In particular, CBA, PRISM, and PART [10] are the algorithms chosen. We tried to be as fair as possible by selecting different high performed well-known algorithms in the literature. The selection of these algorithms arose because they produce rules similar to the proposed algorithm, and they adopt different rule induction mechanisms. Lastly, all experiments have been rum on a computer machine Core i5 with a 3.1 GHz processor and 4.0 GB RAM.

The *minsupp* and *minconf* thresholds for DCAC and CBA have been set in all experiments to 1% and 50% following other scholars in AC literature [9,11,15,21,23,27]. On the other hand, the *minconf* has low impact and was set to 50%. The evaluation measures used to evaluate the pros and cons of DCAC are accuracy, number of rules and training time in ms.

In Table 2, the classification accuracy per dataset has been generated for all the considered algorithms to further evaluate the predictive power of the proposed algorithm. The figures clearly show a consistent domination for DCAC algorithm when compared to the remaining

algorithms. In particular, DCAC won-lost-tie record against PART, and PRISM are 9-8-3, and 17-3-0 respectively. It seems that CBA crashes when the numbers of attributes increase so no results for CBA on twelve out of the twenty datasets can be generated. For the eight datasets that CBA produced results, it outperformed the proposed algorithm on only three of them. The exhaustive search of CBA which is a typical AC algorithm that uses Apriori candidate generation for rule discovery caused a combinatorial explosion especially when the datasets has a dimensionality greater than twelve variables.

The fact that whenever a rule is inserted into the classifier and its covered data are discarded is a definite advantage of DCAC. This is since the classifier constructed contains rules that have no data overlapping and hence ensures that

a) Each training example is covered by only a single rule and is used only once during rule induction phase by that rule. Therefore, an inherited problem from the association

Table 2. The considered algorithms accuracy generated from the 20 UCI datasets

dataset	# of variables	# of cases	PART	DCAC	PRISM	CBA
Arrhythmia	280	452	57.31	60.9	38.5	No results
Balance-scale	5	625	77.28	84.8	63.68	86.08
Cleve	12	690	85.8	82.79	78.97	81.19
Credit-g	21	1000	69.3	70.99	63.8	No results
Cylinder-bands	40	540	59.26	74.57	55.2	No results
Dermatology	35	366	94.81	91.61	84.44	No results
Pima_diabetes	9	768	73.44	72.67	61.08	70.97
Hayes-roth-test	5	28	50	86.72	42.86	82.17
Hayes-roth-train	5	132	74.25	78.84	68.95	72.73
Hepatitis	20	155	80.65	79.73	77.43	No results
Hypothyroid	30	3772	92.74	92.91	91.23	No results
Ionosphere	35	351	87.18	86.51	86.05	No results
Liver-disorders	7	345	62.32	61.83	55.66	63.5
Lung-cancer	57	32	75	74.02	58.38	No results
Lymph	19	148	80.41	78.79	75.69	No results
Mushroom	23	8124	100	99.21	100	No results
Sick	30	3772	97.78	97.56	98.05	No results
Tae	6	151	47.02	57.89	54.98	53.65
Tic-tac-toe	10	958	94.26	91.68	96.46	100
Waveform	41	5000	74.8	78.55	60.58	No results

rule that allows a training example to be used multiple times in inducing rules has been resolved

b) Rules frequencies which are the primary measure for the rule strength (confidence and support) are constantly updated to achieve point (a)'s aim. This safeguards the rule induction phase since insignificant rules are removed despite some of them may have a high rank at the first scan.

Dealing with the rules overlapping problem and the development of rules linked with constantly changing confidence and support values have contributed to the decrease of the one-error rate in the classifiers derived by DCAC. Specifically, the DCAC algorithm outperformed the considered algorithm on average and particularly with a higher average accuracy than PART and PRISM by 4.12% and 9.47% respectively. As a matter of fact, our algorithm ensures each rule is derived from the remaining instances in the training data after removing instances associated with the so far generated rules. This, indeed, only allows rules that have a constant statistical fit to participate in the classifier. These rules are the ones utilised later on during the class prediction step.

The classifier size and time taken to find the rules in milliseconds (ms) per dataset are given in Figure 3. PRISM generates on average larger classifiers than the rest of the considered algorithms, which is due to the fact that PRISM has no pruning. DCAC on average induce less number of rules in the classifier than PART, and PRISM. The proposed algorithm consistently generated smaller classifiers. The rule reduction in DCAC classifier is attributed to two main reasons:

1) Each rule covers large number of training instances because of the removal of training data overlapping among rules
2) The new learning strategy employed by DCAC that allows a rule to cover more training instances

The mechanism of rule learning in DCAC is contributed to a decrease in the final classifier since when each is inserted into the classifier, DCAC reduces the search space of remaining items by only storing those that are linked with acceptable "current" support and "current" confidence values. Existing AC algorithms "must" generate all rules at once then perform the rule pruning whereas our algorithm induce and evaluate each rule at in parallel manner until the dataset gets empty or no item with sufficient data is present. In other words, the removing of the overlapping among rules in the training instances when each rule is generated, has also a positive impact on the classifiers size. In particular, DCAC algorithm ensures that all candidate items frequencies are amended whenever a rule gets produced, which decrease the available numbers of candidate items for the next possible rules.

Finally, the runtime in ms for the considered algorithms on the datasets have been recorded in Figure 4. The figures clearly point out that PRISM is the slowest algorithm to construct classifiers. This has been attributed to that PRISM keeps generating rules as long as they fulfil the expected accuracy. In addition, PART employs additional pruning methods to trim trees before converting them into rule sets and thus it is slower than DCAC. Finally, we applied the CBA algorithm and it generated classifiers from 8 out of the 20 datasets due to the large space of items. The storing large numbers of candidate items on the main memory caused the algorithm to crash in the Weka platform. The number of rules results on the 8 datasets showed that CBA normally generated large classifiers; all of them are larger than those of CBA except on the tic-tac-toe dataset.

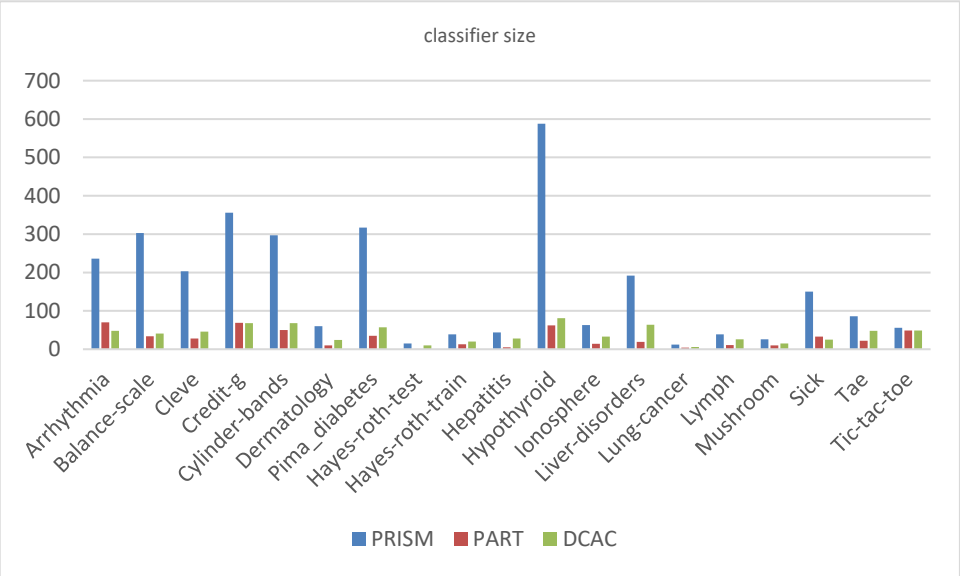

Fig. 3. The considered algorithms classifier size on the UCI datasets

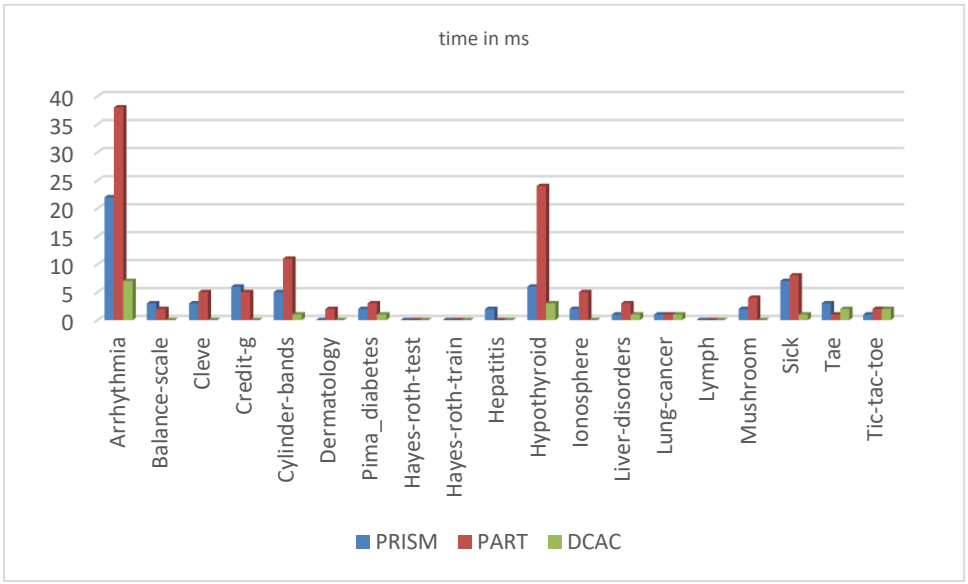

Fig. 4. The considered algorithms training time in ms

5. Conclusions and Future work

Rule discovery and constructing classifier steps contribute to major deficiencies in Associative Classification (AC). These include uncontrollable massive classifiers besides a slow and resource hungry mining process. In this article, we developed a new AC algorithm called Dynamic Covering Associative Classification (DCAC) that integrates these two steps in a single step by continuously inducing rules one by one from the training dataset. Whenever a rule is derived, and its classified training examples are discarded, DCAC builds the next rule from the remaining unclassified training instances. Hence all support and confidence values for the potential rules are amended to guarantee the production of rules that are naturally sorted based on the order that they have been generated. Also, this removes any possible training

examples overlapping among the classifier's rules. These advantages contributed to improving the classification accuracy as well as reducing the classifier size of DCAC when compared to other algorithms. Decision makers can now enjoy a concise highly predictive set of rules in planning. DCAC has been implemented in the Weka environment under "classify" tab page and package "Rules".

Experimental results using 20 datasets with various different sizes and attributes types have been conducted utilising a number of rule based classification and AC algorithms. The results revealed that DCAC is competitive with respect to one error rate and training time when compared to CBA, PRISM and PART and algorithms. Furthermore, DCAC consistently derived a lesser number of rules than these algorithms due to the new prospective learning employed in the rule generation phase. The fact that PART generated more rules than DCAC and less accurate classifiers demonstrates some potential advantages of the proposed algorithm. Normally AC algorithms generate far more rules than rule induction (PRISM) and tree (PART) approaches so having DCAC extracting a smaller classifier is one of the major contributions to AC research.

One possible limitation of DCAC algorithm is that its applicability has not been evaluated on big data applications with unstructured variables. In future research, we intend to extend DCAC to handle applications with big dimensionality possibly under the programming framework of Spark.

References

1. Abdelhamid N., Thabtah F., Multi-label rules for phishing classification. Applied Computing and Informatics 11 (1), 29-46. Elsevier.
2. Abdelhamid N., Thabtah F., Associative Classification Approaches: Review and Comparison. *Journal of Information and Knowledge Management* (JIKM). Vol. 13, No. 3 (2014) 1450027. World Scientific.
3. Abdelhamid N., Thabtah F., Ayesh A. Phishing detection based associative classification data mining. *Expert systems with Applications Journal*, 41 (2014) 5948–5959. Elsevier.
4. Abdelhamid N., Ayesh A., Thabtah F., Ahmadi S., Hadi W. MAC: A multiclass associative classification algorithm. *Journal of Information and Knowledge Management* (JIKM). 11 (2), pp. 1250011-1 - 1250011-10. WorldScinet.
5. Agrawal, R., and Srikant, R. Fast algorithms for mining association rule. Proceedings of the 20th International Conference on Very Large Data Bases- VLDP,pp. 487-499, 1994.
6. Baralis E., Chiusano S., Graza P. A Lazy Approach to Associative classification. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING. vo. 20, num 2. ISSN: 1041-4347.
7. Cendrowska, J. PRISM: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, Vol.27, No.4, 349-370.
8. Cohen W. Fast effective rule induction. In machine learning: Proceedings of the 12th International conference, pp. 115-123. Lake Tahoe, California. Morgan Kaufmann.
9. Costa G., Ortale R., Ritacco E. X-Class: Associative Classification of XML Documents by Structure. ACM Trans. Inf. Syst. 31(1): 3 (2013).
10. Frank, E., and, Witten, I. Generating accurate rule sets without global optimisation. Proceedings of the Fifteenth International Conference on Machine Learning**,** (pp.144–151). Madison, Wisconsin.
11. Han J., Pei J. , Yin Y. Mining frequent patterns without candidate generation, Proceedings of the 2000 ACM SIGMOD international conference on Management of data, p.1-12, May 15-18, 2000, Dallas, Texas, USA.
12. Li X., Qin D, and Yu C. ACCF: Associative Classification Based on Closed Frequent Itemsets. Proceedings of the Fifth International Conference on Fuzzy Systems and Knowledge Discovery - FSKD, pp. 380-384, 2008.
13. Li W., Han J., and Pei. J. CMAR: Accurate and efficient classification based on multiple class-association rules. In ICDM'01, pp. 369–376, San Jose, CA, Nov.2001.

14. Lichman, M. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
15. Liu B., Hsu W., and Ma Y. Integrating classification and association rule mining. In KDD'98, pp. 80–86,New York, NY, Aug. 1998.
16. Lucas J. P., Lirmm A. L., Moreno M. N., Teisseire M. A Fuzzy Associative Classification Approach For Recommender Systems. *Int. J. Unc. Fuzz. Knowl. Based Syst*. 20, 579 (2012). DOI: 10.1142/S0218488512500274.
17. Ma B., Zhang H., Chen G., Zhao Y. Baesens B. Investigating Associative Classification for Software Fault Prediction: An Experimental Perspective. *International Journal of Software Engineering and Knowledge Engineering*, 24, 61 (2014). DOI: 10.1142/S021819401450003X. World Scientific.
18. Mohammad R. M., Thabtah F. and McCluskey L. Intelligent Rule based Phishing Websites Classification, *IET Information Security*, vol. 8, no. 3, pp. 153-160, July 2013-A
19. Qin, XJ, Zhang, Y, Li, X and Wang, Y. Associative classifier for uncertain data. In: , Web-Age Information Management. 11th International Conference on Web-Age Information Management (WAIM 2010), Jiuzhaigou, China, (692-703). 15-17 July 2010.
20. Thabtah F., Hammoud S., Abdeljaber H. Parallel Associative Classification Data Mining Frameworks Based MapReduce. Journal of Parallel Processing Letter. Parallel Process. Lett. 25, 1550002 .World Scientific.
21. Thabtah F., Cowling P., and Peng Y. Multiple Label Classification Rules Approach. *Journal of Knowledge and Information System*, Volume 9:109-129. Springer.
22. Thabtah F., Gharaibeh O., Al-zubaidy R. Arabic Text Mining for Rule based Classification. *Journal of Information and Knowledge Management* (JIKM). Volume: 11, Issue: 1(2012) pp. 1-10. WorldScinet.
    Proceedings of the Principles of Data Mining and Knowledge Discovery - PKDD, pp. 605-612, 2007.
23. Wang X., Yue K., Niu W., and Shi Z. an approach for adaptive associative classification. Expert Systems with Applications: An International Journal, Volume 38 Issue 9, pp. 11873-11883, 2011.
24. Witten I. H. and Frank E. Data Mining: Practical Machine Learning Tools and Techniques.
25. Wua C-H., Wanga J-Y. Associative classification with a new condenseness measure. Journal of the Chinese Institute of Engineers, Vol 38 (4), pages 458-468. Tylor Francis.
26. Yin, X., and Han, J. CPAR: Classification based on predictive association rule. Proceedings of the –the SIAM International Conference on Data Mining -SDM, pp. 369-376, 2003.
27. Yoon Y., Lee G. Efficient implementation of associative classifiers for document classification. Inf. Process. Manage. 43(2): 393-405 (2007).
28. Zaki M., Hsiao CJ CHARM: an efficient algorithm for closed itemset mining. Proceedings of the 2002SIAMinternational conference on data mining (SDM'02), pp 457–473, 2002.