Continuous Features in Inductive Learning and the Effect of RULES Family

Hebah ElGibreen, Mehmet Sabih Aksoy
King Saud University

***Abstract*—**In information system, researchers are usually concerned with understanding the systems. However, due to the excessive growth of computer technologies, handling of large data became a challenge. Thus, simple prediction algorithms can be more helpful than the difficult statistical approaches. Specifically, Inductive Learning can be used to accomplish difficult problem using simple rules or trees. In this field of Machine Learning, methods have been divided into two types: Decision Tree and Covering Algorithms. However, current researchers are starting to focus more on Covering Algorithm due to its outstanding properties and the simplicity of its results. Specifically, one family called RULES was found to be very interesting and its properties seemed appealing. It was found that RULES is one of the most flexible and simplest families and it has high learning rate. Nevertheless, even though RULES is actively improved but it was surprisingly neglected in the conducted surveys, especially with numerical datasets. Yet, complex and real-life problems always contain numerical features. Thus, the purpose of this paper is to extend the Inductive Learning literature and investigate the problem of continuous attributes in RULES and other Inductive Learning families. A theoretical analysis will be conducted to show the effect of numerical values and how it is still an open research area. In addition, an empirical evaluation will also be conducted to show how RULES family can be used as the base of further improvement. Accordingly, this paper can be used as a reference by recent researchers to know what research area is still not covered and need further refinement in Inductive Learning, especially in complex problems that contain numerical values.

***Keywords*—**Inductive Learning, RULES family, Continuous value, Covering Algorithms, Decision Tree

I. INTRODUCTION

IN the past few years, Machine Learning (ML) gained a lot of attention from societies of artificial intelligent. This domain was developed to create autonomous agents who can make the machine act similarly to humans. ML encountered different learning methods depending on the problems that are needed to be solved. One area of ML, where the agent can learn how to train itself in order to make future prediction, is called "Inductive Learning" (IL). Inductive machine learning has been defined in [1] as *" the ability of an agent (like an algorithm) to improve its own performance based on past experience*." In this type of learning the agent is usually provided by previous knowledge as input in order to gain some descriptive knowledge based on the given historical data. Hence, it is a supervised learning paradigm which works as a data analysis tool that uses the knowledge gained through training to produce general conclusion and identify new objects using the produced classifier.

In general, current IL algorithms have been divided into two types: Decision tree (DT) and Covering Algorithm (CA) [2]. Each type of these algorithms has its own purpose, strength, and weakness. DT algorithms are IL methods that discover rules using decision tree. This tree can be used later to represent the rules [3]. On the other hand, CA directly induce rules from the training set based on the concept of separate and conquer. DT attracted a lot of attention in the past few years because of the powerful tree structure. However, researchers started to shift to CA because of the strength of direct rule representation. Thus, several families have been born under the umbrella of CA.

One important family, that is considered as one of the simplest and most flexible CA, is called RULe Extraction System **(**RULES) [4]. RULES family was born in 1995 by Pham and Aksoy [5] to directly induce good rules in a simple manner. It discovers inconsistent rules to allow the coverage of some negative examples, in order to handle noisy data, reduce over-fitting, and increase the flexibility of rule induction. In contrast to the other CA, RULES does not remove covered example but, instead, it marks it as covered. This way, repeating the discovery of the same rule is prevented while coverage accuracy and generality of new rules are preserved. Consequently, this property solves fragmentation and small combination problems; i.e. RULES will avoid data reduction during the learning process and covering small training examples with high error rate.

Nevertheless, it was found that RULES family is surprisingly neglected. Even though researchers have been interested in CA and conducted surveys and empirical studies but RULES was always forgotten. Thus, in [6] a theoretical analysis and empirical study were conducted to compare RULES with other conventional IL methods. The properties of RULES were compared to DT to find that direct induction of rules is more appealing than the use of trees, especially in large and complex data. Moreover, the characteristics of CA families were analyzed and compared to find that RULES can be considered as one of the most flexible families. Hence, it was concluded that RULES is an interesting family, and more research should be done to improve its performance. However, all the test and comparison were only conducted over discrete datasets while numerical values were neglected. Nevertheless, in practical and real-life applications, the collected data usually contain continuous features (attributes). Thus, it is important to extend the study to measure the performance of RULES

Hebah ElGibreen is with IT Department, College of Computer and Information Sciences, King Saud University, hjibreen@ksu.edu.sa
Mehmet Sabih Aksoy is with 2IS Department, College of Computer and Information Sciences, King Saud University, msaksoy@ksu.edu.sa

and other conventional IL methods with numerical values.

Accordingly, the contribution of this paper is to extend the preceding paper conducted in [6] by comparing RULES with other conventional IL methods in numerical datasets. This paper will show the effect of datasets with continuous features in IL, in order to emphasize on what is still missing in this domain. A practical study will also be conducted to analysis and compare the performance of RULES with other classical DT and CA, in order to show how RULES can be used as the base of further improvement and how it surpasses the other families. Based on the theoretical analysis and the empirical study it will be possible to prove the importance of numerical features' problem in CA, and what part of this problem is still an open research area. Hence, this paper can be used as a reference by recent researchers to know what area is still not covered in IL, especially with complex data.

This paper is organized as follows. First, RULES family will be explained and the related work will be discussed to prove the novelty of this paper. After that, continuous features' problem will be theoretically analyzed. Following that, an empirical test will be conducted to test RULES with other DT and CA over numerical values to show its strength and weakness and emphasize the open research questions in the targeted problem. Finally, the paper will be concluded and future work is suggested.

II. RULES FAMILY

RULES family is a CA family that directly induces one rule at a time based on a seed example. It usually applies specialization process to find the best rule. The rule that covers the most positive examples and the least negative examples are chosen as the best rule. Hence, RULES family does not require finding of completely consistent rule. It allows the best rule to cover some negative examples in order to handle noisy data, reduce over-fitting problem, and increase the flexibility of rule induction. Following that, the examples that are positively covered by the discovered rules are marked as covered; however, it is not removed from the training set. This way, repeating the discovery of the same rule is prevented while coverage accuracy and generality of new rules are preserved. At the end, the algorithm is repeated until all examples are covered. Moreover, several versions have been proposed in this family, as follows.

RULES-1 [5]: It is the original RULES algorithm. In this version, not all examples are kept in the memory, and it stored unclassified examples in an array. Thus, its speed was low and badly affected by datasets with a large number of attributes. Moreover, its learning rate was low, where it resulted in a high number of rules.

RULES-2 [7]: It is an improvement of RULES-1, where not all examples are studied together. Instead, it works on one seed example at a time to induce the best rules. Moreover, it allows the user to specify the desired number of rules. Thus, it reduces the resulting rules and it is faster than its predecessor. Finally, additional features were also added to handle missing and numeric values.

RULES-3 [8]: This version contained all the properties of RULES-2 with additional features. It generates more general rules than its predecessor and allows the user to specify the desired level of rule precession.

RULES-3Plus [9]: It is a RULES-3 algorithm with two additional functionality. It searches for the best rule by applying specialization and beam search instead of the exhaustive search. Additionally, it introduced a sorting metric for better selection of candidate rules; where conflict is solved based on the rule generality and accuracy.

RULES-4 [10]: This version is the first incremental version of RULES family. It is a simple algorithm that uses partial memory, where only a portion of the data is stored to induce the new rules. It discovers the rules using RULES-3 Plus and stores it in a Long-Term memory, while rules of the new examples are stored in the Short-Term memory.

RULES-5 [11]: It is the first version that handles continuous attributes without discretization. It defines the interval of each attribute during the rule construction based on the examples' distribution. In particular, with each seed, an attribute interval is chosen to exclude the value of the most similar negative example. This version was also extended in [12] to represent RULES-5+, which improve the performance using a new rule space representation scheme.

RULES-6 [13]: It is a scalable version of RULES family, that is an extension of RULES-3 plus. This version had the advantages of RULES-3 plus in addition to high speed and noise resistant. It scales to large dataset and constrains the search space by employing a beam search and pre-pruning.

RULES-F [14]: It is an extension of RULES-5, where not only continuous attributes are handled but also continuous classes. Fuzzy set theory is applied to handle continuous classes and reduce the overlapping between the rules. It transforms all the attributes into a membership degree to create fuzzy rule set instead of intervals. Moreover, a new rule space representation scheme [12] was also integrated in an a new version called RULES-F+.

RULES-SRI [15]: This version is another scalable RULES algorithm. It selects the best w rules for further search to reduce the search space and make it scalable to a large number of rules. Moreover, rules were pruned based on the hypothesis of general-to-specific partial ordering to further reduce search space.

RULES-IS [16]: It is a new incremental algorithm that was inspired by immune systems. It calls every example an antigen and creates an antibody for each one. The antibody-antigens pairs are stored in Short-Term memory and the size of this partial memory is decided based on the concept of immune system. Thus, every time the algorithm runs incrementally, 5% of the oldest antibody-antigens pairs are removed from the memory.

RULES3-EXT [17]: It is an extension of RULES-3 with four additional features. RULES3-EXT eliminated repeated examples and reduced the number of input file needed to execute the algorithm. Moreover, users were given the ability to change the attribute order when inducing the best rules; and, finally, the algorithm partially discharged rules that cannot fully cover unseen examples.

RULES-7 [18]: It is an extended version of RULES-6, where it does not specialize all the parent rules of the current rule. Instead, it only considers the rules that have greater coverage than a pre-specified value. Moreover, this version applied different control structure to improve its performance, and it removed the duplicate rules during the induction to reduce the algorithm time.

RULES-8[19]: It is an improved version to deal with continuous attribute. RULES-8 discretized the continuous attributes online during the learning process. It re-sorts the dataset by the example class to decide on the split point based on the seed attribute-value pair.

RULES-TL [20, 21]: It is one of the latest versions of RULES family, where incomplete and large datasets are considered. RULES-TL applies Transfer learning to reduce the induction time and handle incomplete data. It was scalable over large and incomplete data.

Ultimately, even though each version has its own properties but most of these versions have some common characteristics. In [6], these characteristics were discussed to find that RULES is very appealing when compared with conventional CA and DT algorithms. However, the case where datasets contains continuous features was not covered. Thus, after the literature review, this type of data will be theoretically and empirically analyzed.

III. Related Work

In the field of CA, several surveys and empirical studies have been conducted through the years. In [22], different CA methods have been tested over several pruning techniques in order to show the effect of pre and post pruning techniques. After that, another survey was conducted in [23], to analyze the characteristics of CA methods. This survey was very interesting and covered a wide range of CA to compare the algorithms based on three biases, namely: language, search, and pruning. However, RULES was neglected, and no empirical test was conducted.

Moreover, in [24], another study was conducted to explain and test different classification techniques. In this work, different ML techniques have been explained separately and then compared together in an empirical study. Alternatively, in [1], a study was conducted to compare DT algorithms with CA and hybrid IL algorithms. Nevertheless, it only tested DataSqueezer with C5 and CLIP4 empirically without any theoretical comparison. In addition, in [25], an experimental evaluation of different CA have been conducted to test its simplicity. However, these methods were explained separately, properties and characteristics were not compared, and RULES family was not considered.

Alternatively, however, Aksoy [4] focused in his study on RULES family and conduct a survey that explained its versions. Nevertheless, this survey was only concerned with RULES family and theoretically explained version one to five only. Hence, no empirical study was conducted and other families of CA were not considered. In [26], different supervised learning techniques were used to test the performance of its algorithm with different discretization techniques. This work was conducted as discretization techniques survey to identify its taxonomy and empirically analyze its performance. However, RULES family was not included in the study. Finally, the preceding version of this paper was conducted in [6], which test RULES family with other conventional families of IL. This study showed the importance of RULES and how it can surpass the other families. However, only discrete datasets was considered while the effect of numerical values was not covered.

Accordingly, it can be noticed from all the studies discussed previously that conventional algorithms of CA and DT have not been studied with RULES and the surveys were either focused on the theoretical or empirical part of the study. Even though the preceding study showed the effect of RULES and other IL families but the data considered was discrete only. Hence, an extended version is needed to consider RULES and compare it with other IL methods in datasets with continuous features. Thus, this paper will theoretically analyze RULES and other IL methods over numerical values, and they will be empirically tested over datasets with continuous features.

IV. Theoretical Study: Continuous Features Problem in Inductive Learning

In rule induction, the dataset can include two types of value: discrete and continuous. The discrete value contains categorical data and has a finite number of values, such as "High, Low, Medium". The continuous value, however, contain a numerical value with an infinite or very large space, such as "1, 2, 3.4, 5.3." Handling of continues values, in general, is a problem and this domain is an active area of research. Consequently, it was suggested to transform continuous attributes into discrete ones. However, deciding what discrete value to assign, the interval length of continuous values for each corresponding discrete value, and the number of intervals is still a problem. Hence, it cannot be said that a perfect method has been found.

Moreover, when it comes to IL algorithms, in specific, they were basically designed to handle discrete values, while continues ones were neglected. Hence, as states in [27], rule induction algorithms perform poorly with continuous values. However, real-life problems mostly contain continuous data rather than discrete. Therefore, different improvements are needed to automatically handle continuous attributes. In IL, the improvements done to deal with continuous features were divided based on the concept of discretization. Specifically, it was possible to divide IL methods that consider continuous features into three types, as follows.

A. *Offline Discretization*

Offline discretization is a pre-processing step that converts continuous attributes to discrete before induction. The main idea is to apply any discretization technique, such as EqualWidth, ChiMerge, or CAIM [26], over the data before applying the rule induction. These discretization methods basically split the values range of the continuous attributes into a fixed number of intervals. Different discretization techniques were applied over CAs; as in SIA [28], ESIA [29], covering and evolutionary algorithms [27], RULES-3+ [30], Prism [31], PrismTCS [32], and supervised dynamic discretization [33]. Nevertheless, it was found that although offline discretization reduces the time, but it can seriously affect the rules' quality resulting from the CA [11]. In specific, there is a great tradeoff between the number of intervals and the consistency of the rule. Such that, choosing small split points would increase the interval size and, hence, reduce consistency; while increasing the split points reduces the interval size and overspecializes the rules.

As a result, there were some attempts to create overlapping intervals, as in EDISC [18]. Nevertheless, it was found that the accuracy result is not efficient enough, and the time complexity of this algorithm is high in some

cases. Hence, creating overlapping intervals using offline discretization might cause more problems. Thus, another method was introduced into rule induction to increase its flexibility. This method was developed based on the fuzzy set theory [34], where the continuous value may belong to multiple intervals, and degree of belonging to a certain interval is measured by a membership function. Different methods have been developed based this concept, as in C4.5 [35] and FR3 [36]. However, it was found that even though fuzzy discretization can improve the accuracy of the rule induction, but it tremendously increases the complexity, and it is usually difficult to understand and apply.

B. *Online Discretization*

Online discretizations assign a fixed number of intervals for the continuous attributes during the learning process. This method tries to solve the problem of offline discretization by increasing flexibility. Online discretization was basically designed for DT algorithms, such as C4.5[37] and CART [38, 39], but at every node it must re-discretize all continuous attributes. Thus, it wastes a lot of time and increases the computational complexity. Consequently, DT algorithms run very slowly with continuous attributes.

As a result, recently online discretization started to occur in CAs. In REP-based family, as Slipper [40] and Ripper [41], an online discretization method was introduced. However, even though they had a good performance, but it needs an intensive computation with every attribute. It encompasses a lot of sorting and needs at least three tables for every attribute. These tables are processed and re-sorted in every loop. Thus, in addition to the computation complexity, this approach might increase the course of dimensionality problem. Additionally, a new family of CA, called Ant-Miner, was developed in [42] to perform a global search over the dataset. It usually uses offline discretization but several improvements have been made to deal with continuous attributes in an online manner; either partially, as in [43], or fully, as in [44, 45]. However, these methods can only deal with dataset that have only continuous attributes. They also need to represent the attributes in a tree before extracting the rules. Moreover, several computations must be conducted and stored to optimize the result. Thus, such method is difficult to understand, highly complex, and might need large memory during the learning process.

Moreover, in [46], online discretization was integrated into RULES-SRI. Instead of examining all the values of an individual, it only examines the boundaries of each attribute during the learning process. Nevertheless, the execution time and re-computation of boundaries with each rule have tremendously increased, regardless of the accuracy improvement. Furthermore, this method does not consider the attributes interdependency and only considered the relationship of the classes. Additionally, RULES was further improved when dealing with continuous attributes through RULES-8. This algorithm increased the computation complexity due to its re-sorting, which might also affect the execution time. Thus, even though it managed to handle noise, but it had the same problems as REP-based family.

C. *None-Discretization*

None-discretization is a new research area that recently started to grow, where intervals are not fixed. It deals with continuous and discrete values in similar manner and tries to discover the best discrete value for an attribute depending on its relationship with other examples and classes. Several attempts have been made to handle continuous attribute in CAs without discretization. A modified version of AQ, called CAQ, was developed in [47] to prove that dealing with continuous attributes as a real number instead of forcing it into discrete representation would lead to more efficient results. However, CAQ did not obtain appropriate ranges because it is affected only by the current example without considering the overall data.

Moreover, RULES family was also improved to directly handle continuous attributes, as in RULE-5. This version defines the interval of each attribute during the rule construction based on the examples' distribution. It was also improved in RULES-5+ to reduce the role of statistical measures by applying a new knowledge representation. However, it was found that the number of rules discovered is too large and caused the problem of overspecialization, and, thus, it became sensitive to noise. In addition, RULES-IS also included a procedure that handled continuous values during the learning process without discretization. During the generation of antibodies, numerical ranges were created with every continuous attribute to cover the positive examples. As a result of the empirical test, it was found that the performance of this method is worse than its predecessor, RULES-3+ and C5.0, over some training data but might be better on future classification. Nevertheless, this algorithm would need a lot of time and computation to match every antigen with all different antibodies.

In [48], however, a system called Brute used a measure of variance to decide on the continuous values' boundaries. It reduced the number of rules by applying rule induction repeatedly over different and overlapping examples using bootstrapping ensemble learning. After that, another version has been developed in [49], which introduce the similarity measure between rules to visualize rule similarity. Nevertheless, this system increased the computational cost due to the reproduction of rules. Moreover, it is questionable if the system can produce stable rules from small data. Additionally, in [50], a new rule based algorithm called uRule was developed to handle uncertain continuous attributes. It is built based on REP-based family and used new heuristics to optimize and prune resulting rules, identify the optimal thresholds, and handle uncertain values. In [51], the empirical result that tested this algorithm was discussed to find that it can handle uncertainty in continuous and discrete attributes. However, it was found that it consumes a lot of time because of the complexity of rule pruning step.

D. *Discussion*

From all above, it can be concluded that even though the preceding study conducted in [6] proved that RULES family has better characteristics than the conventional IL families, and can be empirically better learner but this conclusion was based on discrete data. Moreover, after investigating how continuous features are currently handled, it was found that both RULES and other method in CA and DT is still lacking. In this problem, the methods have some common deficiencies, which can be summarized as follows.

1) **Offline Discretization**: It Does not consider future cases and fixes the intervals in advance. It can cause a major problem in the future, where it is possible that the values of unseen data do not remain in the same distribution. In addition, the update of discretized values can cause another problem with incremental rule induction. It will be difficult to update the interval of older rules and, hence, can reduce the accuracy of the algorithm.

2) **Online Discretization:** It is more accurate than offline discretization. The result of the interval depends on the information gathered during the learning process. Hence, it is context-dependant and can consider the data bias. However, its computational cost is usually very high.

3) **None-Discretization:** It is an area of research that has a bright future, but it has its flaws. The speed of handling continuous attribute is a major problem that could affect its accuracy. It causes overspecialization and makes the results affected by the noise. Moreover, the simplicity of updating the resulting quantization is also an issue; incremental rule induction was not considered.

V. Empirical Study: Continuous Features Evaluation

In the previous section, the significant of continuous features' problem was verified and its gaps were emphasized. However, it is also important to know how RULES family can add in comparison to the other conventional families of IL. Thus, this section will show the result of comparing different versions of RULES family with other IL algorithms in continuous features' problem.

The experiments were conducted on a PC with Intel®Core™ i7 CPU, 2.67 GHz processes, and 6GB RAM. In addition, KEEL tool [52, 53] was used to build the experiments. Moreover, several dataset with different properties are gathered from KEEL repository [53], as illustrated in Table **I**, and the algorithms are validated using 10-fold cross-validation [54]. Note that the preceding paper discretized all datasets in advance to test the algorithms' performance over discrete datasets. However, in this experiment, the datasets are not processed and it contains continuous features.

Table I: Continuous attributes dataset

Dataset	#Examples	#Attributes	#Labels
bupa	345	6	2
cleveland	303	13	5
ecoli	336	7	8
glass	214	9	7
haberman	306	3	2
iris	150	4	3
new-thyroid	215	5	3
pima	768	8	2
vehicle	846	18	4
wisconsin	683	9	2
tic-tac-toe	958	9	2
yeast	1484	8	2

Moreover, four different version of RULES, three DT, and five CA algorithms were included in the experiment. Each algorithm dealt with the continuous features using a different technique, as follows.

- **RULES-5+**: The latest versions of RULES-5, where it deals with continuous features using none-discretization technique.
- **RULES-6:** One of the latest versions of RULES, which is developed to scale over large datasets. This algorithm handle continuous features offline by applying Fayyad and Irani [55] discretization.
- **RULES-SRI:** One of the latest versions of RULES, which is proposed to improve the scalability. It also handles continuous features offline by applying Fayyad and Irani discretization.
- **DataSqueezer [56]:** One of the latest versions of DataSqueezer family that is fast, supervised, greedy, and simple algorithm, but it required the existence of all class labels. Fayyad and Irani offline discretization technique is applied to handles continuous features.
- **Ripper:** A REP-based family algorithm that produced error reduction. This algorithm handles continuous features using online discretization, where it tests all values of an attribute online and choosing the most appropriate one.
- **AQ15 [57]:** The mostly known and used version of AQ to handle noisy and overlapping data. Fayyad and Irani offline discretization technique is applied to handles continuous features.
- **CN2 [58]:** The original version of CN2 that is mostly known and used in CA; it is an algorithm that combines the good properties of AQ and ID3 family. To handle continuous features, Fayyad and Irani offline discretization technique is applied.
- **PRISM:** The original version of PRISM family that is usually used as the base of any other versions of PRISM and was developed as a competitor to ID3 algorithm. To handle continuous features, Fayyad and Irani offline discretization technique is applied.
- **C4.5:** A DT algorithm that is usually used to benchmark the other classification methods. It deals with continuous features online, where it re-discretize all continuous features at every node.
- **PUBLIC [59]:** A DT algorithm that integrates pruning technique during tree construction in order to handle the over-fitting problem. It deals with continuous features online by splitting the nodes while building the tree.
- **CART [38]:** A DT algorithm that integrates regression in addition to classification in order to deal with continuous values without the need for pre-discretization, i.e. it applies online discretization similarly to C4.5.

In addition, after conducting the experiment, and to visualize the result, different statistical analysis measures were recorded. Specifically, to determine the performance of the tested algorithms, the following measures are recorded.

- **Error Rate:** This measure records the error rate of applying the resulting model over the test set; where less error indicates better performance. Hence, it shows if the algorithm is applicable over the test partition.
- **Time Test:** This measure records the time interval, between generating the result of the first and last partition of the dataset. Hence, it indicates the time taken by the algorithm to process the whole dataset.

- **Learning Rate:** This measure records the learning rate of the algorithms by collecting the rule set size at the end of the algorithm. A small rule set size indicates that the result is not overspecialized and, hence, it will be possible to learn new knowledge rather than remembering the data. Therefore, a smaller rule set size indicates better learning rate and better performance.

Based on these measurements, datasets, and algorithms it is possible to assess the performance of RULES in continuous features' problem. Nevertheless, it was noticed that the performance of every algorithm is different depending on the dataset property. Moreover, RULES3-Ext could not handle datasets with a large number of attributes. It took a week to execute the algorithm without finishing. Hence, RULES3Ext is not applicable to datasets with a large number of attributes and, thus, it was excluded in this case.

Starting from the error rate, as shown in Table II, it can be seen that RULES in Bupa, Cleveland, Iris, Tic-Tac-Toe, and Vehicle datasets has the lowest error rate and, accordingly, result in the most accurate model. Specifically, in the first four datasets mentioned before RULES5+ has the best error rate while in Vehicle RULES-6 has the lowest error rate.

Moreover, in Ecoli, Glass, New-Thyroid, Pima, and Wisconsin datasets, RULES family was not far away from the best error rate. Specifically, in Ecoli RULES-5+ has 20% error rate while the least error rate resulted from CN2 is 19%. In Glass dataset, RULES-5+ resulted in 31% error rate while the least error rate resulted from C4.5 is equal to 30%. In New-Thyroid, RULES has 0.06 while the best error rate in CART is 0.05. In addition, in Pima, RULES-5+ has 27% error rate while the least error rate resulted from PRISIM and C4.5 is equal to 25%. Finally, in Wisconsin, RULES-5+ has 4% error rate while the least error rate resulted from PRISM is 2% and also Ripper gave equal error rate to RULES in this dataset.

Nevertheless, when it comes to the rest of the datasets including Haberman and Yeast, RULES is worse than most of the algorithms. For some reason, all versions of RULES family gave worse error rate than some of the other families. Specifically, in Haberman dataset Datasqueezer, CN2, PRISIM, PUBLIC, CART and C4.5 are better than all versions of RULES family. Moreover, in Yeast dataset CN2, Datasqueezer, and PRISIM exceeded RULES accuracy.

Thus, it can be concluded that RULES family can have better performance than DT and other conventional families in CA. When considering the average mean of all datasets, presented in the last row of Table II, it can be noticed that RULES-5+ has the least error rate. Moreover, from the versions presented in the table, it can be concluded that using none-discretization techniques to deal with continuous attributes (as in RULES-5+) can be better than online and offline discretization techniques. Specifically, it can be noted that RULES-5+ exceeded the performance of RULES-6, SRI, and 3EXT (which uses offline discretization) in most datasets. In addition, in Ripper, C4.5, CART and PUBLIC (which use online discretization) RULES-5+ has also better error rate in most datasets.

Nevertheless, having a good accuracy is not enough to measure the total performance. The generality of the resulting rules is also important to know if the learning rate is good enough for future changes and noise in the data. Therefore, the number of rules resulted from each algorithm is also recorded, as in Table III. In general, it can be seen that Datasqueezer has the best learning rate in all datasets. It resulted in only five rules on average, which is relatively low when compared to the other algorithms. This is probably because this family uses only hill climbing and hybrid pruning. Moreover, when considering the average performance of the algorithms over all datasets, it can be noticed that CA usually have better learning rate than DT, except for PUBLIC. However, even though RULES-5+ resulted in the best error rate but its learning rate is badly affected. Thus, it can be concluded that even though the use of none-discretization technique can improve the accuracy of the algorithm, but it can badly affect its learning rate.

In addition to the learning rate, time is also important. Therefore, the time spent in each dataset set was recorded for every algorithm, as illustrated in Fig. 1. Note that CART is not shown in the graph due to its lack of speed. It took it from several minutes to several hours to finish its execution. Thus, its representation affected the clarity of the graph. However, it can be concluded that CART algorithm is a time consuming DT algorithm.

On the other hand, from Fig. 1, it can be noticed that, regardless of the dataset properties, all versions of RULES in addition to PRISIM, AQ, and Datasequeezer have a relatively low execution time. Hence, CA can finish its induction faster than DT algorithms. Only CN2 and Ripper are noticeably affected by the datasets. In CN2, the speed is similar to the other CA in all datasets except of Wisconsin, where it surprisingly has an increase in time. However, in Ripper, the execution time is worse than the other DT and CA in almost all datasets. On average, its execution time is high in all datasets and it further increases on datasets with a larger number of examples or attributes. Finally, even though RULES-5+ has similar speed as the other methods with offline discretization technique, but it actually is slightly slower in total. Specifically, datasets that has a relatively large number of attributes and examples, as in Pima, Vehicle and Yeast, slightly affected its speed. Thus, none-discretization might affect the methods' speed when compared with offline discretization. However, it is still better than the time spent by methods with online discretization techniques.

Nevertheless, when considering DT algorithms it can be noticed that their speed is worse than CA algorithms. In C4.5, the execution time is affected by the type of datasets. Even though the speed is high in some cases, but it becomes slower in Buba, Glass, Pima, Vehicle, and Yeast datasets. Moreover, in PUBLIC, the execution time is relatively high in all datasets except for Bupa, Glass, Iris, and Tic-Tac-Toc. It even got worse with datasets that have a large number of attributes or examples, as in Vehicle and Yeast datasets. Additionally, as stated before, CART algorithm resulted in the worst speed of all algorithms presented. The reason of such behavior is because of the excessive splitting in DT. Thus, it can be said that DT methods' speed is more affected by the datasets properties than CAs.

Table II: Average error rate

Datasets	C45	CART	PUBLIC	PRISM	CN2	AQ	Ripper	DataSqueezer	Rules-6	RULES-SRI	RULES-3Ext	RULES5+
bupa	0.33	0.34	0.35	0.41	0.41	0.58	0.36	0.42	0.62	0.67	0.42	0.30
cleveland	0.48	0.49	0.45	0.45	0.45	0.43	0.54	0.46	0.47	0.45	---	0.41
ecoli	0.21	0.23	0.22	0.22	0.19	0.24	0.27	0.47	0.51	0.37	0.27	0.20
glass	0.30	0.34	0.36	0.31	0.33	0.43	0.38	0.64	0.65	0.60	0.58	0.31
haberman	0.27	0.28	0.29	0.28	0.28	0.74	0.49	0.26	0.31	0.74	0.46	0.31
iris	0.05	0.05	0.06	0.08	0.07	0.15	0.06	0.09	0.19	0.17	0.16	0.03
new-thyroid	0.07	0.05	0.09	0.09	0.08	0.10	0.06	0.13	0.15	0.11	0.08	0.06
pima	0.25	0.31	0.26	0.25	0.28	0.32	0.29	0.35	0.58	0.35	0.48	0.27
vehicle	0.27	0.39	0.29	0.30	0.40	0.35	0.30	0.58	0.32	0.30	0.20	0.24
wisconsin	0.06	0.05	0.05	0.02	0.07	0.06	0.04	0.29	0.34	0.47	---	0.04
tic-tac-toe	0.15	0.26	0.07	0.02	0.30	0.05	0.33	0.43	0.12	0.18	0.08	0.00
yeast	0.42	0.55	0.43	0.25	0.25	0.59	0.50	0.29	0.51	0.71	0.37	0.41
Average	**0.24**	**0.28**	**0.24**	**0.22**	**0.26**	**0.34**	**0.30**	**0.37**	**0.40**	**0.43**	**--**	**0.21**

Table III: Average number of rules

Datasets	C45	CART	PUBLIC	PRISM	CN2	AQ	Ripper	DataSqueezer	Rules-6	RULES-SRI	RULES-3Ext	RULES5+
bupa	29	58.8	2	2	2	2	22	2	2	3	3	45
cleveland	42	51.5	11	71	13	79	41	5	30	20	---	48
ecoli	20	26.9	14	46	18	46	36	8	29	27	69	36
glass	24	12.5	11	40	15	31	22	6	29	21	55	27
haberman	3	90	2	2	2	3	19	2	2	4	4	63
iris	5	14.2	7	8	6	9	6	3	7	7	11	8
new-thyroid	8	23.7	8	16	7	12	6	3	11	11	20	8
pima	24	74.9	5	76	23	66	26	2	23	18	126	80
vehicle	66	76	27	258	45	205	48	4	29	19	240.8	72
wisconsin	13	41.5	10	14	8	10	9	3	101	66	---	17
tic-tac-toe	85	94	25	49	8	61	68	15	27	20	62	23
yeast	161	90	37	45	20	45	140	2	19	15	76	267
Average	**40**	**54.5**	**13**	**52**	**14**	**47**	**37**	**5**	**26**	**19**	**--**	**58**

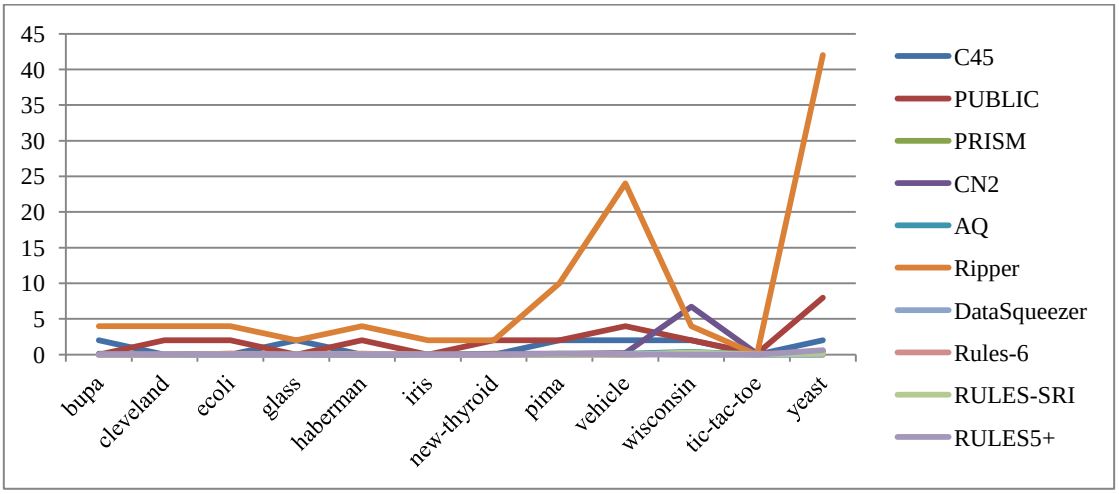

Fig. 1: Average time with continuous attributes in minutes

Ultimately, from all above, it can be said that RULES family is one of the most flexible families in CA. It was previously proven that this family is very promising to be used with discrete datasets. However, when it comes to data with numeric features, the performance is different. From the empirical study conducted in this paper, it was found that CA need further improvement when the dataset contain numerical values. Specifically, when the dataset contain continuous features the following points can be summarized.

1) CA can result in better accuracy, learning rate, and speed comparing to DT, regardless whether the technique used is discretization or none-discretization.

2) When it comes to the error rate, algorithms with none-discretization technique had better accuracy than most methods with offline discretization, especially in the same family. However, its accuracy was similar to the methods that used online discretization techniques.

3) When it comes to the learning rate, it was found that none-discretization techniques could badly affect the learning rate of the algorithm. The number of rules resulted from RULES-5+ was much higher than most algorithms with either online or offline discretization.

4) When it comes to the speed, even though RULES-5+ implementation was not optimized but its execution time was still better than the methods that used online discretization. However, its performance was affected by the datasets properties, where the speed was lower than the methods with offline discretization. Hence, it can be said that none-discretization techniques can be faster than online discretization but it still needs further improvement to surpass the offline discretization speed.

5) RULES-3Ext was not scalable to datasets with a large number of attributes but newer versions such as RULES-5+, 6, and SRI solved this problem.

Accordingly, it can be noticed that the empirical result emphasizes the conclusion discussed in the theoretical analysis, presented in the previous section. Thus, IL needs further improvement when dealing with continuous features. Moreover, because of RULES family properties, discovered in the preceding paper, it would be a good idea to make such improvement using this family. The none-discretization technique developed in RULES family can be further generalized to fill the gaps of continuous features.

VI. Conclusion

Due to the growth of interest in CA, different families have been developed. Based on these families, different surveys were also conducted. However, RULES family was found to be neglected; even though a preceding study was conducted to find the importance of RULES family but discrete dataset was only considered. Hence, in addition to the neglect, the effect of numeric values over difference IL families was also missing. Hence, the contribution of this paper was to extended the preceding version to show the effect of RULES over datasets with numeric values and be used as a reference by recent researchers to know what research area is still not covered and need further refinement in IL. From the theoretical analysis, it was found that even though RULES family showed better properties and characteristics than the other conventional families of IL but it is lacking when applied over datasets with continuous features. Moreover, as a result of the empirical evaluation it was concluded that RULES family can have better performance than the other families in IL due to the use of none-discretization technique. None-discretization is a good option to deal with continuous features, but its speed and generality need further improvement. Thus, continuous features' problem is still an open research area in CA, and RULES can be used as the base of further improvement.

Acknowledgement

This research project was supported by a grant from the "Research Center of the Female Scientific and Medical Colleges", Deanship of Scientific Research, King Saud University. We also thank Dr. Samuel Bigot for his great cooperation in providing us with RULES-5+.

Reference

[1] K. J. Cios, R. W. Swiniarski, W. Pedrycz, L. A. Kurgan, K. Cios, R. Swiniarski, and L. Kurgan, "Supervised Learning: Decision Trees, Rule Algorithms, and Their Hybrids " in *Data Mining*, ed: Springer US, 2007, pp. 381-417.

[2] D. Pham and A. Afify, "Machine-learning techniques and their applications in manufacturing," *Proceedings of the I MECH E Part B Journal of Engineering Manufacture*, vol. 219, pp. 395-412, 2005.
[3] F. Stahl and M. Bramer, "Computationally efficient induction of classification rules with the PMCRI and J-PMCRI frameworks," *Knowledge-Based Systems*, 2012.
[4] M. S. Aksoy, "A review of rules family of algorithms," *Mathematical and Computational Applications*, vol. 13, pp. 51-60, 2008.
[5] D. T. Pham and M. S. Aksoy, "RULES: A simple rule extraction system," *Expert Systems with Applications*, vol. 8, pp. 59-65, 1995.
[6] H. ElGibreen and M. S. Aksoy, "RULES Family: Where does it stand in Inductive Learning?," in *8th International Conference on Computer Engineering and Applications*, Tenerife, Spain, 2014.
[7] D. T. Pham and M. S. Aksoy, "An algorithm for automatic rule induction," *Artificial Intelligence in Engineering*, vol. 8, pp. 277-282, 1993.
[8] D. T. Pham and M. S. Aksoy, "A new algorithm for inductive learning," *Journal of Systems Engenering*, vol. 5, pp. 115-122, 1995.
[9] D. T. Pham and S. S. Dimov, "The RULES-3 Plus inductive learning algorithm," in *In Proceedings of the Third World Congress on Expert Systems*, Seoul, Korea, 1996, pp. 917–924.
[10] D. T. Pham and S. S. Dimov, "An algorithm for incremental inductive learning," *Journal of Engineering Manufacture*, vol. 211, pp. 239-249, 1997.
[11] D. Pham, S. Bigot, and S. Dimov, "RULES-5: a rule induction algorithm for classification problems involving continuous attributes," in *Institution of Mechanical Engineers*, 2003, pp. 1273-1286.
[12] S. Bigot, "A new rule space representation scheme for rule induction in classification and control applications," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 2011.
[13] D. T. Pham and A. A. Afify, "RULES-6: A Simple Rule Induction Algorithm for Supporting Decision Making," presented at the 31st Annual Conference of IEEE Industrial Electronics Society (IECON '05), 2005.
[14] D. T. Pham, S. Bigot, and S. S. Dimov, "RULES-F: A fuzzy inductive learning algorithm," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 220, pp. 1433-1447, 2006.
[15] A. A. Afify and D. T. Pham, "SRI: A Scalable Rule Induction Algorithm," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 220, pp. 537-552, 2006.
[16] D. T. Pham and A. J. Soroka, "An Immune-network inspired rule generation algorithm (RULES-IS)," in *Third Virtual International Conference on Innovative Production Machines and Systems*, WhittlesDunbeath, 2007.
[17] H. I. Mathkour, "RULES3-EXT Improvement on RULES-3 Induction Algorithm," *Mathematical and Computational Applications, Vol. 15, No. 3, pp. , 2010*, vol. 15, pp. 318-324, 2010.
[18] K. Shehzad, "EDISC: A Class-tailored Discretization Technique for Rule-based Classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, pp. 1435-1447, 2012.
[19] D. Pham, "A novel rule induction algorithm with improved handling of continuous valued attributes," Doctor of Philosophy, School of Engineering, Cardiff University, Cardiff, 2012.
[20] H. ElGibreen and M. S. Aksoy, "RULES – TL: A simple and Improved RULES Algorithm for Incomplete and Large Data," *Journal of Theoretical and Applied Information Technology*, vol. 47, 2013.
[21] H. ElGibreen and M. S. Aksoy, "Multi Model Transfer Learning with RULES Family," in *International Conference on Machine Learning and Data Mining MLDM´2013*, New York, 2013.
[22] J. Fürnkranz, "Pruning Algorithms for Rule Learning," *Machine Learning*, vol. 27, pp. 139-172, 1997/05/01 1997.
[23] J. Fürnkranz, "Separate-and-Conquer Rule Learning," *Artificial Intelligence Review*, vol. 13, pp. 3-54, 1999/02/01 1999.
[24] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," presented at the Proceedings of the 2007 conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies, 2007.
[25] U. Ruckert and L. Deraedt, "An experimental evaluation of simplicity in rule learning," *Artificial Intelligence*, vol. 172, pp. 19-28, 2008.
[26] S. Garcıa, J. Luengo, J. A. Saez, V. Lopez, and F. Herrera, "A Survey of Discretization Techniques: Taxonomy and Empirical Analysis in Supervised Learning," *IEEE Transactions on Knowledge and Data Engineering*, 2012.
[27] C. Chiu and N. S. Chiu, "An adapted covering algorithm approach for modeling airplanes landing gravities," *Expert Systems with Applications*, vol. 26, pp. 443-450, 2004.
[28] S. W. Wilson, "Classifier systems and the animat problem," *Machine Learning*, vol. 2, pp. 199-228, 1987.
[29] J. J. Liu and J. T.-Y. Kwok, "An Extended Genetic Rule Induction Algorithm," in *Proceedings of the 2000 Congress on Evolutionary Computation* La Jolla, CA, 2000, pp. 458-463.
[30] T. Pham and S. S. Dimov, "An efficient algorithm for automatic knowledge acquisition," *Pattern Recognition*, vol. 30, pp. 1137–1143, 1996.
[31] J. Cendrowska, "PRISM: An algorithm for inducing modular rules," *International Journal of Man-Machine Studies*, vol. 27, pp. 349-370, 1987.
[32] F. Stahl, M. Bramer, and M. Adda, "PMCRI: A Parallel Modular Classification Rule Induction Framework," in *Machine Learning and Data Mining in Pattern Recognition*. vol. 5632, P. Perner, Ed., ed: Springer Berlin / Heidelberg, 2009, pp. 148-162.
[33] F. Min, Q. Liu, H. Cai, and Z. Bai, "Dynamic Discretization: A Combination Approach," presented at the International Conference on Machine Learning and Cybernetics, Hong Kong, 2007.
[34] L. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338–353, 1965.
[35] M. E. Cintra, M. C. Monard, and H. de Arruda Camargo, "An Evaluation of Rule-Based Classification Models Induced by a Fuzzy Method and Two Classic Learning Algorithms," pp. 188-193, 2010.
[36] J. C. Huhn and E. Hullermeier, "FR3: a fuzzy rule learner for inducing reliable classifiers," *Trans. Fuz Sys.*, vol. 17, pp. 138-149, 2009.
[37] J. R. Quinlan, *C4.5: Programs for Machine Learning*: Morgan Kaufmann, 1993.
[38] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*: Chapman and Hall (Wadsworth, Inc.), 1984.
[39] D. Steinberg, "CART: Classification and Regression Trees," in *The Top Ten Algorithms in Data Mining*, X. Wu and V. Kumar, Eds., ed: Taylor & Francis Group, LLC, 2010, pp. 179-201.
[40] W. W. Cohen and Y. Singer, "A simple, fast, and effective rule learner," in *Sixteenth National Conference on Artificial Intelligence*, 1999, pp. 335-342.
[41] E. Frank and I. H. Witten, "Generating Accurate Rule Sets Without Global Optimization," presented at the Fifteenth International Conference on Machine Learning, 1998.

[42] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas, "An ant colony algorithm for classification rule discovery," *Data Mining: A Heuristic Approach*, vol. 208, 2002.
[43] S. Swaminathan, "Rule induction using ant colony optimization for mixed variable attributes," Master, Computer Science, Texas Tech University, Texas 2006.
[44] F. E. Otero, A. A. Freitas, and C. G. Johnson, "cAnt-Miner: An Ant Colony Classification Algorithm to Cope with Continuous Attributes," presented at the Proceedings of the 6th international conference on Ant Colony Optimization and Swarm Intelligence, Brussels, Belgium, 2008.
[45] K. M. Salama, A. M. Abdelbar, F. E. B. Otero, and A. A. Freitas, "Utilizing multiple pheromones in an ant-based algorithm for continuous-attribute classification rule discovery," *Applied Soft Computing*, vol. 13, pp. 667-675, 2013.
[46] D. T. Pham and A. A. Afify, "Online Discretization of Continuous-Valued Attributes in Rule Induction," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 219, pp. 829-842, 2005.
[47] B. L. Whitehall, S. C. Y. Lu, and R. E. Stepp, "CAQ: A machine learning tool for engineering," *Artificial Intelligent Engineering*, vol. 5, pp. 189–198, 1990.
[48] L. R. Waitman, D. H. Fisher, and P. H.King, "Bootstrapping Rule Induction," in *Third IEEE International Conference on Data Mining (ICDM'03)*, 2003, pp. 677- 680.
[49] L. Waitman, D. Fisher, and P. King, "Bootstrapping rule induction to achieve rule stability and reduction," *Journal of Intelligent Information Systems*, vol. 27, pp. 49-77, 2006.
[50] B. Qin, Y. Xia, R. Sathyesh, S. Prabhakar, and Y. Tu, "uRule: A Rule-based Classification System for Uncertain Data," presented at the IEEE International Conference on Data Mining Workshops (ICDMW), Sydney, NSW, 2010.
[51] B. Qin, Y. Xia, and S. Prabhakar, "Rule induction for uncertain data," *Knowledge and Information Systems*, vol. 29, pp. 103-130, 2010.
[52] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. d. Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, and F. Herrera, "KEEL: A Software Tool to Assess Evolutionary Algorithms to Data Mining Problems," *Soft Computing*, vol. 13, pp. 307-318, 2009.
[53] J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework," *Journal of Multiple-Valued Logic and Soft Computing* vol. 17, pp. 255-287, 2011.
[54] B. Efron and R. Tibshirani, *An Introduction to the Bootstrap*. USA: Chapman & Hall, 1993.
[55] U. M. Fayyad and K. B. Irani, "Multi-interval discretization of continuousvalued attributes for classification learning," presented at the 13th International Joint Conference of Artificial Intelligence, 1993.
[56] L. A. Kurgan, K. J. Cios, and S. Dick, "Highly Scalable and Robust Rule Learner: Performance Evaluation and Comparison," *IEEE SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS*, vol. 36, pp. 32-53, 2006.
[57] R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac, "The Multi-Purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains," in *Fifth National Conference on Artificial Intelligence*, Philadelphia, 1986, pp. 1041-1045.
[58] P. Clark and T. Niblett, "The CN2 induction algorithm," *Machine Learning*, vol. 3, pp. 261-283, 1989.
[59] R. Rastogi and K. Shim, "PUBLIC: A Decision Tree Classifier that Integrates Building and Pruning," *Data Mining and Knowledge Discovery*, vol. 4, pp. 315-344, 2000.