Journal of Theoretical and Applied Information Technology
10th January 2013. Vol. 47 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

RULES – TL: A SIMPLE AND IMPROVED RULES ALGORITHM FOR INCOMPLETE AND LARGE DATA

HEBAH ABDULAZIZ ELGIBREEN[1], MEHMET SABIH AKSOY[2]

[1]IT Department, College of Computer and Information Sciences, King Saud University, Riyadh 11413, Saudi Arabia
[2]IS Department, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia
E-mail: [1] hjibreen@ksu.edu.sa , [2] msaksoy@ksu.edu.sa

ABSTRACT

In every aspect of life, many information is flowing around in which human can take advantage of. Thus, machine learning was born to gather such information and learn how it can autonomously deal with a certain problem. Different domains have grown to serve different purposes in the machine learning field. One of the mostly active domains in this field is Inductive Learning. One algorithm family developed in inductive learning is RULES. Specifically, it is a covering algorithm family where rules are directly induced from a given set of examples. However, two major deficiencies were found in the algorithms that belong to this family. They need to tradeoff between time and accuracy while searching for the best rule and incomplete data were inappropriately handled. Consequently, this paper proposes a new algorithm that is built based on RULES-6. It uses an advance machine learning method called "transfer learning" to gather the knowledge of other agents in different domains and use it as the base knowledge that would reduce the time of search. In addition, the transferred rules are also used to fill missing classes in order to consider not only the labels available in the target task but also the possibility of future cases. Finally, the performance of the proposed algorithm will be tested and compared to other rule induction algorithms to prove that it actually improved the accuracy, reduced the error rate, and consumed a small amount of time.

Keywords – *Rules Induction, Transfer Learning, RULES Family, Covering Algorithms, Inductive Learning*

1. INTRODUCTION

In order to predict future activities and induce general conclusion a field of machine learning, called inductive learning, was introduced. It is a form of data analysis that uses the knowledge gained through training to build a classifier that concludes a general conclusion in the form of rules. Different methods have been proposed to induce classification rules. These methods were divided into two main types: Divide-and-Conquer (Decision Tree) and Separate-and-Conquer (Covering). Divide-and-conquer algorithms, such as ID3 [1] and C4.5 [2], are classification methods that discover rules using decision tree. This tree can be used later to represent the rules [3]. This type of algorithms drew a lot of attention in the past few years because of the simplicity of deriving a rule from the convenient data structure of trees. It only needs to create the rules from each path through the tree. However, handling decision tree created greater problems and badly affected the process of rule induction.

One of the main problems of the divide-and-conquer algorithms is the difficulty of representing some rules in the tree. In specific, it will be difficult to directly induce rules that have no common attributes from the tree and, instead, some attributes will be redundant and irrelevant [4]. Moreover, these algorithms caused the replication problem, where it is possible to re-learn the same sub-trees in different branches. In addition, when the tree is very big, understanding and handling of such a tree will be a headache. Hence, divide and conquer methods might lead to a confusing and large decision tree without any reason [5].

On the other hand, the properties of inducing rules directly from the dataset make it more preferable than the decision tree structure. These properties have been summarized in [6] into four main properties. First, representing the rules as "IF ... THEN" statement makes it more readable than trees, i.e. more understandable and comprehensible by peoples. Moreover, in many cases, it has been empirically proven that rule learning is better than decision trees. Additionally,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

the resulting rules can be easily used in any expert system and can be directly stored in any knowledge-based system. Finally, it is easy to analyze and modify the induced rules due to their independency. Thus, any rule can be understood and validated without the need to reference other rules in the repository.

As a result, researchers have recently tried to improve covering algorithms to be comparative or even better than divide-and-conquer methods. Thus, different families have been born for this purpose, such as AQ [7], CN2[8], RIPPER [9], RULES [10], Prism [11, 12], and ELEM2 [13]. Nevertheless, RULe Extraction System **(**RULES) was distinguished from the others because of its simplicity. It allows for the discovery of inconsistent rules to automatically obtain partial immunity to noise. Furthermore, RULES family is differentiated from the other algorithms because of its preservation of the covered examples, where it is only marked without deletion. Hence, it will avoid repetitive computation, and the result will be more generalized. Thus, it will resist fragmentation and small combination problem. Consequently, it can manage the reduction of data during the learning process and resist rules that cover small training examples with high error rate.

So far, different versions of RULES family have been proposed by different authors. It started with version one to seven in addition to EXT, SRI and IS versions of RULES family. Each version in the RULES family was proposed for a certain purpose. Nevertheless, most of them have one common property. Specifically, when searching for the best rule, a rule is induced starting from empty one, then a specialization process begins to create more specialized rules from the seed example. This process was proposed to improve the accuracy of rule induction, but it consumes a lot of time and needs special pruning procedures to stop the search and reduce its space. Moreover, it is difficult to decide what to specialize and when to stop and as stated in [14], searching with specialization is usually considered as one of the time-consuming methods. Hence, it causes the needs to tradeoff between the accuracy of the rule induction process and its speed during the search for the best rules.

Moreover, another problem that was sought in RULES family is the way it handles incomplete data. Examples that have a missing class are either neglected or filled based on other examples in the training set. However, when the example is neglected it is possible to lose important information and decrease the performance of the algorithm. On the other hand, when the current examples are only considered, then the available classes are only reflected while, in reality, other labels might also be missing from the training set. Hence, not only the current data is important to assign the right label but also future cases should also be considered to increase the resistance to noise and consider future possible cases, even if the data is incomplete.

Nevertheless, past knowledge of other agents can be used to solve the problems of RULES family. It can be learned from different tasks to be used as the base knowledge of the current problem. This base knowledge can reduce the specialization time because it will not be necessary to always start from an empty rule. In addition, it can be used to fill missing classes, where future cases are also considered because the knowledge is taken from different and related domains.

Thus, the purpose of this paper is to propose a new RULES version that improves the searching procedure and incorporates a new scheme to handle incomplete data. Specifically, Transfer Learning [15] will be applied to transfer rules discovered by other agents based on different but related tasks. These transferred rules will be used as the base knowledge to fill the incomplete data with all possible labels, whether it occurred in the training set or in the past knowledge base. Moreover, transfer rules are also used to reduce the time of specialization and guide the searching process by using past knowledge instead of pure pruning.

This paper is organized as follows. First, the background needed to understand the proposed algorithm, including RULES and Transfer Learning, is presented. Then, the related work is discussed to prove the significant of the proposed algorithm. After that, the proposed method is presented and its details are explained. Following that, the algorithm is tested and its empirical result that compares the proposed algorithm with other covering algorithms is explained and discussed. Finally, the paper is concluded and future work is presented.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2. BACKGROUND

This section explains the background needed to understand the proposed algorithm. It discusses RULES family, and the way it works, in addition to Transfer Learning.

2.1 Rule Extraction System - RULES

RULES family is one of the covering algorithms that directly induces rules from the training set based on the concept of separate and conquer. It is considered as one of the simplest and precise families. It induces one rule at a time based on a seed example and then applies specialization process to search for the best rules. The rule that covers the most positive and least negative examples are chosen as the best rule of the current seed example. Hence, RULES family does not require finding of completely consistent rules. 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.

After that, 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 will be preserved. At the end, the algorithm is repeated until all examples are covered. Consequently, considering the whole training set, while training, will make RULES resist fragmentation, where the data reduces during the learning process, and small combination problem, where discovered rules covers only small training examples with high error rate.

Nevertheless, the most important part of RULES searching strategy is the rule forming process [16]. Rule forming process is the searching strategy that aims to create the best rule to cover a certain example. In specific, it searches for the best conjunction of conditions and measures its strength using a certain heuristics; which make this step very costly. Thus, RULES family order the search based on the concept of specialization (general-to-specific search).

The main idea of specialization is to start from the most general rule, i.e. null rule, and then specialize it by adding one attribute at a time as an additional condition. Nevertheless, this process consumes a lot of time and greatly affects the algorithm performance. In this process, a stopping condition must be applied to stop the specialization and reduce the searching time. However, it is difficult to decide what to specialize and when to stop. It is possible that some important specialization might be neglected while focusing on trivial ones because the stop condition is too early, and it is also possible to consume a lot of time to seek good results because the stop condition is far away. Thus, it needs to tradeoff between the speed of the search and its accuracy.

Consequently, it can be concluded, from all above, that RULES family has a good future concerning rule discovery. However, it needs further improvement. Thus, this paper tries to improve the specialization and rule discovery process of RULES family using an advance machine learning method called Transfer Learning, as will be explained next.

2.2 Transfer Learning

Transfer Learning (TL) is one area of machine learning that makes the agent learn by connecting different but related environments [17]. Thus, instead of starting from scratch, the agent will be able to direct its learning rather than randomly explore the problem [18], as shown in Figure 1. For example, in real life, a person can learn how to read a French text using his own knowledge of English characters. Even though French and English are different languages, but they have related context that a person can re-use to minimize the time needed for the learning process.

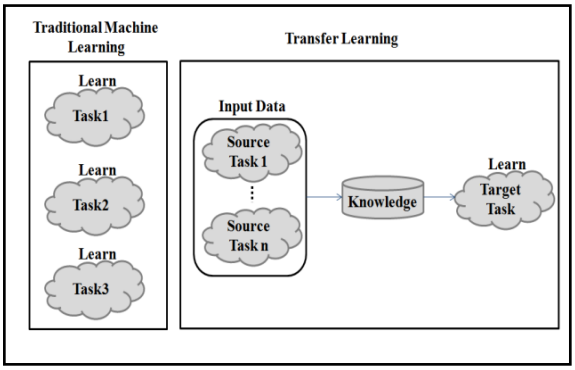

Figure 1: Traditional machine learning vs. TL

The main motivation that caused the birth of this kind of learning was the need to imitate how human could apply previously learned knowledge to improve the lifelong learning methods of machine learning [19]. Such interest started to increase from the mid 90s, but in 2005 TL has been improved to find and apply previously learned skills and knowledge to solve new tasks. Thus, Sinno Jialin and Qiang [15] indicated that the aim

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

of TL is to *"extract the knowledge from one or more source tasks and applies the knowledge to a target task."*

TL approaches, in general, have been divided into different types depending on the state of the class labels, as explained by Sinno Jialin and Qiang [15]. Moreover, different types of transfer can occur depending on what to transfer. It can transfer instances, feature representation, parameters, or even relationship knowledge between the source and target tasks depending on the problem at hand [20].

3. RELATED WORK

In RULES family, different problems have been targeted, and different methods have been proposed along with it. The early versions of RULES family, specifically RULES-1 [21], 2 [22], and 3 [23], were basically proposed to enhance the performance of covering algorithms and induce better rules using a simple algorithm. However, RULES-3+ [24], afterwards, was proposed to improve the performance of RULES-3. This version was very famous and was used as the base for many new algorithms. It is different from the previous version in the way it searches for the best rule and in sorting and selecting the candidate rules, where specialization and H measure were applied.

After that, RULES-4 [25], which is based on RULES3+, and RULES-IS [26] were developed for incremental learning. In addition, RULES-5 [16], RULES-F [27], RULES-5F [28] was also developed based on RULES3+ to handle continuous values during the learning process. After that, scalability became a concern so RULES-6 [29], 7 [30], and SRI [31] were developed to enhance the performance of RULES family and increase its speed.

Nevertheless, even though these methods served different purposes and were designed by different authors, but most of the versions' search was based on the concept of specialization, which caused the need to tradeoff between the accuracy and time. However, one version of RULES family, specifically RULES-IS, did not have this property since it was built based on the immune system network. However, it lost the simplicity property that makes RULES family more appealing than the others. Moreover, in all visions of RULES algorithm, incomplete data were inappropriately handled. Examples that contained missing classes were removed or current examples were used to fill it without considering future cases.

Hence, it can be concluded that RULES family is still lacking in different areas. In general, two main deficiencies can be sought; which is the tradeoff when searching for the best rule and dealing with incomplete data. Specialization, as discussed before, is important to surpass the problem of irrelevant conditions. However, it consumes a lot of time and highly affects the algorithm performance, where it is difficult to decide what to specialize and when to stop. Additionally, incomplete data was inappropriately handled without considering the future.

Nevertheless, past knowledge of other agents can be used to solve these problems. Knowledge that has been learned from other tasks can be used as the base knowledge of the current problem. Hence, TL can be integrated into inductive learning.

In the past, inductive TL has been used in different machine learning techniques for classification. Instances were transferred to improve the classification accuracy, as in [32]. Features' representation, however, were transferred to create a model that can handle unreliable data, especially in images and text mining, as in [33]. Parameters can also be transferred to guide the classification process and, finally, relationship knowledge has been transferred to simplify dealing with a complex target problem, as in [34-36].

Nevertheless, these techniques were mostly targeting certain type of data. Its target was either to improve the accuracy or replace the heuristics, and it was usually used to deal with a scarce target task that is complex to deal with. Thus, most of these algorithms actually increased the space of searching or reduced its accuracy, which can cause greater problems with scalability. However, it was concluded in [37] that transferring the whole rule from the source to the target task can tremendously improve the induction process. Hence, it would be better to take advantage of the good properties of rule transfer while avoiding its drawback.

In our knowledge, no one yet has used Rule Transfer in covering algorithms, specifically RULES family. Consequently, the goal of this paper is to use TL and transfer the rules gathered by other agents in other domains to solve the problems of RULES family. This method will be explained in the next section.

Journal of Theoretical and Applied Information Technology
10th January 2013. Vol. 47 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4. RULES WITH TRANSFER LEARNING

This section explains the proposed algorithm, which is called RULES-TL, and discusses its key ideas. In general, RULES-TL is based on RULES-6, and it is developed to improve the performance of RULES family and scale it further to incomplete data. It integrates relational knowledge transfer (in the shape of rules) with RULES-6. In particular, Inductive TL is applied, which require that the information transferred from the source contains the class label, and no missing information is transferred. These transferred rules are used in a way to reduce the searching space before going through the induction process.

In general, RULES-TL maps the source rules from the source task and transfers it to the target tasks. These rules are then used to fill missing classes and reduce the search space by marking the examples covered by these rules so that no further searching is needed for it.

RULES-TL, as explained in Figure 2, starts by applying RULES-6 over the source tasks to induce the best rules. Then, the resulting rules are mapped to the target task representation. The mapping process first removes any unrelated rules, where an attribute or value in the rule is not found in the target header. Then, its format is transformed based on the format of the target task definition. After that, every mapped rule is taken as ground base to cover existing examples. If the rule covers an example, then the algorithm call Induce-One-Rule of RULES-6 but starting from the mapped rule instead of empty one. Hence, mapped rule is specialized to induce better ones based on the seed example.

Main()

Input: *Tset = Target set, mN = minimum negative, mP = minimum positive, w = beam width*
Output: *RuleSet = induced rules*

```
Trules = read source task rules
MappedRules = MapRules (Trules, Tset)
//Mark covered examples, store unseen rules, and return induced rues
RuleSet = Mark_covered_examples (Tset, mP, mN, w, MappedRules)
//update Tset
Fill_miss_class (Tset, MappedRules)
//Call RULES-6 as in [29]
RulesSet=RulesSet + RULES-6(Tset, mP, mN, w )
```

MapRules (Trules, Tset)

Input*: Trules, Tset*
Output*: Transferred rules*

```
MappedRules = ϕ
For every rule(i) ∈ Trules
//If the rule does not contain unrelated conditions
    If all attribute and values of rule(i) ∈ Tset header
        MappedR(i) = change the format of rules(i) to the same format as Tset
        MappedRules=MappedRules + MappedR(i)
Return MappedRules
```

Mark_covered_examples (MappedRules, Tset, mP, mN, w)

Input*: MappedRules, Tset, mP, mN, w*
Output*: Transferred rules*

```
RuleSet = ∅
Urules = ∅
For every rule(i) ∈ MappedRules
    For every uncovered example in Tset
        Select a seed example (s) from uncovered example
        //Call RULES-6 Induce procedure BUT initial BestRule = rule(i) NOT empty rule
            R = Induce_One_Rule (s, rule(i), Tset, mP, mN, w)
            If R≠ ϕ THEN
            //mark the rule as seen since it covers an example
                rule(i).Unseen = False
                Mark example (s) as covered
                RuleSet = RuleSet + R
    If rule(i).Unseen = True THEN
        Urules = Urules + rule(i)
Store Urules for future use
Return RuleSet
```

Fill_miss_class (Tset, MappedRules)

Input*: MappedRules, Tset*
Output*: Only update the target set by filling its missing classes*

```
BestScore = -1
BestLabel = ϕ
For every example (s) with missing class
    //get the best label based on Mapped rules
    For every rule(i) ∈ MappedRules
        Score = Compute S-Score for rule(i)
        If (score > Threshold) & (score > BestScore) THEN
            BestScore = score
            BestLabel = rule(i).label
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
//If no rule cover the example with missing class
If BestLabel = ϕ THEN
    BestLabel = Label of most similar example
    in Tset

Update s with Label value
```

Figure 2: RULES-TL pseudo code

Following that, covered examples are marked as covered, rules that cover at least one example is marked as seen rule, and rules that does not cover any example is marked as unseen rule for future use. Thus, past knowledge of other agents in a different task is used as the base to discover and induce good rules instead of always starting from an empty rule. Hence, the accuracy and speed are both increased. Thus, it reduces the need to tradeoff between time and accuracy.

After that, incomplete examples, which miss a class, are filled using the mapped rules. It starts to measure the similarity between the incomplete example and mapped rules using S measure [38]. Then, the score is compared with a threshold. If the rule score exceeds this threshold, then the example is considered as covered. However, if the algorithm was not able to find any matching rule, then the most similar example in the training set will be used to fill the missing class. After that, RULES-6 algorithm is applied over the uncovered examples to make sure that all the examples are marked as covered.

Consequently, it can be concluded that the problem of specialization is reduced because of the use of past knowledge of other domains as the base knowledge of new tasks. This way transferred rules can reduce the time of specialization and improve the accuracy. Moreover, incomplete data were accurately handled using such past knowledge instead of neglect important information just because it is incomplete. Hence, it is anticipated that RULES-TL will be more scalable and accurate than the original RULES-6, which will be proven in the experiment later. Nevertheless, to further explain the key ideas of RULES-TL, each one is presented in a separate section as follows.

4.1 Rule Quality Metric

In order to evaluate the rules and search for the best one, a certain quality matrix should be applied. Such matrix measures the quality of a rule based on the available data, specifically number of positive and negative examples covered by a rule. Hence, it measures the quality and coverage of the rules to guide the searching process to the best one. In RULES-TL, two quality matrixes were used depending on purpose of use. When the algorithm wants to measure the rule quality in order to search and induce the best rule of a given example, m-probability-estimate [39] is used. This is because the "induce rule" procedure used in RULES-TL is based on RULES-6, and since it was found that m-probability-estimate outperform other measures when tested with RULES-6 then it was decided to use the same matrix to search for the best rule during the specialization process.

On the other hand, when it comes to searching for the best rule and choosing for the best label to fill the missing class, S measure was used. It is computed using (1), where N is the total number of negative examples, P is the total number of positive examples, n is the number of negative examples covered by the new rule, and p is the number of positive examples covered by the new rule.

$$S = consistancy * gain * (1 - missclassification\ level)$$

$$S = \frac{p}{p+n} \cdot \frac{p}{P\_unclassified} \cdot (1 - \frac{n}{N}) \quad (1)$$

As described by Bigot [38], S measure is a measure that has been developed to improve the rule induction result by considering the classification gain, level, and consistency. This measure has been tested with other covering algorithms' measures to find that it significantly gives better results. Hence, it was decided to choose it when searching for the best label to fill the missing classes.

4.2 Search-space Pruning Rules

As descried previously, two main searching procedures are applied over RULES-TL. The first one is the original searching procedure of RULES-6, which is used to induce the best rule that cover different examples. While the other searching procedure is used to search for the best rule to fill a missing class. Hence, in RULES-TL, two main pruning procedures were used.

When it comes to the search space of rule induction, four main rules were applied based on RULES-6. It reduces the searching space and scale the algorithm to large dataset. These rules took advantage of the nature structure of the specialization hypotheses [1]. This structure indicates that the more conditions added to the rule the fewer examples it will cover. Moreover, to increase the algorithm resistance to future noise the induced rule should not be overspecialized, and it

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

should be general enough to give good results. Thus, the following four rules were applied to constraint the searching space and remove rules that would reduce the quality and speed.

If Rule.Consistency = 100% Then remove Rule

If Rule.PosCoverage ≤ MinPos Then remove Rule

If Rule.Score ≤ BestRule.Score Then remove Rule

If PreRule.NegCoverage - Rule.NegCoverage ≤ MinNeg Then remove Rule

The pruning rules remove fully consistent rule because it will produce overspecialized rules. It removes rules that cover positive examples less than an allowed number specified by the user. It also removes the rules that start to reduce the quality measure. Finally, when the difference between a rule and its predecessor is less than a threshold specified by the user then these rules are also removed.

On the other hand, when it comes to transfer searching space, when missing classes are filled, a noise threshold called T threshold [25] were used. As illustrated in (2), this threshold considers the noise level and the ratio of positive examples in the whole training set. In this threshold, NL is the allowed noise specified by the user, E is the total number of examples in the training set, and Ei is the number of example that belongs to a class i.

$$T = 2 - 2\sqrt{(1 - NL)\frac{E_i}{E}} - 2\sqrt{NL(1 - \frac{E_i}{E})} \quad (2)$$

This threshold is then used to remove rules that are not related to the example under sturdy. Thus, the following rule is applied to decide if the transfer rule can be considered when searching for the best label or not.

If Rule.S-Score ≤ Threshold Then remove Rule

4.3 Discretization Method

In order to handle dataset that contain continuous values, offline discretization was applied. In specific, entropy-based discretization method of Fayyad and Irani [40] were applied over the dataset in order to convert the continuous values into discrete ones. This discretization method were chosen based on the study conducted in [41, 42], where it was found that the, depending on the data and algorithm used the right choice of offline discretization method can be made. In addition, in RULES-6 and RULES-SRI, it was empirically proven that Fayyad and Irani discretization is the most appropriate offline discretization for RULES algorithms. This method split the ranges of continuous values to small intervals to be used as discrete values.

4.4 Missing Attributes

In real life, data is not always complete. Gathered data usually contains missing information and dataset attributes could contain null values. Some researchers choose to neglect such examples, but this is not a good decision since these examples might contain important information that could affect the quality of the rule induction algorithm. In [43], different methods that handle missing attributes have been presented and empirically tested over many rule induction methods and large number of datasets. As a result of this test, it was found that imputation methods, especially Fuzzy K-means (FK-means) [44], are the most suitable methods to handle missing values in rule induction.

In FK-means, each object has a membership function that measures the degree in which the object belongs to a certain value. Hence, it clusters the objects over values clusters based on the membership function. Hence, RULES-TL used FK-means to fill missing values of the attributes. Consequently, important information will not be lost, and incomplete data can be filled.

4.5 Missing Classes

In addition to missing values, labels might also be missed because of natural reasons, such as manually filling of the training set or lack of information. Nevertheless, such examples might be very important and can have sensitive information that would guide the learning process to better result. Thus, missing classes should not be neglected. However, deciding on the best label is a critical procedure and could highly affect the final result. Consequently, only considering the information of training set is not enough.

In general, past knowledge of other agents can be a great help to fill missing labels. Transferring knowledge from another expertise is always a smart choice in any field of human life. However, it cannot be guaranteed that every field will have prior information. Consequently, RULES-TL tries to use TL to transfer past knowledge of other agents but not necessary from the same domain. Thus, it can be guaranteed that filling the missing classes will be guided by past knowledge even if it is from other fields. Hence, new labels will consider the available data in the current

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

training set in addition to the past rules gathered by other agents.

Nevertheless, sometimes it is possible that no transferred rules are found to be a match for the targeted examples. In such a case, it is suggested to use the most similar example in the training set. The example similarity is measured based on D-distance, as illustrated in (3), where V^i_{E1} and V^i_{E2} are the values of the continuous attribute (i) in example E1,E2 respectively, V^i_{min} and V^i_{max} are the minimum and maximum values of attribute (i), and the distance between discrete attributes is computed using equation (4). D-distance, as explained by Pham, Bigot, and Dimov [16] is a distance measure that can compare any type of examples together. In this measure, distance between continuous attributes is considered in addition to the difference between the discrete ones. This way, all types of data is taken into consideration in order to make it possible to find an identical example that belongs to a class so that targeted example also join the same class.

$$D(E1,E2) = \sqrt{\sum_{All\ cont.Attr}\left(\frac{V^i_{E1}-V^i_{E2}}{V^i_{max}-V^i_{min}}\right)^2 + \sum_{All\ disc.Attr} d(A1,A2)} \quad (3)$$

$$d(A1,A2) = \begin{cases} 0 & if\ A1 = A2 \\ 1 & if\ A1 \neq A2 \end{cases} \quad (4)$$

5. EXPERIMENT

In order to test the performance of RULES-TL different experiments were conducted. RULES-TL was first implemented using Java language in JBuilder environment. The experiments were conducted on a PC with Intel®Core™ i7 CPU, 2.67 GHz processes, and 6GB RAM. In addition, KEEL (Knowledge Extraction based on Evolutionary Learning) tool [45, 46] was used to build the experiments and decide on its properties. It was also used to compare the performance between the proposed algorithm and existing methods.

KEEL is an open-source tool developed by Java software to evaluate algorithms proposed for problems of data mining. It includes a large collection of classical rule induction algorithms that can be used to compare the performance of the proposed algorithm. Moreover, it has Statistical Analysis Tools (SAT) that can be used to design an experiment and perform a complete analysis on the algorithms' performance using a simple graphical user interface. Nevertheless, five key elements were defined in order to build the desired experiment. These elements can be described as follows.

5.1 Dataset

In order to show how reliable the proposed algorithm is, eight dataset were used to test RULES-TL. These data set were taken from KEEL dataset repository [46]. The properties of each dataset are specified in Table 1, and each dataset is gathered as a sample from real-life data and serve a certain domain, as follows.

Ecoli: Contains information about measures applied over proteins to predict its localization site.

Yeast: Contains information about Yeast cells to determine its localization site in each cell.

Australian Credit: Contains information about Australian credit card applications to determine if it should be approved or not.

Credit Approval: It is an extended version of Australian Credit with more information.

Cleveland: Contains information about Cleveland heart disease to detect the occurrence of heart disease in the patients of Cleveland Clinic Foundation in Long Beach USA.

Statlog: Contain general information about heart disease to detect its presence or absence.

Bupa: Contains information about liver disorder that might occur due to excessive alcohol consumption in order to decide if a person suffers from alcoholism.

Hepatitis: Contains information about hepatitis patients in order to indicate if he survives or not.

Red Wine: Contains information about red wine to indicate whether it has good quality or not.

White Wine: Contains information about white wine to indicate whether it has good quality or not.

Table 1: Experiment dataset property

Dataset	#Examples	#Attributes	#Labels
ecoli	336	7	8
yeast	1484	8	10
Australian Crd.	690	14	2
Crd. Approval	690	15	2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Dataset	#Examples	#Attributes	#Labels
ecoli	336	7	8
yeast	1484	8	10
Australian Crd.	690	14	2
Crd. Approval	690	15	2
Cleveland	303	13	5
Statlog	270	13	2
Bupa	345	6	2
Hepatitis	155	19	2
Red Wine	1599	11	11
White Wine	4898	11	11

Moreover, each dataset is partitioned to five-pair partitions using the hold-out approach [47]. Each pair includes a test and training data. The training data is used for training the algorithm, and the test data is used to test the result of the algorithm. In specific, for the large dataset (data with more than one thousands examples), the test set include one-third of the data while the training set include the remaining two third. However, with small data (data with less than one thousands examples) the partitioning was repeated five times, and the result was averaged.

5.2 Predecessors

As discussed previously, the attributes values sometimes need further refinement before starting with the learning process. In specific, predecessor procedures need to be applied over all the data to deal with missing and continuous attributes. Hence, like RULES-TL, FK-means and entropy-based discretization method of Fayyad and Irani are applied over the data in order to fill missing attributes and discretize continuous values.

5.3 Comparative Covering Algorithms

In order to prove that the performance of rule induction has improved by using RULES-TL, five different covering algorithms were compared with RULES-TL performance. Specifically, the following algorithms were tested.

RULES-6: The predecessor of RULES-TL, where the transfer learning was not applied.

RULES-SRI: One version of RULES family that was proposed to improve the scalability and make the algorithm more accurate over large data set.

Ripper [48]: An algorithm that is developed to directly generate rules by combining attributes that positively cover as many examples as possible.

C4.5Rules [2]: A decision tree based algorithm that discovers rules using decision tree and then use the resulting tree to represent the rules.

DataSqueezer [6]: A covering algorithm that is a fast, supervised, greedy, and simple algorithm, which required the existence of all class labels.

5.4 Postprocessor

After conducting the experiment, and to visualize the result, different statistical analysis tests were applied. Specifically, three statistical tests were recorded to measure the error rate of each algorithm, their accuracy, and the time spent by RULES-TL. The details of each statistical test can be described as follows.

5x2CV Test [49]: It is an approximate statistical test that compares supervised learning algorithms. It shows the error mean and median of each algorithm based on 2 fold-cross validation and five iterations.

Accuracy Test: It shows the accuracy result of each algorithm by testing the final rules over the test set.

Time Test: It measures the rule induction time spent by RULES-TL, measured by seconds.

5.5 Evaluation Result

Using the resulting statistics of KEEL analysis tool, it was possible to record the algorithms' performance, as illustrated in Table 2. As it can be noticed, the average error rate in RULES-TL is less than all the other algorithms, except for C4.5rules. Specifically, transferring rules to a dataset, with missing classes or not, usually improves the performance. However, comparing to the tree based algorithm (C4.5rules), the average performance has slightly reduced. Nevertheless, in some datasets the performance of RULES-TL has outperformed the C4.5rule algorithm and, on average, the error difference between the two algorithms is only 5%. Hence, it can be said that RULES-TL is comparable to C4.5, not to mention that RULES-TL uses a special procedure to deal with missing classes.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Moreover, the previous statement can be further verified when considering the accuracy of the resulting rules, as illustrated in Table 3. As it can be noticed the accuracy performance of RULES-TL is the same whether the classes are missing or not, and it has more accuracy than most of the algorithms. In addition, in comparison to C4.5 rules, the accuracy of both algorithms in most of the datasets is comparable. On average, C4.5 has 8% more accuracy than RULES-TL. Nevertheless, such percentage is better than the result of past RULES algorithms.

Consequently, it is important to emphasize that, whether the dataset have missing classes or not, RULES-TL outperformed its predecessor RULES-6, and it is more scalable than RULES-SRI. It has better accuracy and less error rate on all tested data. Hence, it can be stated that RULES-TL has improved the scalability of RULES family over complete and incomplete data. This is because RULES scalable algorithms, specifically RULES-6 and RULES-SRI, result in worse performance.

Additionally, when it comes to the speed of RULES-TL, it can be noticed that it does not spend a large amount of time on rule induction. As illustrated in Table 4, it is only a matter of seconds to induce rules. In the most difficult case, when the red-wine dataset has 1599 examples and its target (white wine) has 4898, it only takes 171 seconds when all classes are available and 64 seconds when missing classes exists. Moreover, it must be noted that the performance of the algorithm speed actually improved when missing classes exist due to the use of transferred rules extensively instead of going through most of the examples.

Consequently, RULES-TL reduces the error rate and improves the accuracy, whether it has complete and incomplete datasets. Moreover, it was noted that RULES-TL accuracy performance was the same when the class is missing or not while it reduced the induction time when the classes are missing. Hence, RULES-TL is a good option for large and incomplete data. Ultimately, it can be stated that the purpose of this paper has been met, and RULES-TL is a valid innovate method.

Table 2: Error mean and median using Clas-5X2CV

Dataset		C4.5 rules		Ripper		Data Squeezer		RULES-6		RULES-SRI		RULES-TL			
												Miss Class		No Miss Class	
Source	Target	ẋ	M	ẋ	M	ẋ	M	ẋ	M	ẋ	M	ẋ	M	ẋ	M
ecoli	yeast	0.44	0.44	0.73	0.75	0.71	0.68	0.71	0.71	0.83	0.83	0.63	0.62	0.64	0.64
Australian Crd.	Crd. Approval	0.13	0.14	0.16	0.16	0.32	0.32	0.24	0.23	0.36	0.37	0.19	0.20	0.18	0.17
Cleveland	Statlog	0.18	0.18	0.30	0.31	0.44	0.44	0.30	0.29	0.36	0.40	0.19	0.20	0.19	0.20
Bupa	Hepatitis	0.13	0.12	0.38	0.37	0.16	0.18	0.07	0.06	0.38	0.37	0.13	0.12	0.07	0.06
Red Wine	White Wine	0.50	0.51	0.49	0.49	0.62	0.61	0.53	0.54	0.95	0.95	0.53	0.54	0.53	0.54
Average		**0.27**	**0.27**	**0.41**	**0.41**	**0.45**	**0.44**	**0.37**	**0.36**	**0.57**	**0.58**	**0.33**	**0.33**	**0.32**	**0.32**

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 3: Accuracy result

Dataset		C4.5	Ripper	Data Squeezer	RULES-6	RULES -SRI	RULES-TL	
Source	Target						Miss Class	No Miss Class
ecoli	yeast	0.59	0.30	0.28	0.31	0.16	0.36	0.35
Australian Credit	Credit Approval	0.88	0.91	0.68	0.75	0.65	0.81	0.83
Cleveland	Statlog	0.87	0.79	0.55	0.75	0.66	0.80	0.80
Bupa	Hepatitis	0.90	0.63	0.83	0.92	0.67	0.91	0.92
Red Wine	White Wine	0.52	0.70	0.37	0.46	0.05	0.47	0.46
Average		**0.75**	**0.66**	**0.54**	**0.63**	**0.43**	**0.67**	**0.67**

Table 4: Rule induction execution time in seconds

Dataset		RULES-TL	
Source	Target	Miss Class	No Miss Class
ecoli	yeast	28	26
Australian Crd.	Crd. Approval	5	5
Cleveland	Statlog	1	1
Bupa	Hepatitis	1	1
Red Wine	White Wine	64	171

6. CONCLUSION

In order to predict future activities, RULES family was introduced. It is a covering algorithm family that is used to induce simple rule and identify future activities. However, the algorithms of this family are still lacking when dealing with incomplete data. It needs further improvement to handle missing classes and to reduce the time spent on the specialization process. Hence, this paper has proposed a new algorithm that takes advantage of advance machine learning methods, specifically TL, in order to transfer other agents' knowledge gathered from other domains.

The performance of this algorithm was tested over eight dataset and compared with five rule induction algorithms. As a result, it was found that RULES-TL has improved the performance of RULES family over complete and incomplete datasets. In reality, it actually improves the performance more when the data is incomplete since the accuracy is the same while the time was tremendously reduced especially with large dataset. Nevertheless, RULES family, in general, needs further improvement to beat tree based algorithms like C4.5 rules. However, its current performance is comparable since the difference error rate between RULES-TL and C4.5 was only 5%.

As future work it is suggested to test RULES-TL with larger dataset. In addition, it is also proposed to compare it with more rule induction algorithms in order to further prove its efficiency.

ACKNOWLEDGEMENT

The authors would like to thank the Research Center at college of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia for their support.

REFERENCES

[1] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.
[2] J. R. Quinlan, *C4.5: Programs for Machine Learning*: Morgan Kaufmann, 1993.
[3] F. Stahl and M. Bramer, "Computationally efficient induction of classification rules with the PMCRI and J-PMCRI frameworks," *Knowledge-Based Systems*, 2012.
[4] 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.
[5] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining Practical Machine Learning Tools and*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Techniques, Third ed.: Morgan Kaufmann, 2011.

[6] 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.

[7] R. Michalski, "On the quasi-minimal solution of the general covering problem," in *V International Symposium on Information Processing*, Yougoslavia, Bled, 1969, pp. 128-128.

[8] P. Clark and T. Niblett, "The CN2 induction algorithm," *Machine Learning*, vol. 3, pp. 261-283, 1989.

[9] E. Frank and I. H. Witten, "Generating Accurate Rule Sets Without Global Optimization," presented at the Fifteenth International Conference on Machine Learning, 1998.

[10] M. Aksoy, "A review of rules family of algorithms," *Mathematical and Computational Applications*, vol. 13, pp. 51-60, 2008.

[11] F. Stahl, M. Bramer, and M. Adda, "P-Prism: A Computationally Efficient Approach to Scaling up Classification Rule Induction," in *Artificial Intelligence in Theory and Practice II*. vol. 276, M. Bramer, Ed., ed: Springer Boston, 2008, pp. 77-86.

[12] F. Stahl and M. Bramer, "Induction of Modular Classification Rules: Using Jmax-pruning," pp. 79-92, 2011.

[13] A. An, "Learning classification rules from data," *Computers & Mathematics with Applications*, vol. 45, pp. 737-748, 2003.

[14] H. Theron, "An Empirical Evaluation of Beam Search and Pruning in BEXA," in *Fifth International Conference on Tools with Artificial Intelligence (TAI '93)* Boston, MA, 1993, pp. 132- 139.

[15] P. Sinno Jialin and Y. Qiang, "A Survey on Transfer Learning," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 22, pp. 1345-1359, 2010.

[16] 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.

[17] J. Ramon, K. Driessens, and T. Croonenborghs, "Transfer Learning in Reinforcement Learning Problems Through Partial Policy Recycling: Machine Learning," in *ECML 2007*. vol. 4701, J. Kok, J. Koronacki, R. Mantaras, S. Matwin, D. Mladenic, and A. Skowron, Eds., ed: Springer Berlin / Heidelberg, 2007, pp. 699-707.

[18] M. Taylor, H. B. Suay, and S. Chernova, "Integrating Reinforcement Learning with Human Demonstrations of Varying Ability," in *International Conferance of Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Taipei, Taiwan, 2011.

[19] M. Mahmud, "On Universal Transfer Learning Algorithmic Learning Theory." vol. 4754, M. Hutter, R. Servedio, and E. Takimoto, Eds., ed: Springer Berlin / Heidelberg, 2007, pp. 135-149.

[20] L. Yongqiang, "A Review About Transfer Learning Methods and Applications," in *International Conference on Information and Network Technology IPCSIT*, Singapore, 2011, pp. 7-11.

[21] D. T. Pham and M. S. Aksoy, "RULES: A simple rule extraction system," *Expert Systems with Applications*, vol. 8, pp. 59-65, 1995.

[22] D. T. Pham and M. S. Aksoy, "An algorithm for automatic rule induction," *Artificial Intelligence in Engineering*, vol. 8, pp. 277-282, 1993.

[23] D. T. Pham and M. S. Aksoy, "A new algorithm for inductive learning," *Journal of Systems Engenering*, vol. 5, pp. 115-122, 1995.

[24] 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.

[25] D. T. Pham and S. S. Dimov, "An algorithm for incremental inductive learning," *Journal of Engineering Manufacture*, vol. 211, pp. 239-249, 1997.

[26] 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.

[27] 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.

[28] 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:*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Journal of Systems and Control Engineering, 2011.

[29] 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.

[30] 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.

[31] 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.

[32] W. Pan, E. Zhong, and Q. Yang, "Transfer Learning for Text Mining," pp. 223-257, 2012.

[33] Y.-F. Xie, S.-Z. Su, and S.-Z. Li, "A Pedestrian Classification Method Based on Transfer Learning," presented at the International Conference on Image Analysis and Signal Processing - IASP, Zhejiang, 2010.

[34] J. I. Estévez, P. A. Toledo, and S. Alayón, "Using an Induced Relational Decision Tree for Rule Injection in a Learning Classifier System," presented at the IEEE Congress on Evolutionary Computation New Orleans, LA, 2011.

[35] H. Boström, "Induction of Recursive Transfer Rules," in *Learning Language in Logic*. vol. 1925, J. Cussens and S. Džeroski, Eds., ed: Springer Berlin / Heidelberg, 2000, pp. 369-450.

[36] M. D. Reid, "DEFT Guessing: Using Inductive Transfer to Improve Rule Evaluation from Limited Data," Doctor of Philosophy, School of Computer Science and Engineering, The University of New South Wales, Sydney, Australia, 2007.

[37] P. Ganchev, D. Malehorn, W. L. Bigbee, and V. Gopalakrishnan, "Transfer learning of classification rules for biomarker discovery and verification from molecular profiling studies," *Journal of biomedical informatics*, vol. 44 Suppl 1, pp. S17-23, Dec 2011.

[38] S. Bigot, "A study of specialisation and classification heuristics used in covering algorithms," presented at the IPROM2009 Innovative Production Machines and Systems Fifth I*PROMS Virtual Conference, Cardiff, UK, 2009.

[39] J. Fürnkranz and P. A. Flach, "An analysis of rule evaluation metrics," in *20th International Conference on Machine Learning*, Washington, DC, USA, 2003, pp. 202-209.

[40] 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.

[41] Z. Cai, "Technical Aspects of Data Mining," PhD, University of Wales Cardiff, Cardiff, UK, 2001.

[42] 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.

[43] J. Luengo, S. García, and F. Herrera, "On the choice of the best imputation methods for missing values considering three groups of classification methods," *Knowledge and Information Systems*, vol. 32, pp. 77-108, 2012/07/01 2012.

[44] J. Deogun, W. Spaulding, B. Shuart, and D. Li., "Towards Missing Data Imputation: A Study of Fuzzy K-means Clustering Method," presented at the 4th International Conference of Rough Sets and Current Trends in Computing (RSCTC'04), Uppsala, Sweden, 2004.

[45] 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.

[46] 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.

[47] B. Efron and R. Tibshirani, *An Introduction to the Bootstrap*. USA: Chapman & Hall, 1993.

[48] W. W. Cohen, "Fast Effective Rule Induction," in *Twelfth International Conference on Machine Learning*, 1995, pp. 115-123.

[49] T. G. Dietterich, " Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms," *Neural Computation* vol. 10, pp. 1895-1923, 1998.