Post-Analysis of Learned Rules

Bing Liu and Wynne Hsu

Department of Information Systems and Computer Science
National University of Singapore
Lower Kent Ridge Road, Singapore 119260, Republic of Singapore
{liub, whsu}@iscs.nus.sg

Abstract

Rule induction research implicitly assumes that after producing the rules from a dataset, these rules will be used directly by an expert system or a human user. In real-life applications, the situation may not be as simple as that, particularly, when the user of the rules is a human being. The human user almost always has some previous concepts or knowledge about the domain represented by the dataset. Naturally, he/she wishes to know how the new rules compare with his/her existing knowledge. In dynamic domains where the rules may change over time, it is important to know what the changes are. These aspects of research have largely been ignored in the past. With the increasing use of machine learning techniques in practical applications such as data mining, this issue of post analysis of rules warrants greater emphasis and attention. In this paper, we propose a technique to deal with this problem. A system has been implemented to perform the post analysis of classification rules generated by systems such as C4.5. The proposed technique is general and highly interactive. It will be particularly useful in data mining and data analysis.

1. Introduction

Past research on inductive learning has mostly been focused on techniques for generating concepts or rules from datasets (e.g., Quinlan 1992; Clark & Niblett 1989; Michalski 1980). Limited research has been done on what happens after a set of rules has been induced. It is assumed that these rules will be used directly by an expert system or some human user to infer solutions for specific problems within a given domain. We argue that having obtained a set of rules is not the end of the story. As machine learning techniques are increasingly being used to solve real-life problems, post-analysis of rules will become increasing important.

The motivation for performing post-analysis of the rules comes from realizing the fact that using a learning technique on a dataset does not mean that the user knows nothing at all about the domain and the dataset. This is particularly true if the user is a human being. Typically, the human user does have some pre-conceived notions or knowledge about the learning domain. Hence, when a set of rules is generated from a dataset, naturally he/she would like to know the following: Do the rules represent what I know? If not, which part of my previous knowledge is correct and which part is not? In what ways are the new rules different from my previous knowledge? Past research has assumed that it is the user's responsibility to analyze the rules to answer these questions. However, when the number of rules is large, it is very hard for the user to analyze them manually.

In dynamic domains, rules themselves may change over time. Obviously, it is important to know what have changed since the last learning. These aspects of research have largely been ignored in the past.

Besides enabling a user to determine how well the new rules confirm/deny his/her existing concepts and to determine whether the rules have changed over time, post-analysis of rules also helps to deal with a major problem in data mining, i.e., the interestingness problem (Piatesky-Shapiro & Matheus 1994; Piatesky-Shapiro *et al* 1994). This problem is typically described as follows: In data mining, it is all too easy to generate a huge number of patterns in a database, and most of these patterns (or rules) are actually useless or uninteresting to the user. But due to the huge number of patterns, it is difficult for the user to comprehend them and to identify those patterns that are interesting to him/her. Thus, some techniques are needed to perform this identification task.

This paper proposes a fuzzy matching technique to perform the post-analysis of rules. In this technique, existing rules (from previous knowledge) are regarded as fuzzy rules and are represented using fuzzy set theory (Zimmermann 1991). The newly generated rules are matched against the existing fuzzy rules using the fuzzy matching technique. Different algorithms are presented to perform matching according to different criteria. The matched results will then enable us to answer some of the concerns raised above. In this paper, we focus on performing post-analysis of classification rules generated by induction systems, such as C4.5 (Quinlan 1992). This is because classification rule induction is perhaps the most successful machine learning technique used in practice. However, our proposed technique is not bound to classification rules, it is also applicable to other types of

rules generated by some other learning techniques. The proposed technique is general. It can be used in any domain. It is also highly interactive as it allows the user to modify the existing rules as and when modification is needed. We believe our technique is a major step in the right direction.

2. Problem Definition

Assuming a human user or an intelligent agent has some previous knowledge, or hypotheses, or a set of rules generated from an earlier dataset, about a particular domain. These concepts can be expressed as a set of rules E. There exists a dataset D from this domain. A learning technique T can be applied to D to induce a set of rules B. The rules in E and rules in B have the same syntax and semantics. We are interested in knowing the degree of similarity and difference between set B and set E. Since the focus of this paper is on the classification rules produced by C4.5, T is the decision tree induction technique used in C4.5. The rules in E and B have the same syntax and semantics as the rules produced by C4.5.

The syntax of the rules generated by C4.5 has the following form (we use If-then format, instead of "→" as in C4.5, and we also add a "," between two propositions to facilitate presentation):

If P_1, P_2, P_3, ..., P_n **then** C

where "," means "and", and P_i is a proposition of the form: *attr OP value,* where *attr* is the name of an attribute in the dataset, *value* is a possible value for *attr*, and $OP \in \{=, \neq, <, >, \leq, \geq\}$ is the operator. C is the consequent of the form: *Class* = *value*.

We denote a new rule as $B_i \in B$ and an existing rule as $E_j \in E$. We denote the set of attribute names in the conditional part of B_i as F_i, and the set of attribute names in the conditional part of E_j as H_j. For example, we have

B_i: **If** $Ht > 1.8$, $Wt < 70$ **then** *Class* = *underweight*.

E_j: **If** *Size* = *med*, $Wt \geq 65$, $Wt \leq 70$ **then** *Class* = *fit*.

Then, $F_i = \{Ht, Wt\}$, and $H_j = \{Size, Wt\}$.

We now define what we mean by similarities and differences between E and B. Firstly, we define them intuitively (in this section), and then we define them computationally (in Section 3 and 4). The intuitive definitions can be stated at two levels: the individual-rule level and the aggregate level.

Definitions at Individual-rule Level

Definition 1 (Rule_Similarity): B_i and E_j are similar if both the conditional parts of B_i and E_j, and the consequent parts of B_i and E_j are similar.

Definition 2 (Rule_Difference): B_i and E_j are different if B_i and E_j are far apart in one of the following ways:

(1). **Unexpected consequent:** The conditional parts of B_i and E_j are similar, but the consequents of the two rules are quite different.

(2). **Unexpected conditions:** The consequents of B_i and E_j are similar but the conditional parts of the two rules are far apart. There are two situations:

a). Contradictory conditions: The sets of attribute names in the conditional parts of B_i and E_j overlap to a large extend but their attribute values are quite different.

b). Unanticipated conditions: The sets of attribute names in the conditional parts of B_i and E_j do not overlap.

Definitions at Aggregate Level

Definition 3 (Set_Similarity): The subset of rules in B that are the same or *very close* to some of the rules in E.

Definition 4 (Set_Difference): The subset of rules in B that are *far apart* from the set of rules in E in the sense of unexpected consequent or unexpected conditions as stated in definition 2.

Notice that we have been using fuzzy terms such as *similar*, *very different* (or *far apart*) that we have not quantified. In the next two sections, we will define the computation procedures that measure the similarity and difference. Our technique does not identify the similar rules and different rules. It only ranks them according to the similarity and difference values. The final job of identifying the rules as similar or different is left to the user as the system does not know what degree of match of two rules are considered similar or different.

3. Fuzzy Matching Method

In this section, we present a high level view of the fuzzy matching method. It consists of two main steps:

Step1. The user converts each rule in E to a fuzzy rule. The fuzzy rule has the same syntax as the original rule, but its attribute values must be described using some fuzzy linguistic variables. See the definition below.

Step2. The system matches each new rule $B_i \in B$ against each fuzzy rule $E_j \in E$ in order to obtain the degree of match for each new rule B_i against the set E. The new rules in B are then ranked according to their degrees of match with E. Four matching algorithms are used to perform matching for different purposes.

Before we proceed, let us review the definition of a fuzzy linguistic variable (Zimmermann 1991).

Definition 5: A fuzzy linguistic variable is a quintuple (x, $T(x)$, U, G, $\tilde{M}$) in which x is the name of the variable; $T(x)$ is the term set of x; that is, the set of names of linguistic values of x, with each value being a fuzzy variable denoted generally by x and ranging over a universe of discourse U; G is a syntactic rule for generating the name, X, of values of x; and M is a semantic rule for associating with each value X its

meaning, $\tilde{M}(X)$ which is a fuzzy subset of U. A particular X is called a *term*.

For example, if *speed* is interpreted as a linguistic variable with U = [1,140], then its term set T(*speed*) could be

$$T(speed) = \{slow, moderate, fast, ...\}.$$

$\tilde{M}(X)$ gives a meaning to each term. For example, $\tilde{M}(slow)$ may be defined as follows:

$$\tilde{M}(slow) = \{(u, \mu_{slow}(u)) \mid u \in [1, 140]\}$$

where
$$\mu_{slow}(u) = \begin{cases} 1 & u \in [1,30] \\ -\frac{1}{30}u + 2 & u \in (30,60] \\ 0 & u \in (60,140] \end{cases}$$

$\mu_{slow}(u)$ denotes the degree of membership of u in *slow*.

A window-based user interface has been implemented to simplify the input of semantic rules (i.e. membership values). After the user has specified the semantic rules associated with each term used, the fuzzy matching process can begin.

Let us have an example. Consider the set of classification rules generated from an accident database.

If *Age* > 50, *Loc* = *straight* **then** *Class* = *slight*
If *Age* > 65, *Loc* = *bend*, *Spd* > 50 **then** *Class* = *killed*
If *Age* > 50, *Loc* = *T-junct* **then** *Class* = *slight*

Assume the user believes that an old person crossing at a location with obstructed view is likely to result in a serious accident. S/he specifies the hypothesis as follows:

If *Age* = *OLD*, *Loc* = *OBSTRUCT* **then** *Class* = *BAD*

To confirm or deny this hypothesis, the system must first know how to interpret the semantic meanings of "*OLD*", "*OBSTRUCT*" and *"BAD.*" This is achieved by asking the user to provide the semantic rules associated with these terms. Once the semantic rules have been specified, the matching process is carried out to determine the degrees of match between the hypothesis and the system generated rules. Different ranking algorithms are used for different purposes. For confirmation of the hypothesis, the system will give higher ranking to rules that are similar to the hypothesis. The resulting ranking could be as follows:

1. **If** *Age* > 65, *Loc* = *bend*, *Spd* >50 **then** *Class* = *killed*
2. **If** *Age* > 50, *Loc* = *T-junct* **then** *Class* = *slight*
3. **If** *Age* > 50, *Loc* = *straight* **then** *Class* = *slight*

On the other hand, if our purpose is to find those rules that are contradictory to the hypothesis, then a different ranking algorithm is used and the result is shown below:

1. **If** *Age* > 50, *Loc* = *T-junct* **then** *Class* = *slight*
2. **If** *Age* > 65, *Loc* = *bend*, *Spd* >50 **then** *Class* = *killed*
3. **If** *Age* > 50, *Loc* = *straight* **then** *Class* = *slight*

This shows that rule 1 contradicts the hypothesis because instead of a serious injury, the old person suffers a slight injury. It is important to note that simply reversing the order of ranking for finding similar rules does not work in general for finding different rules.

4. The Rule Matching Computation

Having seen an overview of the proposed technique, we shall now describe the detailed formulas used in the rule matching computation.

Let E be the set of existing rules and B be the set of newly generated rules. We denote W_i as the degree of match between a newly generated rule $B_i \in B$ and the set of user-specified rules E. We denote $w_{(i,j)}$ as the degree of match between $B_i \in B$ and $E_j \in E$. Ranking of the new (or system generated) rules is performed by sorting them in a decreasing order according to their W_i (i.e., the rule with the highest W_i will be at the top).

4.1 Computing $w_{(i,j)}$ and W_i

The computation of $w_{(i,j)}$ consists of two steps: (1) attribute name match, and (2) attribute value match.

1. **Attribute name match** - The attribute names of the conditions of B_i and E_j are compared. The set of attribute names that are common to both the conditions of B_i and E_j is denoted as $A_{(i,j)} = F_i \cap H_j$. Then, the degree of attribute name match of the conditional parts, denoted as $L_{(i,j)}$, is computed as follows:

$$L_{(i,j)} = \frac{|A_{(i,j)}|}{\max(|F_i|, |H_j|)}$$

where $|F_i|$ and $|H_j|$ are the numbers of attribute names in conditional parts of B_i and E_j respectively, and $|A_{(i,j)}|$ is the size of the set $A_{(i,j)}$.

Likewise, the attribute names of the consequents of B_i and E_j are also compared. However, in the case of C4.5, the rules generated all have the same attribute name, i.e., *Class*. Existing rules also use the same attribute. Thus, the attribute names always match. For example, we have

B_i: **If** *Ht* > 1.8, *Wt* < 70 **then** *Class* = *underweight*.
E_j: **If** *Size* = *medium*, *Wt* = *middle* **then** *Class* = *fit*.

The set of common attributes in the conditional parts of the two rules is $A_{(i,j)} = \{Wt\}$. Hence, $L_{(i,j)} = 0.5$.

2. **Attribute value match** - Once an attribute of B_i and E_j matches, the two propositions are compared taking into consideration both the attribute operators and attribute values. We denote $V_{(i,j)k}$ as the degree of value match of the kth matching attribute in $A_{(i,j)}$, and $Z_{(i,j)}$ the degree of value match of the consequents. The computation of the two values is presented in Section 4.2.

We are now in the position to provide the formulas for computing $w_{(i,j)}$ and W_i. Due to the space limitation, we will not give detailed explanation for each formula. Interested readers, please refer to (Liu & Hsu, 1995).

1. Finding similar rules

$$w_{(i,j)} = \begin{cases} \dfrac{Z_{(i,j)} \times L_{(i,j)} \times \sum_{k \in A(i,j)} V_{(i,j)k}}{|A_{(i,j)}|} & |A_{(i,j)}| \neq 0 \\ 0 & |A_{(i,j)}| = 0 \end{cases}$$

The formula for W_i, which is the degree of match of the rule B_i with respect to the set of existing rules, E, is defined as follows (see Figure 1):

$$W_i = \max(w_{(i,1)}, w_{(i,2)}, \ldots, w_{(i,j)}, \ldots, w_{(i,|E|)})$$

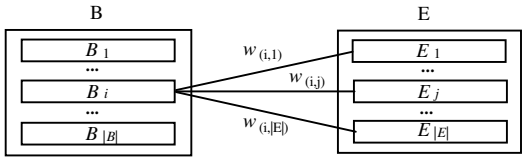

Figure 1. Computing W_i

2. Finding different rules in the following sense:
   1) **Unexpected consequent:** The conditional parts are similar but the consequent parts are far apart.

$$w_{(i,j)} = \begin{cases} \dfrac{L_{(i,j)} \times \sum_{k \in A_{(i,j)}} V_{(i,j)k}}{|A_{(i,j)}|} - (Z_{(i,j)} - 1) & |A_{(i,j)}| \neq 0 \\ -Z_{(i,j)} & |A_{(i,j)}| = 0 \end{cases}$$

   W_i is computed as follows:

$$W_i = \max(w_{(i,1)}, w_{(i,2)}, \ldots, w_{(i,j)}, \ldots, w_{(i,|E|)})$$

   2) **Unexpected conditions:** The consequents are similar but the conditional parts are far apart. Two types of ranking are possible.
      (a) Contradictory conditions
      (rules with $|A_{(i,j)}| > 0$ will be ranked higher)

$$w_{(i,j)} = \begin{cases} Z_{(i,j)} - L_{(i,j)} \times \left(\dfrac{\sum_{k \in A_{(i,j)}} V_{(i,j)k}}{|A_{(i,j)}|} - 1 \right) & |A_{(i,j)}| \neq 0 \\ Z_{(i,j)} & |A_{(i,j)}| = 0 \end{cases}$$

      W_i is computed as follows:

$$W_i = \max(w_{(i,1)}, w_{(i,2)}, \ldots, w_{(i,j)}, \ldots, w_{(i,|E|)}).$$

      (b) Unanticipated conditions
      (rules with $|A_{(i,j)}| = 0$ will be ranked higher)

$$w_{a(i,j)} = \begin{cases} Z_{(i,j)} - L_{(i,j)} \times \left(\dfrac{\sum_{k \in A_{(i,j)}} V_{(i,j)k}}{|A_{(i,j)}|} + 1 \right) & |A_{(i,j)}| \neq 0 \\ Z_{(i,j)} & |A_{(i,j)}| = 0 \end{cases}$$

$$w_{b(i,j)} = \begin{cases} Z_{(i,j)} \times L_{(i,j)} \times \left(\dfrac{\sum_{k \in A_{(i,j)}} V_{(i,j)k}}{|A_{(i,j)}|} + 1 \right) & |A_{(i,j)}| \neq 0 \\ 0 & |A_{(i,j)}| = 0 \end{cases}$$

      W_i is computed as follows:

$$W_i = \max(w_{a(i,1)}, w_{a(i,2)}, \ldots, w_{a(i,j)}, \ldots, w_{a(i,|E|)}) - \max(w_{b(i,1)}, w_{b(i,2)}, \ldots, w_{b(i,j)}, \ldots, w_{b(i,|E|)}).$$

4.2 Computing $V_{(i,j)k}$ and $Z_{(i,j)}$

For the computation of $V_{(i,j)k}$ and $Z_{(i,j)}$, we need to consider both the attribute values and the operators. Furthermore, the attribute value types (discrete or continuous) are also important. Since the computations of $V_{(i,j)k}$ and $Z_{(i,j)}$ are the same, it suffices to just consider the computation of $V_{(i,j)k}$, the degree of match for the *kth* matching attribute in $A_{(i,j)}$. Two cases are considered: the matching of discrete attribute values and the matching of continuous attribute values.

4.2.1. Matching of discrete attribute values

In this case, the semantic rule for each term (X) used in describing the existing rule must be properly defined over the universe of the discrete attribute. We denote U_k as the set of possible values for the attribute. For each $u \in U_k$, the user needs to input the membership degree of u in X, i.e., $\mu_X(u)$. For example, the user gives the following rule:

If *Grade* = *poor* **then** *Class* = *reject*

Here, *poor* is a fuzzy term. To describe this term, the user needs to specify the semantic rule for *poor*. Assume the universe of the discrete attribute *Grade* = {*A, B, C, D, F*}. The user may specify that "*poor*" grade means:

{(*A*, 0), (*B*, 0), (*C*, 0.2), (*D*, 0.8), (*F*, 1)}

where the left coordinate is an element in the universe of the "*Grade*" attribute, and the right coordinate is the degree of membership of that element in the fuzzy set *poor*, e.g., $\mu_{poor}(D) = 0.8$.

When evaluating the degree of match for $V_{(i,j)k}$, two factors play an important role, namely: the semantic rules associated with the attribute value descriptions and the operators used in the propositions. In the discrete case, the valid operators are "=" and "≠". For example, suppose that the two propositions to be matched are as follows:

User-supplied proposition: *attr Opu X*
System-generated proposition: *attr Ops S*

where *attr* is the matching attribute, *Opu*, *Ops* ∈ {=, ≠}, and X and S are the attribute values of *attr*.

The matching algorithm must consider the combination of both operators and attribute values. Four cases result:

Case 1. *Opu* = *Ops* = "=":

$$V_{(i,j)k} = \mu_X(S).$$

Case 2. *Opu* = "=" and *Ops* = "≠":

$$V_{(i,j)k} = \frac{\sum_{\substack{u \in \text{support}(X) \\ u \neq S}} \mu_X(u)}{|U_k| - 1},$$

where support(X) = {$u \in U_k \mid \mu_X(u) > 0$}, and $|U_k|$ is the size of U_k. $|U_k| - 1 = 0$ is not possible.

Case 3. *Opu* = "≠" and *Ops* = "=":

$$V_{(i,j)k} = \mu_{\neg X}(S).$$

Case 4. *Opu* = "≠" and *Ops* = "≠":

$$V_{(i,j)k} = \frac{\sum_{\substack{u \in \text{support}(\neg X) \\ u \neq S}} \mu_{\neg X}(u)}{|U_k| - 1},$$

where if *Ops* = "≠", $|U_k| - 1 = 0$ is not possible.

4.2.2. Matching of continuous attribute values

When an attribute takes continuous values, the semantic rule for the term (X) takes on the form of a continuous function. To simplify the user's task of specifying the

shape of this continuous function, we assume the function has a curve of the form as shown in Figure 2. Thus, the user merely needs to provide the values for *a*, *b*, *c*, and *d*.

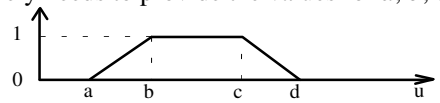

Figure 2. Membership function

For example, the user's pattern is:

If *Age* = *young* **then** *Class* = *accepted*.

Here, *young* is a term for variable *Age*. Suppose that in this case *Age* takes continuous values from 0 to 80. The user has to provide the 4 points using the values from 0 to 80, e.g., $a = 15$, $b = 20$, $c = 30$, and $d = 35$.

In the continuous case, the range of values that the operator can take is expanded to {=, ≠, ≥, ≤, ≤≤}. "≤≤" represents a range: $X_1 \leq attr \leq X_2$. With this expansion, the total number of possible cases to be considered is 25. Due to space limitation, we cannot list all the formulas here. Interested readers may refer to (Liu & Hsu, 1995).

5. Application in Data Mining

We have already mentioned that the proposed technique helps to solve the "interestingness" problem in data mining. We now outline the application in greater detail.

Let *D* be the database on which the data mining technique *T* is applied. Let *B* be the set of patterns (or rules) discovered by *T* in *D*. If we denote *I* as the set of interesting patterns (or rules) that may be discovered in *D*. Then, $I \subseteq B$. Three points to be noted:

- The set of patterns in *B* could easily be in the hundreds or even thousands, and many of the patterns in *B* are useless. But because of the sheer number of patterns, it is very difficult for a user to focus on the "right" subset *I* of patterns that are interesting or useful to him/her.
- Not all patterns in *I* are equally interesting. Different patterns may have different degrees of interestingness.
- *I* may be a dynamic set in the sense that the user may be interested in different things at different points in time.

In general, the size of *B* is much larger than the size of *I*. It is desirable that a system only gives the user the set of interesting patterns, *I*, and ranks the patterns in *I* according to their degrees of interestingness. Hence, the interestingness problem can be defined as follows:

Definition 6: Given *B*, determine *I* and rank the patterns in *I* according to their degrees of interestingness to the user at the particular point in time.

So, how can a computer system know what is useful in a domain and what is considered interesting at a particular moment to a user? What are the criteria used to rank the discovered patterns? Our proposed technique is able to provide a partial answer to these problems.

The interestingness problem has been discussed in many papers (Piatesky-Shapiro & Matheus 1994; Piatesky-Shapiro *et al* 1994; Silberschatz & Tuzhilin 1995). Many factors that contribute to the interestingness of a discovered pattern have also been proposed. They include: coverage, confidence, strength, significance, simplicity, unexpectedness, and actionability (Major & Mangano 1993; Piatesky-Shapiro & Matheus 1994). The first five factors are called objective measures. They can be handled with techniques that do not require user and domain knowledge. They have been studied extensively in the literature (e.g., Quinlan 1992; Major & Mangano 1993). The last two factors, unexpectedness and actionability, are called subjective measures. It has been noted in (Piatesky-Shapiro & Matheus 1994) that although objective measures are useful, they are insufficient in determining the interestingness of a discovered pattern. Subjective measures are needed. Our proposed technique is able to rank the discovered patterns according to their subjective interestingness. In particular, it helps to identify those unexpected patterns.

To date a number of studies have also been conducted on the subjective interestingness and some systems have been built (Major & Mangano 1993; Piatesky-Shapiro & Matheus 1994) with interestingness filtering components to help users concentrate on only the useful patterns. However, these systems handle the subjective interestingness in application specific fashions (Piatesky-Shapiro *et al* 1994; Silberschatz & Tuzhilin 1995). For such systems, domain-specific theories and expert knowledge are hard-coded into the systems, thus making them rigid and inapplicable to other applications.

In our proposed technique, we adopt a simple and effective approach:

1. The user is asked to supply a set of expected patterns, (not necessary a complete set).
2. The user then gives the semantic meanings to the attribute values (as described in Section 3).
3. The newly discovered patterns are then matched against the expected patterns and ranked according to different requirements from the user. Note that our technique does not identify the set of interesting patterns *I*. This task is left to the user.

The assumption of this approach is that some amount of domain knowledge and the user's interests are implicitly embedded in his/her specified expected patterns. With various types of ranking, the user can simply check the few patterns at the top of the list to confirm or to deny his/her intuitions (or previous knowledge), and to find those patterns that are against his/her expectation.

It should be noted that the user does not have to provide all his/her expected patterns at the beginning, which is quite difficult. He/she can actually do this analysis incrementally, and slowly modify and build up the set of expected patterns. The highly interactive nature of our technique makes this possible.

6. Evaluation

The proposed technique is implemented in Visual C++ on PC. A number of experiments have been conducted. A test example is given to evaluate how well the system performs its intended task of ranking the newly generated rules against the existing knowledge. An analysis of the complexity of the algorithm is also presented.

6.1. A test example

The set of rules is generated from a database using C4.5. The attribute names and values have been encoded to ensure confidentiality. Due to the space limitation, only a subset of the rules is listed below for ranking.

Rule 1: A1 <= 41
-> Class NO
Rule 2: A1 <= 49, A3 <= 5.49, A4 > 60
-> Class NO
Rule 3: A1 > 49, A2 = 2
-> Class YES
Rule 4: A1 > 49, A1 <= 50
-> Class YES
Rule 5: A1 > 55
-> Class YES
Rule 6: A1 > 41, A3 > 5.49, A7 > 106, A4 > 60, A10 <= 5.06
-> Class YES
Rule 7: A1 > 41, A4 <= 60
-> Class YES
Rule 8: A1 > 41, A1 <= 47, A3 <= 3.91, A7 > 106, A4 > 60, A10 <= 5.06
-> Class YES

Two runs of the system are conducted in this testing. In the first run, the focus is on finding similar (generated) rules, while in the second run the focus is on finding different rules (or unexpected rules). The system automatically cuts off rules with low matching values.

(I). Finding similar rules

The set of user's rules is listed below with the fuzzy set attached to each term.

User's rule set 1:

Rule 1: A1 <= Mid {a = 30, b = 35, c = 45, d = 50}
-> Class NO {(NO, 1), (YES, 0)}
Rule 2: A1 >= Ret {a = 50, b = 53, c = 57, d = 60}
-> Class YES {(NO, 0), (YES, 1)}

- **Ranking results**:

RANK 1 Rule 5: A1 > 55
-> Class YES
confirming user specified rule 2
RANK 2 Rule 1: A1 <= 41
-> Class NO
confirming user specified rule 1
RANK 3 Rule 3: A1 > 49, A2 = 2
-> Class YES
confirming user specified rule 2
RANK 4 Rule 7: A1 > 41, A4 <= 60
-> Class YES
confirming user specified rule 2
RANK 5 Rule 2: A1 <= 49, A3 <= 5.49, A4 > 60
-> Class NO
confirming user specified rule 1

(II). Finding different rules

The set of user's rules for this test run are listed below, which is followed by three types of ranking for finding different rules.

User's rule set 2:

Rule 1: A7 >= HI, {a =147, b=148, c=152, d = 153}
A4 >= HI {a = 92, b = 93, c = 97, d = 98}
-> Class YES {(NO, 0), (YES, 1)}
Rule 2: A3 >= MI {a =1.75, b =1.8, c = 2.2, d=2.25}
-> Class YES {(NO, 0), (YES, 1)}

- **Unexpected consequent:**

RANK 1 Rule 2: A1 <= 49, A3 <= 5.49, A4 > 60
-> Class NO
contradicting user specified rule 2

- **Contradictory conditions:**

RANK 1 Rule 7: A1 > 41, A4 <= 60
-> Class YES
contradicting user specified rule 1
RANK 2 Rule 6: A1 > 41, A3 > 5.49, A7 > 106, A4 > 60, A10 <= 5.06
-> Class YES
contradicting user specified rule 1
RANK 3 Rule 8: A1 > 41, A1 <= 47, A3 <= 3.91, A7 > 106, A4 > 60, A10 <= 5.06
-> Class YES
contradicting user specified rule 1

- **Unanticipated conditions:**

RANK 1 Rule 3: A1 > 49, A2 = 2
-> Class YES
RANK 2 Rule 4: A1 > 49, A1 <= 50
-> Class YES
RANK 3 Rule 5: A1 > 55
-> Class YES
RANK 4 Rule 7: A1 > 41, A4 <= 60
-> Class YES

6.2 Efficiency analysis

Finally, let us analyze the runtime complexity of the algorithm that implements the proposed technique. Assume the maximal number of propositions in a rule is N. Assume the attribute value matching (computing $V_{(i,j)k}$ and $Z_{(i,j)}$) takes constant time. Combining the individual

matching values to calculate $w_{(i,j)}$ also takes constant time. The computation of W_i is $O(|E|)$. Then, without considering the final ranking, which is basically a sorting process, the worst-case time complexity of the technique is $O(|E||B|N^2)$.

7. Related Work

To the best of our knowledge, there is no reported work on comparing existing rules with the newly generated rules. Most of the work on machine learning focuses on the generation of rules from various types of datasets as well as pruning of the generated rules (e.g., Quinlan 1992; Breiman et al 1984; Clark & Matwin 1993). Some systems also use existing domain knowledge in the induction process (e.g., Ortega & Fisher 1995; Clark & Matwin 1993; Pazzani & Kibler 1992). However, their purpose is mainly for helping the induction process so as to increase learning efficiency and/or to improve prediction accuracy of the generated rules. Clearly, the focus of their research is quite different from ours because our technique is primarily a post-analysis method that aims to help the user or an intelligent agent to analyze the rules generated.

In the areas of data mining research, the interestingness issue (Piatesky-Shapiro & Matheus 1994; Piatesky-Shapiro *et al* 1994) has long been identified as an important problem (see Section 5).

(Piatesky-Shapiro & Matheus 1994) studied the issue of subjective interestingness in the context of a health care application. The system (called KEFIR) analyzes the health care information to uncover interesting deviations (from the norms) or "key findings". A domain expert system is built to identify findings that are actionable and the actions to be taken. KEFIR does not do rule comparison, nor does it handle the unexpectedness problem. The KEFIR approach is also application specific. Its domain knowledge is hard-coded in the system as production rules. Our system is domain independent.

(Silberschatz & Tuzhilin 1995) proposed to use probabilistic beliefs as the framework for describing subjective interestingness. Specifically, a belief system is used for defining unexpectedness. However, (Silberschatz & Tuzhilin 1995) is just a proposal, and no system was implemented. Our proposed approach has been implemented and tested. In addition, our approach allows the user to specify his/her beliefs in fuzzy terms which are more natural than complex probabilities that the user has to assign in (Silberschatz & Tuzhilin 1995).

8. Conclusion

This paper tries to bridge the gap between the user and the rules produced by an induction system. A fuzzy matching technique is proposed for rule comparison in the context of classification rules. It allows the user to compare the generated rules with his/her hypotheses or existing knowledge in order to find out what is right and what is wrong about his/her knowledge, and to tell what has changed since the last learning. This technique is also useful in data mining for solving the interestingness problem. The proposed technique is general, and highly interactive. We do not claim, however, that the issues associated with the rule comparison are fully understood. In fact, much work remains to be done. We believe the proposed technique represents a major step in the right direction.

Acknowledgments: We would like to thank Hing-Yan Lee and Hwee-Leng Ong of Information Technology Institute for providing us the databases, and for many useful discussions. We thank Lai-Fun Mun and Gui-Jun Yang for implementing the system. The project is funded by National Science and Technology Board.

References

Breiman, L., Friedman, J., Olshen, R., and Stone, C. 1984. *Classification and regression trees*. Belmont: Wadsworth.

Clark, P and Niblett, T. 1989. "The CN2 induction algorithm." *Machine Learning* 3, 261-284.

Clark, P. and Matwin, S. 1993. "Using qualitative models to guide induction learning." In *Proceedings of the International Conference on Machine Learning*, 49-56.

Liu, B. and Hsu, W. 1995. *Finding interesting patterns using user expectations*. DISCS Technical Report.

Major, J., and Mangano, J. 1993. "Selecting among rules induced from a hurricane database." In *Proceedings of the AAAI-93 Workshop on KDD*.

Michalski, R. 1980. "Pattern recognition as rule-guided induction inference." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2, 349-361.

Ortega, J., and Fisher, D. 1995. "Flexibly exploiting prior knwledge in empirical learning." *IJCAI-95*.

Pazzani,M. and Kibler,D. 1992. "The utility of knowledge in inductive learning." *Machine learning* 9.

Piatesky-Shapiro, G., and C. J Matheus. 1994. "The interestingness of deviations." In *Proceedings of the AAAI-94 Workshop on KDD*, 25-36.

Piatetsky-Shapiro, G., Matheus, C., Smyth, P. and Uthurusamy, R. 1994. "KDD-93: progress and challenges ...." *AI magazine*, Fall, 1994, 77-87.

Quinlan, J. R. 1992. *C4.5*: *program for machine learning*. Morgan Kaufmann.

Silberschatz, A. and Tuzhilin, A. 1995. "On subjective measures of interestingness in knowledge discovery." In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*.

Zimmermann, H. J. 1991. *Fuzzy set theory and its applications*. Kluwer Academic Publishers.