

Generating Classification Rules According to User's Existing Knowledge *

*Shu Chen and Bing Liu, School of Computing, National
University of Singapore, 3 Science Drive 2, Singapore 117543.
liub@comp.nus.edu.sg*

Abstract

An important problem in applying classification rule induction techniques to practical applications is how to produce rules that are related to the user's existing knowledge about the domain and his/her current interests. Such rules are interesting to the user, and also easily understood and trusted by the user. They can enhance the existing knowledge of the domain and be relied upon in real-world performance tasks. Past research and applications have shown this to be a crucial requirement in many real-life applications. Existing techniques for dealing with this problem typically use sophisticated methods to bias the rule induction process in order to produce rules that are consistent with the existing knowledge. In this paper, we propose a novel and simple approach. It only needs to pre-process the data using the user's existing knowledge. It does not make any modification to the rule induction technique. Practical applications have shown that this simple approach is surprisingly effective and flexible. It demonstrates that to obtain useful results, we do not necessarily need to use sophisticated techniques. Sometimes simple approaches may just be sufficient.

1. Introduction

The user's objective in using a knowledge discovery system (KDS) is to find *interesting* rules that are understandable [e.g., 5, 8, 9, 21, 22, 25, 20]. Rules are easy to understand if

* This paper has been prepared in cooperation with the Society for Industrial and Applied Mathematics.

they conform to or are consistent (fully or at least partially) with the prior knowledge of the domain expert or the user [20]. Rules are interesting if they are *useful* (or *actionable*) [21] and/or *unexpected* [25, 8, 9]. However, for a KDS to know what rules are interesting and understandable to a user is not an easy task. A rule can be interesting to one user but not interesting to another. Thus, whether a rule is interesting and understandable or not is subjective. It depends on the user's prior knowledge about the domain and his/her current interests.

It is now well recognized that applying standard classification rule learning tools in real-life applications is by no means a straight forward task [20] because these tools often produce rules that are completely unrelated to the user's existing concepts about the domain and the user's interests. These rules can be very difficult to understand and/or uninteresting to the user. The root of the problem is that in order to produce a small set of accurate rules to form a model of the domain [23] rule induction systems use various biases in its rule generation process. These biases, however, may not be in agreement with the existing knowledge of the human user, thus resulting in the rule understandability and interestingness problems. That is, many of the rules generated do not make sense to the user and/or are not interesting to the user.

The existing method for dealing with this problem in machine learning is to use the user's knowledge as biases in rule induction in order to generate rules that conform to or are consistent with the existing knowledge [1, 15, 17, 20]. This technique involves sophisticated modifications to the rule induction systems [1, 15, 17, 20]. They also do not deal with the generation of unexpected rules. Unexpected rules are, by definition, interesting.

In data mining research, most current approaches for dealing with the issue of finding interesting rules employ a post-analysis module at the back-end of a mining system [8, 9, 10, 11, 21]. This module uses the user's input knowledge to help him/her identify interesting rules. However, post-analysis cannot find the type of interesting rules studied in this paper because they are not in the set of discovered classification rules. Although there are approaches for finding unexpected rules using the user's existing knowledge, they are for association rule mining [18, 19]. Association rule mining, however, cannot handle numerical attributes, which are very common in classification datasets. To perform association rule mining, numerical attributes have to be discretized. Current discretization techniques, however, do not consider interactions among the attributes and thus tend to destroy the inherent relationships among them.

In this paper, we propose a novel and yet simple approach to generate rules that are related to the user's existing knowledge about the domain and his/her current interests. The essence of the proposed technique is to pre-process the database using user's existing knowledge and then use a existing rule induction system (we use C4.5) to discover the interesting and understandable rules. Our technique does not need any modification to C4.5. The proposed technique has been used in a number of real-life applications, three medical applications and two education applications. The results show that this simple approach is surprisingly effective and flexible. It demonstrated that to solve an important problem, we do not necessarily need to design sophisticated techniques. Some simple approaches may well be sufficient.

2. Preliminary Discussions

Since we will use C4.5 [23] for rule generation, this section first reviews the C4.5 rule generation process. It then discusses the problem that we are going to address.

A database D for C4.5 consists of the descriptions of N objects in the form of tuples. These N objects have been classified into q known classes, $C_1, \dots, C_m, \dots, C_q$. Each object in the database is described by n distinct attributes, $Attr_1, \dots, Attr_i, \dots, Attr_n$. The objective of C4.5 is to find a set of characteristic descriptions (or classification rules) for the q classes. A classification rule in C4.5 has the following form:

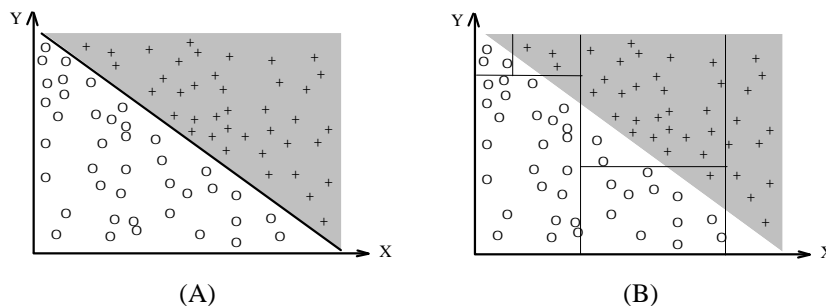
$$P_1, \dots, P_i, \dots, P_r \rightarrow C$$

where “,” means “and”, and P_i is a test on an attribute and C is a class.

To facilitate understanding of the proposed technique, let us review the geometric interpretation of how C4.5 works. C4.5 works by first building a decision tree and then producing a set of classification rules from the tree. Geometrically, we can view the decision tree as specifying how a description space of the tuples is to be carved up into regions associated with the classes. The regions produced by a decision tree are all hyperrectangles [23]. When the task at hand is such that the class regions are not hyperrectangles, the decision tree will approximate the regions with a set of hyperrectangles. This is illustrated by a simple example in Figure 1(A) in which 80 cases (or tuples) of two classes (represented by “o” and “+”) are described by two continuous attributes, X and Y . The intended division of the description space by an oblique line is shown in Figure 1(A), while Figure 1(B) displays the approximation to this division that is found by C4.5.

We first discuss the understandability issue. The main reason that the generated rules are hard to understand is because they are not related to the user’s existing concepts.

The understandability problem, in general, can be explained as follows: Let us use the previous example in Figure 1(A) to illustrate. The user’s concept space is represented as the dark-shaded area in Figure 1(C). Note that there are rectangles formed by C4.5 that do not intersect with the user’s concept space. They represent things that are foreign to the user. Then the rules associated with these rectangles are more difficult to understand (but they could still be interesting, see below). Note also that there are other rectangles formed by C4.5 that cross the boundaries of the user’s concept space. These crossings also make the generated rules difficult to understand because they are confusing to the user. In this paper, we propose a technique that will guide the rule generation process such that the rules generated will be as closely related to the boundaries of the user’s concept space as possible. This is achievable because of the observation that the division produced by C4.5 (e.g., in Figure 1(B)) is by no means unique. Clearly, it is possible to divide the space in many other ways. For example, Figure 1(D) shows another possible division. The proposed technique makes use of this observation to force C4.5 to generate rules that are related to the user’s concepts.



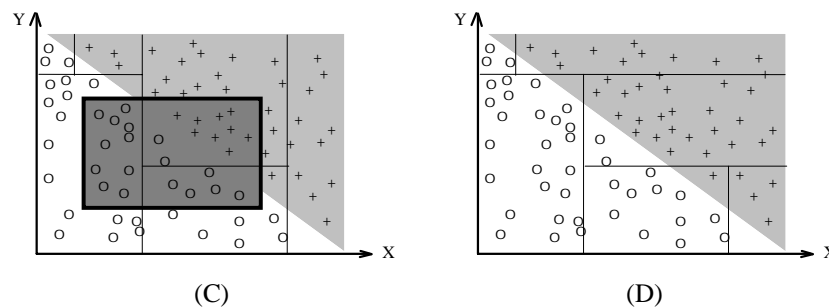


Figure 1. Real and approximate division for an artificial task.

Let us now turn our attention to the meaning of interesting rules. Past research has proposed two main factors that contribute to the subjective interestingness of a rule, unexpectedness and actionability [21, 8, 25].

Unexpectedness: Rules are interesting if they “surprise” the user.

Actionability: Rules are interesting if the user can do something with them to his/her advantage. In other words, rules are useful when they can help to achieve some current goals of the user.

These two measures are not mutually exclusive [25]. Thus, we can classify subjectively interesting rules into three categories according to the above definitions: (1) rules that are both unexpected and actionable; (2) rules that are unexpected but not actionable, and (3) rules that are actionable but expected. In this paper, we handle (1) and (2) by finding unexpected rules and handle (3) by finding the rules that conform to the user’s expectations. Whether a conforming rule or an unexpected rule is actionable or not, it will be decided by the user.

We now use a real example to illustrate the problem we are going to address. This example uses the credit screening database created by Chiharu Sano in UCI machine learning repository [14]. The database has 125 tuples, 10 attributes and two classes *Yes* and *No* representing whether credit has been granted. Without considering any existing knowledge, the set of rules generated by C4.5 is as follow:

R1:	Age > 25, Savings > 7, YR_Work > 2 →	Yes
R2:	Sex = Male, YR_Work > 2 →	Yes
R3:	Jobless = No, Bought = pc →	Yes
R4:	Bought = medinstru, Age ≤ 34 →	Yes
R5:	Sex = Female, Age ≤ 25 →	No
R6:	Savings ≤ 7, M_LOAN > 7 →	No
R7:	YR_Work ≤ 2 →	No

However, the user believes that the following rule (called the user expected rule) is true from experience:

$$\text{Bought} = \text{Jewel}, \text{Sex} = \text{Female} \rightarrow \text{No}.$$

This rule conveys to important messages:

1. The user believes this rule should be true or it is the user’s hypothesis.
2. This rule is closely related to the user’s current interests about the domain.

By looking at the discovered rules above, we have no idea how correct the user expected rule is, and whether there are rules related to the user’s concept because the generated

rules have little relationship with the user's concept.

To find the correctness of the expected rule is easy, e.g., by testing the rule against the database. In this case, the above expected rule is only correct 28.6% of the time (14 tuples satisfy the conditions, but out of these 14 tuples only 4 of them satisfy the conclusion). The question is: can we characterize these conforming and unexpected tuples? This paper proposes such a technique. For the above problem, our proposed technique is able to force C4.5 to produce the following related rules and more (see Section 3.2 for more details):

Rule1: Bought = jewel, Sex = Female, YR_Work > 2 → Yes

Rule2: Bought = jewel, Sex = Female, YR_Work ≤ 2 → No

Rule2 is a conforming rule and Rule1 is an unexpected rule. After seeing these two rules, the user will know exactly what is wrong with his/her existing concept, and also find something unexpected, i.e., Rule1. These rules are easily understood by the user because they are closely related to the user's concept.

The above problem can be illustrated geometrically in Figure 1(C). For example, a user expected rule (e.g., its class is "o") is represented as the dark-shaded area in Figure 1(C). The area crosses the boundaries of 3 regions formed by C4.5. Within the user expected rule, there are "o"s which represent conforming tuples, and "+"s which represent unexpected tuples. Obviously, from the division produced by C4.5 it is hard to know what are the characteristics of the conforming and unexpected tuples. Our proposed technique is able to make C4.5 to produce the conforming and unexpected rules.

3. The Proposed Technique

The key point of the proposed technique is to use the user's concepts to help C4.5 discover interesting rules. The user's concepts are expressed as a set of expected rules, which are in the same format as the generated rules. Using this representation to represent user concepts is natural because when the user is looking for a particular type of rules (e.g., classification rules), his/her expectations are usually of the same type.

Three main steps are involved in the proposed approach:

- Step 1. The user provides a set of expected rules E that he/she expects to find in the database. Each $E_j \in E$ is expressed in the same format as discovered rules.
- Step 2. Pre-process the database D to produce D' . This step (1) computes the correctness of each $E_j \in E$, and (2) assigns a new class *CONFORM* to each tuple that conforms to E and assigns a new class *UNEXPECTED* to each tuple that is unexpected with respect to E . At the conclusion of this step, all the tuples in D are classified as conforming, unexpected or unrelated tuples (Section 3.1).
- Step 3. Run C4.5 on D' . C4.5 will produce three types of rules, conforming rules, unexpected rules and unrelated rules, which give different types of interesting information to the user.

We will not discuss Step 1 further in this paper as it simply asks the user to express his/her previous concepts in the form of a classification rule (see the example in Section 2). (1) of Step 2 is also simple. The correctness ($Corr_j$) of each $E_j \in E$, can be computed as following:

$$Corr_j = \frac{\text{Total number of tuples satisfying both the conditions and conclusion of } E_j}{\text{Total number of tuples satisfying the conditions of } E_j}$$

3.1 Pre-processing the database and running C4.5

During the pre-processing step, we introduce additional classes. A test is carried out on each tuple D_k in the database D . Tuples, that are found to be conforming or unexpected with respect to E , are assigned the *Conform* or *Unexpected* classes respectively (i.e., their original classes are replaced). The remaining tuples (that are unrelated to E) will retain their original classes. The process uses the following definitions.

Definition: D_k conforms to an expected rule $E_j \in E$ if D_k satisfies both the conditions and conclusion of E_j .

Definition: D_k is *unexpected* with respect to $E_j \in E$ if D_k satisfies the conditions of E_j but not its conclusion.

Definition: D_k is *unrelated* to $E_j \in E$ if D_k does not satisfies the conditions of E_j .

Definition: D_k conforms to E if $\exists E_j \in E$, D_k conforms to E_j .

Definition: D_k is *unexpected* with respect to E if $\forall E_j \in E$, D_k does not conform to E_j , and $\exists E_i \in E$, D_k is unexpected with respect to E_i .

Definition: D_k is *unrelated* to E if $\forall E_j \in E$, D_k is unrelated to E_j .

It must be stressed that these definitions are not unique. See the explanation to the algorithm below. We are now in the position to present the overall algorithm for Step 2.

```

1 Initialize RuleNoj and CondNoj to 0, 1 ≤ j ≤ |E|;
2 for each tuple Dk ∈ D do
3   Confm ← FALSE; Unexp ← FALSE;   Class ← the class of Dk;
4   for each Ej ∈ E do
5     if Dk satisfies the conditions of Ej then
6       Increment CondNoj
7       if Dk satisfies the conclusion of Ej then Increment RuleNoj; Confm ← TRUE
8       else Unexp ← TRUE
9     endif;
10  endif;
11  endfor;
12  if Confm then Change the class of Dk to <Class>CONFORM;
13  elseif Unexp then Change the class of Dk to <Class>UNEXPECTED;
14  endif;
15  endfor;
16 for each Ej ∈ E do Corrj = RuleNoj / CondNoj endfor;
```

Notes about the algorithm:

- Lines 1 and 3 are the initialization.
- Line 5-7 prepare the values $RuleNo_j$ and $CondNo_j$ for the computation of the correctness of E_j in Line 16.
- Line 7 indicates that D_k satisfies E_j . We say that D_k conforms to E_j . Line 8 indicates that D_k satisfies only the conditional part of E_j , but not the conclusion. We say that D_k

is unexpected with respect to E_j .

- Lines 12 and 13 assign new classes to tuples that conform to E and that are unexpected with respect to E . It should be noted that there may be contradictory situations, i.e., D_k conforms to E_j but is unexpected with respect to E_m ($j \neq m$). In such situations, the above algorithm and the definitions treat D_k as a conforming tuple. Alternatively, we could treat D_k as an unexpected tuple, or assign it a *CONTRADICTORY* class. All these variations have been implemented in our system as options to the user. In fact, this technique is so flexible that it is also possible not to assign conforming (or unexpected) classes if the user is not interested in the classes (see the example in Section 3.2 below). Note that the new classes introduced are: $\langle \text{Class} \rangle \text{CONFORM}$ and $\langle \text{Class} \rangle \text{UNEXPECTED}$. The interpretation is as follows: $\langle \text{Class} \rangle$ is the original class of D_k and *CONFORM* indicates that D_k is found to be conforming to E . For example, the original class of D_k is *Yes* and D_k is conforming, then the new class of D_k will be *YesConform*.

After the pre-processing step, we obtain the modified D , denoted by D' . Next, we run C4.5 on D' to produce three types of rules: conforming rules, unexpected rules and unrelated rules.

3.2 Why does this simple technique work?

To answer this question, let us look at the situation when there is only one expected rule in E , call it R . When there are multiple rules in E , the situation is similar but more complex.

Basically, the pre-processing divides the tuples in D into three groups: conforming tuples, unexpected tuples and unrelated tuples. Then C4.5 will produce rules to distinguish the new classes $\langle \text{Class} \rangle \text{CONFORM}$, $\langle \text{Class} \rangle \text{UNEXPECTED}$ and the original classes (used by unrelated tuples) thus resulting in the conforming, unexpected and unrelated rules. To illustrate, let us use the geometric interpretation example in Figure 1(C) (reproduced as Figure 2(A)). Since the user rule R has the same format and meaning as the generated rule, then R also represents a hyperrectangle. Let the conditional part of R be the dark-shaded rectangle in Figure 2(A) and the class of R be "o". Using the pre-processing algorithm above, the new division produced by C4.5 is shown in Figure 2(B). Then, region 1 and 3 represents two conforming rules (covering conforming tuples, "c"s), and region 2 represents an unexpected rule (covering unexpected tuples, "u"s). The rest of the regions represents unrelated rules. Obviously, the conforming and unexpected rules are easy to understand by the user because they are related to the user concepts (their boundaries are related), so are the unrelated rules.

The proposed technique is very flexible as it can force C4.5 to produce many interesting rules (as mentioned in Section 3.1). For example, if the user is not interested in conforming rules but only unexpected rules, the situation in Figure 2(C) is produced by C4.5. In this case, there is only one unexpected rule (region 1). This implies that we are able to summarize and present only the relevant rules that are of interest to the user.

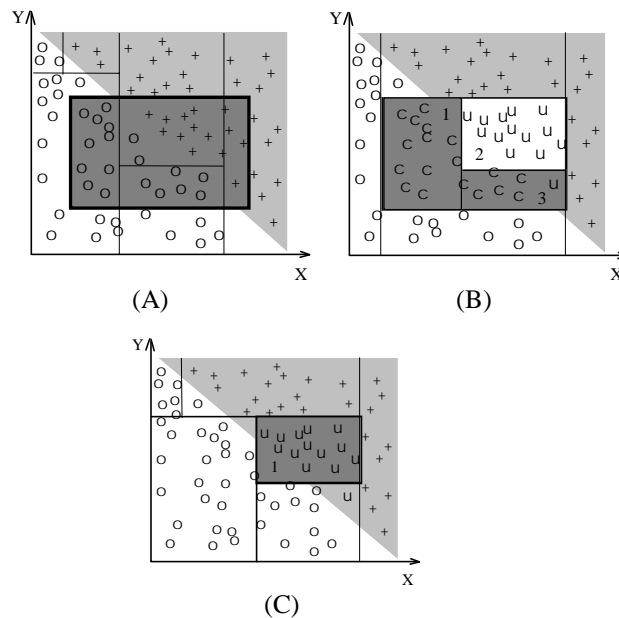


Figure 2. The new divisions because of the user's rule

4. Illustrations

We have tested the system using a number of public domain databases. The system has also been used in 3 real-life medical applications, and 2 education applications. Since there is no reported technique that is able to perform this task, we are unable to do any comparison. Here, we give some test runs of our system using a public domain database and a real-life disease database to illustrate its use.

4.1 Example runs

The first two example runs use the credit screening database created by Chiharu Sano in UCI machine learning repository [14]. We choose this database because it is easy to understand. This database has 125 tuples, 10 attributes and 2 classes *Yes* and *No* representing whether credit has been granted. The meanings of the attributes appeared below are self-explanatory. Without considering any user expected rule, the set of rules generated by C4.5 is as follow (the value in [] following each rule is the predicted accuracy of the rule produced by C4.5):

R1:	Age > 25, Savings > 7, YR_Work > 2 →	Yes [94.8%]
R2:	Sex = Male, YR_Work > 2 →	Yes [93.4%]
R3:	Jobless = No, Bought = pc →	Yes [84.1%]
R4:	Bought = medinstru, Age ≤ 34 →	Yes [82.0%]
R5:	Sex = Female, Age ≤ 25 →	No [56.8%]
R6:	Savings ≤ 7, M_LOAN > 7 →	No [54.6%]
R7:	YR_Work ≤ 2 →	No [54.4%]

Assume the user believes that the following rule should be true from his/her experience:

Age > 23, M_LOAN <= 7 → Yes

By looking at the generated rules above, we have no idea how correct the user expected rule is, and whether there are conforming rules and unexpected rules, etc., because the generated rules have little relationship to the user's concept. This also makes the generated rules hard to understand with respect to the user's concept. However, if we look at the rules (below) produced using the proposed technique, these questions can all be answered easily.

- **Correctness of the expected rule:**

Correctness : 73.3%

From the indicated correctness, we see that the rule is valid about 73.3% of the time.

- **Generated rules using the proposed technique:**

Rule 1: Sex = Female, Age > 23, Age <= 26 →	NoUnexpected [75.8%]
Rule 2: Jobless = Yes, Age <= 40, M_LOAN <= 7 →	NoUnexpected [75.8%]
Rule 3: Age > 65, M_LOAN <= 7 →	NoUnexpected [63.0%]
Rule 4: Jobless = No, Age > 26, Age <= 65, M_LOAN <= 7 →	YesConform [95.0%]
Rule 5: Jobless = No, Sex = Male, Age > 23, Age <= 65, M_LOAN <= 7 →	YesConform [87.9%]
Rule 6: Jobless = No, Age > 21, Savings > 9, M_LOAN > 7 →	Yes [81.9%]
Rule 7: Bought = pc, Age <= 23 →	Yes [70.7%]
Rule 8: Age <= 23, Savings > 10, M_LOAN <= 6 →	Yes [61.2%]
Rule 9: Savings <= 9, M_LOAN > 7 →	No [61.2%]
Rule 10: Age > 32, M_LOAN > 7, YR_Work <= 2 →	No [58.7%]
Rule 11: Age <= 23, M_LOAN > 6 →	No [56.6%]
Rule 12: Age <= 23, Savings <= 30, M_LOAN <= 7, Months > 8 →	No [47.5%]

Let us make some observations about this set of rules:

- The first three rules show the situations where the expected conditions can lead to unexpected conclusion. For example, for the tuples that satisfy Age > 23 and M_LOAN <= 7.5, a person is not granted credit if the person is female and her age is between 23 and 26 (Rule 1). This is certainly unexpected.
- The next two rules show the conforming situations, but with more restrictive conditions. For example, in Rule 4, there are extra conditions on Jobless and Age. After analyzing the first 5 rules, the user would have a much clearer picture about the database with respect to his/her expectation.
- The last 7 rules show the characteristic descriptions of the tuples that are unrelated to the user expectation. They may also be interesting because they are unknown to the user.
- This rule set is relatively easy to understand because the rules are related to the user's concept. We know that the first 5 rules all satisfy the conditions of the user rule, and the last 7 rules do not.
- Finally, the new technique tends to produce more rules and more conditions in each rule. This is because of the extra classes, which require C4.5 to make finer classification.

Evaluation on the database produces the following statistics. *Used* means how many tuples satisfy the rule's conditions, and *Wrong* means how many tuples are classified wrongly when they satisfy the rule's conditions.

Rule	Used	Wrong	Class
1	5	0	NoUnexpected
2	5	0	NoUnexpected
3	3	0	NoUnexpected
4	27	0	YesConform
5	21	1	YesConform
6	46	6	Yes
7	4	0	Yes
8	6	1	Yes
9	6	1	No
10	14	4	No
11	8	2	No
12	17	7	No

Note that these statistics are different from those produced by C4.5. In our case, we are more interested in each individual rule rather than how the whole rule set performs in its predication as in C4.5. Furthermore, in C4.5, the ordering of rules is important, but for us, the ordering is irrelevant. We test every rule against every tuple in the database. From the table, we observe that both the unexpected rules and the conforming rules are quite accurate. With these rules, the user may decide to take some actions. For example, he/she may want to find out why young females are not granted credit and if this is undesirable, he/she can propose corrective procedures to ensure that it does not happen in the future.

Let us next show an example (using the same database) that has more than one expected rule. Suppose the user has two expected rules.

Expected Rule 1:	Sex = Female, Bought = Jewel	→ No
Expected Rule 2:	Age >= 48, Sex = Female	→ No

- **Correctness of the expected rules:**

Correctness: Expected Rule 1: 28.6% Expected Rule 2: 46.7%

- **Generated rules using the proposed technique:** There are 11 generated rules altogether. Below, we only list the conforming and unexpected rules:

Rule 1:	Sex = Female, Age > 52 →	NoConform [82.0%]
Rule 2:	Sex = Female, Age > 47, Age <= 52 →	YesUnexpected [84.1%]
Rule 3:	Bought = Jewel, Sex = Female, Age <= 52 →	YesUnexpected [77.7%]

Evaluation on the database produces the following statistics:

Rule	Used	Wrong	Class
1	7	0	NoConform
2	8	0	YesUnexpected
3	11	1	YesUnexpected

Rule 1 says that only when Sex = Female and Age > 52, the person will not be granted credit. When the age is between 48 (Age > 47) and 52 (Rule 2), the person is granted credit, which is unexpected. When Bought = Jewel, Sex = Female, and Age <= 52 (Rule 3), the person is also given credit, which is also unexpected.

Finally, we show an example run using one of our real-life disease databases. This database has 713 tuples, 9 attributes, and 2 classes, *YES* and *NO*, representing whether the person has the disease. Without considering any user expected rule, the set of rules generated by C4.5 is as follow:

Rule 1:	Sex = FEMALE, Age > 50, DBP <= 72 →	Class = YES [91.7%]
Rule 2:	Sex = FEMALE, Age > 50, LDL > 4.58, SBP <= 160, DBP <= 82 →	Class = YES [90.6%]
Rule 3:	Age > 41, HDL > 1.01, DBP <= 59 →	Class = YES [79.4%]
Rule 4:	Age > 62 →	Class = YES [77.8%]
Rule 5:	Sex = MALE, Age > 49, LDL > 6.73 →	Class = YES [75.8%]
Rule 6:	Sex = FEMALE, Age > 49, LDL > 3.91, LDL <= 4.17 →	Class = YES [75.8%]
Rule 7:	Age > 41, LDL > 5.52, SBP > 112, GLUC <= 4.84 →	Class = YES [75.6%]
Rule 8:	Age > 49, Age <= 50 →	Class = YES [75.6%]
Rule 9:	Age > 49, LDL > 3.24, LDL <= 3.73 →	Class = YES [70.7%]
Rule 10:	Age > 32, HDL <= 0.39, TG <= 2.58 →	Class = YES [64.5%]
Rule 11:	Age <= 32 →	Class = NO [99.2%]
Rule 12:	Age <= 62, LDL <= 3.24 →	Class = NO [98.1%]
Rule 13:	Ethnic = CHINESE, Age <= 41, HDL > 0.39 LDL <= 5.57, GLUC > 3.85 →	Class = NO [98.0%]
Rule 14:	Age <= 49, DBP > 59, TG > 0.61, TG <= 0.84 →	Class = NO [96.0%]
Rule 15:	Sex = MALE, HDL <= 0.8, LDL <= 6.73 →	Class = NO [94.7%]

Now, the doctor believes that the following rule should be true from his/her experience:

Age >= 43, SBP >= 148 → Class = YES

Again, by looking at the generated rules, we have no idea how correct the expected rule is, and whether there are conforming and unexpected rules. The running results using the proposed technique are listed below:

- **Correctness of the expected rule:**

Correctness : 44%

From the indicated correctness, we see that the rule is valid about 44% of the time.

- **Generated rules using the proposed technique:**

The conforming and unexpected rules are listed below. The total number of rules produced is 15 (including unrelated rules).

Rule 1.	Age > 61, LDL > 3.42, SBP > 147 →	Class = YESConform [82.0%]
Rule 2.	LDL > 3.42, SBP > 168, GLUC <= 5.28 →	Class = YESConform [79.4%]
Rule 3.	Age > 42, Age <= 61, LDL > 3.7, SBP > 148, SBP <= 168 →	Class = NOUnexpected [79.4%]
Rule 4.	Age > 42, Age <= 61, SBP > 147, GLUC > 5.28 →	Class = NOUnexpected [66.2%]

Evaluation on the database produces the following statistics.

Rule	Used	Wrong	Class
1	8	0	YESConform
2	7	0	YESConform
3	12	1	NOUnexpected
4	7	1	NOUnexpected

From the above, we can see that Rule 3 and Rule 4 show the situations where expected conditions can lead to unexpected conclusion. The rules are quite accurate. For example, in Rule 3, a person that satisfies $SBP \geq 148$ is not likely to suffer from the disease if he/she is less than 61 years old and the LDL measure is greater than 3.7. This is unexpected.

Rule 1 and Rule 2 show the conforming situations, but with more restrictive

conditions. For example, in Rule 1, it is only when the person's age is greater than 61 years old and his/her LDL measure is greater than 3.42, then he/she is likely to suffer from the disease. After analyzing the 4 interesting rules, the user has a much better understanding of the database with respect to his/her expectation.

4.2. Discussions

Below, we make some observations about the proposed technique and its use.

- The proposed technique works best if in each run there is only one expected rule or all the expected rules are mutually exclusive (i.e., no two expected rules cover a set of common tuples in the database). In this situation, the user can clearly see the conforming and unexpected rules with respect to each individual expected rule. This may be inefficient if the database is very large. However, in many applications of classification rule induction the databases are not that large.
- In certain situations, the number of conforming (or unexpected) rules produced for each expected rule can be large and/or their accuracy may also be quite low. This means that the conforming (or unexpected) tuples covered by the expected rule may be quite random, i.e., scattered in a number of areas. It may also mean that the conditions used by the expected rule are not discriminating.
- The normal methods [23] can still be used to test the accuracy of the conforming and unexpected rules on unseen tuples if the purpose of finding these rules is for future prediction rather than for simply understanding the regularities in the existing data.

5. Related Work

Classification rule induction is an important topic in machine learning. However, the focus of machine learning has been on generating short and accurate rules [e.g., 23, 2, 15, 17, 20]. Systems, that employ existing domain knowledge or theory in the induction process [e.g., 17, 20], mainly use the knowledge to generate more accurate rules or to improve the explainability of the rules [1]. Clearly, this is different from our work, where our primary concern is to generate subjectively interesting rules.

A branch of research in machine learning that is related to our work is theory refinement [12, 15, 24]. In theory refinement, the initial knowledge given is a coarse, perhaps incomplete or incorrect theory of the problem domain. Training examples are used to shape this initial theory into refined, more accurate theory. Refinement operators and hill-climbing search are normally used to modify the existing rules. Operators can specialize rules (e.g., adding new conditions), generalize rules (e.g., deleting some useless conditions), remove old rules and add new rules [24]. There are also other approaches to theory refinement, e.g., neural networks and integrated methods [e.g., 12, 3]. In our context, the user expected rules can be seen as the initial theory. However, our work is different from theory refinement because our focus is on finding subjectively interesting rules (e.g., unexpected rules) with regard to the user's concepts rather than a set of accurate rules that forms an effective theory of the application domain. The rules produced after theory refinement may no longer represent the original user's concept space because of the refinement operations (and the errors in the user's concepts). These rules still have the same problem as the one we discussed in Section 2 (may be to a lesser degree), i.e., the user's concept may intersect a number of rules. Our technique aims to help the user understand his/her existing concepts by carving up his/her existing concept space into conforming and/or unexpected regions (or rules). The proposed pre-processing

method, which simply assigns new classes to tuples (can be in different ways for different purposes, see Section 3.1 and 3.2), is also different from the operator-based rule modification approach and the neural network approach to theory refinement.

In the field of data mining, subjective interestingness [e.g., 21, 25, 8, 9, 10, 11, 18, 19] has long been identified as an important problem. A number of data mining systems have been built with post-analysis modules [e.g., 8, 9] to help the user identify interesting rules. However, post-analysis cannot produce the type of interesting rules studied in this paper because post-analysis does not generate new rules but simply evaluates the interestingness of the discovered rules according to some criteria. The interesting rules studied in this paper need to be generated directly from the database.

There are also a number of approaches that help the user finding interesting association rules. [7] proposes a template-based approach for finding interesting association rules. In this approach, the user specifies interesting and uninteresting association rules using templates. A template describes a set of rules in terms of items occurred in the conditional and the consequent parts. The system then retrieves the matching rules from the set of discovered rules.

[26] proposes an association rule mining algorithm that can take item constraints specified by the user in the rule mining process so that only those rules that satisfy the constraints are generated. [16] extends this approach further to allow much more sophisticated constraints to be specified by the user. It also uses the constraints to optimize the association rule mining process. The idea of using constraints in the rule mining process is important as it avoids generating irrelevant rules.

[18, 19] proposes a method of discovering unexpected association rules that takes into consideration a set of expectations or beliefs about the problem domain. The method discovers unexpected patterns using these expectations to seed the search for patterns in data that contradict the beliefs.

All the above methods for finding interesting association rules cannot be used in our context because association rule mining could not use numeric attributes, which often exist in classification datasets. Although discretization can be applied to partition each numeric attribute into intervals before using an association rule miner, discretization can destroy the inherent relationships of the attributes since each attribute is discretized independently. Such discretization may also produce intervals that make no sense to the user (thus hard to understand). Additionally, association rule mining is more suitable for sparse data such as supermarket transactions. It is not so suitable for classification datasets, which often contain a huge number of associations and can result in combinatorial explosion [10, 28]. When we have too many rules, it becomes very hard, if not impossible, for human to analyze.

[25] proposes to use belief systems to describe unexpectedness. A number of formal approaches to the belief systems are presented, e.g., Bayesian probability and Dempster-Shafer theory. These approaches require the user to provide complex belief information, such as conditional probabilities, which are difficult to obtain in practice.

There are also existing techniques that work in the contexts of specific domains. For example, [21] studies the issue of finding interesting deviations in a health care application. Its data mining system, KEFIR, analyzes health care information to uncover “key findings”. A domain expert system is constructed to evaluate the interestingness (in this case, actionability) of the “key findings”. The approach is, however, application specific. It also does not deal with association rules. Our method is general. It does not make any domain-specific assumptions.

[27, 11] describes a technique for discovering exceptional knowledge based on information theory. This technique does not use any user’s existing concepts. Clearly, it

is different from our work because we are interested in discovering interesting knowledge with respect to the user's existing concepts. Rules discovered by our method may not be exceptions, but general facts, which the user does not know about or has wrong existing concepts of them.

6. Conclusion

In this paper, we proposed a novel and yet simple approach to incorporating user's existing concepts in the learning process to discover a specific type of subjectively interesting rules. This technique only involves pre-processing of the database, i.e., introducing extra classes. It does not modify the underlying rule discovery system. The rule induction power of the underlying rule discovery system itself is employed to generate interesting rules. This approach has been found useful in a number of real-life medical applications.

References

- [1]. P. Clark and S. Matwin. "Using qualitative models to guide induction learning," *Proceedings of ICML-93*, pp. 49-56, 1993.
- [2]. P. Clark and T. Niblett. "The CN2 induction algorithm." *Machine Learning* 3, pp. 261-284, 1989.
- [3]. S. K. Donoho and L. A. Rendell. "Representing and restructuring domain theories: a constructive induction approach," *Journal of Artificial Intelligence Research*, vol. 2, pp. 411-446, July, 1995.
- [4]. R. R. Evans and D. Fisher, Overcoming process delays with decision tree induction, *IEEE Expert*, 9(1), pp. 60-66, 1994.
- [5]. U. Fayyad, G. Piatetsky-Shapiro and P. Smyth. "From data mining to knowledge discovery in databases." *AI Magazine*, pp. 37-54, 1996.
- [6]. M. Kamber, and R. Shinghal. "Evaluating the interestingness of characteristic rules," *KDD-96*, pp. 263-266, 1996.
- [7]. M. Klemetinen, H. Mannila, P. Ronkainen, H. Toivonen, and A.I. Verkamo. "Finding interesting rules from large sets of discovered association rules," *Proceedings of the Third International Conference on Information and Knowledge Management*, pp. 401-407, 1994.
- [8]. B. Liu and W. Hsu. "Post-analysis of learned rules," *AAAI-96*, pp. 828-834, 1996.
- [9]. B. Liu, W. Hsu and S. Chen. "Analyzing discovered classification rules using general impressions," *KDD-97*, 1997.
- [10]. B. Liu, W. Hsu, and Y. Ma. "Pruning and summarizing the discovered associations." *KDD-99*, pp. 125-134, 1999.
- [11]. B. Liu, M. Hu and W. Hsu. "Multi-level organization and summarization of the discovered rules," *KDD-2000*, 2000.
- [12]. J. J Mahoney, and R. J. Mooney. "Comparing methods for refining certainty-factor rule bases," *Proceedings of ICML-94*, pp. 173-180, 1994.
- [13]. J. Major, and J. Mangano. "Selecting among rules induced from a hurricane database," *KDD-93*, pp. 28-41, 1993.
- [14]. C. J. Merz, and P. Murphy. *UCI repository of machine learning database*. [<http://www.cs.uci.edu/~mlearn/MLRepository.html>].
- [15]. R. J. Mooney. "Induction over the unexplained: using overly-general theories to

- aid concept learning,” *Machine Learning*, vol. 10, pp. 79-110, 1993.
- [16]. R. Ng, L. Lakshmanan, J. Han, and A. Pang. “Exploratory mining and pruning optimizations of constrained association rules.” *SIGMOD-98*, 1998.
 - [17]. J. Ortega and D. Fisher. “Flexibly exploiting prior knowledge in empirical learning,” *Proceedings of IJCAI-95*, pp. 1041-1047, 1995.
 - [18]. B. Padmanabhan, and A. Tuzhilin. “A belief-driven method for discovering unexpected patterns.” *KDD-98*, 1998, pp. 74-100.
 - [19]. B. Padmanabhan, and A. Tuzhilin. “Small is Beautiful: Discovering the Minimal Set of Unexpected Patterns.” *KDD-2000*, 2000.
 - [20]. M. Pazzani and D. Kibler. “The utility of knowledge in inductive learning,” *Machine learning*, vol. 9, pp. 57-94, 1992.
 - [21]. G. Piatetsky-Shapiro and C. Matheus. “The interestingness of deviations,” *KDD-94*, pp. 25-36, 1994.
 - [22]. G. Piatetsky-Shapiro, C. Matheus, P. Smyth, and R. Uthurusamy. “KDD-93: progress and challenges ...,” *AI magazine*, Fall (1994), pp. 77-87, 1994.
 - [23]. J. R. Quinlan. *C4.5: program for machine learning*. Morgan Kaufmann, 1992.
 - [24]. B. L. Richard and R. J. Mooney. “Automated refinement of first-order horn-clause domain theories,” *Machine Learning*, vol 19, no. 2, pp. 95-132, 1995.
 - [25]. A. Silberschatz & A. Tuzhilin. “What makes patterns interesting in knowledge discovery systems,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, no. 6, pp. 970-974, 1996.
 - [26]. R. Srikant, Q. Vu, Q. and R. Agrawal. “Mining association rules with item constraints.” *KDD-97*, 1997, 67-73.
 - [27]. E. Suzuki & M. Shimura. “Exceptional knowledge discovery in databases based on information theory.” *KDD-96*, pp. 275-278, 1996.
 - [28]. M. Zaki, “Generating Nonredundant Association Rules,” *Proc. ACM Int’l Conf. Knowledge Discovery & Data Mining*, ACM Press, New York, 2000, pp. 34-43.