Discovering Conforming and Unexpected Classification Rules

Bing Liu, Wynne Hsu and Shu Chen

Department of Information Systems and Computer Science
National University of Singapore
Lower Kent Ridge Road
Singapore 119260

Email: {liub, whsu, chenshu}@iscs.nus.sg

Abstract

One problem in applying machine learning and knowledge discovery techniques to solve real-world problems is how to incorporate the user's concepts about the application domain into the learning process to discover interesting rules to the user. Rules are interesting if they are useful and/or provide new knowledge to the user. Interesting rules are subjective because they depend on the individual user's existing knowledge (concepts or hypotheses) about the domain and his/her interests at a particular point in time. In the applications of classification rule induction techniques to real-world problems, we encounter a number of problems regarding the production of interesting knowledge to the user. In this paper, we address a specific problem, i.e., characterizing the conforming and unexpected tuples in the database with respect to the user's existing concepts, which can be correct, partially correct or entirely incorrect. This helps the user to find interesting rules and enables him/her to have a better understanding of the domain.

1 Introduction

The user's objective of applying a machine learning or knowledge discovery technique to his/her databases is to find *interesting* knowledge in the databases [e.g., Fayyad *et al.*, 1996; Major, and Mangano, 1993; Piatesky-Shapiro and Matheus, 1994; Silberschatz and Tuzhilin, 1996; Liu and Hsu, 1996]. However, for a machine learning system to know what knowledge is interesting is not an easy task. A piece of knowledge (or rule) can be interesting to one user but not interesting to another. Even for the same user, he/she may be interested in different things at different points in time. Thus, whether a piece of knowledge is interesting or not is subjective because it depends on the user's prior knowledge about the domain, and his/her specific interests.

In data mining, the problem of discovering interesting rules is referred to as the interestingness problem [e.g., Fayyad *et al.*, 1996; Piatesky-Shapiro and Matheus, 1994; Silberschatz and Tuzhilin, 1995; Liu and Hsu, 1996]. Past research has proposed many measures of interestingness. These measures can be divided into two categories [Silberschatz and Tuzhilin, 1995; Silberschatz and Tuzhilin, 1996], objective measures - those that depend only on the structure of a rule and the underlying data used in the discovery process, and subjective measures - those that also depend on the users who examine the rule. The focus of this paper is on the subjective aspect of interestingness. Two main subjective interestingness measures are unexpectedness [Silberschatz and Tuzhilin, 1995] and actionability [Piatesky-Shapiro and Matheus, 1994]. Rules are unexpected if they "surprise" the user, and rules are actionable if the user can do something with them to his/her advantage. These two measures are not mutually exclusive. More details on the relationships of the two measures appear in Section 2 (see also [Silberschatz and Tuzhilin, 1996]). In order to produce unexpected and/or actionable rules, the system must know what the user expects, i.e., his/her existing knowledge or concepts about the domain. Different techniques may also be needed to discover different types of interesting rules.

In machine learning, much research has been done on how to use domain knowledge or theory to help the rule generation process [e.g., Ortega and Fisher, 1995; Pazzani and Kibler, 1992]. However, the main purpose of their work is to improve the accuracy of the generated rules by the learning systems. [Clark and Matwin, 1993] also uses the qualitative model of the physical system to aid in the generation of understandable rules. However, not many domains have available qualitative models. Machine learning research also typically assumes that the domain knowledge is correct or at least partially correct. Limited work has been done on incorporating human concepts, which may be correct, partially correct or completely wrong, in a learning system to generate subjectively interesting rules, e.g., unexpected rules.

In data mining, the existing approach for dealing with the subjective interestingness problem is to employ a post-analysis module at the back-end of the knowledge discovery system [Major, and Mangano, 1993; Piatesky-Shapiro and Matheus, 1994; Liu and Hsu, 1996]. This module uses the user's input knowledge about the domain to help him/her identify interesting rules. For example, [Liu and Hsu, 1996] defines 3 types of unexpected rules in the classification rule context, namely, unexpected conclusion rules, unexpected condition rules and unexpected attributes rules. A fuzzy matching technique is proposed to identify these 3 types of unexpected rules. This technique is useful when the number of generated rules are large and it is very difficult for the human user to analyze them manually in order to identify those interesting rules. However, since it is a post-analysis technique, it does not guide the rule generation process.

In our practical applications of rule induction systems, we found that there is another type of rules that is also of interest to the user. The problem can be described as follows: When the user has some existing knowledge (concepts or hypotheses) about the domain, some tuples in the database conform to (i.e., satisfy) the existing knowledge and some tuples are unexpected (i.e., do not satisfy the existing knowledge). The user wishes to know "is it possible to generalize the conforming tuples and unexpected tuples to produce conforming rules and unexpected rules?" These rules are important because they allow the user to know precisely what is right about his/her previous concepts and what is unexpected. This type of knowledge cannot be obtained by post-analysis of the generated rules. Instead it needs the learning system to consider the user's existing concepts in the rule generation process. This research is motivated by the following situations we encountered in our real-life applications.

- There are reported rules about the domain by previously published studies. The user typically wants to verify these rules. This situation happened in all our real-life applications in medical domains. The doctors always wish to check whether the published results can apply to his/her situation, e.g., whether the published results obtained from Caucasian population apply to Asian population, and whether the results obtained from student population apply to the general public. If the existing results are not applicable, then the doctors wish to know in what situations (e.g., with extra conditions) they apply and in what situations they do not apply. In order words, they wish to characterize those conforming and unexpected tuples into rules.
- The user has some previous experiences in the domain and could not believe that his/her knowledge is wrong, or it does not make sense that his/her knowledge is wrong. Again, he/she wishes to characterize those conforming and unexpected tuples. It may turns out that his/her knowledge is correct or partially correct. In many situations, it is typically the user who unconsciously omitted some conditions in his/her expected rules, or do not know about the extra conditions. With the conforming and unexpected rules, the user will have a better understanding of the domain.
- The user notices that there are unexpected tuples in the database (which should not have happened) and wants to characterize the unexpected tuples. For example, in a bank loan domain, it is found that a particular officer approved a number of loans which should not have been approved. Then, the manager wishes to know the characteristics of these unexpected approvals.

In this paper, we propose a novel and yet simple approach to incorporate the user's existing concepts about the domain in the rule induction process to allow the learning system to produce this type of interesting rules. The essence of the proposed technique is to pre-process the database and then to use the rule induction power of C4.5 [Quinlan, 1992] to discover the interesting rules. The pre-processing consists of two steps: (1) accepting the user's description of his/her concepts, (2) categorizing the tuples in the database into a few groups, i.e., conforming tuples, unexpected tuples and unrelated tuples, based on the user's concepts.

The proposed technique has been tested with a number of databases. The results indicate that this simple approach is effective in generating interesting rules.

2 Preliminaries

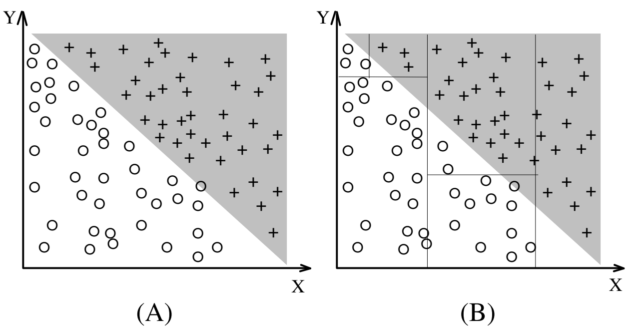
In this research, we use C4.5 [Quinlan, 1992] as our rule induction system. This section first reviews the rule generation process in C4.5, and then presents the problem we will be addressing.

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, …, C_m, ..., C_q. Each object in the database is described by n distinct attributes, $Attr_1$, ..., $Attr_l$, ..., $Attr_n$, so that in an instantiation of object description, an attribute $Attr_l$ takes on the value $a_l \in domain(Attr_l)$. 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:

If $P_1, ..., P_i, ..., P_r$ **then** C (or $P_1, ..., P_i, ..., P_r \rightarrow C$)

where "," means "and", and P_i is a condition of the form, $Attr_i\ OP\ a_i$, where $OP \in \{=, <, >, \leq, \geq\}$ is the operator. C is the conclusion with the format of $Class = C_m$.

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. It is known that the regions produced by a decision tree are all hyperrectangles [Quinlan, 1992]. 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.



(A) (B)

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

We now use an example to illustrate the problem we are attempting to solve. This example uses the credit screening database created by Chiharu Sano in UCI machine learning repository. The database has 125 tuples, 10 attributes and 2 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 →	Class = Yes
R2:	Sex = Male, YR_Work > 2 →	Class = Yes
R3:	Jobless = No, Bought = pc →	Class = Yes
R4:	Bought = medinstru, Age <= 34 →	Class = Yes
R5:	Sex = Female, Age <= 25 →	Class = No
R6:	Savings <= 7, M_LOAN > 7 →	Class = No
R7:	YR_Work <= 2 →	Class = No

Now assume the user believes that the following rule (called the user expected rule) is true from experience:

Bought = jewel, Sex = Female → Class = No.

By looking at the generated rules above, we have no idea how correct the user expected rule is, and whether there are conforming and unexpected rules that can characterize those conforming and unexpected tuples in database, because the generated rules have little relationship with the user's concept. In order words, the discovered rules do not allow the user to check his/her hypotheses, which is an important aspect of knowledge discovery.

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). Hence, there are 4 conforming tuples and 10 unexpected tuples. The question is how to characterize these conforming and unexpected tuples. This paper proposes such a technique. For the above problem, our proposed technique produces the following two related rules (i.e., Rule2 is a conforming rule and Rule1 is an unexpected rule).

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

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

After seeing these two rules, the user will know exactly what is wrong with his/her expectation.

The above problem is illustrated geometrically in Figure 2. For example, a user expected rule (e.g., its class is "o") is represented as the dark-shaded area in Figure 2. The area

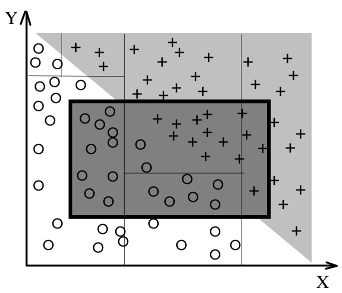


Figure 2. Geometric illustration of the problem

crosses the boundaries of the 4 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. With our proposed technique, we are able to use C4.5 to produce the conforming and unexpected rules.

This problem can be seen as a special case of the general subjective interestingness problem [Piatesky-Shapiro and Matheus, 1994; Silberschatz and Tuzhilin, 1996]. [1] Let us recall the two key measures of subjective interestingness:

1. **Unexpectedness** [Silberschatz and Tuzhilin, 1995]**:** Rules are interesting if they "surprise" the user.
2. **Actionability** [Piatesky-Shapiro and Matheus, 1994]**:** Rules are interesting if the user can do something with them to his/her advantage.

These two measures are not mutually exclusive [Silberschatz and Tuzhilin, 1996]. 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 special case of the general interestingness problem, we can handle (1) and (2) by finding unexpected rules and handle (3) by finding the rules that conform to the user's expectations.

3 The Proposed Algorithm

The key idea of the proposed technique is to use the user's concepts to help C4.5 discover interesting rules. User's concepts are expressed as a set of expected rules, which are in the same format as the generated rules. Using this representation is natural because when the user is looking for a particular type of rules (e.g., classification rules), his/her expectations are usually also 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 as a fuzzy rule. The fuzzy rule has the same syntax as the generated rule, but its attribute values are described using fuzzy sets (see Section 3.1). Using fuzzy sets in the representation of expected rules conforms to human intuition because human knowledge or concepts are normally fuzzy.

Step 2. Pre-process the database D to produce D'. Basically, 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 (see Section 3.2).

[1] [Liu and Hsu, 1996] shows another special case, where the user is not interested in checking his/her hypotheses, but only in identifying the interesting rules from the set of discovered rules.

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 (see Section 3.2.3)

3.1 Specifying expected rules

Now, we describe how a user can specify his/her existing concepts in terms of expected rules. Since fuzzy set theory is used in the specification, we first give a brief overview of the fuzzy set theory.

A fuzzy set [Kandel, 1986] is a class that admits the possibility of *partial membership* in it. Let $X = \{x\}$ denote a space of objects. Then a fuzzy set A in X is a set of ordered pairs: $A = \{(x, \mu_A(x)) \mid x \in X\}$, where $\mu_A(x)$ is termed "the degree of membership of x in A." $\mu_A(x)$ is a number in the interval [0, 1], with the degrees 0 and 1 representing, respectively, non-membership and full membership in a fuzzy set.

In our application, let X be the set of possible values for an attribute. To describe a fuzzy concept A of the attribute, the user needs to input the membership value for each $x \in X$ in the fuzzy concept A, denoted as $\mu_A(x)$. For a discrete value attribute (which can be nominal or ordinal), the specification of the membership values is straightforward. For example, the user gives the following expected rule:

If *ExamScore* = *bad* **then** *Class* = *reject*

Here, *bad* is a fuzzy set representing a user concept. Assume the set of possible values for the discrete attribute *ExamScore* is {*A*, *B*, *C*, *D*, *F*}. The user may specify that *bad* exam score means: {(*A*, 0), (*B*, 0), (*C*, 0.2), (*D*, 1), (*F*, 1)}. The fuzzy set *reject* can also be similarly specified by the user.

For a continuous value attribute, $\mu_A(x)$ takes on the form of a continuous function. To simplify the user's task of specifying the shape of this function, we assume the function has a curve of the form shown in Figure 3. Thus, the user merely needs to provide the values for *a*, *b*, *c* and *d*.

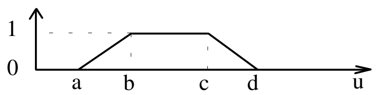


Figure 3. Membership function

For example, consider the user expected rule:

If *Temperature* = *moderate* **then** *Class* = *yes*.

Suppose *Temperature* takes continuous values from 0 to 40. The *moderate* fuzzy concept may be described as: a = 18, b = 20, c = 25, and d = 30.

3.2 Pre-processing the database and running C4.5

3.2.1 Correctness of an expected rule

Naturally, when the user provides an expected rule $E_j \in E$, he/she wishes to know how true E_j is. To obtain this correctness value, the system needs to check E_j against each tuple $D_k \in D$. Since E_j is regarded as a fuzzy rule, deciding whether D_k satisfies E_j is also fuzzy. It returns a value in the range [0, 1]. Thus, to consider a value to be satisfactory (or acceptable), a cutoff value needs to be used.

Let *cutoff* be a value in the range (0, 1] denoting the minimal degree that a data tuple D_k must satisfy the conditional or conclusion part of E_j. The value of *cutoff* is specified by the user. Let r be the number of attributes mentioned in the conditional part of E_j. We denote $V_{f,k,j}$ as the degree that D_k satisfies the fth conditional proposition of E_j, and $Z_{k,j}$ as the degree that D_k satisfies the conclusion of E_j. The computation of $V_{f,k,j}$ and $Z_{k,j}$ is discussed below. We then define

$$M_{cond}^{k,j} = \min(V_{1,k,j}, V_{2,k,j}, \ldots, V_{r,k,j})$$

to be the degree that D_k satisfies the conditional part of E_j. Similarly, we define $M_{concl}^{k,j}$ ($= Z_{k,j}$) to be the degree that D_k satisfies the conclusion part of E_j. The *correctness* of E_j, denoted as $Corr_j$, is defined as:

$$Corr_j = \frac{\text{Total number of tuples with } M_{cond}^{k,j} \geq cutoff \text{ and } M_{concl}^{k,j} \geq cutoff}{\text{Total number of tuples with } M_{cond}^{k,j} \geq cutoff}$$

For the computation of $V_{f,k,j}$ (or $Z_{k,j}$), we need to consider both the attribute value and the operator used in the conditional (or conclusion) proposition. In addition, the attribute value type (discrete or continuous) is also important. Since the computations of $V_{f,k,j}$ and $Z_{k,j}$ are the same, it suffices to just consider $V_{f,k,j}$.

First we consider the case that the attribute takes discrete values. In this discrete case, the valid operators are "=" and "≠". Suppose the condition to be matched in E_j is:

attr Opu A

where *attr* is an attribute name, $Opu \in \{=, \neq\}$, and A is the user's fuzzy value for *attr*. Assume S is the value of *attr* in the data tuple D_k. Two cases result:

Case 1. *Opu* = "=": $V_{f,k,j} = \mu_A(S)$.

Case 2. *Opu* = "≠": $V_{f,k,j} = \mu_{\neg A}(S)$.

When an attribute takes continuous values, the set of valid operators is expanded to {=, ≠, ≥, ≤, ≤≤}. "≤≤" is used to represent: $A_1 \leq attr \leq A_2$. ">" and "<" are not included because they can always be expressed with "≥" and "≤" respectively. With this expansion, the total number of possible cases to be considered is 5. The computation of $V_{f,k,j}$ is listed below:

Case 1. *Opu* = "=": $V_{f,k,j} = \mu_A(S)$.

Case 2. *Opu* = "≠": $V_{f,k,j} = \mu_{\neg A}(S)$.

Case 3. *Opu* = "≥": $V_{f,k,j} = \begin{cases} \mu_A(S) & x \leq b \\ 1 & Otherwise \end{cases}$

Figure 4 shows the meanings of the symbols used in the formula. Note that *min_val* and *max_val* are the minimal and maximal values of the attribute (represented as x) respectively.

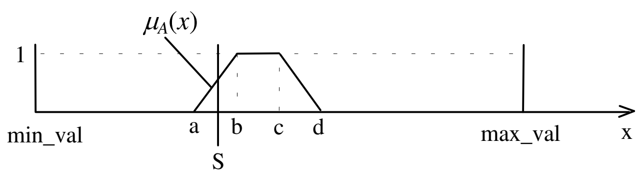


Figure 4. Computing $V_{f,k,j}$ when *Opu* = "≥" or *Opu* = "≤"

Case 6. Opu = "≤": $V_{f,k,j} = \begin{cases} \mu_A(S) & x \ge c \\ 1 & Otherwise \end{cases}$.

Case 7. Opu = "≤≤": $V_{f,k,j} = \begin{cases} \mu_{A2}(S) & x \ge c \\ \mu_{A1}(S) & x \le b \\ 1 & otherwise \end{cases}$.

Figure 5 shows the meanings of the symbols used in the above formula.

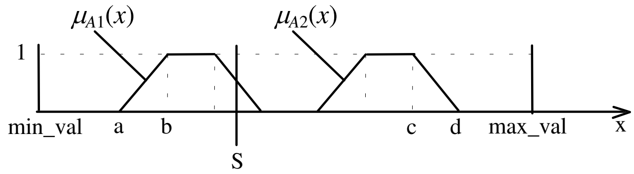


Figure 5. Computing $V_{f,k,j}$ when Opu = "≤≤"

3.2.2 Pre-processing the database

During the pre-processing step, we introduce additional classes. A matching process is carried out on 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 $M_{cond}^{k,j} \ge Cutoff$ and $M_{concl}^{k,j} \ge Cutoff$.

Definition: D_k is *unexpected* with respect to $E_j \in E$ if $M_{cond}^{k,j} \ge Cutoff$ and $M_{concl}^{k,j} < Cutoff$.

Definition: D_k is *unrelated* to $E_j \in E$ if $M_{cond}^{k,j} < Cutoff$.

Definition: D_k *conforms* to E if $\exists E_j \in E$ such that D_k conforms to E_j.

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

Definition: D_k is *unrelated* to E if $\forall E_j \in E$ such that 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 (Figure 6).

1 Initialize $RuleNo_j$ and $CondNo_j$ to 0, $1 \le j \le |E|$;
2 **for** each tuple $D_k \in D$ **do**
3 *Confm* ← FALSE;
Unexp←FALSE;
Class ← the class of D_k;
4 **for** each $E_j \in E$ **do**
5 Compute $M_{cond}^{k,j}$, and $M_{concl}^{k,j}$;
6 **if** $M_{cond}^{k,j} \ge cutoff$ **then**
7 Increment $CondNo_j$
8 **If** $M_{concl}^{k,j} \ge cutoff$ **then**
9 Increment $RuleNo_j$;
10 *Confm* ← TRUE
11 **else** *Unexp* ← TRUE
12 **endif**;
13 **endif**;
14 **endfor**;
15 **if** *Confm* **then** Change the class of D_k to <*Class*>*CONFORM*;
16 **elseif** *Unexp* **then** Change the class of D_k to <*Class*>*UNEXPECTED*;
17 **endif**;
18 **endfor**;
19 **for** each $E_j \in E$ **do** $Corr_j = \frac{RuleNo_j}{CondNo_j}$ **endfor**;

Figure 6. The algorithm for step 2

Notes about the algorithm:

- Lines 1 and 3 are the initialization.
- Line 5-9 compute the degrees that D_k satisfies the conditional and conclusion parts of E_j, and prepare the values $RuleNo_j$ and $CondNo_j$ for the computation of the correctness of E_j in Line 19.
- Line 10 indicates that D_k satisfies E_j. We say that D_k conforms to E_j. Line 11 indicates that D_k satisfies only the conditional part of E_j, but not the conclusion. We say that D_k is unexpected with respected to E_j.
- Lines 15 and 16 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 ($j \ne m$). 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 3.3 below). Note that the new classes introduced are: <*Class*>*CONFORM* and <*Class*>*UNEXPECTED*. The interpretation is as follows: <*Class*> 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*.
- Lines 5-13 takes O(1) computational time (since the number of conditions in each E_j is normally small and does not vary a great deal). Thus, the complexity of the algorithm is $O(|D||E|)$, where $|D|$ and $|E|$ are the number of tuples in D and the number of rules in E respectively.

3.2.3 Running C4.5 with the modified database

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. Conforming rules can be those rules that are actionable but expected. Unexpected rules represent "surprises" to the user. Unrelated rules represent knowledge that are not related to the user's expectations, which may also be interesting because they are unknown to the user.

3.3 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 <*Class*>*CONFORM*, <*Class*>*UNEXPECTED* and the original classes (used by unrelated tuples) thus resulting in the conforming, unexpected and unrelated rules. To illustrate, let us use the example in Figure 2 (reproduced as Figure 7(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 7(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 7(B). Then, region 1 and 3 represent 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. The proposed technique is flexible as it can bias C4.5 in many ways. For example, if the user is not interested in conforming rules but only unexpected rules, the situation in Figure 7(C) is produced. In this case, there is only one unexpected rule (region 1).

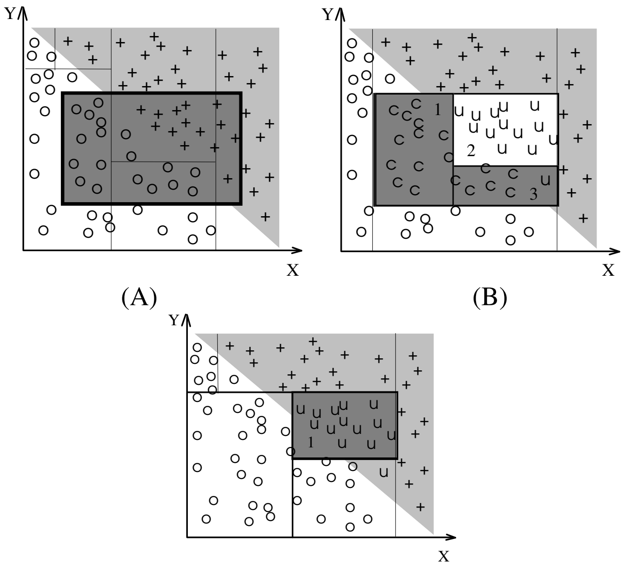


Figure 7. The new divisions because of the user's rule

4 An Example

We have tested the system using a number of public domain databases. The system has also been applied to 3 real-life databases in medical domains. The 3 real-life databases involved have 713, 2200, and 11,312 tuples respectively. A test run of our system is provided to illustrate the use of the technique.

4.1 An example run

The real-life disease database used in this example has 713 tuples, 9 attributes, and 2 classes, *YES* and *NO*, representing whether the person has the disease. Due to confidentiality of the data, we could not provide more details about the data. Without considering any user expected rule, the set of rules generated by C4.5 is as follow (the value in [ ] after each rule is the predicted accuracy of the rule produced by C4.5):

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

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

Age >= mid_age	{a = 40, b = 45, c = 50, d = 55}
SBP >= high	{a = 145, b = 150, c = 160, d = 180}
-> Class = disease	{(YES, 1), (NO, 0)}

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 because the generated rules bear little relationship to the user's concept. However, if we look at the rules produced using the proposed technique below, these questions can all be answered easily. The running results using the proposed technique are shown below:

- **Correctness of the expected rule:**

  Correctness (*cutoff* value is 0.6): 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 unrelated rules are not listed as they are not our fo-

cus). The total number of rules produced is 15.

R1. Age > 61, LDL > 3.42, SBP > 147
$\rightarrow$ Class = YESConform [82.0%]

R2. LDL > 3.42, SBP > 168, GLUC <= 5.28
$\rightarrow$ Class = YESConform [79.4%]

R3. Age > 42, Age <= 61, LDL > 3.7, SBP > 148, SBP<=168 $\rightarrow$ Class = NOUnexpected [79.4%]

R4. Age > 42, Age <= 61, SBP > 147, GLUC > 5.28
$\rightarrow$ Class = NOUnexpected [66.2%]

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

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

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.

We now make some observations about the 4 interesting rules:

- R3 and R4 show the situations where the expected conditions can lead to unexpected conclusion. The rules are quite accurate. For example, in R3, a person that satisfies *SBP >= high* 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.
- R1 and R2 show the conforming situations, but with more restrictive conditions. For example, in R1, 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 would have a clear picture about the disease with respect to his/her expectation.

4.2 Discussions

Let us make some observations about the proposed technique and its use.

- A question that one may ask is "is it possible to achieve the same results by only passing those conforming and unexpected tuples to C4.5 without the rest of the tuples?" The answer is no because the rules produced this way will not be able to discriminate the conforming (or unexpected) tuples from the rest of the tuples.
- 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 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 [Quinlan, 1992] 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

Although many machine learning systems [e.g., Ortega and Fisher, 1995; Pazzani and Kibler, 1992] can use existing domain knowledge or theory in the learning process, their purpose is mainly for producing more accurate rules. Limited work has been done on incorporating human concepts in the learning process in order to discover subjectively interesting rules, e.g., conforming and unexpected rules.

In the field of data mining, a number of systems have been built which can help the user to identify interesting rules. Most of them use post-analysis modules or interestingness filters [e.g., Major, and Mangano, 1993; Piatesky-Shapiro and Matheus, 1994; Liu and Hsu, 1996] to help the user filter out uninteresting rules from a set of discovered ones. However, post-analysis cannot produce the types of interesting rules studied in this paper.

[Piatesky-Shapiro and Matheus, 1994] studies the issue of subjective interestingness in the context of a healthcare application. The system (called KEFIR) analyzes the health care information to uncover interesting deviations (from the norms). A domain expert system is built as a post-analysis module to identify findings that are actionable and the actions to be taken.

[Liu and Hsu, 1996] reports a post-analysis technique based on fuzzy matching to help the user identify certain types of unexpected rules (i.e., unexpected conclusion rules, unexpected condition rules and unexpected attribute rules) from the set of discovered rules.

[Silberschatz and Tuzhilin, 1995] proposes to use belief systems for defining unexpectedness. It also suggests to employ an interestingness engine in the knowledge discovery system to discover interesting rules in the first place rather than going through post-analysis. However, [Silberschatz and Tuzhilin, 1996] is mainly a proposal, and no detailed technique is designed.

6 Conclusion

In this paper, we proposed a novel yet simple approach to incorporate user's concepts in the knowledge discovery process in order to discover subjectively interesting rules. The approach is found to be useful in real-life applications.

This research also raises an important question in machine learning, e.g., how to perform subjective learning. We believe that the human learning experience is essentially subjective. For example, given the same situation (or data) to different people, they typically learn different things because they have different background knowledge, different hypotheses and different personal interests. This paper touches on a specific aspect of subjective learning. In our future work, we will explore this further.

Acknowledgments

We are grateful to Dr. Hing-Yan Lee, Ms. Hwee-Leng Ong and Ms. Angline Pang from Information Tehcnology Institute, and Dr. King-Hee Ho and Dr. Poh-San Lai from National University Hospital for many useful discussions, for providing us the databases, and for their help in the testing of our system.

References

[Agrawal *et al.,* 1993] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases, *SIGMOD-1993*, pages 207-216, 1993.

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

[Fayyad *et al*., 1996] U. Fayyad, G. Piatesky-Shapiro and P. Smyth. Knowledge discovery and data mining: towards a unifying framework, *KDD-96*, pages 82-88, 1996.

[Han *et al*., 1993] J. Han, Y. Cai, and N. Cercone. Data driven discovery of quantitative rules in relational database. *IEEE Trans. Knowl. and Data Eng.*, 5:29-40, 1993.

[Kamber, and Shinghal, 1996] M. Kamber, and R. Shinghal, Evaluating the interestingness of characteristic rules. *KDD-96*, 1996.

[Kandel, 1986] A. Kandel. *Fuzzy Mathematical Techniques with Applications*. Addison-Wesley, 1986.

[Klemetinen *et al*., 1994] 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,* pages 401-407, 1994.

[Liu and Hsu, 1996] B. Liu and W. Hsu. Post-analysis of learned rules. *AAAI-96*, pages 828-834, 1996.

[Liu *et al*., 1997] B. Liu, L-P. Ku and W. Hsu. Discovering Interesting Holes in Data. To appear in *IJCAI-97*, 1997.

[Major, and Mangano, 1993] J. Major, and J. Mangano. Selecting among rules induced from a hurricane database. *KDD-93*, 1993.

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

[Ortega and Fisher, 1995] J. Ortega and D. Fisher. Flexibly exploiting prior knowledge in empirical learning. *IJCAI-95*, pages 1041-1047, 1995.

[Pazzani and Kibler, 1992] M. Pazzani and D.Kibler. The utility of knowledge in inductive learning. *Machine learning,* 9:57-94, 1992

[Piatesky-Shapiro and Matheus, 1994] G. Piatesky-Shapiro and C. Matheus. The interestingness of deviations. *KDD-94*, pages 25-36, 1994.

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

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

[Silberschatz and Tuzhilin, 1995] A. Silberschatz and A. Tuzhilin. “On subjective measures of interestingness in knowledge discovery. *KDD-95*, pages 275-281, 1995.

[Silberschatz and Tuzhilin, 1996] A. Silberschatz & A. Tuzhilin, “What makes patterns interesting in knowledge discovery systems,” *IEEE Transactions on Knowledge and Data Engineering*, 8(6):970-974, 1996.