Using General Impressions to Analyze Discovered 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
{liub, whsu, chenshu}@iscs.nus.edu.sg

Abstract

One of the important problems in data mining is the evaluation of subjective interestingness of the discovered rules. Past research has found that in many real-life applications it is easy to generate a large number of rules from the database, but most of the rules are not useful or interesting to the user. Due to the large number of rules, it is difficult for the user to analyze them manually in order to identify those interesting ones. Whether a rule is of interest to a user depends on his/her existing knowledge of the domain, and his/her interests. In this paper, we propose a technique that analyzes the discovered rules against a specific type of existing knowledge, which we call *general impressions*, to help the user identify interesting rules. We first propose a representation language to allow general *impressions* to be specified. We then present some algorithms to analyze the discovered classification rules against a set of general impressions. The results of the analysis tell us which rules conform to the general impressions and which rules are unexpected. Unexpected rules are by definition interesting.

1. Introduction

The aim of data mining is to discover *useful* or *interesting* rules (Fayyad, Piatesky-Shapiro, and Smyth 1996) for the user. However, past applications have found that it is easy to generate a large number of rules from a database, and most of them are not useful to the user (e.g., Piatesky-Shapiro and Matheus 1994a; Piatesky-Shapiro et al. 1994b; Silberschatz and Tuzhilin 1996; Liu and Hsu 1996). The presence of the huge number of rules makes it difficult for the user to analyze them and to identify those that are of interest to him/her. Some automated assistance is needed.

Identifying interesting rules from a set of discovered rules is not a simple task because a rule could be interesting to one user but not interesting to another. The interestingness of a rule is essentially subjective (e.g., Piatesky-Shapiro et al. 1994b; Klemetinen et al. 1994; Silberschatz and Tuzhilin 1996; Liu and Hsu 1996) because it depends on the user's existing concepts about the domain, and his/her interests. There is also an *objective* aspect of interestingness, which is not studied here. Interested readers, please refer to (Major and Mangano 1993; Silberschatz and Tuzhilin 1996). Two main measures of subjective interestingness are *unexpectedness* (Silberschatz and Tuzhilin 1996) and *actionability* (Piatesky-Shapiro and Matheus 1994a). They are defined as follows:

- **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.

These two measures are not mutually exclusive (Silberschatz and Tuzhilin 1996). Thus, subjectively interesting rules can be classified into three categories: (1) rules that are both unexpected and actionable; (2) rules that are unexpected but not actionable, and (3) rules that are actionable but expected. (1) and (2) can be handled by finding unexpected rules, and (3) can be handled by finding the rules that conform to the user's existing concepts.

In (Liu and Hsu 1996), a fuzzy matching approach is reported to analyze the discovered rules against the user's existing concepts. The existing concepts are expressed as a set of expected fuzzy rules. A fuzzy matching algorithm is then used to compare the discovered rules against the expected rules to help the user identify those interesting rules. One limitation of this technique is that too much reliance is being placed on the user's ability to supply the set of fuzzy expectations. In many situations, users do not know enough about their domains to supply the expected rules. Instead, we find that even if the users cannot supply the set of fuzzy expectations, they do have certain *general impressions* (*GI*) about their domains. This paper first proposes a specification language for the user to specify his/her *GIs* and then presents two matching algorithms to analyze the discovered rules against the *GIs* (which can be correct, partially correct or completely wrong). Through this analysis, the user is able to find the interesting rules easily.

2. Preliminaries

Assume a human user has some previous concepts about the domain represented by the database *D*. These concepts can be correct, partially correct or entirely wrong. In this work, we distinguish two types of existing concepts:

***General impressions* (*GI*)**: The user does not have detailed concepts about the domain, but does have some vague feelings. For example, in a housing loan domain, the user may feel that having a high monthly salary increases one's chance of obtaining a loan.

***Reasonably precise knowledge* (*RPK*)**: The user has more definite idea. For example, in the same loan domain, the user may believe that if one's monthly salary is over \$5,000, one will be granted a loan. Of course, the user may not be so sure that it is exactly \$5,000. There is a fuzziness surrounding the value \$5000 in his/her mind.

(Liu and Hsu 1996) studied the rule analysis against *RPK*. This paper focuses on *GIs*. In the situation where one has some *RPK* about certain aspects of the domain, but only *GIs* about the others, a combined approach may be used.

In this paper, we analyze classification rules produced by C4.5 (Quinlan 1992), which have the form:

$$at_1\ OP_1\ v_1,\ \ldots,\ at_n\ OP_n\ v_n \rightarrow Class,$$

where at_i is an attribute in D, v_i is a possible value of at_i, $OP \in \{=, <, >, \leq, \geq\}$, and $Class$ is a class value in D.

3. Representing General Impressions

We now present the specification language that allows a user to express his/her general impressions (*GIs*) about a domain. This specification focuses on representing the general impressions related to classification rules. Two types of *GIs* are defined: Type_1 *GI*s and Type_2 *GI*s.

Let $A = \{A_1, ..., A_s\}$ be the set of attributes in D, and $C = \{C_1, ..., C_m\}$ be the set of possible classes in D.

Definition 1: An *impression term* (*IPT*) is of the form: a ID, where $a \in A$, and $ID \in \{<, >, <<, |, [\text{a set}]\}$ is an *impression descriptor*. [a set] represents a subset of values of a discrete (and/or nominal) attribute.

Definition 2: A Type_1 *GI* (*T1*) is of the following form,

$$a_1\ ID_1,\ \ldots,\ a_w\ ID_w \rightarrow C_j \quad (\text{or } IPT_1,\ \ldots,\ IPT_w \rightarrow C_j),$$

where $I = \{a_1, \ldots, a_w\} \subseteq A$, $I \neq \emptyset$, and $a_p \neq a_q$ if $p \neq q$.

The meanings of *T1*s can be illustrated as follows:

- $a < \rightarrow C_j$: This represents the impression that a smaller value of a will result in a higher likelihood of being in class C_j. It can be used to specify *T1*s such as "the smaller the period of loan repayment is, the more likely will the loan application be approved."
- $a > \rightarrow C_j$: This represents the impression that a larger value of a will result in a higher chance of leading to class C_j. For example, "the more savings there are, the more likely will the loan application be approved."
- $a << \rightarrow C_j$: This represents the impression that if the value of a is within some range, then class C_j is likely to be the result. For example, "if one is neither too young nor too old, then the loan application is likely to be approved."
- $a \mid \rightarrow C_{sub}$: This represents the impression that there exist some relationships between the attribute a and the classes in C_{sub} ($\subseteq C$). However, the exact relationships are not known. $a \mid \rightarrow C_{sub}$ is a short form for $a \mid \rightarrow c_1$, ..., $a \mid \rightarrow c_f$, where $C_{sub} = \{c_1, \ldots, c_f\}$. For example, "we know that one's number of years of work plays a part in determining if a loan will be approved, but we do not know how."
- $a\ [S] \rightarrow C_j$: This represents the impression that if the value of a is an element in the set S, it is more likely to lead to class C_j.

For convenience, we organize the *T1*s into L levels such that each *T1* at level z has z *IPTs*.

- Each level-1 *T1* expresses the user's belief on how a single attribute may affect a class. For example, in a loan application domain, we may have the following four *T1*s at level 1:
  (1) *saving* $> \rightarrow$ *approved*,
  (2) *age* $\mid \rightarrow$ {*approved*, *not_approved*},
  (3) *jobless* {*no*} $\rightarrow$ *approved*,
  (4) *jobless* {*yes*} $\rightarrow$ *not_approved*.

  Level-2 and above *T1*s express the user's beliefs on how combinations of *IPTs* may affect the classes. Examples of level-2 *T1*s are:
  (5) *saving* $>$, *age* $<< \rightarrow$ *approved*,
  (6) *saving* $>$, *jobless* {*yes*} $\rightarrow$ *approved*.

  Note that (6) says that if one has sufficient saving, even if he/she does not have a job, he/she may still be granted a loan. This is possible because although "*jobless* {*yes*}" is not favorable for one to obtain a loan (refer to (4)), in the special circumstance where one has a large amount of saving, one may still be granted a loan. It is also possible that a combination of *IPT*s may act antagonistically to lead to an unexpected class. For example, IPT_1 and IPT_2 both lead to class *Yes* individually but their combination leads to class *No*. In such situations, we say that the combined *T1* *shadows* the lower level *T1*s involving IPT_1 and IPT_2. Such exceptions can be expressed using level-2 and above *T1*s.
- Type_1 *GI*s make the following assumption: If two or more *T1*s lead to the same class and they have no common attributes, then their combinations (except those *T1* combinations that are shadowed by some higher level *T1*s) also lead to the same class. For example, with the above 6 *T1*s, it is assumed that the following,

  saving $>$, *age* $<<$, *jobless* {*no*} $\rightarrow$ *approved*,

  also holds. Hence, there is no need to specify the above as a level-3 *T1*. Any impression that cannot be composed with a proper combination of *T1*s are considered unexpected. We say that a *T1* is a *minimal impression*. The assumption is justified because it conforms to human intuitions, and those exceptional cases can be expressed with higher level *T1*s.

Note that we have not considered the impression whose left-hand side is empty, i.e., $\rightarrow C_j$. This impression means that the user believes that there is only one valid class C_j for D. This case is simple and we will not study it in the paper.

In the case where the user has more definite idea that a specific combination of *IPT*s is sufficient to lead to some class, then a Type_2 *GI* may be used.

Definition 3: A Type_2 *GI* (*T2*) is of the following form, which has two parts separated by a "&":

$$a_1\ ID_1,\ \ldots,\ a_k\ ID_k\ \ \&\ \ a_{k+1}\ ID_{k+1},\ ...,\ a_w\ ID_w \rightarrow C_j,$$

where

(1) $I = \{a_1, \ldots, a_w\} \subseteq A$, $I \neq \emptyset$, and $a_p \neq a_q$ if $p \neq q$.
(2) The first part ($a_1\ ID_1,\ \ldots,\ a_k\ ID_k$) is called the *core* and must be non-empty [1], and the second part is

[1] If there is no core, it should be specified as a Type_1 *GI*.

called the *supplement*. Let F_1 ($\neq \varnothing$) denote the core and F_2 denote the supplement. We have:

i) If $F_2 \neq \varnothing$, we call such a *T2* a *maximal impression*. It means that the user believes that (a_1 ID_1, …, a_k ID_k) and any subset of {(a_{k+1} ID_{k+1}), ..., (a_w ID_w)} may lead to C_j. Anything more than the F_1 and F_2 combination is unexpected.

ii) If $F_2 = \varnothing$, we call such a *T2* an *exact impression*. It means that the user believes that (a_1 ID_1, …, a_k ID_k) should lead to C_j. Anything more or less is unexpected.

We now present the technique for analyzing a set of discovered rules against *T1*s and *T2*s.

4. Matching and Ranking Discovered Rules Against *GIs*

We first give a high-level view of the proposed technique before presenting the computational details. This technique consists of two main steps:

1. The user specifies all the general impressions (both *T1*s and *T2*s) that he/she has about the domain using the above specification language.
2. The system analyzes the discovered rules by matching them against the *T1*s and *T2*s in various ways for finding different types of interesting rules. The discovered rules are then ranked according to the matching results. With these rankings, identification of the interesting rules becomes simple.

Note that this paper does not address issues such as whether the discovered rules are consistent, whether subsumption relations exist among them, etc. For such issues, see (Major and Mangano 1993). This paper assumes that such analyses have been done on the discovered rules and the *GIs* before they are analyzed by our matching algorithms.

4.1 Matching and ranking discovered rules against Type_1 *GI*s

Let X be the set of Type_1 *GI*s (*T1*s) and R be the set of discovered rules. Our goal is to rank the rules in R in a number of ways such that three types of interesting rules can be identified.

1. **Conforming rules**: Both the conditions and the class of $R_i \in R$ match a subset of *T1*s in X.

   To rank the rules in R according to their conformity to *T1*s in X, we need to compare each $R_i \in R$ with the subset of *T1*s that lead to the same class as R_i. Let K be this subset. Two cases can arise during the comparison:

   (1). All the attributes used in R_i appear in X, i.e., no unanticipated attributes. Unanticipated attributes mean that the user did not know that these attributes are relevant to the classification.

   (2). A subset of attributes in R_i do not appear in X, i.e., $\exists$ some unanticipated attributes.

   In (2), those conditions in R_i whose attributes do not appear in X are first removed. The resulting rule is then handled in the same way as in (1). Note that the rules falling in (1) and (2) are ranked separately. For all the rules falling in (1), there is only one ranking (according to the degree of conformity). For rules falling in (2), we impose a two-level ranking system. The rules are first partitioned according to the number of unanticipated attributes. Then, within each partition, the rules are ranked as in (1). This two-level ranking system for (2) also applies to the remaining two types of analysis.

   We denote the number of attributes used in the conditional part of R_i as N (assume all unanticipated attributes in R_i, if any, have been removed). The degree of conforming match is defined as follows:

$$T1Cfm_i = T1match(R_i, 1, K) / N.$$

   This formula uses the algorithm *T1match*(R_i, *FirstT1*, K) below, which matches R_i against K. *FirstT1* is the index of the starting *T1* in K. Since *T1*s are minimal impressions, the algorithm basically tries to find a subset of *T1*s (which have no common attributes) in K, whose combined set of attributes is the same as that in R_i, that gives the best match value. The rankings of the rules in R are done according to those two cases ((1) and (2)) and by sorting them according to their $T1Cfm_i$ values in a decreasing order.

Some notes about the algorithm:

- The computation of this algorithm depends on the number of *T1* combinations in K that have the same set of attributes as R_i. In the worse case, it is the number of possible partitions of the set of attributes in R_i (Roberts 1984). However, the number of *T1*s specified by the user is typically small, and hence, the number of possible partitions that can be formed is also small. Thus, computational cost is low.

```
Algorithm T1match(Ri, FirstT1, K)
    end := FALSE;    nextT1 := FirstT1;
    cval := 0;       maxval := 0;
    while (end = FALSE) do
        nextT1 := find the next T1 in K whose set of
                  attributes is a subset of that in Ri and
                  return its index;
                   /* return 0 if nothing is found */
        if nextT1 = 0 then  end := TRUE
        else split Ri into two parts, subR1 and subR2,
                where subR2 contains the part of Ri
                whose attributes are not in K[nextT1];
            subCval := T1match(subR2, nextT1+1, K);
            cval := subCval + match(subR1, K[nextT1]);
            if cval > maxval then maxval := cval endif
        endif
        Increment nextT1
    endwhile
    return(maxval)
```

- The above algorithm uses the procedure *match*, which is specified as follows:

$$match(rule, T1) = \sum_{1}^{n} Smatch(P_j, IPT_j)$$

  where P_j is a condition of *rule* and is of the form: at_j OP_j v_j; n is the number of conditions in *Rule*; IPT_j (from *T1*) is of the form: a_j ID_j; and $a_j = at_j$. *Smatch* is defined as:

```
procedure Smatch((a OP v), (a ID))
  if OP = "=" then
    if v ∈ ID then M := 1 else M := 0 endif
  elseif (ID = "<<" and (OP = "<" or OP = ">"))
      or ((ID ="<" or ID =">") and OP = "<<") then
    M := 0.5     /* "≥" and "≤" for OP are considered
                    the same as ">" and "<" respectively */
  elseif ID ≠ "|" then
     if OP = ID then M := 1 else M := 0 endif
  else M := 0.2
  endif
  return(M)
```

Let us have an example of a conforming match. Assume we have the discovered rule,

jobless = *no*, *saving* > 10 → *approved*,

and the 6 *T1*s in Section 3. Of the 6 *T1*s, only (1), (2), (3), (5) and (6) have the class *approved*. Our rule matches (6) partially ("*saving* > 10" matches "*saving* >" but "*jobless* = *no*" does not match "*jobless* {yes}"). At the same time, it matches completely the combined *T1* formed using (1) and (3) ("*saving* > 10" matches "*saving* >" in (1), "*jobless* = *no*" matches "*jobless* {no}" in (3)). Since the algorithm looks for the best match, the rule's conforming match value is 1 (a complete match).

2. **Unexpected conclusion rules**: The conditions of $R_i \in R$ match the conditions of a subset of *T1*s in *X*, but not its class.

   Since in this case we look for rules in *R* with unexpected conclusions, then

   $K = \{x \in X \mid x$ has a different class from $R_i\}$.

   The degree of unexpected conclusion match is computed with:

   $T1UexClu_i = T1match(R_i, 1, K) / N$.

   Notice that *T1match* does not consider *shadowing*. The reason is that in the event of a partial match, it is difficult for the system to know whether shadowing has occurred. Instead, we display both the degrees of conforming match and unexpected conclusion match of each rule in the ranking to alert the user (see the example in Section 5).

   Based on the context of the previous example, an unexpected conclusion rule is,

   jobless = *no*, *saving* > 10 → *not_approved*,

   because "*saving* >" in (1), "*jobless* {no}" in (3) both lead to class *approved*, whereas the rule's class is *not_approved*.

3. **Unexpected condition rules**: It is not known from *T1*s in *X* that the conditions of $R_i \in R$ will lead to its class.

   This is the opposite to finding conforming rules. Its rankings are the reverse of the conforming rule rankings.

   Again, refer to the context of the previous example, an unexpected condition rule is,

   age < 20, *saving* < 10 → *approved*,

   because we have no *T1* relating "*age* <" and "*saving* <" to class *approved*. Hence, there is no match for the conditional part, and we say that the rule is an unexpected condition rule.

4.2 Matching and ranking discovered rules against Type_2 *GI*s

Let *Y* be the set of Type_2 *GI*s (*T2*s). Following Section 4.1, we present the rankings for finding different kinds of interesting rules. However, the rankings here are performed with respect to each *T2* in *Y*, which are different from the rankings discussed for Type_1 *GI*s. Of course, the types of rankings discussed in Section 4.1 can also be carried out here. But they are less informative.

1. **Conforming rules with respect to a *T2* ∈ *Y***: Both the conditions and the class of $R_i \in R$ match those of the *T2*.

   The degree of conforming match is computed using the *T2Match* algorithm below. It takes in three parameters, F_1, F_2 (which make up the *T2*, see below) and *Q* (a set of discovered rules to be ranked) and produces a set of values, $T2M_i$.

   For conforming match, $Q = \{r \in R \mid r$ has the same class as the $T2\}$. Those discovered rules not in *Q* will not be considered. The conforming match value is $T2M_i$. The ranking of rules in *Q* is done by sorting them according to their $T2M_i$ values in a decreasing order.

```
1 Algorithm T2Match(F1, F2, Q)
2   for each Qi ∈ Q do
3       M := coreMatch(Hi, F1);
4       if M > 0 then
5           T2Mi := (M + supMatch(Hi, F2)) / max((|F1|+|F2|), |Hi|)
6       endif
7   endfor
```

   Notes about the algorithm:

   - We first explain the terms used in the algorithm. *T2* (∈ *Y*) is of the form:

     $a_1\ ID_1, \ldots, a_k\ ID_k\ \ \&\ \ a_{k+1}\ ID_{k+1}, \ldots, a_w\ ID_w \rightarrow C_j$.

     F_1 denotes the core and F_2 denotes the supplement. A rule $Q_i \in Q$ is of the form:

     $at_1\ OP_1\ v_1, \ldots, at_n\ OP_n\ v_n \rightarrow Class$.

     H_i denotes the set of conditions of Q_i, $\{(at_1\ OP_1\ v_1), \ldots, (at_n\ OP_n\ v_n)\}$.
   - Some Q_is do not have $T2M_i$ values (line 4), and they are not ranked. When $T2M_i$ has a value, it indicates that Q_i satisfies (as defined in *coreMatch* below) the core (F_1) of *T2*. This means that the set of attributes in F_1 is a subset of those in H_i. If Q_i does not satisfy the core of *T2*, it does not have a $T2M_i$ value.
   - Procedures *coreMatch* and *supMatch* are listed below. Due to space limitations, we are unable to provide explanations to these procedures.

```
procedure coreMatch(Hi, F1)
  M := 0;
  for each (ag IDg) ∈ F1 do
     if ∃(atm OPm vm) ∈ Hi s.t atm = ag then
        P := Smatch((atm OPm vm), (ag IDg));
        if (P = 0) or (M = 0.01) then M := 0.01
        else M := M + P endif
     else  M := 0; EXIT-LOOP
     endif
  endfor
  return(M)
```

```
procedure supMatch(H_i, F_2)
    M := 0;
    for each (a_g ID_g) ∈ F_2 do
        if ∃(at_m OP_m v_m) ∈ H_i s.t at_m = a_g then
            M := M + Smatch((at_m OP_m v_m), (a_g ID_g))
        endif
    endfor
    return(M)
```

- Let w be the maximal size of the left-hand side of all the *T2*s. The complexity of the whole computation is $O(|Y||R|w)$ if the ranking process is not considered.

2. **Unexpected conclusion rules with respect to a *T2* $\in$ *Y***: The conditions of $R_i \in R$ match those of the *T2* but not the class. For this ranking, we still use the *T2Match* algorithm, but

   $Q = \{r \in R \mid r$ has a different class from the $T2\}$.

   The match value of each $Q_i \in Q$, is $T2M_i$.
3. **Unexpected condition rules with respect to a *T2* $\in$ *Y***: The class of $R_i \in R$ matches that of the *T2*, but not its conditions. This is the opposite to finding conforming rules.

With this, we have finished presenting the proposed technique. Before giving an example, let us answer the question: *Are the rankings optimal*? This question assumes that there exists an optimal ranking, which is doubtful. We believe that it is unlikely to have an optimal ranking due to the subjective nature of interestingness. The important issue is the ability to analyze the discovered rules against some vague impressions, and through such analysis bring out those conforming and unexpected rules to the attention of the user. Our proposed ranking techniques have been tested using 5 real-life databases (from which 18-183 rules are produced) involving our industry partners.

5. An Example

We have conducted many tests with the proposed technique using public domain databases and our real-life databases. Since no existing technique is able to perform the task reported here, we are unable to do a comparison. Below, we provide an example to illustrate the use of our technique.

We choose the credit screening database created by Chiharu Sano in UCI ML repository for illustration because it is easy to understand. This database has 125 cases, 10 attributes and 2 classes *Yes* and *No* representing whether credit is granted. The set of rules generated by C4.5 is:

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

5.1 Ranking using Type_1 *GI*s

The user specified *T1*s are as follows:

1. Jobless {No} → Yes	2. Savings > → Yes
3. Age > → Yes	4. Age < → No
5. YR_Work > → Yes	6. YR_Work < → No
7. Bought | → {Yes, No}	8. M_LOAN | →{Yes, No}

- **Conforming rankings**: The numbers within [] are the contributing *T1*s. The number before [] is the conforming match value and after is the unexpected conclusion match value (note that the match value 1.00 indicates a complete match and 0.00 indicates no match). If both the numbers are large, it could mean a possible *shadowing*. In the rankings below, rules with very low match values are removed to save space.

  Rankings against impressions with **Yes** class:

  Without unanticipated attribute ranking:

  Rank 1 R1: Age>25, Savings >7, YR_Work>2 → Yes
  (1.00) [2,3,5] (0.00)

  Rank 2 R3: Jobless = No, Bought = pc → Yes
  (0.60) [1, 7] (0.10)

  With one unanticipated attribute ranking:

  Rank 1 R2: <u>Sex = Male</u>, YR_Work > 2 → Yes
  (1.00) [5] (0.00)

  From this ranking we see that *Sex* also plays a role which was not known previously.

  Rankings against impressions with **No** class:

  Without unanticipated attribute ranking:

  Rank 1 R7: YR_Work <= 2 → No
  (1.00) [6] (0.00)

  With one unanticipated attribute ranking:

  Rank 1 R5: <u>Sex = Female</u>, Age <= 25 → No
  (1.00) [4] (0.00)

- **Unexpected conclusion rankings**: The number before [] is the unexpected conclusion match value and after is the conforming match value.

  Rankings against impressions with **Yes** class: None

  Rankings against impressions with **No** class:

  Without unanticipated attribute ranking:

  Rank 1 R4: Bought = medinstru, Age <= 34 → Yes
  (0.60) [4,7] (0.10)

  This rule's conclusion is unexpected to some extent as it is against the user's impression, *Age* < → *No*.

  No rule in the unanticipated attribute ranking.

- **Unexpected condition rankings**: The number before [] is the conforming match value and after is the unexpected conclusion match value of each rule.

  Rankings against impressions with **Yes** class:

  Without unanticipated attribute ranking:

  Rank 1 R4: Bought = medinstru, Age <= 34 → Yes
  (0.10) [4, 7] (0.60)

  This rule's condition, *Age* <= 34, is unexpected because the user does not expect that (*Age* <) leads to class *Yes*.

  No rule in the unanticipated attribute ranking.

  Rankings against impressions with **No** class:

  Without unanticipated attribute ranking:

  Rank 1 R6: Savings <= 7, M_LOAN > 7 → No
  (0.10) [8] (0.10)

  This rule's condition, *Savings* <= 7, is unexpected because the user does not know that little saving will lead to class *No*.

  No rule in the unanticipated attribute ranking.

5.2 Ranking using Type_2 *GI*s

Here, we use only one *T2 GI* in the example as the rankings are against each *GI* separately. The user specified *T2* is:

Age > & YR_Work >, Jobless {No} > → Yes

- **Conforming ranking:** The number in () is the conforming match value.

  Rank 1 R1: Age>25, Savings>7, YR_Work>2→Yes (0.67)

  Clearly, R1 is to some extent conforming.
- **Unexpected conclusion ranking:**

  There is no unexpected conclusion rule.
- **Unexpected condition ranking:** The number in () is also the conforming match value.

  Rank 1 R4: Bought = medinstru, Age <= 34 → Yes (0.00)

With these rankings, the user can simply check the few rules at the top of the lists to confirm or to deny his/her *GI*s, and to find those unexpected rules. When the number of rules is large the above rankings will be of great help to the user in his/her analysis of the discovered rules.

6. Related Work

Although many classification rule induction systems can make use of domain knowledge or theory in the discovering process, their purpose is to use the domain theory to help produce more accurate rules (e.g., Ortega and Fisher 1995; Evans and Fisher 1994) or improve the rule explainability (Clark and Matwin 1993). Clearly, they are different from our work, which aims to help the user analyze the discovered rules in order to identify those interesting ones.

In data mining, subjective interestingness (e.g., Piatesky-Shapiro and Matheus 1994a; Piatesky-Shapiro et al. 1994b; Major and Mangano 1993; Klemetinen et al. 1994) has long been identified as an important problem. (Piatesky-Shapiro and Matheus 1994a) studied the issue in a health care application. The system (called KEFIR) analyzes the health care information to uncover interesting deviations. However, KEFIR does not perform rule comparison. Its approach is also application-specific. It is clearly different from our work. Our system compares discovered rules against the user's *GIs*. It is also application independent.

(Silberschatz and Tuzhilin 1996) proposed to use belief systems to describe unexpectedness. A number of formal approaches to beliefs were presented. However, these approaches require the user to provide complex belief information, such as conditional probabilities, which are difficult to obtain in practice. It does not handle *GI*s. The paper also suggested to use an interestingness engine to help the discovery system produce interesting rules in the first place. This is an ideal approach. However, the approach requires the user to supply all his/her existing knowledge to the system in advance, which is difficult in most situations. This is analogous to the problem of *knowledge acquisition* in expert systems (Boose 1993). Our post-processing technique encourages interactive and iterative analysis of the discovered rules. The iterative approach is not suitable for the interestingness engine technique because a knowledge discovery process is normally computational intensive.

(Liu and Hsu 1996) reported a technique for rule analysis against user's expectations. It requires the user to provide reasonably precise knowledge, which was found to be difficult for the user to supply in many applications. The proposed technique overcomes this shortcoming.

7. Conclusion

This paper studies the problem of analyzing discovered rules against a particular form of existing concepts, namely *general impressions* (*GI*s). A specification scheme for representing *GI*s is proposed and two matching algorithms for analyzing discovered rules are presented. This technique is useful for solving the interestingness problem.

Acknowledgments

We would like to thank H-Y. Lee, H-L. Ong, A. Pang, K-H Ho and P-S Lai for many discussions, for providing us the databases, and for their help in the testing of our system.

References

Boose, J. 1993. A survey of knowledge acquisition techniques and tools. In B. Buchanan & D. Wilkins (ed.) *Knowledge Aacquisition and Learning*, 39-56.

Clark, P., and Matwin, S. 1993. Using qualitative models to guide induction learning. *ICML-93*, 49-56.

Evans, R., and Fisher, D. 1994. Overcoming process delays with decision tree induction. *IEEE Expert* 9(1):60-66.

Fayyad, U., Piatesky-Shapiro, G., and Smyth, P. 1996. From data mining to knowledge discovery in databases. *AI Magzine*, 37-54.

Kamber, M., and Shinghal, R. 1996. Evaluating the interestingness of characteristic rules. *KDD-96*, 263-266.

Klemetinen, M., Mannila, H., Ronkainen, P., Toivonen, H., and Verkamo, A.I. 1994. Finding interesting rules from large sets of discovered association rules. *Proceedings of the Third International Conference on Information and Knowledge Management*, 401-407.

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

Liu, B., Hsu, W., and Chen, S. 1997. Discovering conforming and unexpected classification rules. *IJCAI-97 Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, August 23, Nagoya, Japan.

Liu, B., Ku, L-P., and Hsu, W. 1997. Discovering interesting holes in data. To appear in *IJCAI-97*, August 23-29, Nagoya, Japan.

Major, J., and Mangano, J. 1993. Selecting among rules induced from a hurricane database. *KDD-93*, 28-41.

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

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

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

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

Roberts, F. 1984. Applied combinatorics. Pentice Hall.

Silberschatz, A., and Tuzhilin, A. 1996. What makes patterns interesting in knowledge discovery systems. *IEEE Trans. on Know. and Data Eng.* 8(6): 970-974.