Finding Interesting Patterns Using User Expectations

Bing Liu, Wynne Hsu, Lai-Fun Mun, and Hing-Yan Lee[*]

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

[*] Information Technology Institute
11 Science Park Road, Singapore Science Park II,
Singapore 117685, Republic of Singapore
hingyan@iti.gov.sg

Technical Report: TRA7/96

Department of Information Systems and Computer Science

National University of Singapore

Finding Interesting Patterns Using User Expectations

Bing Liu, Wynne Hsu, Lai-Fun Mun, and Hing-Yan Lee*

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

* Information Technology Institute
11 Science Park Road, Singapore Science Park II,
Singapore 117685, Republic of Singapore
hingyan@iti.gov.sg

Abstract

One of the important issues in data mining is the interestingness problem. This problem is described as finding the interesting patterns from a large number of discovered patterns. Typically, in a data mining application, it is all too easy to discover a huge number of patterns. Most of these patterns are actually useless or uninteresting to the user. But due to the huge number of patterns, it is difficult for a user to comprehend all the patterns and to identify those interesting to him/her. To prevent the user from being overwhelmed by the large number of patterns, techniques are needed to analyze and to rank the patterns according to their interestingness. This paper proposes such a technique. It performs post-analysis of the discovered patterns to help the user identify those interesting ones. The technique is based on fuzzy matching of the discovered patterns with a set of user-specified patterns. The degrees of match are then used to rank the discovered patterns according to various interestingness measures, such as unexpectedness and actionability. Experiments have shown that the proposed technique is able to help a user focus on a small subset of discovered patterns that has the most application value.

1. Introduction

In knowledge discovery, techniques are constantly being developed and improved for discovering various types of patterns in databases. While these techniques were shown to be useful in numerous applications, new problems have also emerged. One of the major problems is that, in practice, it is all too easy to discover a huge number of patterns in a database. Most of these patterns are actually useless or uninteresting to the user. But due to the huge number of patterns, it is difficult for the user to comprehend and to identify those patterns that are interesting to him/her. To prevent the user from being overwhelmed by the large number of patterns, techniques are needed to rank them according to their interestingness.

So far, a number of papers have discussed the interestingness issue [e.g., 4, 8, 11, 12, 13, 14]. The main factors that contribute to the interestingness of a discovered pattern have also been proposed. They include: coverage, confidence, strength, statistical significance, simplicity, unexpectedness, actionability [e.g., 4, 8, 11, 12]. The first five factors are called objective measures [15]. They can be handled with techniques requiring no application and domain knowledge. They have been studied extensively in the literature [e.g., 12, 8]. The last two factors are called the subjective measures [11, 15], which measure the subjective interestingness of a pattern to the user. They are defined as follows:

1. Unexpectedness: Patterns are interesting if they are unexpected or previously unknown to the user [4].
2. Actionability: Patterns are interesting if the user can do something with them to his/her advantage [4,11].

It has been noted in [13, 11] that although objective measures are useful in many respects, they are insufficient in determining the interestingness of the discovered patterns. Subjective measures are needed. Subjective interestingness is the focus of this paper. The proposed technique is for ranking the discovered patterns according to their subjective interestingness. It assumes that some other techniques have performed the pattern discovery task and have filtered out those patterns that do not meet the objective requirements.

To design a general ranking technique(s) using subjective interestingness measures is a difficult task. Some of the reasons are: (1) in different domains (or applications), people are interested in different things; (2) given the same database and the patterns discovered, different users may be interested in different subsets of the patterns; (3) even for the same user, at different points in time, his/her interests may also vary due to the specific situation he/she is in at the particular moment. In order to identify and/or to rank the discovered patterns, it is obvious that the system must have a great deal of knowledge about the database, the application domain and the user's interests at a particular time.

To date a number of studies [e.g., 4, 8, 11, 12, 156] have been conducted on the subjective interestingness issues and some systems have also been built [8, 11] with interestingness filtering components to help users focus on the useful patterns. However, these systems mostly handle the subjective interestingness in application/domain-specific fashions [15].

In this paper, we propose a general approach to determine the subjective interestingness (unexpectedness and actionability) of a discovered pattern. The technique is characterized by asking the user to specify a set of patterns according to his/her previous knowledge or intuitive feelings. This specified set of patterns is then used by a fuzzy matching algorithm to match and rank the discovered patterns. The assumption of this technique is that some amount of domain knowledge and the user's interests are implicitly embedded in his/her specified patterns. In general, we can rank the discovered patterns according to their conformities to the user's knowledge or their unexpectedness, or their actionabilities. With such rankings, a user can simply check the few patterns on the top of the list to confirm his/her intuitions (or previous knowledge), or to find those patterns that are against his/her expectation, or to discover those patterns that are actionable.

The proposed approach is simple, effective and highly interactive. Though we do not claim that this technique solves the interestingness problem completely, we believe this is a major step towards the right direction.

2. Problem Definition

From a user's point of view, he/she wants to find patterns from one or more databases, denoted by *D,* that are useful or interesting to him/her. From a discovery system's point of view, a technique *Q* is used to discover all the patterns from *D* that are discoverable by *Q*. Let $B_{(Q,D)}$ be the set of patterns discovered by *Q* on *D*. We denote $I_{(Q,D)}$ as the set of interesting patterns in $B_{(Q,D)}$. Thus, $I_{(Q,D)} \subseteq B_{(Q,D)}$. Three points to be noted:

- $I_{(Q,D)}$ may not be the complete set of interesting patterns that can be discovered from D. It is simply the set of interesting patterns that can be discovered by technique Q on D.
- Not all patterns in $I_{(Q,D)}$ are equally interesting. Different patterns may have different degrees of interestingness to the user.
- $I_{(Q,D)}$ may be a dynamic set in the sense that the user may be interested in different things at different points in time. The degree of interestingness of each pattern may also vary.

In general, $B_{(Q,D)}$ is much larger than $I_{(Q,D)}$. This implies that many patterns discovered by Q are uninteresting or useless. It is desirable that a system only gives the user the set of interesting patterns, $I_{(Q,D)}$, and ranks the patterns in $I_{(Q,D)}$ according to their degrees of interestingness. Hence, we define the interestingness problem as follows:

Definition: Given $B_{(Q,D)}$, the set of patterns discovered by Q on D, determine $I_{(Q,D)}$ and rank the patterns in $I_{(Q,D)}$ according to their degrees of interestingness to the user at the particular point in time.

In practice, this is difficult to achieve because the definition of interestingness is domain (or application) dependent and also user and his/her situation dependent. To simplify our task, we only rank all the patterns discovered (i.e., $B_{(Q,D)}$). Our assumption is that $I_{(Q,D)}$ will be a small subset of the top-ranked patterns. The final identification task is left to the user.

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

3. The Proposed Technique

This section describes the proposed method. Slightly different procedures are used for finding unexpected patterns and for finding actionable patterns.

3.1 Finding unexpected patterns and confirming user's knowledge

Patterns are unexpected if they are previously unknown to the user [4]. Unexpected patterns are, by definition, interesting because they provide new information to the user [4, 15]. Apart from finding unexpected patterns, sometimes the user also wishes to know whether his/her existing knowledge about the database is correct. For these two purposes, the proposed method has the following two steps:

1. The user is asked to provide a set of patterns E (with the same syntax as the discovered patterns) that he/she expects to find in the database D based on his/her previous knowledge or intuitive feelings. These user patterns are regarded as fuzzy patterns (also called user-expected patterns), which are described with the help of fuzzy linguistic variables [17].

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

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

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

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

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

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

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

Thus, in this step, the user needs to input (1) his/her expected patterns, and (2) the fuzzy set $\tilde{M}(X)$ for each term X used in his/her expected patterns.

2. The system then matches (in a number of ways) each discovered pattern in $B_{(Q,D)}$ against the patterns in E using a fuzzy matching technique. The discovered patterns are then ranked according to their degrees of match with E.

Note that, though the proposed technique is application/domain independent, it is not discovery technique independent. This is because different knowledge discovery techniques discover different types of patterns (e.g., classification patterns, association patterns, sequence patterns, time series patterns, etc.). For different pattern types, fuzzy pattern matching methods may not be the same. For example, the matching technique for sequence patterns and the matching technique for classification patterns (or rules) should be different.

In short, the matching algorithm must be customized for different pattern types. Thus, each general discovery tool can have a suitable implementation of the proposed method, which may be used for any domain and any application. Section 4 and 5 describe such an implementation.

Now let us consider an example of how the technique works. Suppose we have the following set of discovered (classification) patterns from an accident database ("," denotes "and").

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

The user-specified expected pattern is:

If *P_Age* = *OLD*, *Loc* = *BAD_VISIBILITY* **then** *Class* = *BAD_ACCIDENT*

Before matching can be performed, the system must know how to interpret the semantic meanings of "*OLD*", "*BAD_VISIBILITY*" and "*BAD_ACCIDENT.*" This is achieved by asking the user to provide the fuzzy sets associated with these terms. A graphical user-interface has been built to make this process of supplying the fuzzy sets easy and simple.

Having specified the semantic meanings, a matching algorithm is then executed to determine the degrees of match between the discovered patterns and the user-specified expected pattern. Different ranking algorithms are used for different purposes. If our purpose is to confirm a hypothesis, the system will rank the discovered patterns such that the pattern with the highest degree of match is ranked first. The results of such a ranking could be as follows:

A1. **If** *P_Age* > 65, *Loc* = *bend*, *Speed* > 50 **then** *Class* = *killed*

A2. **If** *P_Age* > 50, *Loc* = *T-junct* **then** *Class* = *slight*

A3. **If** *P_Age* > 50, *Loc* = *straight* **then** *Class* = *slight*

This confirms the user's belief that an old person involved in an accident at some bad location will result in a serious injury. On the other hand, if our purpose is to find unexpected patterns in the sense of an unexpected consequent, then a different ranking will result as shown below:

B1. **If** *P_Age* > 50, *Loc* = *T-junct* **then** *Class* = *slight*

B2. **If** *P_Age* > 65, *Loc* = *bend*, *Speed* > 50 **then** *Class* = *killed*

B3. **If** *P_Age* > 50, *Loc* = *straight* **then** *Class* = *slight*

This shows that pattern B1 is against the user's expectation because instead of a serious injury, the old person suffers a slight injury. It is important to note that simply reversing of the order for conformity is in general not the right method for ranking patterns according to their unexpectedness. In fact, the unexpectedness of a pattern could be described in a number of ways. Details on this will be discussed in Section 4.

From the example, we can see that by determining the degrees of match between the discovered patterns and the user-specified expected patterns in various ways, and ranking the patterns accordingly, it is possible to help the user focus on the appropriate subsets of patterns based on his/her purpose.

It can also be observed that the working of this method depends on the following assumption.

Assumption: The user knows the database and has some intuitive feelings or previous knowledge about the kinds of pattern that might be found in the database.

We believe this assumption is realistic because in real life, after working on a particular domain and its database for some time, the user generally develops good intuitive sense regarding the kinds of patterns that can be found in the database. We have tested this on our industrial partner. Even if the user is new to the database, database visualization tools are available to help the user obtain a good initial feel of the kinds of patterns in the database. With this as a starting point, the user can then incrementally add more patterns to aid in the ranking process. It is important to note, however, that this method does not require the user to provide the complete set of his/her expected patterns at the beginning, which is quite difficult. Due to the interactive nature of the technique, he/she may try something simple at the beginning and slowly build up the set of expected patterns.

3.2 Finding actionable patterns

Patterns are actionable if the user can do something with them to his/her advantage [4, 11]. The key here is the usefulness to the user. It has been recognized that many unexpected patterns are also actionable. Hence, the method presented above, in some sense, is also able to find some actionable patterns. However, for specific cases in which the user knows what are the possible actions that he/she can take and in what situations to take them, a variation of the above method is proposed to identify actionable patterns. This method consists of three steps.

1. The user specifies all the possible actions Y that he/she (or his/her organization) can take.
2. For each action $Y_q \in Y$, the user specifies the situations under which he/she is likely to take the action. The situations are represented by a set of fuzzy patterns Act^q (similar to the expected patterns in E). The patterns in Act^q are called the user-specified action patterns.

3. The system then matches each discovered pattern in $B_{(Q,D)}$ against the patterns in Act^q using a fuzzy matching technique. The results of this matching are used to rank the discovered patterns in $B_{(Q,D)}$. For each action Y_q (or Act^q), a separate ranking will be produced.

Note that:

- In finding actionable patterns, the user does not provide what he/she expects as for finding unexpected patterns, but rather the situations under which he/she may take some actions. These situations may or may not be what the user expects.
- This technique associates patterns with actions to be taken in response to the patterns. Thus, more information is given to the user, i.e., not only the actionable patterns, but also the actions to be taken.

Let us illustrate this method with an example. Considering the following discovered patterns:

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

In this example, we consider two actions:

Action 1. Educate people to be more careful at locations with good visibility. Assume there is only one user-specified action pattern for which action 1 is to be taken:

If *Loc* = *GOOD_VISIBILITY* **then** *CLASS* = *SLIGHT*

Action 2. Install speed cameras at locations with bad visibility. Again assume there is only one user-specified action pattern for which action 2 is to be taken.

If *Loc* = *BAD_VISIBILITY*, *Speed* ≥ *FAST* **then** *Class* = *BAD_ACCIDENT*

With these and the user specified fuzzy sets for *GOOD_VISIBILITY*, *BAD_VISIBILITY*, *FAST*, *SLIGHT* and *BAD_ACCIDENT*, the ranking results are:

Action 1:	Rank 1	1.	**If** *P_Age* > 50, *Loc* = *straight* **then** *Class* = *slight*
	Rank 2	3.	**If** *P_Age* > 50, *Loc* = *T-junct* **then** *Class* = *slight*
	Rank 3	2.	**If** *Loc* = *bend*, *Speed* > 50 **then** *Class* = *killed*
Action 2:	Rank 1	2.	**If** *Loc* = *bend*, *Speed* > 50 **then** *Class* = *killed*
	Rank 2	3.	**If** *P_Age* > 50, *Loc* = *T-junct* **then** *Class* = *slight*
	Rank 3	1.	**If** *P_Age* > 50, *Loc* = *straight* **then** *Class* = *slight*

This ranking helps the user to identifying those patterns supporting the actions. With this, the user may decide to educate old people to be more careful, and/or to install speed cameras at bends. Note that the actions themselves are not ranked according to the possible benefits they may bring to the user, which will be more helpful. This will be part of our future work.

4. An Implementation of the Proposed Technique

In this and the next section, we will describe a particular implementation of the proposed technique. The patterns assumed in this implementation is as follows:

If $P_1, P_2, P_3, \ldots, P_n$ **then** C

where "," means "and", and P_i is a proposition of the following format:

attr OP value

where *attr* is the name of an attribute in the database, and *value* is a possible value for the attribute *attr* and $OP \in \{=, \neq, <, >, \leq, \geq\}$ is the operator.

C is the consequent, which has the same format as P_i. However, its *attr* does not have to be an attribute name in the database. For example, in the C4.5 system [14], *Class* is used for the consequent. The above representation is very common for classification patterns (rules) and association patterns. We now present the computational formulas used in the methods discussed in Section 3.1 and 3.2.

4.1. Confirming user knowledge and finding unexpected patterns

Before presenting the detailed computation, we first define some notations. Let E be the set of user-expected patterns and B (previously $B_{(Q,D)}$) be the set of discovered patterns. We denote W_i as the degree of match of a discovered pattern $B_i \in B$ with respect to the set of expected patterns E. We denote $w_{(i,j)}$ as the degree of match between a discovered pattern $B_i \in B$ and an expected pattern $E_j \in E$. Ranking of the discovered patterns is performed by sorting them in a decreasing order according to their W_i (i.e., the pattern with the highest W_i will be on top). Let us now discuss the computation of $w_{(i,j)}$ and W_i. $w_{(i,j)}$ is computed in two steps:

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

   $$L_{(i,j)} = \frac{|A_{(i,j)}|}{\max(|e_j|, |b_i|)}$$

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

   Likewise the attribute names of the consequents of B_i and E_j are also compared. $R_{(i,j)}$ denotes the degree of match for the consequent parts. $R_{(i,j)}$ is either 0 or 1. This is because we assume that there is only one consequent for each pattern. Hence, either the consequent attributes of the two patterns are the same ($R_{(i,j)} = 1$), or different ($R_{(i,j)} = 0$).

   For example, we have

   B_i: **If** *Height* > 1.8, *Weight* < 70 **then** *Health_con* = *underweight*.

   E_j : **If** *Build* = *medium*, *Weight* > 65, *Weight* < 70 **then** *Health_con* = *fit*.

   The set of common attributes in the conditional parts is $A_{(i,j)} = \{Weight\}$. The consequent parts have the same attribute *Health_con*. Hence,

   $L_{(i,j)} = 0.5$, and $R_{(i,j)} = 1$.

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

Here, we present the computation of $w_{(i,j)}$ and W_i. As mentioned in Section 3, the proposed method can be used for confirming user's hypothesis and also for finding unexpected patterns. For these two purposes, different formulas are used for computing $w_{(i,j)}$ and W_i. Note that we do not claim these formulas are optimal. But a large number of experiments have shown that they produce rankings that closely model human intuition of subjective interestingness.

1. Confirming user's knowledge

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

Explanation: $\dfrac{L_{(i,j)} \times \sum_{k \in A(i,j)} V_{(i,j)k}}{|A_{(i,j)}|}$ computes the degree of match of the conditional parts of B_i and E_j. $R_{(i,j)} \times Z_{(i,j)}$ computes the degree of match of the consequent parts of B_i and E_j. $w_{(i,j)}$ gives the degree of match of pattern B_i with pattern E_j.

The formula for computing W_i, which is the degree of match of the discovered pattern B_i with respect to the set of expected patterns E, is defined as follows (see also Figure 1):

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

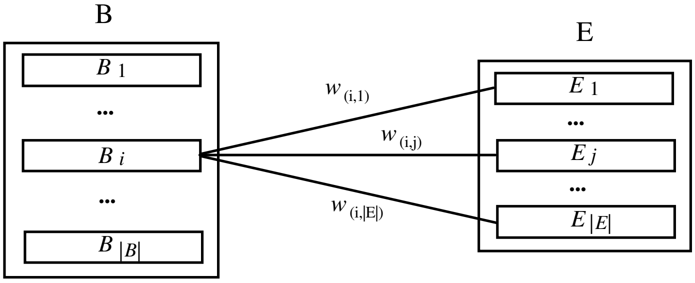


Figure 1. Computing W_i

2. Finding unexpected patterns

For this purpose, the situation is more complex. We can have a number of ways to rank the patterns according to the types of unexpectedness.

1) **Unexpected consequent:** The conditional parts of B_i and E_j are similar, but the consequents of the two patterns are far apart. Two types of ranking are possible depending on the user's interest.

(a) Contradictory consequent (patterns with $R_{(i,j)} = 1$ will be ranked higher).

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

Explanation: Since this ranking is to find those patterns whose conditional parts are similar, but the consequents are contradictory, we need to give higher $w_{(i,j)}$ value for B_i whose consequent part has the same attribute name as E_j, hence the expression $R_{(i,j)} \times (Z_{(i,j)} - 1)$.

W_i is computed as follows:

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

(b) Unanticipated consequent (patterns with $R_{(i,j)} = 0$ will be ranked higher).

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

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

Explanation: A higher value is given to $w_{a(i,j)}$ when the attribute names of the consequent parts of B_i and E_j do not match. However, B_i may match well with another expected pattern E_r. Thus, $w_{b(i,j)}$ is needed to take this into consideration.

W_i is computed as follows:

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

2) **Unexpected reason:** The consequents are similar but the conditional parts of B_i and E_j are far apart. Again two types of ranking are possible.

(a) Contradictory conditions (patterns with $|A_{(i,j)}| > 0$ will be ranked higher)

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

Explanation: Since this ranking is to find those patterns whose consequents match well, but the conditional parts are contradictory, we need to give a higher $w_{(i,j)}$ value for B_i whose conditional part has good attribute name match with E_j. Therefore, we have the expression $L_{(i,j)} \times \left(\dfrac{\sum_{k \in A(i,j)} V_{(i,j)k}}{|A_{(i,j)}|} - 1 \right)$.

W_i is computed as follows:

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

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

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

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

Explanation: A higher value is given to $w_{a(i,j)}$ when the attribute names of the conditional parts of B_i and E_j do not match well. However, B_i may match well with another expected pattern E_r. Thus, $w_{b(i,j)}$ is needed to take this into consideration.

W_i is computed as follows:

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

3) **Totally unexpected patterns:** Both conditional and consequent parts of B_i and E_j are very different in the sense that the attribute names in B_i are unexpected.

$$w_{(i,j)} = \begin{cases} 4 - L_{(i,j)} \times \left(\dfrac{\sum_{k \in A(i,j)} V_{(i,j)k}}{|A_{(i,j)}|} + 1 \right) - R_{(i,j)} \times (Z_{(i,j)} + 1) & |A_{(i,j)}| \neq 0 \\ 4 - R_{(i,j)} \times (Z_{(i,j)} + 1) & |A_{(i,j)}| = 0 \end{cases}$$

Explanation: Since this ranking is to find those patterns whose attribute names (both in the conditional parts and consequent parts) have little (or no) intersection with the set of attribute names mentioned in E, we give higher $w_{(i,j)}$ value for B_i whose attribute names do not match well with those of E_j.

W_i is computed as follows:

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

4.2 Finding actionable patterns

For finding actionable patterns, the notations in Section 4.1 still apply except E, which is replaced with Act^q. The formula for matching the discovered pattern B_i and the user-specified action pattern $Act^q_j \in Act^q$ is the same as the one for confirming user's knowledge in Section 4.1. However, the computation of $Z_{(i,j)}$ and $V_{(i,j)k}$ are slightly different from those used in Section 4.1, which will be discussed in the next section.

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

Explanation: The matching formula for conformity is used because the purpose of matching here is the same as the matching for confirming the user's knowledge.

We denote $W_{(i,q)}$ as the degree of match of B_i with respect to the set of action patterns Act^q. $W_{(i,q)}$ is computed as follows:

$$W_{(i,q)} = \max(w_{(i,1)}, w_{(i,2)}, \ldots, w_{(i,j)}, .., w_{(i,|Act^q|)})$$

The discovered patterns in B are ranked for each action $Y_q \in Y$ according to their $W_{(i,q)}$ values.

5. Fuzzy Matching of Attribute Values

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

5.1. Matching of discrete attribute values

In this case, the semantic rule for each term (X) used in describing the user-specified patterns must be properly defined over the universe (or domain) of the discrete attribute. We denote U_k as the set of possible values for the attribute. For each $u \in U_k$, the user needs to input the membership value of u in X, denoted as $\mu_X(u)$. In the discrete case, the formulas for computing $V_{(i,j)k}$ and $Z_{(i,j)}$ for finding unexpected patterns and for finding actionable patterns are the same.

Let us have a example. The user gives the following pattern:

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

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

{(F, 1), (D, 0.8), (C, 0.2)}

where the left coordinate is an element in the universe (or domain) of the "*Grade*" attribute, and the right coordinate is the degree of membership of that element in the fuzzy set *poor*, e.g., $\mu_{poor}(D) = 0.8$. It is assumed that all the other attribute values not mentioned in the set have the degree of membership of 0.

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

User-specified proposition: *attr Opu X*
System-discovered proposition: *attr Ops S*

where *attr* is the matching attribute, *Opu* and *Ops* belong to the set {=, ≠}, and X and S are the attribute values of *attr*. Since the matching algorithm must take into consideration the combination of both the operators and attribute values, four cases result:

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

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

Case 2. *Opu* = "=" and *Ops* = "≠":

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

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

Case 3. *Opu* = "≠" and *Ops* = "=":

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

Case 4. *Opu* = "≠" and *Ops* = "≠":

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

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

5.2. Matching of continuous attribute values

When an attribute takes continuous values, the semantic rule for the term (X) takes on the form of a continuous function. To simplify the user's task of specifying the shape of this continuous function, we assume that the function has a curve of the form as shown in Figure 2. Thus, the user merely needs to provide the values for a, b, c, and d.

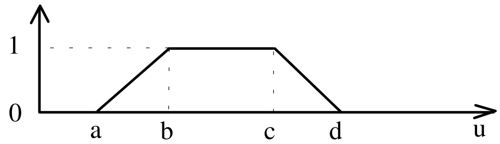


Figure 2. Membership function

For example, the user's pattern is:

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

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

In the continuous case, the set of values that the operator in a proposition can take is {=, ≠, ≥, ≤, ≤≤}. "≤≤" is used to represent a range like this: $X_1 \leq attr \leq X_2$. With this expansion, the total number of possible combinations to be considered is 25. All the formulas are listed in appendix.

In this continuous case, the formulas used for finding unexpected patterns (Section 4.1) and for finding actionable patterns (Section 4.2) are slightly different. The difference is that for finding unexpected patterns, it compares two propositions to see how different they are. But, for finding actionable patterns, it checks to see whether the proposition used in the discovered pattern is covered by the proposition used in the user-specified pattern or vice versa. For example, we have

User-specified proposition: $A \geq 5$

System-discovered proposition: $A \geq 10$

For the fuzzy term 5 (in $A \geq 5$), assume a = b = c = d = 5. In the case of finding unexpected patterns, $V_{(i,j)k}$ will be evaluated to be less than 1 because the two propositions do not cover the same area. However, in the case of finding actionable patterns, $V_{(i,j)k} = 1$.

6. Evaluation

The proposed technique is implemented in Visual C++ on PC. A test example is given below. An analysis of the complexity of the algorithm is also presented.

6.1. A test example

This sub-section gives a test example. The set of patterns is generated from a real database using C4.5. All the attribute names and also some attribute values have been encoded to ensure confidentiality of the data. Since for pattern generation in this test we used C4.5, which generates classification patterns and uses only one attribute as the class (or as the consequent), we cannot test the rankings for *unanticipated consequent* and *totally unexpected patterns*. To save space, only a small subset of the patterns generated by C4.5 is listed below for ranking.

Pattern 1: A1 <= 41

-> Class NO

Pattern 2: A1 <= 49, A3 <= 5.49, A4 > 60

-> Class NO

Pattern 3: A1 > 49, A2 = 2

-> Class YES

Pattern 4: A1 > 49, A1 <= 50

-> Class YES

Pattern 5: A1 > 55

-> Class YES

Pattern 6: A1 > 41, A3 > 5.49, A7 > 106, A4 > 60, A10 <= 5.06

-> Class YES

Pattern 7: A1 > 41, A4 <= 60

-> Class YES

Pattern 8: A1 > 41, A1 <= 47, A3 <= 3.91, A7 > 106, A4 > 60, A10 <= 5.06

-> Class YES

Three runs of the system are conducted in this testing. In the first run, the focus is on confirming the user's knowledge, while in the second run the focus is on finding unexpected patterns. In the third run, the focus is on finding actionable patterns.

6.1.1 Confirming user's knowledge

The set of user expected patterns is listed below with the fuzzy set attached to each term (attribute value used in the user's patterns).

User expected pattern set 1:

Pattern 1: A1 <= Md_A1 {a = 30, b = 35, c = 45, d = 50}

-> Class NO {(NO, 1), (YES, 0)}

Pattern 2: A1 >= Re_A1 {a = 50, b = 53, c = 57, d = 60}

-> Class YES {(NO, 0), (YES, 1)}

- **Ranking results**:

RANK 1 Pattern 5: A1 > 55

-> Class YES

confirming user specified pattern 2

RANK 2 Pattern 1: A1 <= 41

-> Class NO

confirming user specified pattern 1

RANK 3 Pattern 3: A1 > 49, A2 = 2

-> Class YES

confirming user specified pattern 2

RANK 4 Pattern 7: A1 > 41, A4 <= 60

-> Class YES

confirming user specified pattern 2

RANK 5 Pattern 2: A1 <= 49, A3 <= 5.49, A4 > 60

-> Class NO

confirming user specified pattern 1

The rest of the patterns are cut off because of their low matching values.

6.1.2 Finding unexpected patterns

The set of user expected patterns for this test run is listed below, which is followed by three types of ranking for finding unexpected patterns.

User expected pattern set 2:

Pattern 1:	A7 >= 150,	{a = 147, b = 148, c = 152, d = 153}
	A4 >= 95	{a = 92, b = 93, c = 97, d = 98}
	-> Class YES	{(NO, 0), (YES, 1)}
Pattern 2:	A3 >= 2	{a = 1.75, b = 1.8, c = 2.2, d = 2.25}
	-> Class YES	{(NO, 0), (YES, 1)}

- **Unexpected consequent:**

  RANK 1 Pattern 2: A1 <= 49, A3 <= 5.49, A4 > 60
  -> Class NO

  contradicting user specified pattern 2

  The rest of the patterns are cut off because of their low matching values.

- **Contradictory conditions:**

  RANK 1 Pattern 7: A1 > 41, A4 <= 60
  -> Class YES
  contradicting user specified pattern 1

  RANK 2 Pattern 6: A1 > 41, A3 > 5.49, A7 > 106, A4 > 60, A10 <= 5.06
  -> Class YES
  contradicting user specified pattern 1

  RANK 3 Pattern 8: A1 > 41, A1 <= 47, A3 <= 3.91, A7 > 106, A4 > 60, A10 <= 5.06
  -> Class YES
  contradicting user specified pattern 1

  The rest of the patterns are cut off because of their low matching values.

- **Unanticipated conditions:**

  RANK 1 Pattern 3: A1 > 49, A2 = 2
  -> Class YES

  RANK 2 Pattern 4: A1 > 49, A1 <= 50
  -> Class YES

  RANK 3 Pattern 5: A1 > 55
  -> Class YES

  RANK 4 Pattern 7: A1 > 41, A4 <= 60
  -> Class YES

  The rest of the patterns are cut off because of their low matching values.

6.1.3 Finding actionable patterns

For simplicity, we use only two actions. One action has one user-specified action pattern, while the other has two. Due to the confidentiality, we cannot provide the real actions, but use First_Action, and Second_Action to represent them.

User's actions and patterns:

Action 1: First_Action
User patterns: Pattern 1: A1 >= Re_A1 {a = 48, b = 50, c = 58. d = 60}
-> Class YES {(NO, 0), (YES, 1)}

Action 2: Second_Action
User patterns: Pattern 1: A3 >= 2 {a = 1.75, b = 1.8, c = 2.2, d = 2.25}
-> Class YES {(NO, 0), (YES, 1)}
Pattern 2: A7 >= 150 {a = 147, b = 148, c = 152, d = 153}
A4 >= 95 {a = 92, b = 93, c = 97, d = 98}
-> Class YES {(NO, 0), (YES, 1)}

- **Ranking results**:

Action 1: First_Action

RANK 1 Pattern 5: A1 > 55
-> Class YES
actionable according to user-specified pattern 1.

RANK 2 Pattern 3: A1 > 49, A2 = 2
-> Class YES
actionable according to user-specified pattern 1.

RANK 3 Pattern 4: A1 > 49, A1 <= 50
-> Class YES
actionable according to user-specified pattern 1.

RANK 4 Pattern 7: A1 > 41, A4 <= 60
-> Class YES
actionable according to user-specified pattern 1.

The rest of the patterns are cut off because of their low matching values.

Action 2: Second_Action

RANK 1 Pattern 6: A1 > 41, A3 > 5.49, A7 > 106, A4 > 60, A10 <= 5.06
-> Class YES
actionable according to user-specified pattern 2.

RANK 2 Pattern 8: A1 > 41, A1 <= 47, A3 <= 3.91, A7 > 106, A4 > 60, A10 <= 5.06
-> Class YES
actionable according to user-specified pattern 2.

The rest of the patterns are cut off because of their low matching values.

6.2 Efficiency analysis

Finally, let us analyze the runtime complexity of the proposed technique. Here, we only analyze the algorithm for finding unexpected patterns. For finding actionable patterns, the basic algorithm is the same. Assume the maximal number of propositions in a pattern (a user-expected pattern or a discovered pattern) is N. Assuming the attribute value matching (computing $V_{(i,j)k}$ and $Z_{(i,j)}$) takes constant time. Combining the individual matching values to calculate $w_{(i,j)}$ also takes constant time. The computation of W_i is O($|E|$). Then, without considering the final ranking which is basically a sorting process, the worst-case time complexity of the technique is O($|E||B|N^2$).

7. Related Work

Although the interestingness has long been identified as an important issue in data mining [4], most of the data mining techniques and tools do not deal with this problem. Instead, their primary concern is to discover all the patterns in the given databases [4, 11, 15].

To date, some studies have been performed on the interestingness problem [1, 4, 8, 11, 12, 13, 15]. A number of interestingness measures have also been proposed. These measures can be classified into two classes: objective measures and subjective measures. Objective measures typically involve analyzing the discovered patterns' structures, their predictive performances, and their statistical significance [4, 8, 12]. Examples of objective measures are: coverage, certainty factor, strength, statistical significance and simplicity [3, 6, 8, 11].

It has been noted in [11], however, that objective measures are insufficient for determining the interestingness of the discovered patterns. Subjective measures are needed. Two main subjective measures are: unexpectedness, and actionability [4, 11].

[8] defined pattern interestingness in terms of performance, simplicity, novelty, significance, etc. Most of the measures are objective measures with the exception of novelty. However, no general method was proposed for handling novelty. Instead, domain-specific theories are coded to aid in filtering out the uninteresting patterns.

[11] studied the issue of interestingness in the context of a health care application. A knowledge discovery system, KEFIR, is built. The system analyzes health care information to uncover "key findings". Key findings refer to the important deviations from the norms for various indicators such as cost, usage, and quality. The degree of interestingness of a finding is estimated by the amount of benefit that could be realized if an action can be taken in response to the finding. Domain experts provide the recommended actions to be taken for various types of findings. Once a finding is discovered, the system computes the estimated benefit for taking an recommended action.

The method used in KEFIR presents a good approach for incorporating the subjective interestingness into an application system. However, the approach is application specific. Its domain knowledge (from domain experts) is hard-coded in the system as production rules. The system cannot be used for any other application. In contrast, our method is general. It does not make any domain-specific assumptions. A pattern analysis system based on our technique can be attached to each data mining tool to help the user identify the interesting patterns. Though domain-specific systems such as KEFIR are still the most effective method for ranking patterns and actions, the cost of building such a system is very high.

[15] proposed to use probabilistic beliefs and belief revision as the framework for describing subjective interestingness. Specifically, a belief system is used for defining unexpectedness. A belief is represented as an arbitrary predicate formula. Associated with each belief is a confidence measure. Two types of beliefs are presented, hard and soft beliefs. Basically, hard beliefs cannot be changed even in the face of new evidences, while soft beliefs are modifiable when a new evidence arrives. If a pattern contradicts the hard beliefs of the user, then this pattern is unexpected and interesting. The unexpectedness of a pattern is also defined with respect to a soft belief. However, [15] is just a proposal. No system has been implemented that utilizes this approach. For an actual implementation, a great deal of details have to be studied. [15] also does not handle pattern actionability. Our proposed approach has been implemented and tested. In addition, our approach allows the user to specify his/her beliefs (expectations) in fuzzy terms which are more natural and intuitive than complex conditional probabilities that the user has to assign in [15].

8. Conclusion

In this paper, we study the subjective interestingness issue in data mining from a domain independent perspective. A general method for ranking the discovered patterns according to their interestingness is proposed. An particular implementation has also been done. This method is characterized by asking the user to input his/her expected or action patterns and then the system ranks the discovered patterns by matching them against the expected or the action patterns. This method can be used to confirm user's knowledge, to find unexpected patterns, or to discover actionable patterns. Besides these applications, the proposed technique may also be used to discover interesting trends by periodically analyzing the deviations of the newly discovered patterns against the old patterns. This can be done simply by using the old patterns as the user-specified patterns.

The proposed method is simple and effective. It is also highly interactive and allows the user to identify interesting patterns incrementally. We do not claim, however, that the issues associated with interestingness are fully understood. Much further research is still needed, e.g., we still do not have a good understanding of how objective interestingness measures such as coverage and confidence interact with subjective interestingness measures, and how actions themselves may be ranked to give the user more information.

Acknowledgment: We would like to thank Gui-Jun Yang for implementing the user interface of the system. We thank Hwee-Leng Ong and Angeline Pang of Information Technology Institute for many useful discussions. The project is funded by National Science and Technology Board.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Database mining: a performance perspective. *IEEE Trans. on Knowl. and Data Eng.* 5(6), pg. 914-925.

[2] I. Bhandari, 1993. Attribute focusing: machine-assisted knowledge discovery applied to software production process control. In *Proceedings of AAAI-93 Workshop on Knowledge Discovery in Databases*, 1993.

[3] V. Dhar and A Tuzhilin, Abstract-driven pattern discovery in databases. *IEEE Trans. Knowl. Data Eng.* 5(6), 1993.

[4] W. J. Frawely, G. Piatesky-Shapiro, C. J. Matheus. Knowledge discovery in databases: an overview. In G. Piatetsky-Shapiro and W.J Frawley (eds), *Knowledge Discovery in Databases*, AAAI/MIT Press, 1991, pg 1-27.

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

[6] J. Hong and C. Mao. Incremental disocvery of rules and structure by hierachical and parallel clustering. In G. Piatetsky-Shapiro and W.J Frawley (eds), *Knowledge Discovery in Databases*, AAAI/MIT Press, 1991.

[7] W. Klosgen. Problems for knowledge discovery in databases and their treatment in the statistics interpreter explora. *International Journal of Intelligent Systems*, 7(7), 1992, pg. 649-673.

[8] J. A. Major, J. J. Mangano. Selecting among rules induced from a hurricane database. In *Proceedings of AAAI Workshop on Knowledge Discovery in Databases*, 1993, pg 30-31.

[9] C. J. Matheus, P. K. Chan, and G. Piatetsky-Shapiro. Systems for knowledge discovery in databases. *IEEE Trans. Knowl. & Data Eng.*, 5(6), 1993.

[10] C. J. Matheus, G. Piatetsky-Shapiro, and D. McNeill. An application of KEFIR to the analysis of healthcare information. In *Proceedings of AAAI-94 Workshop on Knowledge Discovery in Databases*, 1994.

[11] G. Piatesky-Shapiro and C. J Matheus. The interestingness of deviations. In *Proceedings of AAAI Workshop on Knowledge Discovery in Databases*, 1994, pg. 25-36.

[12] G. Piatesky-Shapiro. Discovery, analysis, and presentation of strong rules. In G. Piatetsky-Shapiro and W.J Frawley (eds), *Knowledge Discovery in Databases*, AAAI/MIT Press, 1991, pg 231-233.
[13] G. Piatetsky-Shapiro, C. Matheus, P. Smyth, and R. Uthurusamy. KDD-93: progress and challenges ... *AI magazine*, Fall, 1994, pg. 77-87.
[14] J. Ross Quinlan. *C4.5*: *program for machine learning*. Morgan Kaufmann, 1992.
[15] A. Silberschatz and A. Tuzhilin. On subjective measures of interestingness in knowledge discovery, In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, 1995, pg. 275-281.
[16] R. Uthurusamy, U.M Fayyad, and S. Spangler. Learning useful rules from inconclusive data. In G. Piatetsky-Shapiro and W.J Frawley (eds), *Knowledge Discovery in Databases*. AAAI/MIT Press, 1991.
[17] H. -J. Zimmermann. *Fuzzy set theory and its applications*. Second Edition, Kluwer Academic Publishers, 1991.

Appendix

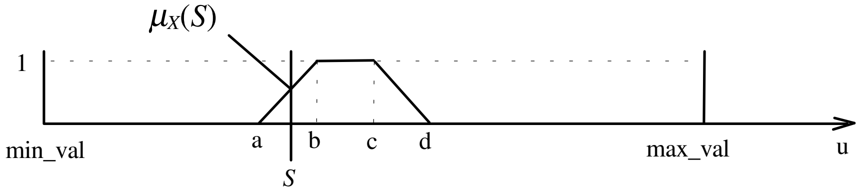
Here we list all the formulas for value matching in the continuous case. Since a particular proposition from either a user-expected pattern or a discovered pattern can use any one of the operators {=, ≠, ≤, ≥, ≤≤), there are altogether 25 formulas. Discovered patterns using < or > are treated in the same way as ≤ or ≥ in our computation. Assume the propositions from the user-expected pattern and the discovered pattern are as follows:

User-expected: $attr\ Opu\ X$ or $X_1 \leq attr \leq X_2$
Discovered: $attr\ Opd\ S$ or $S_1 \leq attr \leq S_2$

Let U = [min_val, max_val] be the domain or the universe of the attribute *attr*, where min_val and max_val are the minimal and the maximal values of the attribute respectively.

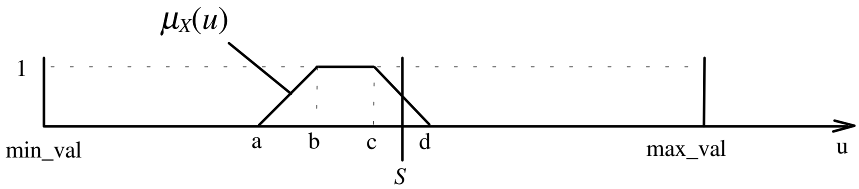
1. *Opu* = *Opd* = "=":

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



2. *Opu* = "=" and *Opd* = "≠":

Let $U_X = \{\, u \mid a \leq u \leq d\}$.



$$V_{(i,\,j)k} = \frac{\int_{U_X} \mu_X(u)du}{\int_U du}$$

3. *Opu* = "=" and *Opd* = "≥":

Let $U_X = \{\, u \mid a \leq u \leq d\}$ and $U_S = \begin{cases} \{u | u \geq b\} & b \leq S \leq c \\ \{u | u \geq S\} & otherwise \end{cases}$.

$$V_{(i,j)k} = \frac{\int_{U_X \cap U_S} \mu_X(u)\,du}{\int_{U_X \cup U_S} du}$$

4. Opu = “=“ and Opd = “≤“:

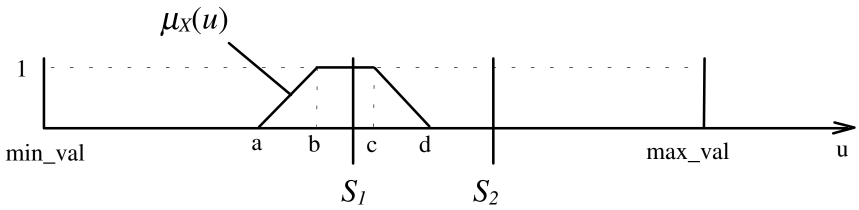
Let $U_X = \{\, u \mid a \le u \le d\}$ and $U_S = \begin{cases} \{u | u \le c\} & b \le S \le c \\ \{u | u \le S\} & otherwise \end{cases}$.

$$V_{(i,j)k} = \frac{\int_{U_X \cap U_S} \mu_X(u)\,du}{\int_{U_X \cup U_S} du}$$

5. Opu = “=“ and Opd = “≤≤“:

Let $U_X = \{\, u \mid a \le u \le d\}$, $S_1' = \begin{cases} b & b \le S_1 \le c \\ S_1 & otherwise \end{cases}$, $S_2' = \begin{cases} c & b \le S_2 \le c \\ S_2 & otherwise \end{cases}$, and

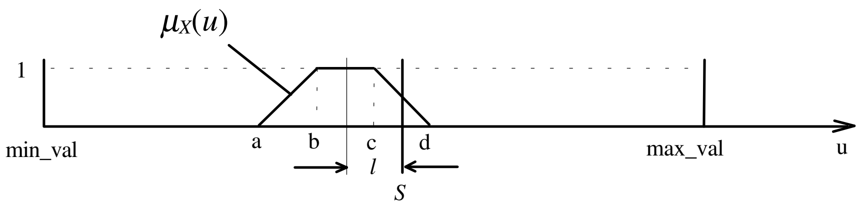
$U_S = \{\, u \mid S_1' \le u \le S_2' \}$.



$$V_{(i,j)k} = \frac{\int_{U_X \cap U_S} \mu_X(u)\,du}{\int_{U_X \cup U_S} du}$$

6. $Opu = Opd$ = “≠“:

Let $U_X = \{\, u \mid a \le u \le d\}$.



$$V_{(i,j)k} = \left(1 - \frac{\int_{U_X} \mu_X(u)\,du}{\int_{U} du}\right) \times \left(1 - \frac{l}{\max\_val - \min\_val}\right)$$

7. Opu = “≠“ and Opd = “=“:

$$V_{(i,j)k} = \left(1 - \frac{l}{\max\_val - \min\_val}\right) \times w,$$

where w is small fixed value.

8. Opu = “≠“ and Opd = “≥“:

Let $U_S = \begin{cases} \{u | u \geq b\} & b \leq S \leq c \\ \{u | u \geq S\} & otherwise \end{cases}$.

$$V_{(i,\,j)k} = \frac{\int_{U_S} \mu_{\neg X}(u)du}{\int_U du}$$

9. *Opu* = "≠" and *Opd* = "≤":

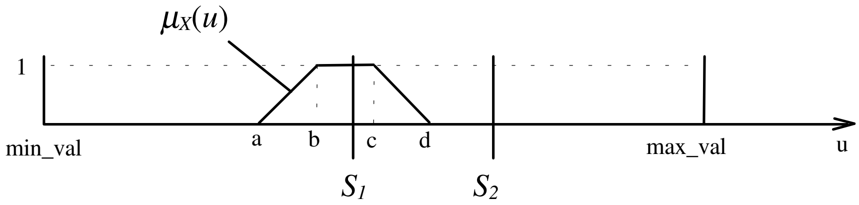
Let $U_S = \begin{cases} \{u | u \leq c\} & b \leq S \leq c \\ \{u | u \leq S\} & otherwise \end{cases}$.

$$V_{(i,\,j)k} = \frac{\int_{U_S} \mu_{\neg X}(u)du}{\int_U du}$$

10. *Opu* = "≠" and *Opd* = "≤≤":

Let $S_1^{'} = \begin{cases} b & b \leq S_1 \leq c \\ S_1 & otherwise \end{cases}$, $S_2^{'} = \begin{cases} c & b \leq S_2 \leq c \\ S_2 & otherwise \end{cases}$, and

$U_S = \{\, u \mid S_1^{'} \leq u \leq S_2^{'} \,\}$.

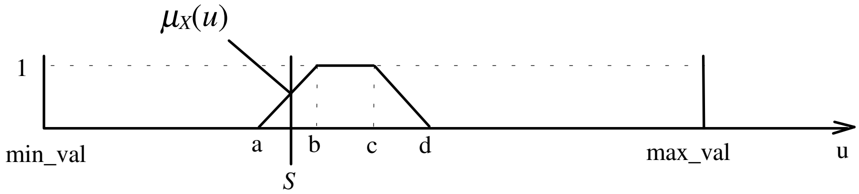


$$V_{(i,\,j)k} = \frac{\int_{U_S} \mu_{\neg X}(u)du}{\int_U du}$$

11. *Opu* = *Opd* = "≥":

Let $f(u) = \begin{cases} 0 & u < a \\ \mu_X(u) & a \leq u < b \\ 1 & u \geq b \end{cases}$,

$U_X = \{u \mid u \geq a\}$, and $U_S = \begin{cases} \{u | u \geq b\} & b \leq S \leq c \\ \{u | u \geq S\} & otherwise \end{cases}$.



$$V_{(i,j)k} = \frac{\min(\int_{U_X} f(u)du, \int_{U_S} du)}{\max(\int_{U_X} f(u)du, \int_{U_S} du)}$$

12. *Opu* = "≥" and *Opd* = "=":

$$V_{(i,j)k} = \begin{cases} 0, & S \le a \\ w, & otherwise \end{cases},$$

where w is a fixed small value.

13. *Opu* = "≥" and *Opd* = "≠":

Let $f(u) = \begin{cases} 0 & u < a \\ \mu_X(u) & a \le u < b \\ 1 & u \ge b \end{cases}$,

$U_X = \{u \mid u \ge a\}$.

$$V_{(i,j)k} = \frac{\int_{U_X} f(u)du}{\int_U du}$$

14. *Opu* = "≥" and *Opd* = "≤":

Let $f(u) = \begin{cases} 0 & u < a \\ \mu_X(u) & a \le u < b \\ 1 & u \ge b \end{cases}$,

$U_X = \{u \mid u \ge a\}$, and $U_S = \begin{cases} \{u \mid u \le c\} & b \le S \le c \\ \{u \mid u \le S\} & otherwise \end{cases}$.

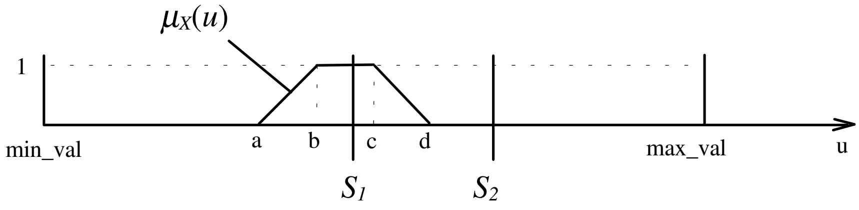
$$V_{(i,j)k} = \frac{\int_{U_S \cap U_X} f(u)du}{\int_U du}$$

15. *Opu* = "≥" and *Opd* = "≤≤":

Let $f(u) = \begin{cases} 0 & u < a \\ \mu_X(u) & a \le u < b \\ 1 & u \ge b \end{cases}$,

$S_1' = \begin{cases} b & b \le S_1 \le c \\ S_1 & otherwise \end{cases}$, $S_2' = \begin{cases} c & b \le S_2 \le c \\ S_2 & otherwise \end{cases}$,

$U_S = \{ u \mid S_1' \le u \le S_2' \}$, and $U_X = \{u \mid u \ge a\}$.

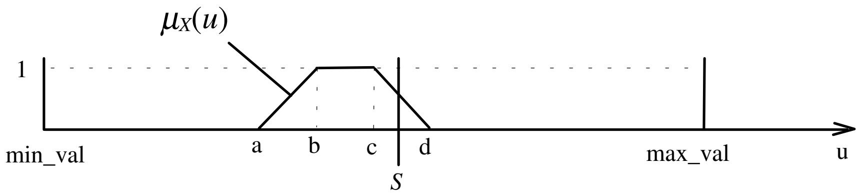


$$V_{(i,\,j)k} = \frac{\int_{U_S} f(u)du}{\int_{U_S \cup U_X} du}$$

16. *Opu* = *Opd* = “≤“:

Let $f(u) = \begin{cases} 1 & u \le c \\ \mu_X(u) & c < u \le d \\ 0 & otherwise \end{cases}$,

$U_X = \{u \mid u \le d\}$, and $U_S = \begin{cases} \{u | u \le c\} & b \le S \le c \\ \{u | u \le S\} & otherwise \end{cases}$.



$$V_{(i,\,j)k} = \frac{\min(\int_{U_X} f(u)du, \int_{U_S} du)}{\max(\int_{U_X} f(u)du, \int_{U_S} du)}$$

17. *Opu* = “≤“ and *Opd* = “=“:

$$V_{(i,\,j)k} = \begin{cases} 0, & S \ge d \\ w, & otherwise \end{cases},$$

where *w* is a fixed small value.

18. *Opu* = “≤“ and *Opd* = “≠“:

Let $f(u) = \begin{cases} 1 & u \le c \\ \mu_X(u) & c < u \le d \\ 0 & otherwise \end{cases}$,

$U_X = \{u \mid u \le d\}$.

$$V_{(i,\,j)k} = \frac{\int_{D_X} f(u)du}{\int_{U} du}$$

19. *Opu* = “≤“ and *Opd* = “≥“:

Let $f(u) = \begin{cases} 1 & u \le c \\ \mu_X(u) & c < u \le d \\ 0 & otherwise \end{cases}$,

$U_X = \{u \mid u \le d\}$, and $U_S = \begin{cases} \{u | u \ge b\} & b \le S \le c \\ \{u | u \ge S\} & otherwise \end{cases}$.

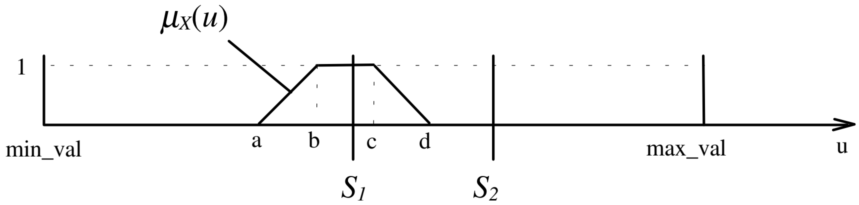
$$V_{(i,\,j)k} = \frac{\int_{U_S \cap U_X} f(u)\,du}{\int_U du}$$

20. *Opu* = “≤“ and *Opd* = “≤≤“:

Let $f(u) = \begin{cases} 1 & u \le c \\ \mu_X(u) & c < u \le d \\ 0 & otherwise \end{cases}$,

$S_1^{'} = \begin{cases} b & b \le S_1 \le c \\ S_1 & otherwise \end{cases}$, $S_2^{'} = \begin{cases} c & b \le S_2 \le c \\ S_2 & otherwise \end{cases}$,

$U_S = \{\, u \mid S_1^{'} \le u \le S_2^{'} \}$, and $U_X = \{u \mid u \le d\}$.



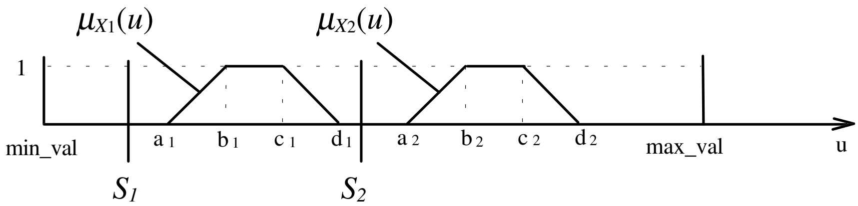
$$V_{(i,\,j)k} = \frac{\int_{U_S} f(u)du}{\int_{U_S \cup U_X} du}$$

21. *Opu* = *Opd* = “≤≤“:

Let $f(u) = \begin{cases} \mu_{X1}(u) & a_1 \le u < b_1 \\ 1 & b_1 \le u < c_2 \\ \mu_{X2}(u) & c_2 \le u < d_2 \\ 0 & otherwise \end{cases}$,

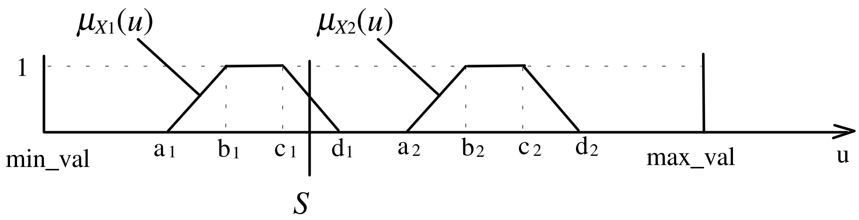
$S_1^{'} = \begin{cases} b_1 & b_1 \le S_1 \le c_1 \\ S_1 & otherwise \end{cases}$, $S_2^{'} = \begin{cases} c_2 & b_2 \le S_2 \le c_2 \\ S_2 & otherwise \end{cases}$,

$U_S = \{u \mid S_1^{'} \le u \le S_2^{'} \}$, and $U_X = \{u \mid a_1 \le u \le d_2\}$.



$$V_{(i,\,j)k} = \frac{\int_{U_S} f(u)\,du}{\int_{U_S \cup U_X} du}$$

22. *Opu* = “≤≤“ and *Opd* = “=“:



$$V_{(i,j)k} = \begin{cases} 0 & d_2 \le u \quad \text{or} \quad u \le a_1 \\ w & otherwise \end{cases},$$

where w is a fixed small value.

23. Opu = "≤≤" and Opd = "≠":

Let $f(u) = \begin{cases} \mu_{X1}(u) & a_1 \le u < b_1 \\ 1 & b_1 \le u < c_2 \\ \mu_{X2}(u) & c_2 \le u < d_2 \\ 0 & otherwise \end{cases}$,

$U_X = \{u \mid a_1 \le u \le d_2\}$.

$$V_{(i,j)k} = \frac{\int_{U_X} f(u)du}{\int_U du}$$

24. Opu = "≤≤" and Opd = "≥":

Let $f(u) = \begin{cases} \mu_{X1}(u) & a_1 \le u < b_1 \\ 1 & b_1 \le u < c_2 \\ \mu_{X2}(u) & c_2 \le u < d_2 \\ 0 & otherwise \end{cases}$,

$U_X = \{u \mid a_1 \le u \le d_2\}$, and $U_S = \begin{cases} \{u \mid u \ge b_1\} & b_1 \le S \le c_1 \\ \{u \mid u \ge S\} & otherwise \end{cases}$.

$$V_{(i,j)k} = \frac{\int_{U_S \cap U_X} f(u)du}{\int_{U_S \cup U_X} du}$$

25. Opu = "≤≤" and Opd = "≤":

Let $f(u) = \begin{cases} \mu_{X1}(u) & a_1 \le u < b_1 \\ 1 & b_1 \le u < c_2 \\ \mu_{X2}(u) & c_2 \le u < d_2 \\ 0 & otherwise \end{cases}$,

$U_X = \{u \mid a_1 \le u \le d_2\}$, and $U_S = \begin{cases} \{u \mid u \le c_2\} & b_2 \le S \le c_2 \\ \{u \mid u \le S\} & otherwise \end{cases}$.

$$V_{(i,j)k} = \frac{\int_{U_S \cap U_X} f(u)du}{\int_{U_S \cup U_X} du}$$