Accepted by *IEEE Transactions on Knowledge and Data Engineering*, 1998.

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.edu.sg

* Kent Ridge Digital Labs
11 Science Park Road
Singapore Science Park 2
Singapore 117685, Republic of Singapore
hingyan@krdl.org.sg

Correspondence: Bing Liu.

Abstract

One of the major problems in the field of knowledge discovery (or data mining) is the interestingness problem. Past research and applications have found 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 them 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 rank them according to their interestingness. In this paper, we propose such a technique, called the *user-expectation* method. In this technique, the user is first asked to provide his/her expected patterns according to his/her past knowledge or intuitive feelings. Given these expectations, the system uses a fuzzy matching technique to match the discovered patterns against the user's expectations, and then rank the discovered patterns according to the matching results. A variety of rankings can be performed for different purposes, such as to confirm the user's knowledge and to identify unexpected patterns, which are by definition interesting. The proposed technique is general and interactive.

Index Terms:

Knowledge discovery, interesting patterns, unexpectedness,
post-analysis of patterns, pattern ranking

Acknowledgments:

We would like to thank Hwee-Leng Ong, Angeline Pang, King-Hee Ho, Poh-San Lai, Kok-Seng Pun, Kheng-Cheong See, Gui-Jun Yang, Shu Chen and 6 of our undergraduate project students for providing us data, testing our system and giving us feedback. This project is funded by Singapore National Science and Technology Board.

Finding Interesting Patterns Using User Expectations

Abstract

One of the major problems in the field of knowledge discovery (or data mining) is the interestingness problem. Past research and applications have found 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 them 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 rank them according to their interestingness. In this paper, we propose such a technique, called the *user-expectation* method. In this technique, the user is first asked to provide his/her expected patterns according to his/her past knowledge or intuitive feelings. Given these expectations, the system uses a fuzzy matching technique to match the discovered patterns against the user's expectations, and then rank the discovered patterns according to the matching results. A variety of rankings can be performed for different purposes, such as to confirm the user's knowledge and to identify unexpected patterns, which are by definition interesting. The proposed technique is general and interactive.

Index Terms:

Knowledge discovery in databases, interesting patterns, unexpectedness,
post-analysis of patterns, pattern ranking

1. Introduction

It has long been recognized in the knowledge discovery research and applications that a discovery system can produce a glut of patterns from databases, most of which are not useful or interesting to the user [7, 8, 16, 22, 23, 24, 28, 33]. Due to the huge number of patterns produced, it is difficult for a user to focus on the "right" subset of patterns that are interesting to him/her. This problem is known as the "interestingness" problem [8, 22, 28, 32, 33]. In order to prevent the user from being overwhelmed by the large number of patterns, techniques are needed to identify the interesting patterns and to rank them according to their degrees of interestingness.

This interestingness problem has been discussed in many papers [e.g., 7, 8, 13, 16, 22, 23, 24, 28, 32, 33]. Many measures of interestingness have also been proposed. They include: coverage [30], confidence [24, 30], strength [5], performance [24], statistical significance [24], simplicity [24], unexpectedness [32, 22], and actionability [28]. The first five measures are the so-called objective measures [32, 33, 28]. They can be handled with techniques that do not require application or domain knowledge. They have been studied extensively in the literature [e.g., 1, 5, 11, 24]. The last two measures (unexpectedness [32], and actionability [28]) are called the subjective measures [32]. It has been noted in [28] that although objective measures are useful, they are insufficient in determining the interestingness of a discovered pattern. 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, in particular, unexpectedness [32, 22, 23]. It is assumed that some knowledge discovery techniques have performed the pattern discovery task and have filtered out those patterns that do not meet the objective requirements. For convenience, we will use "interestingness" and "subjective interestingness" interchangeably in this paper.

Most existing knowledge discovery systems do not provide any component for helping the user to identify interesting patterns [e.g., 1, 5, 8, 10]. Those that do [e.g., 28, 24] handle subjective interestingness in application/domain-specific fashions, and thus are not applicable to other domains and applications. More details about the existing work will be discussed in Section 6.

In this paper, we focus on the issue of unexpectedness and propose a general technique to help the user identify unexpected patterns, which are by definition interesting [32]. In order to identify unexpected patterns, obviously we must know what are expected or what the user knows or what the user believes. In other words, the user must let the system know what he/she expects to find from the database. By comparing the user-expected patterns with the newly discovered patterns, we can rank the newly discovered patterns according to their degrees of unexpectedness. This is the key point of the proposed technique. We name it the *user-expectation* method. In this technique, the user is first asked to supply a set of expected patterns. A fuzzy matching algorithm is then used to analyze and rank the discovered patterns. Discovered patterns can be ranked according to how well they conform to the user's expectations or how different they are from the user's expectations. With such rankings, the user can simply check the few patterns at the top of the list to confirm or to deny his/her intuitions (or previous knowledge), and to find those patterns that are against his/her expectations (i.e., unexpected patterns). Naturally, the proposed technique may also be used to periodically analyze the deviations of the newly discovered patterns against the old patterns. This can be done by simply using the old patterns as the user-expected patterns.

In this paper, we focus on analyzing and ranking classification patterns (or rules) [30] produced by systems such as C4.5 [30]. For other types of patterns, such as associations, sequence patterns and deviations, etc., the detailed formulas presented in the paper need to be customized because different types of patterns have different meanings and thus require different matching techniques. However, we believe the idea of using the user's expectations to help him/her identify unexpected patterns is general. The proposed technique for analyzing classification patterns is also domain and application independent, and highly interactive because the user can add and modify expected patterns easily, and fuzzy matching and ranking can be done incrementally and efficiently.

The paper is organized as follows: Section 2 defines the interestingness problem. Section 3 gives a high level view of the proposed technique. Section 4 presents the detailed computations. Section 5 evaluates the proposed technique. Section 6 discusses the related work and Section 7 concludes the paper.

2. Problem Definition

Let D be the database. A knowledge discovery (or data mining) technique T is applied to D to discover a set of patterns. A subset B of the patterns also satisfies user-specified objective requirements such as coverage, confidence [24], strength [5] and statistical significance [24], etc. Those patterns that do not satisfy these objective interestingness measures are discarded. However, not all patterns in *B* are subjectively interesting to the user. Let us denote *I* as the set of subjectively interesting patterns in *B*, then we have

$$I \subseteq B$$

Three more points to be noted:

- Given the same database *D* and the discovered patterns *B*, different users may be interested in different subsets of the patterns in *B*.
- Not all patterns in *I* are equally interesting. Different patterns may have different degrees of interestingness to the user.
- *I* may be a dynamic set in the sense that the same user may be interested in different things at different points in time.

In general, *B* is much larger than *I*. This implies that many patterns discovered by *T* are uninteresting or useless. It is desirable that a system only gives the user the set of interesting patterns, *I*, and ranks the patterns in *I* according to their degrees of interestingness. Hence, we define the interestingness problem as follows:

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

In order to identify *I* from *B* and rank the patterns in *I*, we need some concrete measures. Past research have identified two key measures of subjective interestingness, i.e., unexpectedness [32] and actionability [28]. Let us review them here.

1. **Unexpectedness:** Patterns are interesting if they "surprise" the user.
2. **Actionability:** Patterns are interesting if the user can do something with them to his/her advantage. In other words, patterns are useful when they can help to achieve some current goals of the user.

[8] also mentions another subjective measure of interestingness based on novelty of patterns, but does not explore this concept in detail. In this work we mainly deal with unexpectedness, i.e., helping the user identify unexpected patterns.

Note that the above definition of unexpectedness is informal. The detailed definitions of various unexpectedness in the context of classification patterns are presented in Section 4. For a more theoretical definition of the general unexpectedness with respect to a belief system, please refer to [33].

Before presenting our technique, let us describe the type of patterns that we deal with. In this paper we concentrate on developing the user expectation method for classification patterns (or rules) as produced by induction systems such as C4.5 [30]. Let us now show the syntax of the rules generated by C4.5 (since classification patterns are common called classification rules, so we will use these two terms interchangeably from now on):

If $P_1, P_2, P_3, ..., P_n$ **then** C (or $P_1, P_2, P_3, ..., P_n \rightarrow C$)

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

attr OP value

where *attr* is the name of an attribute in the database (or more precisely in a single relation of a 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 form:

Class = *value*.

3. The User-Expectation Technique

This section gives a high level view of the proposed user-expectation technique. The detailed computation formulas will be presented in the next section.

3.1. The overall approach to knowledge discovery

Our proposed method is an interactive post-analysis technique in the sense that it helps the user analyze the patterns discovered by a knowledge discovery system (KDS) in order to identify those interesting ones. It does not affect the knowledge discovery process. Before presenting our proposed method, we first describe where our method fits in the overall knowledge discovery (or data mining) process.

Figure 1(a) shows the typical existing approach to knowledge discovery. It has no tool available to help the user identify the interesting patterns. Figure 1(b) shows the use of an interestingness filter at the back-end of a KDS to filter out those uninteresting patterns (both Figure 1(a) and Figure 1(b) are adopted from [32]). The approach in Figure 1(c) is proposed in [32]. This approach uses an interestingness engine to guide the discovering process so that only the interesting patterns are discovered in the first place. We propose the Figure 1(d) approach, which is a generalization of the one in Figure 1(b).

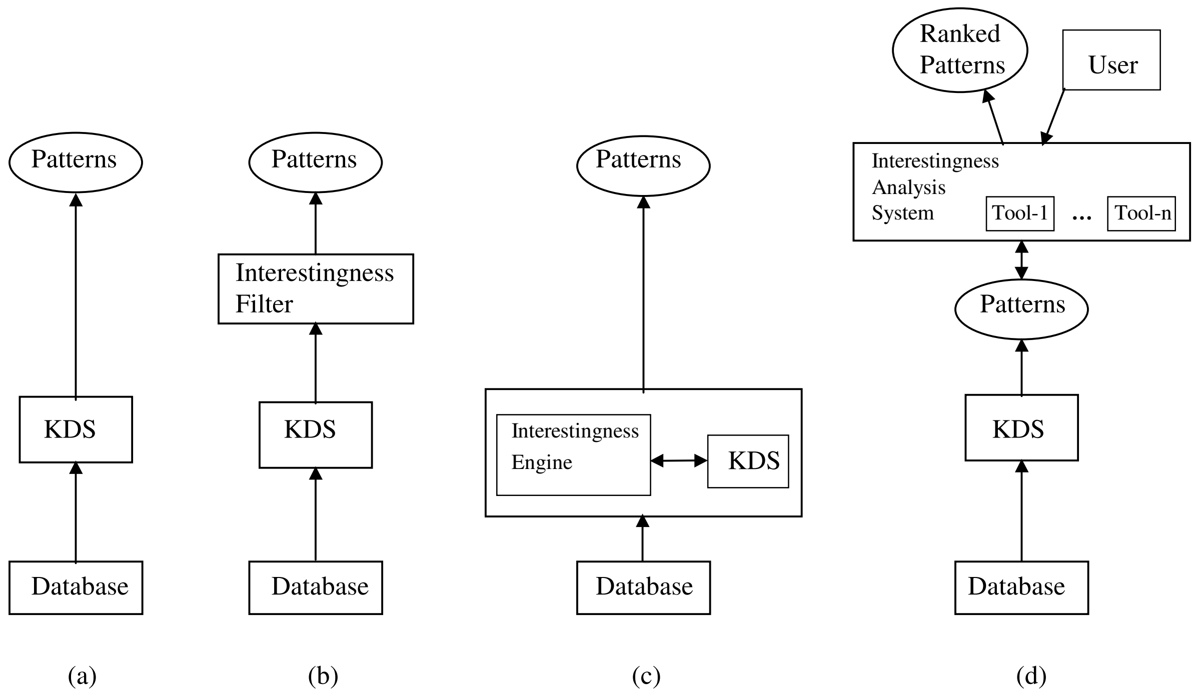


Figure 1. Approaches for Knowledge Discovery

Below, we discuss some of the problems associated with the approaches in Figure 1(a), (b) and (c):

- The approach in Figure 1(a) suffers from the drawback of overwhelming the user with a large number of uninteresting patterns [8, 32].
- The approach in Figure 1(b) is more practical and useful. When a set of patterns is discovered, a filtering system analyzes the discovered patterns to filter out those uninteresting ones. However, this approach does not emphasize user's involvement and user's interaction with the system, which we believe are crucial for practical applications (see the discussion on Figure 1(c)). It should also allow multiple filters that can work independently and/or in parallel to identify different types of interesting patterns.
- The approach in Figure 1(c) is perhaps the most attractive approach of all because it discovers only the interesting patterns in the first place (and/or avoids generating many non-interesting patterns [32]). It is also potentially efficient because domain knowledge and the user's interests can be used to prune the search space [32]. Although theoretically attractive, it suffers from a major problem, i.e., the problem of *knowledge acquisition*.

  In general, it is very difficult, if not impossible, to obtain from the user the complete set of existing knowledge that he/she has about the domain and his/her interests at a particular moment. This is the well-known problem of *knowledge acquisition* in expert systems [4, 6, 9, 31]. The difficulties of knowledge acquisition are well documented in artificial intelligence and expert systems literature [e.g., 4, 6, 9, 31]. Typically, the user (or the expert) may know a great deal, but it is very hard for him/her to tell what he/she knows consistently, let alone to tell everything he/she knows in one go. Due to this problem, using an interestingness engine may not be practical in many situations because if the user does not provide the complete set of knowledge at the beginning, the discovered interesting patterns will not be complete.

  Typically, the user could only supply a small subset of his/her knowledge at the beginning. After seeing some discovered patterns, he/she is then reminded of more existing knowledge that he/she has. He/she then incrementally and interactively adds the new knowledge in and/or modifies the knowledge already given. Through this iterative process, a more complete set of existing knowledge is built up. This is analogous to the iterative knowledge acquisition process in expert systems development [4, 6, 9]. However, this incremental and iterative process of providing the existing knowledge by the user is not compatible with the approach in Figure 1(c) because it is not efficient for the KDS to execute whenever the user remembers another piece of knowledge and/or modifies an piece of already given knowledge since a KDS is, in general, computationally intensive.

  Another problem with the approach in Figure 1(c) is that when a number of users with different existing knowledge and different interests, wish to find the interesting patterns, the KDS will need to be run many times by each user. It will be better to find the complete set of patterns first and then let each user analyze them according to his/her interests.

  However, the Figure 1(c) approach is ideal if the user is able to provide the complete and consistent set of existing knowledge about the domain and his/her interests.

Our proposed approach in Figure 1(d) uses a post-analysis component, called Interestingness Analysis System

(IAS) to help the user identify interesting patterns. It is a generalization of the approach in Figure 1(b), and it has the following main features:

1. The KDS is only run once to discover the patterns in the database. These discovered patterns can be used by any number of users and for any number of times to identify the interesting patterns according to individual needs.
2. The IAS consists of a set of interactive tools to help the user incrementally and interactively analyze the discovered patterns. The user-expectation method proposed in this paper is one of such tools. The interactive process is the key to allow the user to incrementally supply a more and more complete set of existing knowledge.
3. Each tool in IAS ranks the discovered patterns according to one interestingness measure so that the user can identify the interesting patterns easily. Each of the tools is domain and application independent.

We believe that this approach is more practical for real-world applications where the complete set of existing knowledge is not well-defined and/or not easily available. In a particular application, if the approach of Figure 1(c) can be easily achieved with satisfactory results, then it should be used.

The difference between our approach in Figure 1(d) and the approach in Figure 1(b) is that our approach generalizes the one in Figure 1(b). Instead of using a filter to filter out those uninteresting rules, our framework uses a set of interactive and domain independent tools to help the user analyze the discovered patterns in order to identify those interesting ones. User's involvement and interactive pattern analysis are the key features of our approach. Furthermore, it allows parallel application of different tools on the discovered patterns. Using multiple tools is also described in [35]. Some operations on combining different analysis results are proposed in [35].

The main difference between the proposed approach and the approach in Figure 1(c) is that each analysis using an IAS tool is typically much more efficient than running the KDS because the number of discovered patterns is much smaller than the size of the database, and an IAS tool is typically more efficient than the KDS.

3.2. Overview of the proposed technique

To identify unexpected patterns, obviously we must know what the user expects or believes. With this set of beliefs or expectations as input, we can evaluate the conformity of the discovered patterns with respect to this set of user-expected patterns in order to help the user find those unexpected patterns. This is the essence of the proposed technique. We now briefly discuss the steps involved in this technique.

Step1. The user is asked to provide a set of patterns E that he/she expects to find in the database D. These user-expected patterns are regarded as fuzzy patterns. A fuzzy pattern has the same syntax as the discovered pattern, but its attribute values must be described using some fuzzy linguistic variables [37]. See the definition below.

Step2. The system matches each discovered pattern $B_i \in B$ against each expected pattern $E_j \in E$ in order to obtain the degree of match for each discovered pattern B_i against the set E. The discovered patterns in B are then ranked according to their degrees of match with E. Four matching algorithms are proposed to perform matching and ranking for different purposes for helping the user to find conforming patterns and unexpected patterns.

Note that this technique does not identify the conforming and unexpected patterns, but only ranks them according to the conformity and unexpectedness measures. The final job of identifying the conforming patterns and unexpected patterns is left to the user as the system does not know what degree of matching of two patterns can be considered conforming or unexpected.

3.3. Representation of user-expected patterns

For the proposed technique to work, we first need to represent the user's expectations. A simple observation shows that when the user is looking for one type of patterns using a KDS, then his/her expected patterns are also of the same type. For example, if one is looking for classification patterns from a database using a KDS, then one's expectations are automatically classification patterns, rather than associations or sequence patterns.

Since the discovered patterns and the user-expected patterns are of the same type, the user's expectations can be naturally represented in the same way (or format) as the discovered patterns. This, however, does not mean that the discovered pattern format can represent all types of existing knowledge that the user may have about the domain. It is possible that some other types of user knowledge may allow different types of interesting patterns to be identified. For example, in an application of discovering classification patterns, the user may have a piece of existing knowledge or intuition that two attributes X and Y in the database are functionally related, e.g., $X = Y + 5$. However, the KDS discovers the following two patterns:

If $X > 5$ **then** *Class* = *Yes*, and **If** $Y > 10$ **then** *Class* = *No*.

These two patterns are clearly unexpected because they contradict the user's intuition, i.e., they indicate that X and Y are not functionally related. However, the functional relation $X = Y + 5$ cannot be represented as an expected classification pattern. In general, it is difficult to enumerate all the possible situations. Representing and using various types of human knowledge is still a very difficult problem in Artificial Intelligence [31]. There is no one general method that can represent all types of knowledge. That is also why we advocate the approach in Figure 1(d), i.e., having a set of tools to allow the user to express various types of expectations (and/or existing knowledge) and to analyze the discovered patterns in various ways to identify those known (or frequently used) types of interesting patterns.

In this paper, we only focus on analyzing discovered patterns with respect to a set of user-expected patterns represented in the same format as the discovered patterns, i.e., classification patterns. In our implemented system, we also have other tools to handle situations where the user wishes to express expected coverage or confidence of a pattern. These situations are handled using objective interestingness analysis tools.

Another important issue in representing the user's expectations is that the human knowledge or hypotheses are typically imprecise. Thus, some kind of fuzzy representation is required. Fuzzy linguistic variables in fuzzy sets theory fits in naturally. The proposed technique adopts this representation.

Before we proceed, let us review the definition of a fuzzy linguistic variable [37].

Definition: A fuzzy linguistic variable is a quintuple $(x, T(x), U, G, \tilde{M})$ in which x is the name of the variable; $T(x)$ is the term set of x; that is, the set of names of linguistic values of x, with each value being a fuzzy variable denoted generally by x and ranging over a universe of discourse U; G is a syntactic rule for generating the name, X, of values of x; and M is a semantic rule for associating with each value X its meaning, $\tilde{M}(X)$ which is a fuzzy subset of U. A particular X is called a *term* [37].

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:

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

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

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

This may look complicated. However, the user only needs to input (1) his/her expected patterns, and (2) the fuzzy set $\tilde{M}(X)$ for each term X used in his expected patterns. A window-based user interface has been implemented to simplify this input process. After the user has specified the semantic rules associated with the fuzzy linguistic variables, the fuzzy matching process can begin.

3.4. An example

Let us consider an example of how the proposed technique may work in practice. Suppose we have the following set of discovered (classification) patterns from a traffic accident database.

Pattern 1 **If** *P_Age* > 50, *Loc* = *straight* **then** *Class* = *slight*

Pattern 2 **If** *P_Age* > 65, *Loc* = *bend*, *Speed* > 50 **then** *Class* = *killed*

Pattern 3 **If** *P_Age* > 50, *Loc* = *T-junct* **then** *Class* = *slight*

The user-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 the term "*OLD*", "*BAD_VISIBILITY*" and "*BAD_ACCIDENT*." Having specified the semantic meanings, a matching algorithm is then executed to determine the degrees of match between the discovered patterns and the user-expected pattern. Different ranking algorithms are used for different purposes. If our purpose is to confirm the user's hypothesis, the system will rank the discovered patterns accordingly, i.e., the pattern with the highest degree of match is ranked first. The result of such a ranking could be as follows:

Rank 1 Pattern 2. **If** *P_Age* > 65, *Loc* = *bend*, *Speed* > 50 **then** *Class* = *killed*

Rank 2 Pattern 3. **If** *P_Age* > 50, *Loc* = *T-junct* **then** *Class* = *slight*

Rank 3 Pattern 1. **If** *P_Age* > 50, *Loc* = *straight* **then** *Class* = *slight*

It shows that discovered pattern 2 confirms the user's belief that an old person involved in an accident at a bad location will result in a serious injury. In addition, it tells the user that *speed* also plays a role.

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:

Rank 1 Pattern 3. **If** *P_Age* > 50, *Loc* = *T-junct* **then** *Class* = *slight*

Rank 2 Pattern 2. **If** *P_Age* > 65, *Loc* = *bend*, *Speed* > 50 **then** *Class* = *killed*

Rank 3 Pattern 1. **If** *P_Age* > 50, *Loc* = *straight* **then** *Class* = *slight*

This shows that discovered pattern 3 is against the user's expectation because instead of a serious injury, the old person suffers a slight injury (*T-junct*s are considered to have bad visibility). 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. The unexpectedness of a pattern actually can be described in a number of ways (see Section 4.1).

We see from this example that by determining the degrees of match between the discovered patterns and the user-expected patterns, and ranking the discovered patterns accordingly, it is possible to help the user to focus on the appropriate subsets of patterns depending on his/her purpose.

Besides using the user-expectation method to confirm the user's expectations and to find those patterns that are against the user's expectations, this method may also be used to analyze interesting deviations. A typical scenario could be as follows. A company collects data constantly. At the end of every month, a knowledge discovery process is performed to discover patterns in the database collected this month. The newly discovered patterns may be compared with the previously discovered patterns to find deviations. This can be achieved through our user-expectation method by using the previous month's discovered patterns as the expected patterns, and match them with the newly discovered patterns.

4. Fuzzy Pattern Matching Computation

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

Let E be the set of user-expected patterns and B be the set of discovered patterns. We denote W_i as the degree of match between a discovered pattern $B_i \in B$ and 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).

4.1 Computing $w_{(i,j)}$ 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 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. However, in the case of C4.5, the classification rules generated all have the same attribute name, i.e., *Class*. Expected rules also use the same attribute. Thus, the attribute names always match.

For example, we have

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

E_j: **If** *Build* = *medium*, *Weight* = *middle* **then** *Class* = *fit*.

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

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

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

We now present the formulas for computing $w_{(i,j)}$ and W_i. As mentioned in Section 3, the user-expectation method can be used to rank the discovered patterns for the purposes of confirming the user's hypothesis and of finding unexpected patterns. For these two types of ranking, different formulas are used to compute $w_{(i,j)}$ and W_i.

1. **Conforming pattern ranking:** B_i and E_j are conforming if both the conditional parts of B_i and E_j, and the consequent parts of B_i and E_j are similar.

   Since the user's expectations may not be correct, this ranking allows him/her to verify the validity of his expectations. The $w_{(i,j)}$ value is computed as follows:

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

   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. $Z_{(i,j)}$ computes the degree of match of the consequent parts of B_i and E_j. $w_{(i,j)}$ gives the degree of match of B_i with 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 2):

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

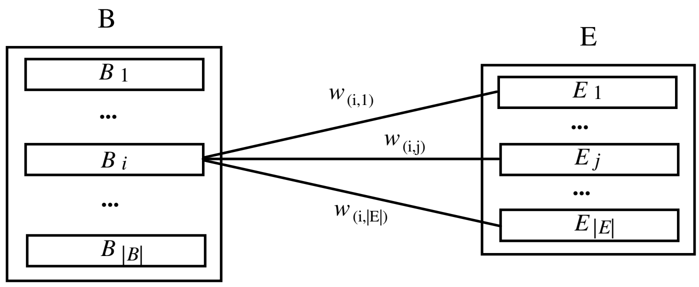


Figure 2. Computing W_i

2. Unexpected pattern rankings

For this purpose, the situation is more complex. We can have a number of ways to rank the discovered patterns in B according to different 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.

This ranking shows the user that his/her knowledge may not be correct because the discovered patterns indicate that for the similar user's assumption (or condition), it leads to a different consequence. This is certainly of interest to the user.

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

Explanation: Since this ranking is for finding those patterns whose conditional parts are similar, but the consequents are contradictory, we need to give a higher $w_{(i,j)}$ value for B_i whose consequent part does not match the consequent part of E_j, hence the expression $Z_{(i,j)}$ - 1.

W_i is computed as follows:

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

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

(a) **Unexpected conditions**: The sets of attribute names in the conditional parts of B_i and E_j overlap to a large extent but their attribute values are quite different. That is, the patterns with $|A_{(i,j)}| > 0$ will be ranked higher.

This ranking serves to tell the user that some attribute value ranges (e.g., *attribute*1 > 5) specified in his/her expected patterns are not quite complete because there are other values of the attributes (e.g., *attribute*1 > 2) that also lead to the same consequences.

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

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 $L_{(i,j)} \times \left(\frac{\sum_{k \in A_{(i,j)}} V_{(i,j)k}}{|A_{(i,j)}|} - 1 \right)$.

W_i is computed as follows:

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

(b) **Unexpected attributes**: The set of attributes used in the conditional part of B_i are unexpected with respect to E_j, i.e., the conditional parts of B_i and E_j share few common attributes.

This ranking allows the user to see that there are unexpected attributes that affect the outcome (or the consequence). In other words, there exist some factors that influence the outcome but the user is not aware of these factors. The computation is more complex in this case because B_i's conditional attributes may be unexpected with respect to E_j, but match E_k very well. In this case, B_i's conditional attributes are not unexpected. To deal with this situation, we compute two match values, i.e., $w_{a(i,j)}$ and $w_{b(i,j)}$.

$$w_{a(i,j)} = 2Z_{(i,j)} - Z_{(i,j)}L_{(i,j)}$$

$$w_{b(i,j)} = Z_{(i,j)}L_{(i,j)}$$

Explanation: A higher value is given to $w_{a(i,j)}$ when the attributes of the conditional parts of B_i and E_j do not match well (but their consequents match well). On the contrary, a higher value is given to $w_{b(i,j)}$ when the attributes of the conditional parts of B_i and E_j do match well and so do their consequents. As mentioned, we cannot simply use $w_{a(i,j)}$ alone to decide the unexpected attributes match. Thus, the unexpected attributes match value W_i is computed in the way as shown below. The purpose is to rank those patterns whose attributes are unexpected to some expected patterns and do not match any other expected patterns.

W_i is computed as follows:

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

It should be stated that we do not claim that the above formulas are the best formulas. We believe that it is unlikely that there exists a set of optimal formulas that can produce rankings agreed upon completely by everyone because the rankings that we are dealing with are inherently subjective. In the process of designing the appropriate formulas, we have tried a number of alternatives. From the results of our experiments, we have selected the set of formulas presented here whose rankings our human subjects agree to be the most conforming to their intuition. Details of our experiments are in Section 5.

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

For the computation of $V_{(i,j)k}$ and $Z_{(i,j)}$, we need to consider both the attribute values and the operators. In addition, the attribute value types (discrete or continuous) are also important. Since the computation of $V_{(i,j)k}$ and

$Z_{(i,j)}$ are the same, it suffices to just consider the computation of $V_{(i,j)k}$, the degree of match for the kth matching attribute in $A_{(i,j)}$. Two cases are considered: the matching of discrete attribute values and the matching of continuous attribute values.

4.2.1 Matching discrete values

When an attribute takes only discrete values, the semantic rule for each term (X) used in describing the expected pattern 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 k. For each $u \in U_k$, the user needs to input the membership value of u in X, denoted as $\mu_X(u)$.

For example, the user gives the following pattern (or rule):

If *Education_level* = *low* **then** *Class* = *reject*

Here, *low* is a fuzzy term. To describe it, the user needs to specify the semantic rule for *low*. Assume the universe (or domain) of the discrete attribute *Education_level* = {*primary*, *secondary*, *university*, *graduate*}. The user may specify that "*low*" *education_level* means:

{(*primary*, 1), (*secondary*, 0.5)}

where the left coordinate is an element in the universe (or domain) of the "*Education_level*" attribute, and the right coordinate is the degree of membership of that element in the fuzzy set *low*, e.g., $\mu_{low}(secondary) = 0.5$.

It is assumed that all the other attribute values not mentioned in the set have the degree of membership 0. The fuzzy term *reject* can also be similarly described.

When performing the matching, these semantic rules and operators used in the propositions play an important part. In the discrete case, only two operators are possible, i.e., "=" and "≠". Assuming the propositions from a user-expected pattern and a discovered pattern are:

User-expected: *attr Opu X*

Discovered: *attr Opd S*

There are four possible combinations. The formulas for the four cases are listed below.

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

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

Case 2. *Opu* = "=" and *Opd* = "≠":

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

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

Case 3. *Opu* = "≠" and *Opd* = "=":

$$V_{(i,\,j)k} = \mu_{\neg X}(S), \text{ where } \mu_{\neg X}(S) = 1 - \mu_X(S).$$

Case 4. *Opu* = "≠" and *Opd* = "≠":

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

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

4.2.2 Matching continuous values

When an attribute takes continuous values, the semantic rule for the term (X) takes 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 3, which our users find reasonable and reflect their fuzzy concepts. Thus, the user merely needs to provide the values for a, b, c, and d.

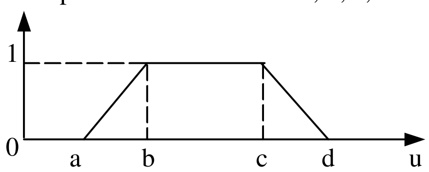


Figure 3. Membership function

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 instance, *young* may be expressed by a = 15, b = 20, c = 30, and d = 35.

For continuous values, the operator used by a user-expected pattern can be any one of the set {=, ≠, ≥, ≤, ≤≤}. "≤≤" is used to represent a range like this: $X_1 \leq attr \leq X_2$. Notice that ">" and "<" are not included. This is because as far as our matching algorithm is concerned, the user can always use a fuzzy proposition with "≥" (or "≤") to express one with ">" (or "<"). Thus, the total number of possible operator combinations is 25. The computation formulas are listed in the appendix.

5. Evaluation

The proposed technique is implemented in Visual C++ on a PC with a graphical user interface. We have conducted a large number of experiments to evaluate its performance. An example test run and some simple analysis of the ranking results are given below to illustrate its capability. The complexity of the algorithm that implements those formulas and various rankings is also presented.

5.1. System testing

Since the proposed technique deals with subjective interestingness, it is difficult to have an objective measure of its performance. Hence, we have carried out a number of experiments involving our industrial partners and other people to check whether the rankings do reflect people's intuitions of subjective interestingness, in particular, unexpectedness.

The people involved in the experiments are 6 undergraduate students, 3 postgraduate students and 5 members from our industry partners. All of them are not involved in the development process of the project. The experiments are conducted on the system using patterns discovered from two real-life databases contributed by our industry partners and one public domain database. The public domain database used is the credit screening database created by Chiharu Sano in UCI machine learning repository. This database has 10 attributes (5 continuous and 5 nominal attributes), and 125 tuples. The number of generated rules is 7. This database is very small. It is mainly used for initial practice by the subjects. We are unable to use many public domain databases in our experiments because most databases in UCI machine learning repository are hard to understand. For our experiments, the subject must be able to understand the domain and the database. Otherwise, it will be very hard for him/her to provide any expectations. The two real-life databases are a disease database and an accident database. The disease database has 10 attributes (7 continuous and 3 nominal attributes) and 713 tuples. The number of discovered rules is 21. This number is just right for the subjects to check carefully whether each ranking is indeed intuitive. The accident database has 12 attributes (4 continuous and 8 nominal attributes) and 3579 tuples. The number of rules generated is 183. This number of 183 gives a flavor of a large number of discovered rules such that the subjects could only check the top few rules in each ranking to find those interesting ones.

Below, we report some observations of the experiments.

- Different subjects have different expectations, and they are also interested in different things.
- Just as expected, it is difficult for a subject to list all his/her expected patterns at the beginning. It is only through an interactive and iterative process that the set of expected patterns is built up. Normally, a subject starts with one or two expected patterns. The system then ranks the discovered patterns according to the one/two expected patterns. After studying the ranked patterns, the subject may derive additional expected patterns. Gradually, he/she builds up this set of expected patterns, which is used to do the final ranking of the discovered patterns. However, some subjects prefer to do ad hoc analysis, i.e., always using one or two expected patterns. The interactive nature of the proposed technique facilitates all these activities.
- Some subjects do occasionally disagree with the relative ranking, e.g., they believe that one pattern should be ranked above another. However, that (i.e., slight different relative ranking) is not a problem. We do expect such minor disagreements because we are dealing with a subjective issue here. The important thing is that everyone agrees that the techniques are able to bring those interesting patterns to the top of the list.

The databases we used in these experiments may not be that large. However, that is not the key point because our proposed technique does not go through the database in its analysis process. What is important is the number of discovered patterns. When the number of discovered patterns is large, manual analysis becomes difficult.

Apart from the above experiments, we ourselves also tested the system used another two public domain databases, i.e., the soybean database (35 attributes and 683 tuples) and the flag database (30 attributes and 194 tuples) in UCI machine learning repository. We and one of our industry partners also tested it using two market survey databases. The first one has 9366 tuples and 30 attributes. The number of rules generated is 95. The other has 10890 tuples and 27 attributes. The number of rules generated from this database is 105. These real-life databases could not be used in the above experiments because they are confidential. The proposed technique has proven to be useful. As a further indication of its usefulness, one of our industry partners has already incorporated our system into their existing data mining and data visualization toolkits.

5.2. A test example

Here, we give a test example. The set of rules (or patterns) is generated from our real-life disease database using C4.5. To ease understanding, only a subset of the rules generated by C4.5 is used for ranking. It is assumed that all the discovered rules satisfy the objective interestingness measures. The discovered rules and the user-expected rules are listed below.

Discovered rules

Rule 1: Age <= 41 -> Disease = NO
Rule 2: Age <= 49, LDL <= 5.49, DBP > 60 -> Disease = NO
Rule 3: Age > 49, Sex = FEMALE -> Disease = YES
Rule 4: Age > 49, Age <= 50 -> Disease = YES
Rule 5: Age > 55 -> Disease = YES
Rule 6: Age > 41, LDL > 5.49, SBP > 106, DBP > 60, GLUC <= 5.06 -> Disease = YES
Rule 7: Age > 41, DBP <= 60 -> Disease = YES
Rule 8: Age > 41, Age <= 47, LDL <= 3.91, SBP > 106, DBP > 60, GLUC <= 5.06 -> Disease = YES

User-expected rules (with a fuzzy set attached to each term)

Rule 1:	SBP >= 150,	{a = 147, b = 148, c = 152, d = 153}
	DBP >= 95	{a = 92, b = 93, c = 97, d = 98}
	-> Disease = YES	{(NO, 0), (YES, 1)}
Rule 2:	LDL >= 2,	{a = 1.75, b = 1.8, c = 2.2, d = 2.25}
	DBP >= 60	{a = 50, b = 55, c = 65, d = 68)
	-> Disease = YES	{(NO, 0), (YES, 1)}

The ranking results are listed below. Note that no matching value (i.e., W_i) is attached to each rule in the rankings because although the rankings are effective, the absolute degree of conformity and/or unexpectedness of each rule can be quite subjective, and the value can be confusing to the user.

- **Conforming rule ranking**

RANK 1 Rule 6: Age > 41, LDL > 5.49, SBP > 106, DBP > 60, GLUC <= 5.06 -> Disease = YES
conforming to user specified rule 2 and 1.

RANK 2 Rule 8: Age > 41, Age <= 47, LDL <= 3.91, SBP > 106, DBP > 60, GLUC <= 5.06
-> Disease = YES
conforming to user specified rule 2 and 1.

RANK 2 Rule 7: Age > 41, DBP <= 60 -> Disease = YES conforming to user specified rule 2.

The rest of the rules are cut off because of their low matching values, i.e., there is 0 match either on the conditional part or the conclusion part.

From this ranking, we can see that discovered rule 6 and 8 conforms to user-expected rule 1 and 2 to certain extent.

- **Unexpected consequent rule ranking**

RANK 1 Rule 2: Age <= 49, LDL <= 5.49, DBP > 60 -> Disease = NO
contradicting user specified rule 2

The rest of the rules are cut off because of their low matching values, i.e., there is 0 condition match or there is a complete conclusion match.

We see that discovered rule 2 is unexpected with respect to user-expected rule 2 because to a large extent the conditional parts of the two rules match, but their consequents are completely opposite. This also allows the user to realize that *Age* plays an important role in determining whether a person has the disease. There is no unexpected rule for user-expected rule 1, which means the discovered rules that satisfy its conditions always have the consequent Disease = YES.

- **Unexpected conditions rule ranking**

RANK 1 Rule 7: Age > 41, DBP <= 60 -> Disease = YES
contradicting user specified rule 2

RANK 2 Rule 6: Age > 41, LDL > 5.49, SBP > 106, DBP > 60, GLUC <= 5.06 -> Disease = YES
contradicting user specified rule 1

RANK 3 Rule 8: Age > 41, Age <= 47, LDL <= 3.91, SBP > 106, DBP > 60, GLUC <= 5.06
-> Disease = YES
contradicting user specified rule 1

The rest of the rules are cut off because of their low matching values, i.e., there is 0 match of attributes in conditional part or there is 0 match on the conclusion.

From this ranking, we can see some discovered rules that have the same consequent as the user-expected rules but their conditional parts contradict (rule 7) to or have non-overlapping value ranges (rule 6 and 8) with the user-expected rules. Thus, these conditions are unexpected to the user.

- **Unexpected attributes rule ranking**

RANK 1 Rule 3: Age > 49, Sex = FEMALE -> Disease = YES
RANK 2 Rule 4: Age > 49, Age <= 50 -> Disease = YES
RANK 3 Rule 5: Age > 55 -> Disease = YES
RANK 4 Rule 6: Age > 41, LDL > 5.49, SBP > 106, DBP > 60, GLUC <= 5.06 -> Disease = YES
RANK 5 Rule 8: Age > 41, Age <= 47, LDL <= 3.91, SBP > 106, DBP > 60, GLUC <= 5.06
-> Disease = YES

RANK 6 Rule 7: Age > 41, DBP <= 60 -> Disease = YES

The rest of the rules are cut off because of their low matching values, i.e., there is 0 match in the conclusion part or the whole match value (i.e., W_i) is less than 0.

Clearly, from this ranking we can see that the conditions of these rules are unanticipated with respect to the set of user-expected rules. They serve to inform the user that there are some other attributes other than those mentioned in the expected rules that affect the outcome (whether a person has the disease).

From this example run, we can see that through these rankings and the analysis we can obtain a great deal of knowledge about the discovered rules and the database in relation to the user's previous intuitions. In particular, when the number of rules is large (which is often the case in knowledge discovery applications), the above rankings will be helpful to the user in identifying interesting patterns.

5.3 Efficiency analysis

Finally, let us analyze the runtime complexity of the algorithm that implements the proposed technique. We did not list the algorithm explicitly in the paper, but only the computation formulas involved. The algorithm that realizes those formulas is obvious.

Assume the maximal number of propositions in a pattern (a user-expected pattern or a discovered pattern) is N. Assume the attribute value matching (computing $V_{(i,j)k}$ and $Z_{(i,j)}$) takes constant time (since the fuzzy set specified for each value by the user is normally small). Thus, the computation of $w_{(i,j)}$ is $O(N)$, and the computation of W_i is $O(|E|)$. Without considering the final ranking which is basically a sorting process, the worst-case time complexity of the technique is $O(|E||B|N)$.

The above analysis is for an algorithm that computes $w_{(i,j)}$ and W_i with the complete sets B and E. In fact, the algorithm is naturally incremental because when new expected patterns are added or some existing expected patterns are modified or removed, there is no need to re-compute those $w_{(i,j)}$ whose expected patterns are not modified, but only those $w_{(i,j)}$ with the newly added or modified expected patterns. This requires the system to remember only the previous $w_{(i,j)}$ (and $w_{a(i,j)}$ and $w_{b(i,j)}$) results. When there is only addition of new expected patterns but no modification or removal of the existing expected patterns, the system only needs to remember the previous maximal $w_{(i,j)}$ (and $w_{a(i,j)}$ and $w_{b(i,j)}$) results. After all the patterns in B are matched with the newly added (or modified) expected patterns, W_i can be evaluated again (which is only $O(|E|)$).

6. Related Work

Although the interestingness has long been identified as an important issue in knowledge discovery in databases [8], most of the knowledge discovery techniques and tools only deal with this problem in a limited way. Instead, their primary concern is to discover all the patterns in the given databases [e.g., 1, 5, 10, 30].

Past research has, however, identified many measures to evaluate the degree of interestingness of a discovered pattern. These measures can roughly be classified into two classes: objective measures and subjective measures [33]. Objective measures typically involve analyzing the discovered patterns' structures, predictive performances, and statistical significance. Examples of objective measures are: coverage [30], certainty factor [11], strength [5], statistical significance [24] and simplicity [24].

It has been noted in [28] that objective measures are insufficient for determining the interestingness of the discovered patterns. Subjective measures are needed. Two main subjective measures are: unexpectedness [33, 22], and actionability [28]. While objective interestingness has been well studied and used in practical applications, limited work has been done on subjective interestingness.

[24] 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.

[28] studied the issue of interestingness in 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. For each action, the expert also estimates the probability of success. Once a finding is discovered, the system computes the estimated benefit for taking a 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 other applications.

In contrast, our method is general. It does not make any domain-specific assumptions. A pattern analysis system based on the proposed framework can be attached to each knowledge discovery tool to help the user identify interesting patterns. This paper only focuses on one type of patterns, i.e., the classification patterns. All the matching techniques are designed for this type of patterns. For other types of patterns, the matching techniques may be customized. However, we believe the general idea of using user-expectation to analyze the discovered patterns and to identify unexpected patterns can be applied to other types of patterns. Though domain-specific systems such as KEFIR are still the most effective method for ranking patterns, the cost of building such systems is high. As mentioned, it also suffers from the problem of being difficult to keep up with the changes in the application domain.

[32, 33] proposed to use beliefs and belief revision as the framework for describing subjective interestingness. Specifically, a belief system is used to define unexpectedness. A number of approaches to beliefs are discussed and compared, e.g., Bayesian approach, Dempster-Shafer approach, frequency approach, statistical approach. Two types of beliefs are also proposed, hard beliefs 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. A pattern is unexpected and interesting with respect to the user's hard beliefs if it contradicts the hard beliefs. A pattern is unexpected and interesting with respect to the user's soft beliefs if it changes the degrees of the soft beliefs. The more the pattern "affects" the degrees of soft beliefs, the more interesting the pattern is. However, [32, 33] just presents some proposals. To the best of our knowledge, no practical system has been implemented that utilizes this approach. Our proposed approach has been implemented and tested with both real-life databases and public domain databases.

With regard to the applications of fuzzy sets, the representation of fuzzy rules in this research is basically the same as the typical fuzzy rules used in fuzzy expert systems [19, 14] and/or fuzzy control systems [e.g., 12, 34]. However, the fuzzy expert rules and fuzzy control rules are used for fuzzy inferencing, i.e., given a fact (or a measurement) they can draw some conclusions. This is different from our fuzzy matching technique because our technique compares two rules rather than draw any inference from the rules using given facts. Our fuzzy matching technique is also different from fuzzy pattern recognition [15, 2] (where patterns normally refer to images of various kinds). In fuzzy pattern recognition, a number of features is typically selected and fuzzified to represent each pattern class. A function is then defined based on the fuzzification for classifying unknown patterns. This is clearly different from our work because our technique is not for recognizing any unknown patterns (or images). There are also many other fuzzy sets applications in domains such as fuzzy databases [21], fuzzy information retrieval [27], fuzzy decision making [38], etc. The key idea of these existing fuzzy applications is to use a set of pre-defined fuzzy specifications (expert rules, control rules, features in images, etc.) to arrive at some fuzzy conclusions given a set of facts, which may be facts in the fuzzy expert systems, or measurements in fuzzy control systems, or unknown images. Our application of fuzzy sets is different because our technique matches and compares rules not to draw conclusions but to see whether they conform to or are unexpected with respect to each other.

7. Conclusion

In this paper, we study the subjective interestingness issue in knowledge discovery in databases from a domain independent perspective. A general method for ranking the discovered patterns according to their interestingness (in particular, unexpectedness) is proposed. This method is characterized by asking the user to input his/her expected patterns and then the system ranks the discovered patterns by fuzzy matching them against the expected patterns. This method can be used for confirming user's knowledge, for finding unexpected patterns and for analyzing pattern deviations. The method is efficient and allows the user to identify interesting patterns incrementally and interactively. We do not claim, however, that the problems associated with the interestingness or even the unexpectedness are fully understood. Further research is needed. In the paper, we did not study the actionability issue. We aim to explore it in our future work.

References

[1] R. Agrawal, T. Imielinski, and A. Swami, "Database mining: a performance perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, no. 6, 1993, pp. 914-925.

[2] J. C. Bezdek and S. K. Pal, eds. Fuzzy models for pattern recognition: methods that search for patterns in data. IEEE press, 1992.

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

[4] B. Buchanan and D. Wilkins (eds.). *Readings in knowledge acquisition and learning*. Morgan Kaufmann, 1993.

[5] V. Dhar and A Tuzhilin, "Abstract-driven pattern discovery in databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, no. 6, 1993, pp. 926-938.

[6] J. Durkin. *Expert Systems: design and development*. Prentice Hall, 1994.

[7] U. Fayyad, G. Piatesky-Shapiro & P. Smyth, "From data mining to knowledge discovery: an overview," In: *Advances in knowledge discovery and data mining*, U. Fayyad, G. Piatesky-Shapiro, P. Smyth and R. Uthurusamy, (eds.), AAAI/MIT Press, 1996, 1-34.

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

[9] J. Giarratano and G. Riley. *Expert systems: principles and programming*. PWS Publishing Company, 1994.

[10] J. Han, Y. Cai, and N. Cercone, "Data driven discovery of quantitative rules in relational database," *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, 1993, pp. 29-40.

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

[12] M. Jamshidi, N. Vadiee, and T. Ross, eds. Fuzzy logic and control. Pretice Hall, 1993.

[13] M. Kamber, & R. Shinghal. Evaluating the interestingness of characteristic rules. *KDD-96* (1996) 263-266.

[14] A. Kandel. (ed). Fuzzy expert systems. CRC press, Boca Raton, 1991.

[15] A. Kandel. Fuzzy techniques in pattern recognition. John Wiley, 1982.

[16] M. Klemetinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo, "Finding interesting rules from large sets of discovered association rules," *Proceedings of the First International Conference on Information and Knowledge Management*, 1994.

[17] G. Klir and B Yuan. Fuzzy sets and fuzzy logic: theory and applicaitons. Prentice Hall, 1995.

[18] G. Klir and T. Folger, Fuzzy sets, uncertainty, and information. Prentice Hall, 1988.

[19] R. Kruse, E. Schwecke and J. Heinsohn. Uncertainty and Vagueness in knowledge based systems: numerical methods. Springer-verlag. 1991.

[20] W. Klosgen, Problems for knowledge discovery in databases and their treatment in the statistics interpreter explora," *International Journal of Intelligent Systems*, vol. 7, no. 7, 1992, pp. 649-673.

[21] D. Y. Li and D. B. Liu, A fuzzy Prolog database system, John Wiley, 1990.

[22] B. Liu and W. Hsu, "Post-analysis of learned rules," *Proceedings of the Thirteenth National Conference on Artificial Intelligence* (*AAAI-96*), Portland, Oregon, 1996, pp. 828-834.

[23] B, Liu, W. Hsu, and S. Chen, "Using general impressions to analyze discovered classification rules." *KDD-97*, 1997, pp. 31-36.

[24] J. A. Major, J. J. Mangano, "Selecting among rules induced from a hurricane database," *Proceedings of the AAAI-93 Workshop on Knowledge Discovery in Databases*, 1993, pp. 28-41.

[25] C. J. Matheus, P. K. Chan, and G. Piatetsky-Shapiro, "Systems for knowledge discovery in databases,"

IEEE Transactions on Knowledge and Data Engineering, vol. 5, no. 6, 1993.

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

[27] S. Miyamoto. Fuzzy sets in information retrieval and cluster analysis. Kluwer, 1990.

[28] G. Piatesky-Shapiro and C. J Matheus, "The interestingness of deviations," *Proceedings of the AAAI-94 Workshop on Knowledge Discovery in Databases*, 1994, pp. 25-36.

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

[30] J. Ross Quinlan, *C4.5*: *program for machine learning*. Morgan Kaufmann, 1992.

[31] S. Russell and P. Norvig. *Artificial Intelligence: a modern approach*. Prentice Hall, 1995.

[32] A. Silberschatz and A. Tuzhilin, "On subjective measures of interestingness in knowledge discovery," *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, 1995, pp. 275-281.

[33] A. Silberschatz & A. Tuzhilin, "What makes patterns interesting in knowledge discovery systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, no. 6, 1996, pp. 970-974.

[34] M. Sugeno, eds. Industrial applications of fuzzy control. North-holland, 1985.

[35] A. Tuzhilin, "A pattern discovery algebra," *SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 1997, pp. 71-76.

[36] R. Uthurusamy, U.M Fayyad, and S. Spangler, "Learning useful rules from inconclusive data," *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro and W.J Frawley, eds. AAAI/MIT Press, 1991.

[37] H. -J. Zimmermann, *Fuzzy set theory and its applications*, Second Edition, Kluwer Academic Publishers, 1991.

[38] H. -J. Zimmermann, *Fuzzy sets, decision making, expert systems.* Kluwer Academic Publishers, 1987.

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:

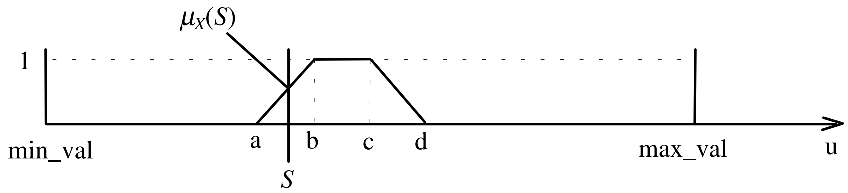
User-expected: $attr\ Opu\ X$ or $X_1 \le attr \le X_2$

Discovered: $attr\ Opd\ S$ or $S_1 \le attr \le S_2$

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.

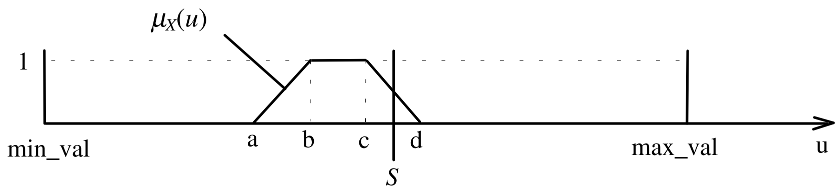
1. $Opu = Opd$ = "=":

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



2. Opu = "=" and Opd = "≠":

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



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

3. Opu = "=" and Opd = "≥":

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

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

4. Opu = "=" and Opd = "≤":

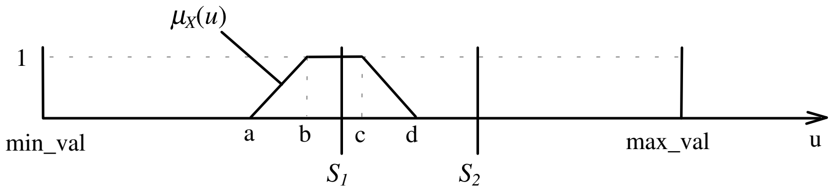
Let $U_X = \{ u \mid a \le u \le d\}$ and $U_S = \begin{cases} \{u \mid u \le c\} & b \le S \le c \\ \{u \mid u \le S\} & otherwise \end{cases}$.

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

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

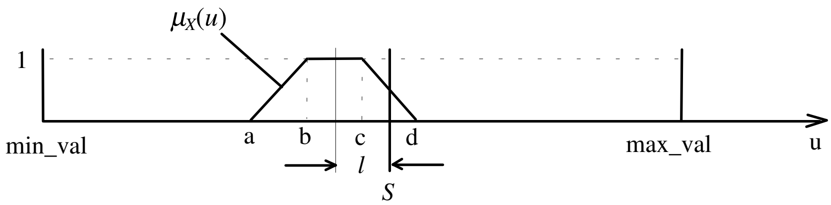
$U_S = \{ u \mid S_1^{'} \le u \le S_2^{'} \}$.



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

6. *Opu* = *Opd* = "≠":

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



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

7. *Opu* = "≠" and *Opd* = "=":

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

where w is small fixed value.

8. *Opu* = "≠" and *Opd* = "≥":

Let $U_S = \begin{cases} \{u \mid u \ge b\} & b \le S \le c \\ \{u \mid u \ge S\} & otherwise \end{cases}$.

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

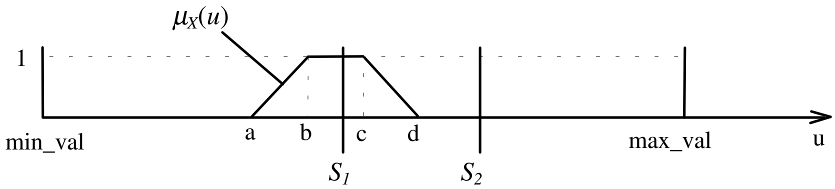
9. *Opu* = "≠" and *Opd* = "≤":

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

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

10. *Opu* = "≠" and *Opd* = "≤≤":

Let $S_1^{'} = \begin{cases} b & b \le S_1 \le c \\ S_1 & otherwise \end{cases}$, $S_2^{'} = \begin{cases} c & b \le S_2 \le c \\ S_2 & otherwise \end{cases}$, and $U_S = \{ u \mid S_1^{'} \le u \le S_2^{'} \}$.

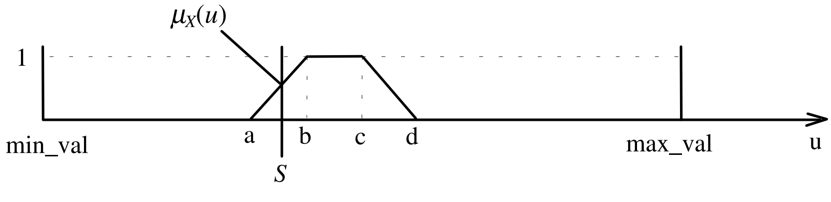


$$V_{(i,j)k} = \frac{\int_{Us} \mu_{\neg X}(u)du}{\int_{U} du}$$

11. *Opu* = *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 $U_S = \begin{cases} \{u | u \ge b\} & b \le S \le c \\ \{u | u \ge S\} & otherwise \end{cases}$.



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

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 $U_S = \begin{cases} \{u \mid u \le c\} & b \le S \le c \\ \{u \mid u \le S\} & otherwise \end{cases}$.

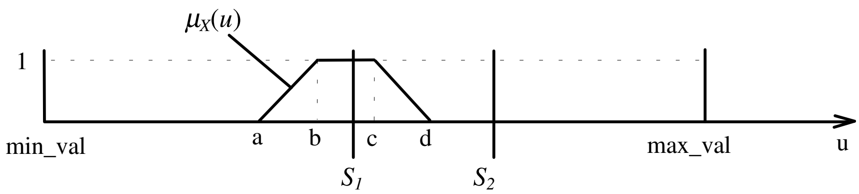
$$V_{(i,\,j)k} = \frac{\int_{U_S \cap U_X} f(u)\,du}{\int_U du}$$

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

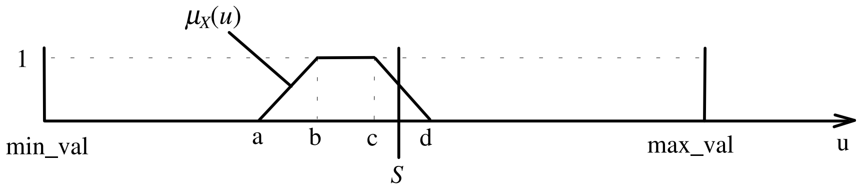


$$V_{(i,\,j)k} = \frac{\int_{U_S} f(u)\,du}{\int_{U_S \cup U_X} du}$$

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

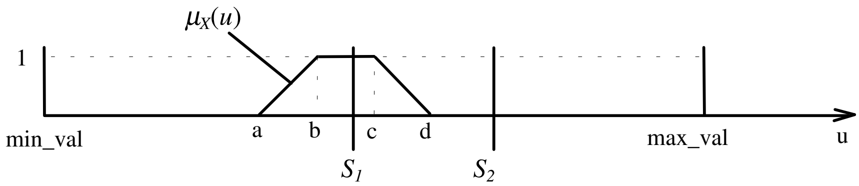
$$V_{(i,\ j)k} = \frac{\int_{U_S \cap U_X} f(u)du}{\int_U du}$$

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



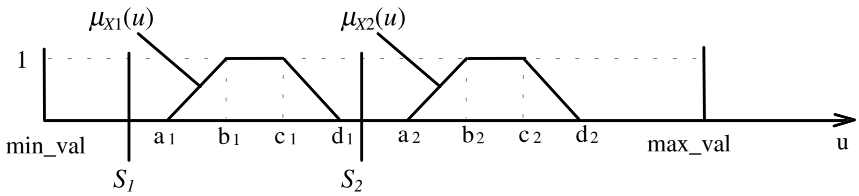
$$V_{(i,j)k} = \frac{\int_{U_S} f(u)du}{\int_{U_S \cup U_X} du}$$

21. *Opu* = *Opd* = “≤≤“:

Let $f(u) = \begin{cases} \mu_{X_1}(u) & a_1 \le u < b_1 \\ 1 & b_1 \le u < c_2 \\ \mu_{X_2}(u) & c_2 \le u < d_2 \\ 0 & otherwise \end{cases}$,

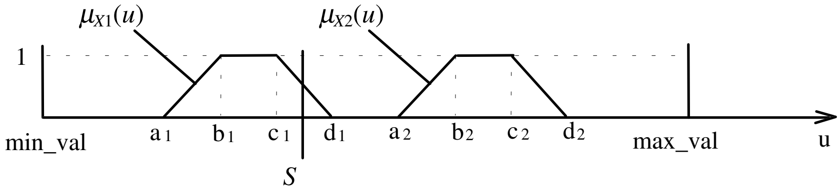
$S_1^{'} = \begin{cases} b_1 & b_1 \le S_1 \le c_1 \\ S_1 & otherwise \end{cases}$, $S_2^{'} = \begin{cases} c_2 & b_2 \le S_2 \le c_2 \\ S_2 & otherwise \end{cases}$,

$U_S = \{u \mid S_1^{'} \le u \le S_2^{'} \}$, 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 $U_S = \begin{cases} \{u | u \ge b_1\} & b_1 \le S \le c_1 \\ \{u | u \ge S\} & otherwise \end{cases}$.

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

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 $U_S = \begin{cases} \{u | u \le c_2\} & b_2 \le S \le c_2 \\ \{u | u \le S\} & otherwise \end{cases}$.

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