Mining Actionable Patterns by Role Models

Ke Wang
Simon Fraser University
wangk@cs.sfu.ca

Yuelong Jiang
Simon Fraser University
yjiang@cs.sfu.ca

Alexander Tuzhilin
New York University
atuzhili@stern.nyu.edu

Abstract

Data mining promises to discover valid and potentially useful patterns in data. Often, discovered patterns are not useful to the user. "Actionability" addresses this problem in that a pattern is deemed actionable if the user can act upon it in her favor. We introduce the notion of "action" as a domain-independent way to model the domain knowledge. Given a data set about actionable features and an utility measure, a pattern is actionable *if it summarizes a population that can be acted upon towards a more promising population observed with a higher utility. We present several pruning strategies taking into account the actionability requirement to reduce the search space, and algorithms for mining all actionable patterns as well as mining the top k actionable patterns. We evaluate the usefulness of patterns and the focus of search on a real-world application domain.*

1 Introduction

1.1 Background

Knowledge discovery in databases is the non-trivial process of identifying *valid, novel*, potentially *useful*, and ultimately *understandable* patterns in data [4]. The literature has mostly focussed on the "valid" and "understandable" measures that can be defined in *objective* terms, such as the confidence/support measure [2] and the size/number of patterns. Due to the lack of modeling user knowledge, discovered patterns often are known or not useful to the user. To model the "novel" and "useful" aspects, *subjective* measures are needed [7, 10]. [10] classified subjective measures into *unexpectedness* and *actionability*. A pattern is *unexpected* if it is surprising to the user. A pattern is *actionable* if the user can act upon it to her advantage.

Despite the efforts of initial works on actionability mining (more details in Related Work), it turned out to be very hard to capture formally the "elusive" nature of actionability in all of its manifestations and across different types of applications and settings. The following factors contribute to the difficulties of this problem. First, there is the dilemma that user knowledge should be acquired to define actionability, but the ability to provide such knowledge tends to invalidate the need for mining actionable patterns. Second, it is very hard to model the "usefulness" of a pattern. Strictly speaking, this information is not available until after patterns are deployed in the real world; on the other hand, the data mining algorithm needs to tell if a pattern is actionable *before* it is deployed. Finally, the search space is likely to be large and a scalable method must exploit the notion of actionability to prune the search as early as possible.

1.2 Our Contributions

We propose an actionability mining approach that addresses three goals. (1) Find the right balance between providing enough domain knowledge and making the problem scalable, tractable and non-trivial. (2) Model actionability in a domain-independent manner and provide a way to estimate the success of actionable patterns. (3) Utilize user knowledge early in the search of actionable patterns. The main idea is as follows. Suppose that a database contains historical observations about several *features* and a success measure called the *utility*. Also, suppose that we know some *actions* can influence features in certain (simple) ways. If some features are correlated with the utility in some population of the data, a change in those features would imply a change in the utility. Now if some actions can influence those features, this influence will cascade to the utility through the correlation. We are interested in the patterns that summarize those actions and populations where such cascaded influences increase the utility. An example explains these points.

Example 1.1 (Phone call charge example) Consider a customer database in the long distance phone call application:

$$C(CustID, Rate, Married, \cdots, U).$$

U is the utility representing the profit generated on a customer. For simplicity, U has the domain $\{Low, High\}$. The feature $Rate$ represents the rate charged to a customer and has the domain $\{Normal, Special\}$, with $Normal$ denoting the normal rate and $Special$ denoting the promotional rate. Suppose that we observed the following two patterns in the data

$$D : Rate = Normal, Married = No \rightarrow U = Low,$$
$$D' : Rate = Special, Married = No \rightarrow U = High.$$

The customers in D' generated a higher profit because they made more calls at the lower rate. Comparing these two patterns reveals that the telephone company can make more profit by taking the action of offering the special rate to the customers in D. The increase is expected because a higher profit was observed on the customers (i.e., D') who shared similar characteristics (i.e., unmarried) but were offered the special rate. ■

The example conveys several points. (1) In many real life applications, the user knows some simple action/feature relationship (e.g., offering special rate can change $Rate$ from $Normal$ to $Special$). (2) The user is interested in finding the action/utility relationship, i.e., how actions would affect the utility. Since several actions and features could be involved in a complex relationship to affect the utility, finding such relationships requires examining both the action knowledge and the data, therefore, is non-trivial. (3) A pattern is actionable if it summarizes a population that can be affected, by taking actions, toward another population observed with a higher utility. In a sense, the second population serves the "role model" of the first one.

Can this problem be solved by a standard method, such as a classification algorithm to learn actions that influence the utility? Unfortunately, standard learning methods cannot easily incorporate the action knowledge. Without this knowledge, these methods can learn the rules that summarize the data, but not the rules that change the state of the data. This comment equally applies to association rule mining [2] and other rules learnt purely from the data. In principle the actionability problem is a learning problem, but the presence of action knowledge and the focus on changes make it a whole new learning problem that standard methods are not designed to handle.

The contributions of this paper are as follows. We introduce the notion of "action" as a domain-independent way to model some important domain knowledge. An action describes the condition under which it applies and the effect it has on a feature. We formulate a new data mining problem where actions are the first class objects and actionable patterns are summarizations of opportunities for actions to boost the utility. We show that mining actionable patterns is at least as hard as mining frequent itemsets of [2]. We present several pruning strategies taking into account the actionability requirement to reduce the search space, and algorithms for mining all actionable patterns as well as mining the top k actionable patterns. We evaluate the usefulness of patterns and the focus of search on a real-world application domain.

In the rest of the paper, we review related work in Section 2, introduce our action model in Section 3, define the actionability problem in Section 4, present mining algorithms in Section 5, and evaluate our approach in Section 6. Finally we conclude the paper.

2 Related Work

Some initial work has been done on actionability mining. [7] defined actionability of a key finding (pattern) in terms of the estimated benefits in taking corrective action(s) that restores the deviation back to its norm. This approach worked well for the specific healthcare application, but could not be generalized to other application domains. A domain independent approach was presented in [1], where a hierarchy of actions was introduced and certain types of patterns are assigned to the nodes in the hierarchy, and actionable patterns were discovered by executing pattern templates at the nodes of the hierarchy.

Kleinberg et al. [6] presented the microeconomic view of data mining. They regarded data mining as an optimization problem where the "utility" of decisions is the objective function: partition the customer base C into k parts $C_1, \cdots, C_k$ to maximize the sum of the optima $\Sigma_1^k max_{x \in D} \Sigma_{i \in C_j} c_i \cdot x$, where $c_i \cdot x$ denotes the utility of a decision x on a customer i. In our setting, however, $c_i \cdot x$ is unknown because it corresponds to the actionability of *increasing* the utility on a customer i by taking a set of actions x. Modeling this notion of actionability is our contribution.

Ras et al. [8] presented an algorithm for finding action rules. They considered *stable* attributes whose values cannot be changed, and *flexible* attributes whose values can be changed. An "action rule" is a pair of regular rules that agree on all stable attributes and differ in some flexible attributes and the utility. They did not model actions nor attributes where actions affect only certain ranges, and could not optimize the set of actions for a better actionability. The search of action rules is done in a post-processing after finding all regular rules.

The action model was first proposed in [5] with the focus on constructing a model for optimizing the actionability for future cases. The work in [12] presented a profit-motivated product recommender approach, in the presence of an inverse correlation between the chance of selling an item and the profit generated from the sales. The work in [13] assumed that the user is able to provide the "cost" for changing a feature value. In practice, obtaining such cost will be the bottleneck. For example, it is difficult for the user to determine the "cost" for converting a customer from the attrition state to the loyal state. A key and non-trivial issue in actionability mining is acquiring the user knowledge in a *scalable* manner without overloading the user. We believe that our action model is an effective way to address this issue.

3 The Action Model

An action model for representing the user knowledge about actions was recently introduced in [5]. Consider a data set stored in a table

$$C(F_1, ..., F_m, U).$$

F_i are *features* and U is the *utility* to be maximized. We consider a *ranked domain* for every feature F_j and the utility U, consisting of a small number of linearly ordered scales, represented by a few ordinals 0,1,... For example, donation scale, skill level, letter grade, performance evaluation, all can be abstracted into a ranked domain. $dom(F_j)$ and $dom(U)$ denote the domain of F_j and U. Note that the average of several utility values can be a decimal. $c[F_j]$ and $c[U]$ denote the value of a case c in C for F_j and U. $|x|$ denotes the number of elements in a set x. $SAT(x)$ denotes the set of cases in C satisfying a condition x.

3.1 The Influence Matrix

Consider several *actions*, denoted $A_1, \cdots, A_n$. An action can influence the values of features $F_1, \cdots, F_m$. The influence is specified by the *influence matrix* $\{E_{ij}\}$, $1 \le i \le n$, $1 \le j \le m$. E_{ij}, the *influence range* of A_i on F_j, has the form $[a, b]$, where

1. $a = b = -$, if A_i has no influence on F_j.
2. $a < b$, if A_i increases the current value f in $[a, b]$ for F_j to some value in the *destination range* $[f, b]$.
3. $a > b$, if A_i decreases current value f in $[a, b]$ for F_j to some value in the *destination range* $[b, f]$.

By default, if A_i has no influence on F_j, or if f is not in $[a, b]$, A_i changes the current value f to the destination range $[f, f]$.

Example 3.1 (Teaching example) We shall use the real life teaching evaluation domain in Figure 1 as our case study. Each F_i represents the score on one aspect of teaching. U represents the overall score. All scores are in the 0-6 discrete scale, the higher the better. Each A_i represents a possible action. For example, if attending the communication workshop (A_3) increases any communication skill level (F_3) f in the range $[1, 5]$ to a new level in $[f, 5]$, we represent this knowledge by the entry $E_{33} = [1, 5]$ in the influence matrix, with the following implications.

Features:		
	F1:	Present course materials in an organized fashion
	F2:	Stimulate student interest in the course
	F3:	Explain difficult concepts effectively
	F4:	Respond well to questions and comments
	F5:	Enthusiastic
	F6:	Provide practical examples
	F7:	Show an interest in students
	F8:	Grade fairly
	F9:	Provide sufficient feedback
	F10:	There was sufficient class participation
	F11:	The class sessions were worthwhile
	F12:	This was a demanding course
	F13:	The material I learned will be useful to me
	F14:	The text and other reading materials were worthwhile
Utility		
	U:	Overall, I would recommend this instructor
Actions		
	A1:	Post all materials on the Web before class
	A2:	Include some materials beyond the textbook
	A3:	Take part in communication skill workshops
	A4:	Increase office hour
	A5:	Provide the instructor with more TA support
	A6:	Change the content of the course
	A7:	Change the textbook of the course

Figure 1. The teaching evaluation case study

(1) The action has no influence on the communication skill level less than 1 or more than 5. (2) The new level is in the range $[f, 5]$, but the exact level is not specified. Such range-based specification is essential because typically only some range is known. (3) No prior assumption is made about the distribution of the new value in $[f, 5]$. One approach is asking the user to provide this information, at the cost of more burden on the user. Our approach is letting the data itself provide this information. (4) This knowledge does not specify the effect on the utility U; the influence matrix is only intended to capture the simple action/feature relationships known to the user. ■

Remarks. The influence matrix is usually sparse in that the user may not have any knowledge for many entries. The user only has to specify those entries that have a known action/feature relationship. The above action model implicitly assumes that the influence of an action on a feature is independent of the state of other features and the influence of other actions. We make this *independence assumption* because the human user tends to work better when dealing with one thing at a time. In practice, the user knowledge is largely imprecise and approximating due to the human bottleneck, and capturing complete user knowledge is neither necessary nor tractable. The naive Bayes classifier [3] is an example of making an independence assumption and achieving good results.

3.2 The Influence of Actionsets

We now consider the influence of a set of actions, or an *actionset*. Under the independence assumption, the destination range of an actionset is the "union" of the destination ranges of all actions in the set. For example, if the destination range of $f = 3$ under A_1 is $[2, f]$ and the destination range of f under A_2 is $[f, 5]$, the destination range of f under the actionset $\{A_1, A_2\}$ is $[2, 5]$. Note that there is no gap in the "union" range because f must be in all ranges.

Definition 3.1 (Destination range) Consider an actionset α and a feature value $F = f$. Let $[l_i, h_i]$ be the destination range of $F = f$ under an action $A_i \in \alpha$. Let $l(\alpha, f)$ denote $\min(l_i)$ and let $h(\alpha, f)$ denote $\max(h_i)$. The *destination range* of $F = f$ under α is $[l(\alpha, f), h(\alpha, f)]$. $A_i \in \alpha$ is a *bounding action* of $F = f$ under α if either $l_i = l(\alpha, f)$ or $h_i = h(\alpha, f)$. ■

In other words, the bounding actions completely determine the destination range. We shall consider only bounding actions for a destination range. Under the independence assumption, we can further define the influence of an actionset on several features.

Definition 3.2 (Destination space) Consider an actionset α and a feature vector $\hat{f} = (f_1, \dots, f_k)$ on the features $F_1, \dots, F_k$. The *destination space* of $\hat{f} = (f_1, \dots, f_k)$ under α is $DS_{1,k} = ([l_1, h_1], \dots, [l_k, h_k])$, where $[l_j, h_j]$ is the destination range of $F_j = f_j$ under α. ■

Let α_j be the bounding actions of $F_j = f_j$ under α, $1 \leq j \leq k$. Let α^k denote $\alpha_1 \cup \dots \cup \alpha_k$. α can be replaced with α^k without affecting $DS_{1,k}$ because $[l_j, h_j]$ is determined completely by the bounding actions α_j. In other words, $l(\alpha, f_j) = l(\alpha^k, f_j)$ and $h(\alpha, f_j) = h(\alpha^k, f_j)$. From now on, we consider only an actionset α of the form α^k.

Example 3.2 Consider the influence matrix in Table 1. $dom(F_1)$ is in the 1-5 scale and $dom(F_2)$ is in the 1-6 scale. Consider a vector $(F_1 = 3, F_2 = 3)$. A_1 increases $F_1 = 3$ to a value in $[3, 5]$ and increases $F_2 = 3$ to a value in $[3, 6]$. A_2 decreases $F_1 = 3$ to a value in $[2, 3]$ and has no effect on F_2. Thus, the destination space of $(F_1 = 3, F_2 = 3)$ under $\alpha = \{A_1, A_2\}$ is $([2, 5], [3, 6])$ on F_1 and F_2. In words, by applying A_1 and A_2 to a case matching $(F_1 = 3, F_2 = 3)$, the new feature values of the case fall into $([2, 5], [3, 6])$ on F_1 and F_2. A_1 and A_2 are the bounding actions of $F_1 = 3$ under α; A_1 is the bounding action of $F_2 = 3$ under α. ■

	F_1	F_2
A_1	[1,5]	[1,6]
A_2	[4,2]	[-,-]

Table 1. An example of influence matrix

4 Patterns and Actionability

An actionset changes the values of features. Ultimately, we are interested in the actionability of this change in terms of increasing the utility U. Consider the population P in the data C described by the vector $\hat{f} = (f_1, \dots, f_k)$ on features $F_1, \dots, F_k$. An actionset α will change any case c in P to the destination space $DS_{1,k} = ([l_1, h_1], \dots, [l_k, h_k])$ defined in Definition 3.2. To estimate the new utility of c due to the change, our approach is examining the existing cases in the destination space $DS_{1,k} = ([l_1, h_1], \dots, [l_k, h_k])$ and using their utility as the estimate. The assumption is that the user trusts the historical data for such estimation. If the destination space is correlated to a higher utility, we have a "profitable" change. We now formalize this notion of actionability.

4.1 Patterns

Consider the data population described by

$$B : (u, f_1, \cdots, f_k) \tag{1}$$

where u is a utility value and each f_j is a value for a distinct feature F_j. An actionset α will change each case in this population to the destination space

$$M : ([l_1, h_1], \cdots, [l_k, h_k]) \tag{2}$$

where $l_j = l(\alpha, f_j)$ and $h_j = h(\alpha, f_j)$. If a higher average utility $u' > u$ was observed in the population described by M, the triple (B, M, α) summarizes an opportunity for actions: by applying α to the population described by B, their expected utility will increase from u to u'. How much this statement is valid depends on how much the user believes her knowledge and the historical estimate u'. We are interested in finding all patterns represented by the triples (B, M, α) such that $u' - u$ is above some minimum threshold.

The above (B, M, α) representation, however, does not convey the "growth" of a longer pattern from a shorter one, which is important for organizing the search space in the next section. Below, we present an equivalent representation that facilitates such organization. The idea is representing a pattern (B, M, α) as a set of "items", where each item describes the actions and the change on one feature.

Definition 4.1 (Item) An *item* for a feature F_j has the form

$$(f_j, \alpha_j, [l_j, h_j]),$$

where f_j is a value for F_j, $l_j = l(\alpha_j, f_j)$ and $h_j = h(\alpha_j, f_j)$, and α_j contains only the bounding actions of f_j (under α_j). $Item_{F_j}()$ denotes the set of all items for F_j. ■

Suppose that there are p actions. For each value f_j of F_j, there are at most $p+1$ distinct l_j and at most $p+1$ distinct h_j. 1 is added for f_j itself being l_j or h_j. Therefore, there are at most $(p+1)^2$ distinct ranges $[l_j, h_j]$ for f_j. Each range $[l_j, h_j]$ uniquely determines the set of bounding actions α_j. Therefore, there are at most $(p+1)^2$ items of the form $(f_j, \alpha_j, [l_j, h_j])$ involving f_j.

Example 4.1 Consider Example 3.2. $Item_{F_1}()$ contains the following items, ordered by f_j:

$(1, \{A_1\}, [1,5])$, $(1, \emptyset, [1,1])$,
$(2, \{A_1, A_2\}, [2,5])$, $(2, \{A_2\}, [2,2])$,
$(3, \{A_1\}, [3,5])$, $(3, \{A_2\}, [2,3])$, $(3, \{A_1, A_2\}, [2,5])$, $(3, \emptyset, [3,3])$,
$(4, \{A_1\}, [4,5])$, $(4, \{A_2\}, [2,4])$, $(4, \{A_1, A_2\}, [2,5])$, $(4, \emptyset, [4,4])$,
$(5, \{A_1\}, [5,5])$. ■

Definition 4.2 (Itemset) An *itemset* of length k ($k \geq 0$) has the form

$$\{u \mid I_1, \cdots, I_k\},$$

where u is a utility value and $I_j = (f_j, \alpha_j, [l_j, h_j])$ is an item from $Item_{F_j}()$ for a distinct feature F_j. α^k is called the *actionset* of the itemset. ■

To ensure that an itemset $\{u \mid I_1, \cdots, I_k\}$, where $I_j = (f_j, \alpha_j, [l_j, h_j])$, represents a valid triple (B, M, α) as described in Equation (1) and (2), $([l_1, h_1], \cdots, [l_k, h_k])$ must be the destination space of $(f_1, \cdots, f_k)$ under $\alpha = \alpha^k$. This requirement is stated below.

Definition 4.3 (Bounding-preservation) An itemset $\{u \mid I_1, \cdots, I_k\}$, where $I_j = (f_j, \alpha_j, [l_j, h_j])$, is *bound-preserving* if, for $1 \leq j \leq k$,

$$l(\alpha^k, f_j) = l(\alpha_j, f_j) = l_j, h(\alpha^k, f_j) = h(\alpha_j, f_j) = h_j. ■$$

In other words, $\{u \mid I_1, \cdots, I_k\}$ is bound-preserving if α_j has represented the influence of α^k on F_j. Hence, $[l_j, h_j]$ will remain unchanged whether α_j or α^k is considered. Since only bound-preserving itemsets represent valid triples (B, M, α) as described in Equation (1) and (2), we shall consider only bound-preserving itemsets.

Example 4.2 Refer to Table 1. Let

$I_1 = (F_1 = 3, \{A_1\}, [3,5]), I_2 = (F_2 = 3, \emptyset, [3,3])$.

Note that $\alpha_1 = \{A_1\}$ and $\alpha_2 = \emptyset$. The itemsets $\{u \mid I_1\}$ and $\{u \mid I_2\}$ are bound-preserving. $\{u \mid I_1, I_2\}$ is not bound-preserving because the presence of α_1 affects $[3,3]$ of I_2: $h(\alpha_1 \cup \alpha_2, F_2 = 3) = 6 \neq 3$. This example shows that adding an item does not preserve the bound-preserving property. One can verify that $\{u \mid I'_1, I'_2\}$ is bound-preserving, where

$I'_1 = (F_1 = 3, \{A_1\}, [3,5]), I'_2 = (F_2 = 3, \{A_1\}, [3,6])$, or
$I'_1 = (F_1 = 3, \{A_1, A_2\}, [2,5]), I'_2 = (F_2 = 3, \{A_1\}, [3,6])$. ■

Definition 4.4 (Pattern) A *pattern* is a bound-preserving itemset. Let P be a pattern:

$$\{u \mid I_1, \cdots, I_k\}, \tag{3}$$

where $I_j = (f_j, \alpha_j, [l_j, h_j])$. α^k is the actionset of P. $B(P) = (u, f_1, \cdots, f_k)$ is called the *base* of P, and $SAT(B(P))$ is called the *base population*. $M(P) = ([l_1, h_1], \cdots, [l_k, h_k])$ is called the *model* of P, and $SAT(M(P))$ is called the *model population*. ■

The pattern P above is interpreted as follows. Assuming that the user knowledge given by the influence matrix E is valid, the actionset α^k would transform a case c matching the profile $B(P)$ to fit the profile $M(P)$; further assuming that the observed average utility u' on the model population $SAT(M(P))$ is a valid estimate, this transformation would produce the change of $u' - u$ on the utility for a random case c drawn from the base population $SAT(B(P))$. If $u' - u$ is large enough, P suggests a way to boost the utility from u to u', i.e., by applying α^k to the cases in $SAT(B(P))$.

Example 4.3 Continue with teaching evaluation data in Figure 1. Suppose that A_4 and A_5 improve F_9, and A_5 improves F_8, i.e., $E_{49} = E_{58} = E_{59} = [0,6]$. Consider the pattern P:

$$\{0 \mid I_1, I_2\} \qquad (0.1\%, 94.67\%, 5.12),$$

where $I_1 = (F_8 = 0, \{A_5\}, [0,6])$ and $I_2 = (F_9 = 0, \{A_4, A_5\}, [0,6])$.

P says that, by increasing the office hour (A_4) and providing more TA support (A_5), an instructor matching the profile $B(P) : (U = 0, F_8 = 0, F_9 = 0)$ is able to increase the overall score from 0 to 5.12 on average. 5.12 is the average overall score observed for the instructors matching the profile $M(P) : (F_8 = [0,6], F_9 = [0,6])$. $SAT(B(P))$ and $SAT(M(P))$ account for 0.1% and 94.67% of the data, respectively. ■

4.2 Problem Statements

Definition 4.5 Given a table C and a pattern P, $\{u \mid I_1, \cdots, I_k\}$,

- the *base support* of P, $bs(P)$, is $|SAT(B(P))|/|C|$,
- the *model support* of P, $ms(P)$, is $|SAT(M(P))|/|C|$,
- the *utility support* of P, $us(P)$, is $\Sigma c[U]/|C|$, where $c \in SAT(M(P))$,

- the *actionability degree* of P, $ad(P)$, is $us(P)/ms(P) - u$. ■

A larger $ms(P)$ means a more statistically valid model population. A larger $bs(P)$ means a larger base population for actions. $us(P)$ measures the utility sum of the model population. $ad(P)$ measures the expected increase of the utility by applying the actionset of the pattern to a case in the base population. P is interesting if $bs(P)$, $ms(P)$ and $ad(P)$ are large enough as perceived by the user.

Definition 4.6 (Actionability mining) Given a table C, an influence matrix E, and thresholds σ_1, σ_2 in $[0, 1]$ and $\sigma_3 \geq 0$, a pattern P is *actionable* if $bs(P) \geq \sigma_1$, $ms(P) \geq \sigma_2$, and $ad(P) \geq \sigma_3$. The *actionability mining problem* is finding all actionable patterns. ■

The frequent itemset mining [2] has attracted a great deal of attention in database and data mining research. We show that actionability mining is as least as hard as frequent itemset mining in that the latter is a special case of the former. Consider the special case that $dom(U)$ has the single value 0 and there is no action. In this special case, every item I_j now has the form $(f_j, \emptyset, [f_j, f_j])$, and $P = \{0 \mid I_1, \cdots, I_k\}$, $M(P) = (f_1, \cdots, f_k)$ and $B(P) = (0, f_1, \cdots, f_k)$ are reduced to $\{f_1, \cdots, f_k\}$.

Theorem 4.1 $\{f_1, \cdots, f_k\}$ on the features $F_1, \cdots, F_k$ is frequent for a minimum support σ_1 as defined in [2] if and only if $\{0 \mid I_1, \cdots, I_k\}$ is actionable for $\sigma_1 = \sigma_2$ and $\sigma_3 = 0$, where $I_j = (f_j, \emptyset, [f_j, f_j])$. ■

Sometimes, the user is interested in only k most actionable patterns for some user-specified k. If k is small, it is not efficient to search all actionable patterns and perform the filtering at the end of the search. A better strategy is exploiting the top-k requirement to prune the search space as early as possible.

Definition 4.7 (Top-k actionability mining) Given a table C, an influence matrix E, σ_1, σ_2 in $[0, 1]$ and $\sigma_3 \geq 0$, and an integer k, the *top-k actionability mining problem* is finding the k, or all if there are less than k, actionable patterns P with largest $ad(P)$. ■

5 The Mining Algorithms

One obvious approach is enumerating all combinations (B, M, α) of a base population B, a model population M and an actionset α. This approach is not efficient because many combinations do not represent patterns (see Example 4.2), due to the bound-preservation requirement, or are not actionable, due to the threshold requirement. Essentially, this approach ignores the "structure" of (B, M, α) as a pattern and the "relationship" between patterns. We present an algorithm that can *focus* on actionable patterns by taking into account these information. It boils down to how to enumerate patterns and how to prune the search space.

5.1 Enumeration Strategies

We need a procedure to enumerate all patterns. Since each pattern is an itemset, we can enumerate patterns as itemsets. The problem is that many itemsets do not represent patterns because they do not satisfy the bound-preservation requirement, and enumerating all itemsets is an over-kill. Below, we first discuss the depth-first enumeration of all itemsets, and then modify it to enumerate *all and only* patterns.

Itemsets can be enumerated by the standard *depth-first enumeration* based on a pre-determined lexicographical order on itemsets [9]. Consider the enumeration in the space of three features A, B, C in Figure 2. $\{u_1 \mid I_a\}$ for UA, $\{u_1 \mid I_a, I_b\}$ for UAB, $\{u_1 \mid I_a, I_b, I_c\}$, $\cdots$, $\{u_1 \mid I_a, I_b, I'_c\}$ for $UABC$. This finishes all itemsets starting with u_1, I_a, I_b. Similarly, it enumerates other itemsets starting with u_1, I_a, I'_b. Next, it enumerates all itemsets starting with $u_1, I_a, I_c, \cdots, u_1, I_a, I'_c$, finishing all itemsets starting with u_1, I_a. Similarly, it enumerates all itemsets starting with u_1, I'_a for other I'_a. Next, it enumerates all itemsets starting with u_1, I_b, $\cdots$, u_1, I'_b, then all itemsets starting with u_1, I_c, $\cdots$, u_1, I'_c. By now, all itemsets starting with u_1 are enumerated. All itemsets starting with other u_i are enumerated similarly.

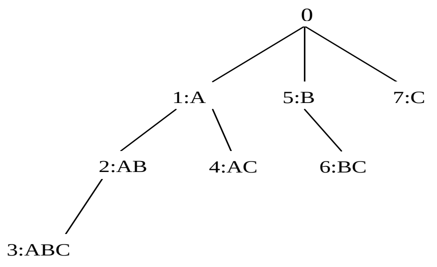


Figure 2. The lexicographic tree

Below, we modify the above enumeration to enumerate only and all patterns. First observe that adding an item I_{k+1} from $Item_{F_{k+1}}()$ to an existing pattern $\{u \mid I_1, \cdots, I_k\}$ does not guarantee that $\{u \mid I_1, \cdots, I_k, I_{k+1}\}$ is a pattern, i.e, bound-preserving. Example 4.2 is an example.

To ensure that adding I_{k+1} to a pattern $P_k = \{u \mid I_1, \cdots, I_k\}$ leads to a pattern $P_{k+1} = \{u \mid I_1, \cdots, I_k, I_{k+1}\}$, we consider only those items I_{k+1} in $Item_{F_{k+1}}()$ such that P_{k+1} is bound-preserving.

Definition 5.1 ($Item_{F_{k+1}}(P_k)$) Assume that P_k is a pattern not involving a feature F_{k+1}. $Item_{F_{k+1}}(P_k)$ denotes the set of items I_{k+1} from $Item_{F_{k+1}}()$ such that $P_k \cup \{I_{k+1}\}$ is bound-preserving. ■

The depth-first pattern enumeration. We modify the above depth-first enumeration by replacing $Item_{F_{k+1}}()$ with $Item_{F_{k+1}}(P_k)$ when expanding a pattern P_k. This modification enumerates only patterns. Two questions must be answered: can $Item_{F_{k+1}}(P_k)$ be efficiently computed? More importantly, does this modification guarantee to enumerate all patterns? We answer these questions below.

Computing $Item_{F_{k+1}}(P_k)$. Let $I_{k+1} = (f_{k+1}, \alpha_{k+1}, [l_{k+1}, h_{k+1}])$ and $I_j = (f_j, \alpha_j, [l_j, h_j])$ for $1 \leq j \leq k$. Let $\alpha^k = \alpha_1 \cup \cdots \cup \alpha_k$. For $P_{k+1} = P_k \cup \{I_{k+1}\}$ to be bound-preserving, the newly added item I_{k+1} must not "interfere" with the items $I_j, 1 \leq j \leq k$ in the prefix in terms of the bound-preservation requirement. Precisely, the following two conditions must hold:

- *Forward Non-Interference (FNI)*: α_{k+1} contains only the actions A_i such that, for $1 \leq j \leq k$, α_j remains the bounding actions of f_j after adding A_i, i.e.,

$$l_j = l(\alpha^k \cup \{A_i\}, f_j),$$
$$h_j = h(\alpha^k \cup \{A_i\}, f_j).$$

  Let $Action(P_k)$ denote the set of such A_i. Note that $Action()$ contains all actions.

- *Backward Non-Interference (BNI)*: α_{k+1} remains the bounding actions of f_{k+1} in P_{k+1}, i.e.,

$$l_{k+1} = l(\alpha^k \cup \alpha_{k+1}, f_{k+1}),$$
$$h_{k+1} = h(\alpha^k \cup \alpha_{k+1}, f_{k+1})$$

In computation, $Item_{F_{k+1}}(P_k)$ is the set of items $I_{k+1} = (f_{k+1}, \alpha_{k+1}, [l_{k+1}, h_{k+1}])$ from $Item_{F_{k+1}}()$ such that α_{k+1} contains only those actions from $Action(P_k)$, and BNI holds.

Inductively, $Action(P_{k+1})$ is computed from $Action(P_k)$ as follows. First, $Action(P_{k+1})$ is a subset of $Action(P_k)$ because if an action violates FNI on P_k, it also violates FNI on P_{k+1}. In addition, to ensure FNI on the new item I_{k+1} in P_{k+1}, $Action(P_{k+1})$ contains only those actions A_i from $Action(P_k)$ such that α_{k+1} remains the bounding actions of f_{k+1} after adding A_i, i.e.,

$$l_{k+1} = l(\alpha^{k+1} \cup \{A_i\}, f_{k+1}),$$
$$h_{k+1} = h(\alpha^{k+1} \cup \{A_i\}, f_{k+1}),$$

where $\alpha^{k+1} = \alpha^k \cup \alpha_{k+1}$ is the actionset of P_{k+1}. Note that $\alpha_j (1 \leq j \leq k)$ still remains the bounding actions since $A_i \in Action(P_k)$.

Example 5.1 Refer to $Item_{F_1}()$ and $Item_{F_2}()$ in Example 4.1. Initially, $Action() = \{A_1, A_2\}$. Let $P_1 = \{u \mid I_1\}$, where $I_1 = (3, \{A_1\}, [3,5])$ is an item from $Item_{F_1}()$. $Action(P_1) = \{A_1\}$ because A_2 interferes with the range $[3,5]$ of I_1: $l(\{A_1, A_2\}, 3) = 2 \neq l(\{A_1\}, 3) = 3$, violating FNI. $Item_{F_2}(P_1)$ contains the items $(f_2, \{A_1\}, [f_2, 6])$ from $Item_{F_2}()$, $1 \leq f_2 \leq 6$, because these items pass both FNI and BNI checking. $Item_{F_2}(P_1)$ does not contain any item $(f_2, \emptyset, [f_2, f_2])$ from $Item_{F_2}()$, $1 \leq f_2 \leq 5$, because these items fail to pass BNI. ■

Theorem 5.1 (Soundness and completeness) The depth-first pattern enumeration enumerates all and only patterns (The proof is given in the full paper [11]).

5.2 Pruning Strategies

Even though the proposed enumeration has focused on patterns, many patterns do not pass the interestingness thresholds and should be pruned. We say that P_{k+1} is a *child pattern* of P_k if $P_{k+1} = P_k \cup \{I_{k+1}\}$. The descendant relationship is the transitive closure of the child relationship. The general idea is establishing some necessary condition of actionable patterns so that the failure by a pattern implies the failure by all descendant patterns. Following the "anti-monotonicity" [2] of support (i.e., the support of a child pattern is no more than the support of a parent pattern), we have

Theorem 5.2 (Base/model support pruning) If $bs(P) < \sigma_1$, $bs(P') < \sigma_1$ for all descendants P' of P. If $ms(P) < \sigma_2$, $ms(P') < \sigma_2$ for all descendants P' of P. ■

If P fails to pass either σ_1 or σ_2, we do not need to search into the subtree below P. If P passes both σ_1 and σ_2, we check whether it passes the actionability constraint, $u' - u \geq \sigma_3$, where $u' = us(P)/ms(P)$ is the average utility of the cases in $SAT(M(P))$. Unfortunately, since the average u' could either increase or decrease at a child pattern, even if P fails this constraint, we cannot prune the subtree below P.

Our approach is to replace the average u' with a *larger* estimate u'' that is *non-increasing* at a child pattern. If $u'' - u \geq \sigma_3$ fails on a pattern P, it fails on all descendants of P because u'' is non-increasing at a child pattern, and $u' - u \geq \sigma_3$ fails on these descendants as well because $u' \leq u''$. In short, if P fails to pass $u'' - u \geq \sigma_3$, all descendants of P are not actionable and can be pruned.

The maximum utility in $SAT(M(P))$ has the property required of u'', but it could over-estimate u' so much that many patterns failing $u' - u \geq \sigma_3$ can satisfy $u'' - u \geq \sigma_3$, which means little pruning. To have a tighter (i.e., smaller) u'', we consider all of the top $\lceil \sigma_2 |C| \rceil$ utility values in $SAT(M(P))$ and use their average utility as u''. We choose the number $\lceil \sigma_2 |C| \rceil$ because $SAT(M(P))$ must contain at least this number of cases in order to pass the threshold σ_2. In other words, $u'' = us'(P)/\mu$, where $us'(P)$ denote the sum of the top $\lceil \sigma_2 |C| \rceil$ utility values in $SAT(M(P))$ (with duplicates counted), and $\mu = \lceil \sigma_2 |C| \rceil$. Note that μ is a constant. The following lemma shows that u'' is an over-estimate of $u' = us(P)/ms(P)$ and has the anti-monotonicity.

Lemma 5.1 Assume that $ms(P) \geq \sigma_2$. (1) $us'(P)/\mu \geq us(P)/ms(P)$. (2) If P' is a child pattern of P, $us'(P') \leq us'(P)$. ■

We give an example to convey the intuition behind Lemma 5.1(1). Suppose that $SAT(M(P))$ contains 5 cases with the utility values $\{4, 4, 3, 2, 1\}$. $us(P)/ms(P) = (4{+}4{+}3{+}2{+}1)/5 = 2.8$. Suppose that $\mu = 3$. The average of the top 3 utility values, $us'(P)/3$, is $(4{+}4{+}3)/3 = 3.6$, which is larger than the actual average in $SAT(M(P))$. Lemma 5.1(2) follows because the sum of the top μ utility values shrinks on a subset.

Theorem 5.3 (Actionability pruning) If $us'(P)/\mu - u < \sigma_3$, P and its descendants are not actionable.

Proof. Assume that $ms(P) \geq \sigma_2$ (otherwise we are done). From Lemma 5.1(1), $us'(P)/\mu \geq u'$. Since $us'(P)/\mu - u < \sigma_3$, $u' - u < \sigma_3$, therefore, P is not actionable. The second part follows from the first part and Lemma 5.1(2). ■

A pattern not pruned by Theorem 5.3 may still fail $u' - u \geq \sigma_3$. The size of this gap depends on how closely the average of the top μ utility values in $SAT(M(P))$ approximates the actual average u'. The larger the minimum model support σ_2 is and the smaller the variance of the utility in $SAT(M(P)$ is, the closer the approximation is.

Below, we consider the top-k actionability mining problem. Since typically k is small, it is not efficient to find all actionable patterns and filtering all but the top k patterns. A better strategy is maintaining the minimum ad required to be among the top k actionable patterns and using it as the minimum bound for u''. We can keep the set of k largest ad of the actionable patterns seen so far, denoted by AD. On generating a new pattern P, if $ad(P) > min(AD)$, we replace the minimum element in AD with $ad(P)$; if $ad(P) \leq min(AD)$, we apply the following pruning to check if the subtree at P can be pruned.

Theorem 5.4 (Top-k pruning) If $us'(P)/\mu - u < min(AD)$, P or all its descendants cannot be among the top k actionable patterns.

Proof. Note that $ad(P) = us(P)/ms(P) - u$. From $us'(P)/\mu - u < min(AD)$ and Lemma 5.1(1), $ad(P) < min(AD)$. From Lemma 5.1(2), for any descendant P' of P, $us'(P')/\mu \leq us'(P)/\mu$. By repeating the same argument as for P, we have $ad(P') < min(AD)$. So, P and P' cannot be among the top-k patterns.■

5.3 Algorithms

To compute $bs(P)$, $ms(P)$, $us(P)$ and $us'(P)$, we maintain the base and model populations for P on the current path in the depth-first enumeration, denoted (B, M). For the shortest patterns $\{u \mid\}$, B is the set of cases having the utility value u and M is the whole database C. At a child pattern $P' = P \cup \{I\}$, where $I = (f, \alpha, [l, h])$ is from $Item_F(P)$, (B', M') for P' is constructed by scanning B and M. B' contains the cases in B that have the value f on F, and M' contains the cases in M that have the values in the range $[l, h]$ on F. We can create (B', M') for all the child patterns P' in the same scan of B and M, assuming that the memory can hold all such (B', M'). Alternatively, B and M can be stored as the partitions $B_1, \cdots, B_p$ and $M_1, \cdots, M_p$, where B_i and M_i correspond to the value i of F, and only related partitions are scanned to create (B', M'). The algorithm for finding the top k actionable patterns is similar, except that it also applies Theorem 5.4 for pruning as discussed in Section 5.2. The detailed algorithms are reported in [11].

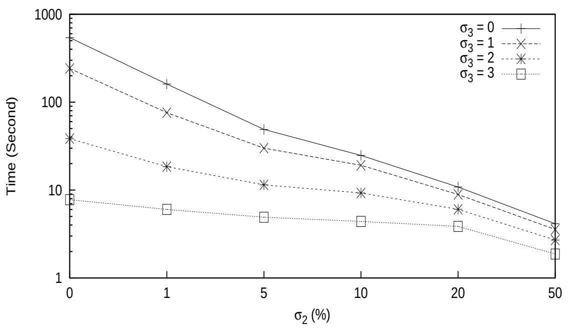
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
A1	[0,6]													
A2						[0,6]				[0,6]				
A3			[0,6]		[0,6]									
A4									[0,6]					
A5								[0,6]	[0,6]					
A6												[0,6]	[0,6]	
A7														[0,6]

Figure 3. The influence matrix

6 Evaluation

The evaluation focused on answering two questions, namely, whether the proposed method can find useful patterns and whether it scales up on large data sets. The algorithm was implemented in VC++ on a Pentium 4 with 2.4GHz CPU and 512MB memory running Windows XP.

It is hard to find a data set together with domain experts to provide the knowledge on actions. Hence, we conducted our experiments on the teaching evaluation domain for which we are experts ourselves. The data set was collected from Fall 1996 to Spring 2003 from a major university in North America, with 5,895 evaluations on 14 features $F_1 - F_{14}$ and the utility U. Figure 1 describes the features, utility, and actions. Figure 3 shows the influence matrix. All scores are in the 0-6 scale. We choose the maximum influence range $E_{ij} = [0, 6]$ because it represents the most general knowledge of boosting score and creates a larger search space. Other choices of influence ranges will be examined as well. Figure 6 shows the distribution of the utility U in



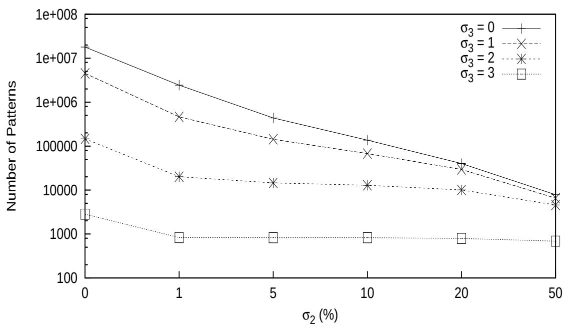
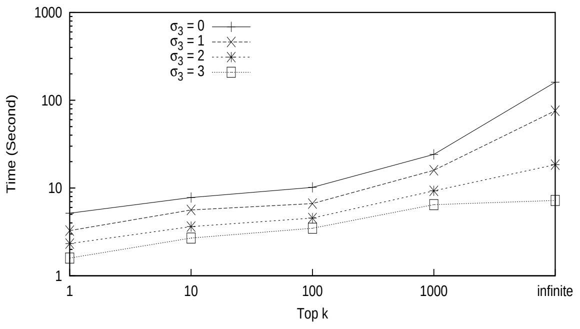


Figure 4. Time and pattern No. enumerated, $\sigma_1 = 0.1\%$



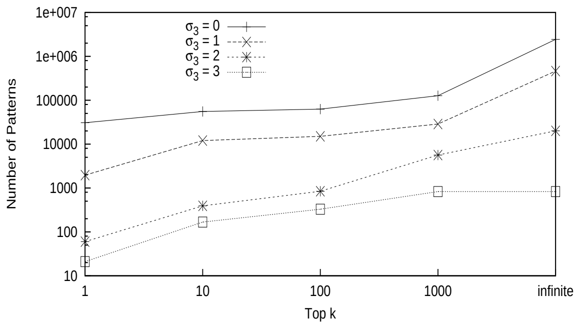


Figure 5. Top-k pattern mining: Time and pattern No. enumerated, $\sigma_1 = 0.1\%$ and $\sigma_2 = 1\%$

log scale. We observed a similar distribution for the features $F_1, \cdots, F_{14}$. Therefore, there is a strong correlation between the utility and features, as expected in this domain.

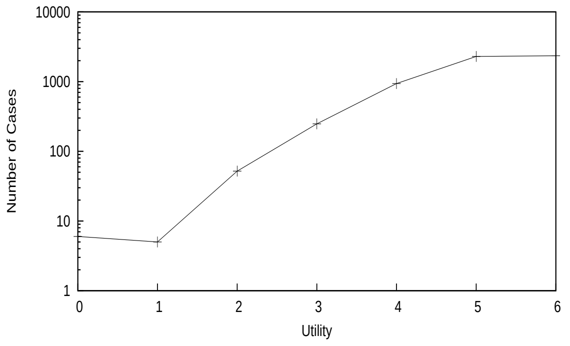


Figure 6. The distribution of utility

6.1 Usefulness

It is typical to use a small base minimum support σ_1 but a large model minimum support σ_2 for identifying "outlier" bases that have "overwhelming" role models. Given that only 6 cases have the utility value 1, $\sigma_1 = 0.1\%$ is the maximum to find actionable patterns for such cases. So, we reported our findings with the settings of $\sigma_1 = 0.1\%$ and $\sigma_2 = 1\%$. Here are the five actionable patterns that come at the top, ranked by $ad(P)$:

P_1: $\{0 \mid (F_9 = 0, \{A_4, A_5\}, [0, 6])\}$ $(0.1\%, 98.68\%, 5.12)$
P_2: $\{0 \mid (F_8 = 0, \{A_5\}, [0, 6])\}$ $(0.1\%, 94.67\%, 5.12)$
P_3: $\{0 \mid (F_8 = 0, \{A_5\}, [0, 6]),\ (F_9 = 0, \{A_4, A_5\}, [0, 6])\}$ $(0.1\%, 94.67\%, 5.12)$
P_4: $\{2 \mid (F_{10} = 3, \{A_2\}, [3, 6]),\ (F_{12} = 5, \{A_6\}, [5, 6]),$ $(F_{14} = 4, \{A_7\}, [4, 6])\}$ $(0.1\%, 10.89\%, 3.71)$
P_5: $\{2 \mid (F_{12} = 5, \{A_6\}, [5, 6]),\ (F_{14} = 4, \{A_7\}, [4, 6])\}$ $(0.1\%, 10.98\%, 3.70)$.

In general, these patterns identify a base of a small utility value and a role model of overwhelming support. In other words, for the small number of instructors not doing well, many instructors can serve as a role model for them to follow, and each pattern suggests the concrete actions to take. Patterns further down the list tend to have a larger base support but a smaller actionability degree.

Let us examine P_1 in details. This pattern suggests that, by increasing office hours (i.e., A_4) and providing more TA support (i.e, A_5), an instructor doing poorly in the feedback evaluation (i.e., $F_9 = 0$) and the overall evaluation (i.e., $U = 0$) could increase the overall evaluation to 5.12 on average. First of all, these actions indeed address the right problems, according to the knowledge in this domain.

At first glance, the increase of 5.12 seems a bit too drastic. However, if we consider what the user has in mind and what the data has recorded, this increase makes sense. In fact, the user believes that A_4 and A_5 can change F_9 from 0 to some value in $[0, 6]$. When examining the instructors in the range $F_9 = [0, 6]$, we find the average overall evaluation of 5.12. If the user trusts her knowledge and the historical data, she should believe that the estimation 5.12 provided by the data is achievable.

6.2 Efficiency

6.2.1 Pruning strategies

We are not aware of any algorithms that model the notion of actions and push the actionability requirement to prune the search space. The closest strategy that can be compared with is the Apriori pruning based on the anti-monotonicity of support [2]. In our setting, this strategy has the form of the base and model support pruning, given by Theorem 5.2. Therefore, we focus on the *additional* benefits of the actionability pruning (Theorem 5.3) and the top-k pruning (Theorem 5.4).

Actionability pruning. Figure 4 plots the execution time and the number of patterns enumerated (both in log scale) for $\sigma_1 = 0.1\%$. σ_2 is varied from 0% to 50% along the x-axis. The level of actionability pruning is represented by the four curves corresponding to different σ_3 thresholds. In particular, $\sigma_3 = 0$ corresponds to turning off the actionability pruning. For a larger σ_3, the time spent is significantly shorter and much fewer patterns are enumerated. For example, with $\sigma_3 = 3$ it took only 10 seconds for all σ_2, whereas with $\sigma_3 = 0$ it took up to 700 seconds. The speedup is 70 times! As σ_2 increases, this speedup decreases because there are fewer patterns. This study clearly shows that the actionability pruning is highly effective, especially for a small minimum model support σ_2.

Top-k pruning. We evaluated the benefit of the top-k pruning (Theorem 5.4) in Figure 5 where the x-axis represents the k value. Let us compare the case of $k = 10$ in Figure 5 with the case of $\sigma_2 = 1\%$ in Figure 4. They both use $\sigma_1 = 0.1\%$ and $\sigma_2 = 1\%$, but one applies the top-k pruning and one does not. The comparison shows that the top-k pruning substantially reduces the time and search space. This advantage is also observed by comparing $k = 10$ with $k = \infty$ in Figure 5, where $k = \infty$ has the effect of turning off the top-k pruning. The advantage slowly reduces as k increases. In practice, however, k is typically a small number. The top-k pruning has substantially reduced the gap among different σ_3. For example, even with $\sigma_3 = 0$, it took about only 10 seconds for $k = 100$.

In summary, the above experiments show the actionability pruning has additional benefits on top of the support-based pruning, and the top-k pruning has additional benefits on top of all other pruning.

7 Conclusion

Data mining based on actionability is an important but under-studied topic. Unless this topic is addressed properly, the usefulness issue of data mining results will keep haunting data mining researchers and practitioners. A main contribution of this work is the introduction of "actions" as a way to address this issue. We presented a new concept of actionability and the algorithms for its discovery. On a real-world application domain, our approach demonstrates the effectiveness of finding useful patterns and applying pruning strategies. In the future work, we intend to enhance empirical studies on more datasets and application domains.

References

[1] Adomavicius and Tuzhilin. Discovery of actionable patterns in databases: The action hierarchy approach. In *KDD*. AAAI/MIT Press, 1997.

[2] R. Agrawal, T. Imilienski, and A. Swami. Mining association rules between sets of items in large datasets. In *SIGMOD*, pages 207–216, 1993.

[3] R. Duda and P. Hart. Pattern classification and scene analysis. Wiley, 1983.

[4] S. Fayyad, Piatetsky-Shapiro. From data mining to knowledge discovery: An overview. In *Advances in Knowledge Discovery and Data Mining, in Fayyad, Piatetsky-Shapiro, Smyth, Uthurusamy*, pages 1–34. AAAI/MIT Press, 1996.

[5] Y. Jiang, K. Wang, A. Tuzhilin, and A. Fu. Mining patterns that respond to actions. In *ICDM*, 2005.

[6] J. Kleinberg, C. Papadimitriou, and P. Raghavan. A microeconomic view of data mining. *Journal of Knowledge Discovery and Data Mining*, 2:311–324, 1998.

[7] G. Piatesky-Shapiro and C. Matheus. The interestingness of deviations. In *KDD*, 1994.

[8] Z. Ras and A. Wieczorkowska. Action-rules: how to increase profit of a company. In D. Zighed, J. Komorowski, and J. Zytkow, editors, *Principles of Data Mining and Knowledge Discovery*, pages 587–592, 2000.

[9] R. Rymon. Search through systematic set enumeration. In *Principles of Knowledge Representation and Reasoning*, pages 539–550, 1992.

[10] A. Silberschatz and A. Tuzhilin. On subjective measures of interestingnessin in knowledge discovery. In *KDD*, pages 275–281, 1995.

[11] K. Wang, Y. Jiang, and A. Tuzhilin. Mining actionable patterns by role models. In *Technical Report*. School of Computing Science, Simon Fraser University, 2005.

[12] K. Wang, S. Zhou, and J. Han. Profit mining: from patterns to actions. In *EDBT*, 2002.

[13] Q. Yang and J. Yin. Postprocessing decision trees to extract actionable knowledge. In *ICDM*, 2003.