Rule Interestingness Analysis Using OLAP Operations

Bing Liu, Kaidi Zhao
Department of Computer Science
University of Illinois at Chicago
851 S. Morgan St., Chicago, IL 60607
{kzhao, liub}@cs.uic.edu

Jeffrey Benkler
Motorola, Inc
600 N. U.S. Highway 45
MD: AS220, Libertyville, IL 60048
jeffbenkler@motorola.com

Weimin Xiao
Motorola Labs
1301 E. Algonquin Rd.
Schaumburg, IL 60196. USA
awx003@motorola.com

ABSTRACT

The problem of interestingness of discovered rules has been investigated by many researchers. The issue is that data mining algorithms often generate too many rules, which make it very hard for the user to find the interesting ones. Over the years many techniques have been proposed. However, few have made it to real-life applications. Since August 2004, we have been working on a major application for Motorola. The objective is to find causes of cellular phone call failures from a large amount of usage log data. Class association rules have been shown to be suitable for this type of *diagnostic data mining* application. We were also able to put several existing interestingness methods to the test, which revealed some major shortcomings. One of the main problems is that most existing methods treat rules individually. However, we discovered that users seldom regard a single rule to be interesting by itself. A rule is only interesting in the context of some other rules. Furthermore, in many cases, each individual rule may not be interesting, but a group of them together can represents an important piece of knowledge. This led us to discover a deficiency of the current rule mining paradigm. Using none-zero minimum support and none-zero minimum confidence eliminates a large amount of context information, which makes rule analysis difficult. This paper proposes a novel approach to deal with all of these issues, which casts rule analysis as *OLAP* operations and *general impression* mining. This approach enables the user to explore the knowledge space to find useful knowledge easily and systematically. It also provides a natural framework for visualization. As an evidence of its effectiveness, our system, called *Opportunity Map*, based on these ideas has been deployed, and it is in daily use in Motorola for finding actionable knowledge from its engineering and other types of data sets.

Categories and Subject Descriptors

H.2.8 [Information Systems]: Data Management – Data Mining. 1.3.m [Computer Graphics]: Miscellaneous – Visualization

General Terms: Human Factors, Management, Design.

Keywords: Diagnostic data mining, Interestingness analysis, general impressions, class association rules, OLAP.

SIGKDD'06 August 20-23, 2006, Philadelphia, USA.

1. INTRODUCTION

It is well known that many existing data mining techniques often produce a large number of rules, which make it very difficult for manual inspection of the rules to identify the interesting ones. This is called the *interestingness* problem. Over the years, many techniques have been proposed to deal with this problem in order to help the user find *useful* knowledge. However, despite these efforts, interestingness remains a difficult problem. Few existing techniques have made it to real life applications. The difficulty is often attributed to the fact that interestingness is highly subjective. It depends on the user's current needs and his/her existing domain knowledge. While this is true, in this paper we also argue that another reason for the limited success is that we perhaps have looked in the wrong direction. Furthermore, the research is also somewhat misguided by the current rule mining paradigm, which tends to fragment the knowledge space, creates a large number of holes in the space and makes it difficult for the user to find interesting knowledge.

Since August 2004, we have been working on a major application for Motorola. The data set contains cellular phone call records. The raw data has more than 600 attributes and millions of records. After some pre-processing by our domain experts, we are left with just over 200 attributes. The data set is like any classification data set. Some of the attributes are continuous and some are discrete. One attribute indicates the final disposition of the call such as *failed during setup*, *dropped while in progress*, and *ended successfully*. This attribute is the class attribute in classification with discrete values. We note that this data set is only the data set that we began with. Our deployed system has been successfully used to analyze eleven (11) different data sets so far for entirely different applications in Motorola. Thus, we speak with some level of generality rather than based on only a single application.

Two types of mining are usually performed with this kind of data:

1. *Predictive data mining*: The objective is to build predictive or classification models that can be used to classify future cases or to predict the classes of future cases. This has been the focus of research of the machine learning community.
2. *Diagnostic data mining*: The objective here is usually to understand the data and to find causes of some problems in order to solve the problems. No prediction or classification is needed. In the above example, the problems are *failed during setup* and *dropped while in progress*. A large number of data mining applications in engineering domains are of this type because product improvement is the key task.

Our above application falls into the second type. The objective is not prediction, but to better understand the data and to find causes of call failures or to identify situations in which calls are more likely to fail. That is, the user wants interesting and actionable knowledge. Interestingness evaluation of rules is thus the key. Clearly, the discovered knowledge has to be understandable.

As the data set is a typical classification data set, rules that characterize product problems are of the following form

$$X \rightarrow y,$$

where X is a set of conditions and y is a class, e.g., for our above example $y \in$ {*failed-during-setup*, *dropped-while-in-progress*, *ended-successfully*}. In this paper, we focus on helping the user identify interesting knowledge based on such rules. These rules basically give the conditional probabilities of $\Pr(y \mid X)$, which are exactly what a diagnostic data mining application is looking for. Moreover, such rules are easily understood.

It is easy to see that such rules are *classification rules*, which can be produced by classification algorithms such as decision trees and rule induction, and class association rule mining. However, traditional classification techniques such as decision trees and rule induction are not suitable for the task due to three main reasons:

1. A typical classification algorithm only finds a very small subset of the rules that exist in data [25]. Most of the rules are not discovered because their objective is to find only enough rules for classification. However, the subset of discovered rules may not be useful in the application. Those useful rules are left undiscovered. We call this the *completeness* problem.
2. Due to the completeness problem, the context information of rules are lost (see below), which makes rule analysis later very difficult as the user does not see the complete information.

2. Since the rules are for classification purposes, they usually contain many conditions in order to achieve high accuracy. Long rules are, however, of limited use according to our experience because the engineers can hardly take any action based on them. Furthermore, the data coverage of long rules is often so small that it is not worth doing anything about them.

Class association rule mining [17] is found to be more suitable as it generates all rules in data that satisfy the user specified minimum support and minimum confidence thresholds. Class association rules are a special type of association rules [1] with only a class on the right-hand-side of each rule.

Using the Motorola application, we were able to put several interestingness techniques to the test. We found that most existing interestingness techniques are useful to some extent, but they are “good to have” techniques rather than essential techniques. Thus, they cannot form the core of a rule interestingness analysis system to help the user systematically identify interesting knowledge. To our great surprise, we also discovered that the current rule mining paradigm itself poses a major obstacle for this interestingness analysis task. Below we first summarize the main shortcomings of the current interestingness techniques:

- Lack of contexts: Most existing methods treat rules individually. However, a key discovery from our interactions with domain experts is that a single rule is seldom interesting by itself no matter what its support and confidence values are. It is only interesting if it deviates significantly from its siblings. That is, a rule is only interesting in a *meaningful context* and in *comparisons* with others. The user wants to see both the rule and the context.
- Do not find generalized knowledge from rules (meta-mining): Each individual rule may not be interesting by itself. A group of related rules together may represent an important piece of knowledge. For example, a set of rules from an attribute may show some interesting trend, i.e., as the values of the attribute go up, a call is more likely to fail. We call such knowledge *general impressions*. Our domain experts said that such knowledge is much more useful than individual rules because they may reveal some hidden underlying principles.
- Lack of knowledge exploration tools: Due to the subjective nature of interesting knowledge, a systematic method is required for the user to explore the rule space in order to find useful knowledge. Our experiences show that the user-driving interactive discovery is the best approach. Although there are many existing techniques for visualizing rules, they mostly also treat and visualize rules individually, which we found in our applications, was not very effective.

Context is the key to dealing with all the above problems. However, the existing rule mining paradigm eliminates a large amount of contextual information. Let us see why:

- In the mining of class association rules, user-specified *minimum support* (*minsup*) and *minimum confidence* (*minconf*) values are used to ensure that the computation is feasible. Those rules that do not meet the minsup or minconf requirements are not generated. However, they can form important context information for other rules and generalized knowledge. Such contextual information is thus lost.

For example, an attribute B has three possible values, a, b, d. Due to the minsup we only find the rule $B = a \rightarrow c$, where c is a class. The other two possible rules, $B = b \rightarrow c$ and $B = d \rightarrow c$, which form the context for $B = a \rightarrow c$, are not found because they do not satisfy the minsup. We call them *holes* (or *gaps*) in the knowledge space. Then rule $B = a \rightarrow c$ does not have a context. We also cannot find any generalized knowledge about the attribute due to incomplete information or the holes. Hence, we say that the current mining paradigm *fragment*s the knowledge space and creates *discontinuity* in the space, which make the understanding and exploration of knowledge by human users very difficult.

In this paper, we propose a novel approach to deal with all these problems. We show that a major part of rule exploration can be casted as an OLAP problem (On-line Analytical Processing) [7] based on *rule cubes* whereby well established techniques in OLAP, e.g., slicing, dicing, drilling down and rolling up, can be used to explore rules to *systemically* discover useful knowledge. Methods are also proposed to help the user find *general impressions*, which are also critical as they are what the users are very interested in. To the best of our knowledge, this is the first time that OLAP operations have been used for rule interestingness analysis. The OLAP framework also provides a natural way for visualization with contexts. Together with the mining of general expressions, the proposed technique represents an effective and systematic approach. The mining support is provided by a class association rule miner. The full system, called *Opportunity Map*, based on this approach has been deployed and is in daily use in Motorola. Originally, the intended task was to identify causes of cellular phone call failures, but the system has been found to be generic and has been used to mine many other data sets for the diagnostic type of applications and for data understanding.

2. RELATED WORK AND PROBLEMS

This work is related to three areas of research, rule mining, rule interestingness analysis and rule visualization. As we have discussed the shortcomings of the current rule mining paradigm in the introduction, we will not repeat it here. Below we only focus on rule interestingness analysis and visualization**.**

There are several existing interestingness methods that can help the user find interesting knowledge.

Unexpectedness: In this method, the user is asked to give some existing knowledge and the system then finds the unexpected rules [11][19][23][26][33]. This did not work well because our users were not sure what to expect. They wanted the system to find interesting knowledge for them. [4] studies neighborhood unexpectedness of rules. The neighborhood of a rule, which is similar to the concept of context, is a set of syntactically similar rules, i.e., involving similar items. This is not applicable to us. We need a different definition and also different rule mining. The idea of general impressions is first proposed in [16]. However, they need to be given by the user for finding unexpected rules. In this work, we mine general impressions from discovered rules.

Rule ranking: Ranking rules according to some interestingness measures [2][9][28]. Our experiences show that almost all top ranked rules represent some artifacts of the data rather than any useful patterns. Moreover, giving only individual rules without contexts to compare with is not appropriate as we discussed earlier. This method does not find generalized knowledge.

Querying and filtering: In [6][21], some data mining query languages are proposed to select the right data to mine different types of rules. [14][30][31] also report several rule query languages to enable the user to specify what rules that he/she needs and the system then retrieves the relevant rules. We tried this approach, but our users did not know what to ask. In [29], a set of rule post-processing operators is defined to allow the user to filter unwanted rules, select rules of interest, group rules, etc. This is a good approach. However, it stops short of providing contexts to rules and mining generalized knowledge.

We also considered several other methods [8][24][27]. All of these and above methods had some use but were not sufficient. The first version of our system [35] was based on class association rules, rule ranking and grouping, and sophisticated visualization. However, the techniques were not effective enough because they did not provide a systematic approach to enable the user to explore the knowledge space, and could not produce the kind of knowledge that the user needed. Thus, it was not a final product that could be deployed. The approach still requires extensive human effort to interpret the results, so was not considered satisfactory.

Regarding data mining results visualization, our work is related to rule visualization [13]. [10] proposes interactive mosaic plots to visualize the contingency tables of association rules. In [5], classification rules are visualized using rule polygons. [34] visualizes the temporal behavior of rules. In [12], a post-processing environment is proposed to browse and visualize association rules so that the user can divide a large rule set into smaller ones. In [22], important rules in terms of support and confidence values are highlighted with a grid view. In [20], ordering of categorical data is studied to improve visualization.

The above approaches mainly help to visualize individual rules. They do not actively help the user find useful knowledge. They visualize rules without sufficient contextual information. Thus, they differ from our approach in terms of both the goal and the visualization. Our visualization is based on rule cubes and OLAP with a set of well established and systematic operations.

3. THE CONCEPTUAL FRAMEWORK

Before building the current system we spent more than a year building the first system [35]. As discussed in the related work, it was not deployable although the system was useful to some extent. After several demonstrations, it became clear to us that a new approach was needed as our domain experts were not willing to use it. However, our extensive interactions with the domain experts during the process were very valuable. We found that

1. Presenting rules individually and ranking them is not very useful. The highly ranked rules are mostly artifacts of the data rather than useful patterns. Rules need to be visualized within some meaningful contexts so that related rules can be compared. It is through the comparison that the user will discover useful knowledge. The keyword is "*comparison*". Without a meaningful comparison, nothing is interesting.

2. Users are interested in generalized knowledge because that is how they usually describe knowledge. For example, in our meetings with domain experts, they often said such things as "when the values of attribute A_i increase, the call is more likely to fail", and "attribute A_i is an important attribute". We call such knowledge *general impressions* because they are summaries of many underlying rules and are also imprecise. General impressions need to be mined from rules.

3. Users are mostly interested in short rules, seldom more than two conditions because they said that it was impractical for them to do laboratory tests to meet the conditions of long rules. Furthermore, rules with many conditions often cover so few cases that it is not worthwhile doing anything about them.

4. Users want to explore rules to discover knowledge based on their engineering background. They do not want the system to dictate what is interesting because the domain knowledge is so complex that there is little chance that an automatic system is able to decide interestingness. Thus, a simple and systematic exploration tool is needed to facilitate this process.

OLAP based on *rule cubes* provides a generic and convenient environment for addressing all of these issues. By design, OLAP is an exploration tool. We will also see that interesting rules and their contexts can be visualized in the OLAP framework. General impressions can also be mined in the framework. Thus, interactive rule exploration and interesting knowledge finding are naturally integrated. This turned out to be a powerful approach, which is used in Opportunity Map. Class association rule mining provides the backbone mining support for the whole system.

The conceptual framework of the proposed approach is shown in Fig. 1. The class association rule miner produces both rule cubes for OLAP and also long rules (rules with many conditions). We will discuss them shortly. GI miner mines general impressions (GI) from rules in rule cubes. The visualization and the user interaction are powered by the usual OLAP operations, which also show the GI mining results and long rules.

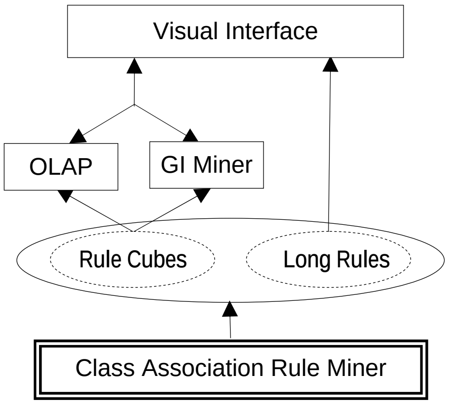


Fig. 1: The conceptual framework of Opportunity Map

4. RULE EXPLORATION AS OLAP OPERATIONS

OLAP is a database technology. The idea is to organize the data using a multi-dimensional *data cube*. Each attribute and its values in the data represent a dimension. There is also a measurement attribute. Its values are the cell values in the cube. The well-known example is the sales data across different regions in different time periods [7]. The measurement attribute is the sales volume of each region in a particular quarter. Each dimension may also have a concept hierarchy, which allows the user to see the sales volume at different levels of hierarchy. The main OLAP operations are roll-up, drill-down, slice, and dice. Below, we introduce these operations in the context of rules. We will see that these operations for data analysis are also very powerful for rule analysis. The OLAP framework thus presents a systematic methodology for the exploration of knowledge.

4.1. From Rules to Rule Cubes

Rules are not data. The problem is how to transform rules into cubes so that OLAP operations can be applied. We discuss it now.

Let the set of attributes in the data D be $A = \{A_1, A_2, \ldots, A_n\}$. Let the class attribute be C. Let the domain or the set of possible values of an attribute A_i be $dom(A_i)$.

We want to convert class association rules into cubes. Recall that class association rules (CAR) [17] are a special kind of association rules [1] with only a class on the right-hand-side of each rule. They are of the form: $X \rightarrow y$, where X is a set of conditions, and $y \in dom(C)$ is a class. A condition is an attribute value pair of the form: A_i=a_{ij} ($a_{ij} \in dom(A_i)$). Every condition uses a distinctive attribute. CAR mining requires every attribute in the data to be discrete. This is not a problem as there are many existing discretization algorithms that can be used to discretize a continuous attribute into intervals.

Given any set of attributes, $\{A_{i1}, \ldots, A_{ip}\} \subseteq A$, we use S to denote the set of all possible rules using the set of attributes:

$$S = \{(A_{i1}=v_1, \ldots, A_{ip}=v_p \rightarrow C = c_k) \mid v_1 \in dom(A_{i1}), \ldots, v_p \in dom(A_{ip}), c_k \in dom(C)\}$$

The supports and confidences of the rules are omitted here.

It is easy to see that all the rules using the attributes can be represented as a cube with p+1 dimensions (1 being the class attribute). We call this a *rule cube*. The measurement attribute, which we do not have explicitly, is the support/frequency count of data records that satisfy each rule. Thus, each cell of the rule cube is represented with

$$A_{i1}=v_1, \ldots, A_{ip}=v_p, C = c_k$$

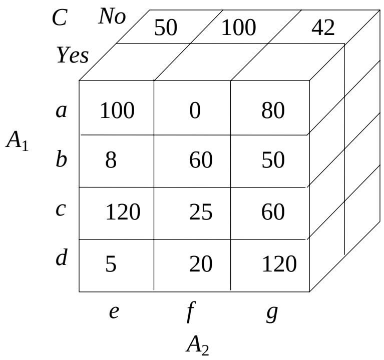
Its cell value is the number of data points that contains (A_{i1}=v_1, …, A_{ip}=v_p, $C = c_k$), which is simply the support count of the rule:

$$A_{i1}=v_1, \ldots, A_{ip}=v_p \rightarrow C = c_k$$

The confidence of the rule can be computed with

$$conf\,(A_{i1}=v_1,\ldots,A_{ip}=v_p \rightarrow C=c_k) = \frac{\sup(A_{i1}=v_1,\ldots,A_{ip}=v_p, C=c_k)}{\sum_{j=1}^{|dom(C)|}\sup(A_{i1}=v_1,\ldots,A_{ip}=v_p, C=c_j)}, \qquad (1)$$

where the function *sup* gives the support count of a cube cell. Let us see an example. We have a data set with three attributes. One of them is the class attribute C, which has two values, *yes* and *no*. The other two attributes are A_1 and A_2. A_1 has four possible values a, b, c, d, and A_2 has three possible values e, f, g. Assume that the data set has 1158 data points. The rule cube is shown in Fig. 2, which represents 24 rules (3×4×2).



A_1	e	f	g
a	100	0	80
b	8	60	50
c	120	25	60
d	5	20	120

Fig. 2: A rule cube example representing 24 rules.

As an example, the rule, $A_1 = a$, $A_2 = e \rightarrow C = yes$, has the support of 100/1158, and the confidence of 100/(100+50). The rule, $A_1 = a$, $A_2 = f \rightarrow C = yes$, has the support of 0 and the confidence of 0. We see that the support and the confidence of each rule can be easily computed if we have the rule cube. In visualizing rules, we will have both values computed and visualized.

4.2. OLAP Operations on Rule Cubes

Let us now see how various OLAP operations are performed on the rule cube.

Roll-up: The roll-up operation performs aggregation on the rule cube, either by claiming up a concept hierarchy for a dimension or by dimension reduction. In application, it is easy to build a concept hierarchy for an attribute based on the domain knowledge or requirement. For example, we may group the values a and b together and call it *ab*, and group c and d together and call it *cd*. Likewise, we can group *ab* and *cd* to create a concept *abcd*. This gives a 3-level hierarchy. In an actual application, suitable names can be given to the concepts. Here, we only use an example of dimension reduction to show the working of roll-up. For instance, if we remove attribute A_1, we obtain the sub-cube in Fig. 3.

C	*No*	105	310	95
	Yes	233	105	310
		e	*f*	*g*
			A_2	

Fig. 3: Roll-up: after removing the dimension A_1

This roll-up operation gives us 6 one-conditional rules:

$A_2 = e \rightarrow C = yes$ $\quad$ $A_2 = e \rightarrow C = no$
$A_2 = f \rightarrow C = yes$ $\quad$ $A_2 = f \rightarrow C = no$
$A_2 = g \rightarrow C = yes$ $\quad$ $A_2 = g \rightarrow C = no$

Computing their supports and confidence is straightforward.

Drill-down: Drill-down is the reverse of roll-up. It goes from less detailed data to more detailed data. Drill-down can be stepping down a concept hierarchy for a dimension or introducing additional dimensions. For example, if our current cube only has two dimensions, A_2 and *C*, then adding the A_1 dimension constitutes a drill-down operation, which give more detailed information. For example, we can drill down from Fig. 3 to Fig. 2.

Slice: The slice operation performs a database selection on one dimension of the given cube, resulting in a sub-cube. For example, if we select using the criterion $A_1 = a$, we obtain the sub-cube in Fig. 4.

C	*No*	50	100	42
	Yes	100	0	80
Slice for $A_1 = a$:		*e*	*f*	*g*
			A_2	

Fig. 4: A slice operation

This operation gives us the following 2-conditional rules:

$A_1 = a, A_2 = e \rightarrow C = yes$ $\quad$ $A_1 = a, A_2 = e \rightarrow C = no$
$A_1 = a, A_2 = f \rightarrow C = yes$ $\quad$ $A_1 = a, A_2 = f \rightarrow C = no$
$A_1 = a, A_2 = g \rightarrow C = yes$ $\quad$ $A_1 = a, A_2 = g \rightarrow C = no$

Dice: The dice operation defines a sub-cube by performing a selection on two or more dimensions. For example, if we do the selection, ($A_1 = a$ or $A_1 = b$) and ($A_2 = e$ or $A_2 = f$), we obtain the sub-cube in Fig. 5.

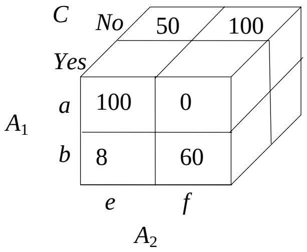


Fig. 5. A dice operation

This operation results in the following rules:

$A_1 = a, A_2 = e \rightarrow C = yes$ $\quad$ $A_1 = a, A_2 = e \rightarrow C = no$
$A_1 = a, A_2 = f \rightarrow C = yes$ $\quad$ $A_1 = a, A_2 = f \rightarrow C = no$
$A_1 = b, A_2 = e \rightarrow C = yes$ $\quad$ $A_1 = b, A_2 = e \rightarrow C = no$
$A_1 = b, A_2 = f \rightarrow C = yes$ $\quad$ $A_1 = b, A_2 = f \rightarrow C = no$

There are other OLAP operations. However, the operations above have been shown sufficient for finding interesting knowledge.

Our practical experience shows that OLAP operations are useful for interestingness analysis due to the following reasons:

1. They allow the user to see rules in context because all relevant rules are contained in the same cube. Note that although in our examples above we did not include rules confidences, they can be easy added in visualization as we will see in Section 6.
2. The cube representation of rules gives us a natural way to visualize rules. Our visualization system is based on matrices, which are sufficient for visualizing the complete information of two and three dimensional rule cubes.
3. They allow general impressions about rules to be easily mined based on cubes and also displayed in the same visualization. We will discuss the details in the next section.
4. The well defined operations of OLAP enable the user to *systematically* explore the knowledge space to find interesting knowledge, which is proven to be critical in our applications.

The downside is that we cannot build a huge rule cube to cover all attributes in the data as the cube size grows exponentially with the number of attributes. Fortunately, we do not need such a huge rule cube in practice because our application experiences show that users are seldom interested in long rules. In this work, we only build all 3-dimensional rule cubes (the class attribute is a dimension in every rule cube). This can be done using a class association rule miner, which also generates longer rules at the same time. Cube generation and rule mining are thus integrated.

4.3. Mining Class Association Rules

To support the cube operations using a class association rule (CAR) miner, we need to find all possible rules. This means that we have to set both minsup and minconf to 0. However, this causes combinatorial explosion. We need a compromise.

- We set minsup and minconf to 0 in mining rules with 1 and 2 conditions so that we can generate all 3-dimensional rule cubes, which represent all possible 2-condition rules (the class attribute forms a dimension). This usually does not cause memory problem. For example, we are able to use our data set of over 200 attributes to generate rules to populate all 3-dimensional rule cubes. In our application, 2-condition rules are usually sufficient as we stated earlier.
- We set non-zero minsup and minconf for rules with more conditions. This prevents combinatorial explosion. Multiple minimum supports give additional flexibilities [18], which deal with skewed data and also enable the system not to generate unwanted rules [15]. These are useful in practice.

All these can be done quite easily in a CAR miner such as CBA [17] by setting some parameters. We have re-implemented CBA to accommodate the required flexibility. For more details on class association rule mining, please refer to [17][15].

We also note that rule cubes do not need to be materialized. The tree structure (we use a hash tree) for generating and storing CAR rules can be used directly as virtual cubes by OLAP operations. Our current implementation takes this approach. However, it is possible to materialize the cubes and store them on disk if there is not enough main memory for generating long rules.

5. MINING GENERAL IMPRESSIONS

As we discussed in Section 4, domain experts usually look for general knowledge which is broadly applicable and exception rules in some contexts. As we mentioned earlier, they often say

- "Attribute A is an important attribute": We call such attributes *discriminative attributes.*
- "When the values of attribute A_i increase (or decrease), the phone call is more likely to fail": We call such a behavior of an attribute a *trend*. This kind of knowledge is only applicable to ordinal and numeric attributes.
- "When the attribute A_i takes value a_{ij}, the phone call is more likely to fail" We call such rules *context exceptions* (or simply *exceptions*) as they are found in certain contexts. Clearly, this type of rules is particularly useful for categorical attributes.

These pieces of knowledge are called general impressions (GI). The question is how to let a data mining system find such knowledge in the first place. This is the topic of this section.

General impressions are mined from the following construct:

$$X, A_i \rightarrow C$$

where X is a set of conditions, which can be empty (i.e., $|X| \geq 0$), A_i is an attribute, and C is the class attribute. Each condition in X is an attribute value pair expressed as $A_k = a_{kj}$. We also call X a *data constraint* as it defines a subset of the data $D_{DC} \subseteq D$ that satisfies the constraint. Attribute A_i is called the *active attribute* as we will consider all its values. Let the set of attributes used in X be X_A. We have $|X| = |X_A|$ (each attribute in X is distinct) and $X_A \cap \{A_i\} = \emptyset$. Thus, $X, A_i \rightarrow C$ represents a set of rules:

$$\{(X, A_i = a_{ij} \rightarrow c_k) \mid a_{ij} \in dom(A_i), c_k \in dom(C)\}.$$

These rules can be easily obtained by an OLAP operation. In the examples in Section 4, if $X = \emptyset$, these rules can be obtained by a *roll-up* operation in Fig. 2. If X contains $A_1 = a$, the rules can be obtained by a *slice* operation in Fig. 4.

5.1. Rules in Contexts and Exceptions

As we discussed above, a rule is only interesting in a meaningful context. The context of a rule is simply its sibling rules, which can be defined in various ways. Our applications show that the following definition is a very useful one.

Definition (context of a rule): The context of the rule ($X, A_i = a_{ij} \rightarrow c_k$) consists of the following rules, which cover all the values of the active attribute A_i:

$$X, A_i = a_{i1} \rightarrow c_k,$$
$$\ldots,$$
$$X, A_i = a_{ir} \rightarrow c_k$$

where $\{a_{i1}, \ldots, a_{ir}\}$ is the domain of A_i. These rules are also called *sibling rules* of one another.

In this context, we can compare all rules and detect values of A_i that represent exceptions, i.e., highly correlated (positively or negatively) to the class c_k. We compute exceptions based on confidences, but can be done in other ways.

Definition (degree of exception in confidence): The *degree of exception*, denoted by $DE(X, A_i{=}a_{ij}, c_k)$, of the rule ($X, A_i = a_{ij} \rightarrow c_k$) is measured by how the observed confidence is different from the expected confidence:

$$DE(X, A_i = a_{ij}, c_k) = \left| Conf_O(X, A_i = a_{ij}, c_k) - Conf_E(X, c_k) \right|$$

where $Conf_O(X, A_i{=}a_{ij}, c_k)$ is the observed confidence of the rule, and $Conf_E(X, c_k)$ is the class c_k's prior probability in D_{DC}.

5.2. Discriminative Attributes

We now compute the discriminative power of an attribute.

Definition (discriminative power of an attribute): The discriminative power of attribute A_i given X is the ability of this attribute to distinguish data of all classes in D_{DC}:

$$DP(X, A_i) = \sum_{k=1}^{m} w_{c_k} \times \sum_{j=1}^{r} sup(X, A_i = a_{ij}, c_k) \times DE(X, A_i = a_{ij}, c_k)$$

where r is the number values of attribute A_i, m is the number of classes, $sup(X, A_i = a_{ij}, c_k)$ is the support count of the rule: $X, A_i = a_{ij} \rightarrow c_k$, and w_{c_k} is a weight parameter reflecting the importance of class c_k. This weight parameter is useful because it enables the user to focus on the class that he/she is interested in. We note that there are other ways to define discriminative power of an attribute, e.g., information gain [25].

Ranking can be performed based on the results.

5.3. Trend Behaviors

Recall a trend represents a piece of knowledge that when the values of an attribute increase/decrease, a particular class is more/less likely to occur. The trend analysis is only applicable to continuous attributes and ordinal attributes.

Definition (unit trend of A_i on class c_k): A *unit trend* of attribute A_i with respect to class c_k and fixed conditions X, denoted by $UT(X, A_i{=}[a_{ix}, \ldots, a_{iy}], t_l, c_k)$, is a trend t_l of the rule confidences of a set of consecutive values $[a_{ix}, \ldots, a_{iy}]$ of attribute A_i with respect to class c_k in D_{DC}. t_l (also called a *trend type*) is decided by a statistical test, and takes one of the types from the set {*increasing-trend*, *decreasing-trend*, *stable-trend*}. The value range from a_{ix} to a_{iy} is such that any larger range will not have the trend t_l. That is, the trend is *maximal*.

For an attribute A_i and class c_k, there can be multiple unit trends over the values of the attribute. For example, it can have an increasing trend from *value1* to *value5*, and a decreasing trend from *value6* to *value10*. An example trend is:

> *In a particular salary range, as the salary of the borrower goes higher, a loan is more likely to be approved.*

Here, "*loan approved*" is the class c_k that we are interested in. The attribute "*salary*" is the trend attribute A_i, which shows an increasing trend t_l as its values go from small a_{ix} to large a_{iy}.

A unit trend is derived from a set of rules based on their confidences. For example, the above simple trend may be derived from the following rules:

Rule 1: *salary = 20K* $\rightarrow$ *loan_approved, Confidence: 20%*
Rule 2: *salary = 30K* $\rightarrow$ *loan_approved, Confidence: 35%*
Rule 3: *salary = 40K* $\rightarrow$ *loan_approved, Confidence: 40%*

By using the trend relationship, all the three rules (or even more) can be summarized into a single piece of knowledge. This unit trend has the value range from 20K to 40K. Our domain experts really like this notion because that is what they are interested in and is exactly the format that they are familiar with.

The type of a trend is calculated using a statistical test called *reverse arrangement test* [3]. For example, to test whether it is an increasing trend, we first calculate how many times that a later value is strictly greater than an earlier value. Each time that happens, we call it a *reversal*. If there are a lot of reversals (more than are likely from pure chance with no trend), we have significant evidence of an increasing trend. If there are too few reversals we have significant evidence of decreasing trend. Formally, it works as follows:

Given r ordered values as $v_1, v_2,..., v_r$, for all the possible pairs, we count a reversal each time when $i < j$ and $v_i < v_j$. We keep the total count as reversal count R.

For r values, the maximum possible number of reversals is $r(r-1)/2$. For random data, on average we would expect to have $r(r-1)/4$ reversals.

With value r and R, we can lookup the statistic tables [3] to decide the count whether we can conclude a significant trend in the data.

For each unit trend, we also compute two statistical properties.

Support: It is the sum of the data points this trend covers (can be the raw count). It can also be expressed as a percentage of the total number of data points in D_{DC}.

Confidence: It is used to indicate how likely the trend is correct. A value of 1.0 indicates a perfect trend of that type without exception. Otherwise, it is calculated based on how many data values and their covered data points satisfy the trend behavior.

$$Confidence = 1.0 - Abnormal\_data / Support$$

where *Abnormal_data* is the data count which does not follow the trend.

Definition (trend value of an attribute on class c_k): The trend value of attribute A_i given X, denoted as $TV(X, A_i, t_l, c_k)$, with respect to class c_k for trend type t_l in data D_{DC} is defined by:

$$TV(X, A_i, t_l, c_k) = \sum_{j=1}^{k} Support_{UT_j} \times Confidence_{UT_j}$$

where UT_j is an unit trend of attribute A_i and there are k such unit trends.

Finally, we can use the trend values (TV) to rank attributes according to different types of trends (*increasing-trend*, *decreasing-trend* or *stable-trend*). The user can see them in the visualization and determine which trends are interesting. We also note that there are other ways for computing the trend value of an attribute. The above method works quite well for our applications.

6. SYSTEM AND EVALUATION

We first give a brief description of our system. The main focus is on the visualization sub-system. We then discuss the evaluation.

6.1. Opportunity Map

The Opportunity Map system consists of five main components: a discretizer, a CAR rule generator, a GI miner, and a visualizer. Given a data set, continuous attributes are first discretized using the discretizer (a manual discretization option is also available). The discretized data is fed into the CAR rule generator. The resulting rules form 3-dimentional virtual rule cubes. The user uses the visualizer to explore the rule space based on OLAP operations. GI miner is called when requested based on the sub-cube shown on screen. Below, we give more details on rule cube visualization.

Due to the use of rule cubes and OLAP operations, the visualization is simple. In our system, every visualization screen is a 2-dimensional matrix. Each grid in the matrix visualization visualizes one or more cells in the rule cube.

1. For a 2-dimensional cube, each grid represents a cube cell. Inside the grid, a rule is visualized using a bar. The height of the bar represents the confidence of the rule, with the support count written beside it.
2. For a 3-dimensional cube, each grid represents the cube cells in the third dimension. Thus, it contains multiple rules, with each rule visualized as a bar.

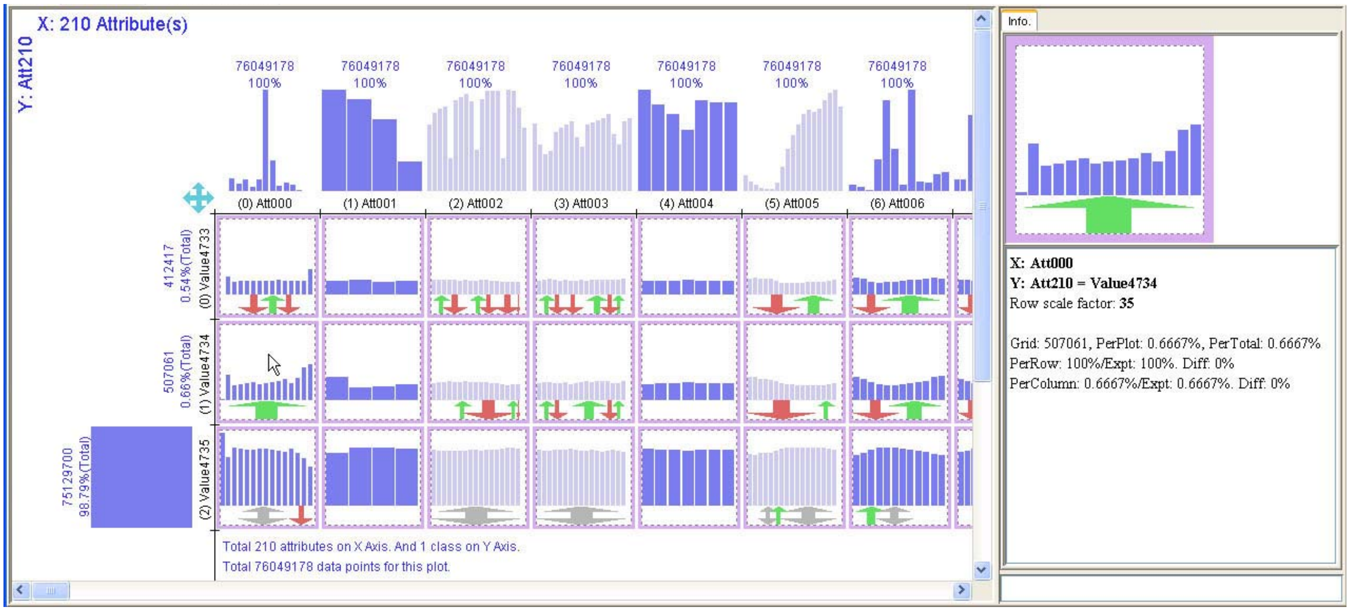
Note that rules with more than two conditions are not visualized in our current system as they are seldom used in our applications (our engineers said that it is very hard to do anything about long rules). However, if the user wants to see longer rules related to a cube cell, the relevant rules will be listed in a separate window.

6.2. Evaluation and a Case Study

Since the proposed system helps the user find subjectively interesting/useful knowledge, it is difficult to have an objective measure of its effectiveness. As an evidence of its effectiveness, our system has been deployed and is in regular use in Motorola. The original intended application was to find causes of cellular phone call failures from the call log data. Due to its success, the system has been used to analyze more than 11 large data sets for entirely different applications by Motorola engineers. Many pieces of actionable knowledge have also been put to practice use.

Here, we give a case study using the call log data to show how the user interacts with the system to find useful knowledge. The goal is to discover possible causes of call failures. This is a large and high dimensional data set. It has seventy six million data records, and about 600 attributes. Sampling was used to reduce the data size. After some initial analysis by domain experts, the number of attributes was reduced. This version of the data contains 211 attributes, in which one attribute is the class attribute with three values. The majority class covers a very large proportion of the data. Due to confidentiality, we could not disclose the exact percentage. All the attribute names and values are also replaced by generic names and values.

The visualization uses a matrix layout, and has two main modes, *overall visualization mode*, and *detailed visualization mode*. In the overall visualization mode (Fig. 6), the X axis is associated with all attributes in the data. The Y axis is associated with all the classes. For each attribute (a column), each grid shows all one conditional rules of the corresponding class value. Each rule is visualized as a thumbnail bar. The height of the bar is the rule confidence value. Thus, this screen simply shows all the 2-



Total 210 attributes on X Axis. And 1 class on Y Axis.
Total 76049178 data points for this plot.

X: Att000
Y: Att210 = Value4734
Row scale factor: **35**

Grid: 507061, PerPlot: 0.6667%, PerTotal: 0.6667%
PerRow: 100%/Expt: 100%. Diff: 0%
PerColumn: 0.6667%/Expt: 0.6667%. Diff: 0%

Fig. 6. Initial visualization showing general knowledge: all 2-dimensional rule cubes.

dimensional rule cubes. Each rule cube is formed by the class attribute and one other attribute. Each column shows one cube.

The system supports automatic scaling among classes to address the class imbalance issue. Scaling increases relative proportions. In Fig. 6, it has already been applied. Otherwise, we will not see anything for the minority classes (the first two classes on the Y axis), which are the classes that our users are interested in.

Blue color is used by default. Some attributes may have so many possible values (e.g., Att002, Att003, etc.) that the grid size may be inadequate to draw them all. Light blue is used to indicate this. To see all values, the user can either increase the grid size, or use a detailed visualization (see below). Various support counts and proportions are written on the screen or in the Information Panel on the right when the user moves the mouse over the screen.

This overall visualization mode is able to summarize and show a number of important properties of the data immediately:

1. The data distribution of each attribute is illustrated by the distribution bars at the top of each column above the X axis. For the class attribute, they are on the left of the Y axis.
2. One-conditional class association rules are visualized for all attributes. Each rule is represented as a small bar in the visualization, with its context information: rules of all other values of the attribute (X direction in each grid) and all other classes (Y direction across rows) are visualized side by side.
3. Trends are detectable from the shape in each grid. Strong unit trends are indicated using color arrows: red for decreasing, green for increasing and gray for stable trends.

After seeing the overall visualization, the user may be interested in trends and want to see which attributes are strongly correlated with the classes. For example, if he is interested in seeing increasing trends with respect to the first class (first row, class Value4733), using a sorting command from the menu, he gets the visualization in Fig. 7. Attributes are sorted so that those with strongest increasing trends on class Value4733 appear first on the X axis. With this screen, the user can easily confirm his previous knowledge, or find new knowledge. For example, it clearly shows that as the values of attribute Att951 increase from small to large, the chance of occurrence of failure class Value4733 increases. This may be a vital piece of knowledge that could be used in product design. Visualization of sorted trend attributes for other trend types and classes is viewable using similar commands.

Another sorting shows the discriminative attributes (Fig. 8). Note that the pink color is used to indicate that the scaling produced some very small values that can hardly be seen on screen (for that grid). The top ranked attributes all have strong discriminating powers on the classes. For example, the values of the first attribute (Att021) clearly discriminate data into different classes. Attribute Att020 also has strong discriminating power except for the middle 4 values. Our users confirmed that this kind of knowledge is very useful. Also crucial is the fact that the user can view all the details side-by-side. This provides context for visual comparison and discovery, which is vital in practice.

Visualizing all the 2-dimensional rule cubes in the overall visualization is very useful to get the users started and to pick the right attributes for further study using the *detailed visualization*.

A detailed visualization shows either a larger version of a 2-dimensional rule cube (Fig. 9), or a 3-dimensional rule cube (Fig. 10). The X axis shows the values of a given attribute. The Y axis shows another attribute (usually the class attribute).

1. Fig. 9 shows a detailed visualization of one attribute (Att001, on the X axis) with all classes (the Y axis). This is simply a 2-dimensional rule cube. This visualization reveals the following detailed pieces of knowledge: The exact trends of this attribute with respect to all classes (if the attribute is ordinal). It also shows the exact counts and percentages which are not shown in the overall visualization.
2. Exceptions of this attribute. The dotted dark blue horizontal lines show the expected confidences for the classes (scaling is applied). Those bars that deviate from their expected

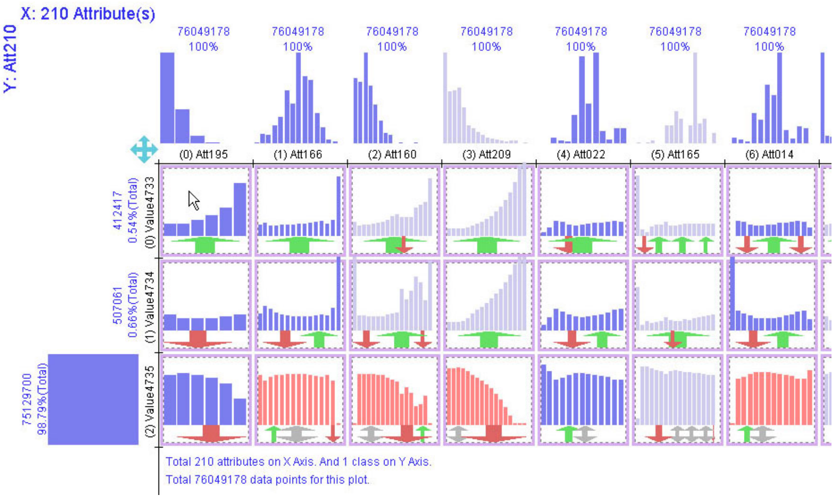


Fig. 7. Trend attributes sorted for class Value4733

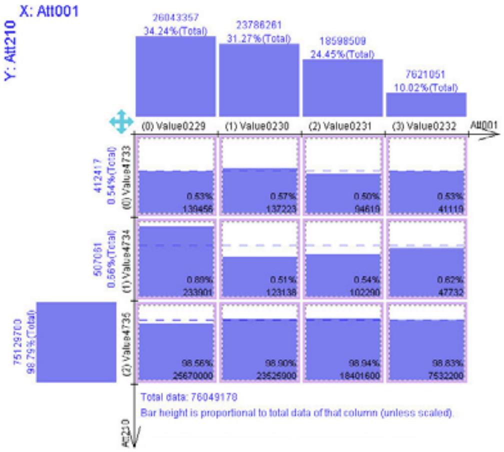


Fig. 9. Detailed visualization of one 2-dimensional rule cube

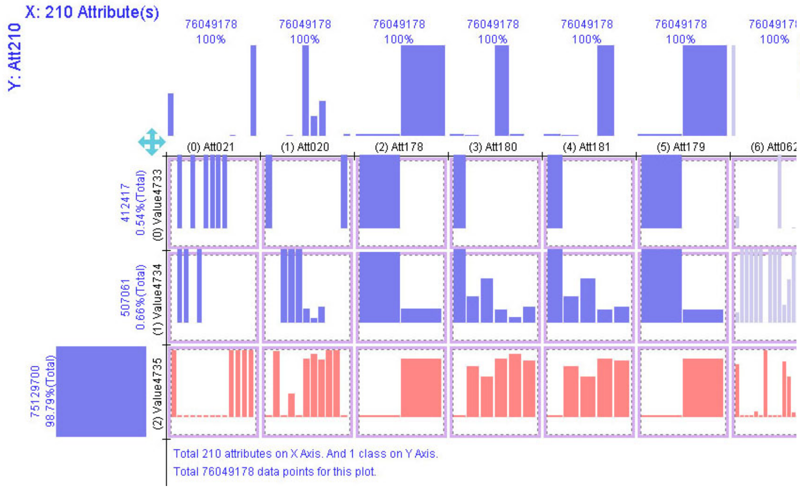


Fig. 8. Discriminate attributes sorted

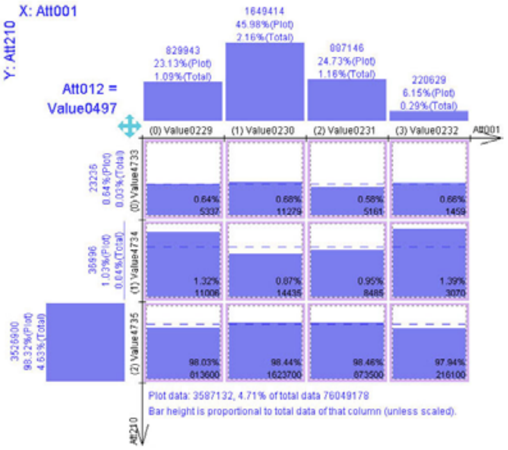


Fig. 10. Detailed visualization of one 3-dimensional data cube (a slice)

confidences represent important exceptions. Values (on the X axis) can be sorted according to their degrees of exception.

For each piece of knowledge discovered here, the user is able to see all the related classes (in the vertical direction) and all the related values (in the horizontal direction) from the visualization. They act as the context. Now, suppose the user is interested in seeing what will happen if another condition is added. This is an OLAP slice operation. It is visualized by adding a data constraint to the visualization so that the subset of data can be visualized and studied. Fig. 10 shows the result of using attribute "Att012 = Value0497" as the data constraint (Value0497 is a phone model). Thus, all the information shown in this visualization has 2 attributes involved (plus the class attribute), i.e., two conditional rules. For example, the rule in the grid of row 2 and column 2 is:

Att012=Value0497, Att001=Value0232 → Att210=Value4734, 1.39%

Comparing with Fig. 9, we can see that the slice operation affects the last column greatly. It changed the class Value4734 from below expected confidence in Fig. 9 to far above expected confidence in Fig. 10.

Visualizing 3-dimensional rule cubes (or sub-cubes) using detailed visualization can be employed for comparative study, as shown in Fig. 11. It "stacks" two or more detailed visualizations by drawing the bars for each corresponding grid side by side into one grid, creating a clear contrast view in each grid. Since the Y axis is the class attribute and the data constraints represent two different phones, this visualization shows that the second phone (Value0498) has a lower failure rate than the first one, by having consistently tall bars in the row of Value4735 (the class for "success calls"). Other rows and bars show how the products perform on different values of Att012 as well as the comparisons between them. This visualization is produced by a slice operation. Due to space limitation, we are unable to show many other types of analysis that can be performed on the system.

In summary, the overall visualization provides a summary of all the 2-dimensional rule cubes. Various sorting of attributes give the user different general impressions. This allows the user to pin down interesting attributes easily. Detailed visualizations provide further information and enables detailed analysis. What is crucial is that all pieces of information and rules are shown in context

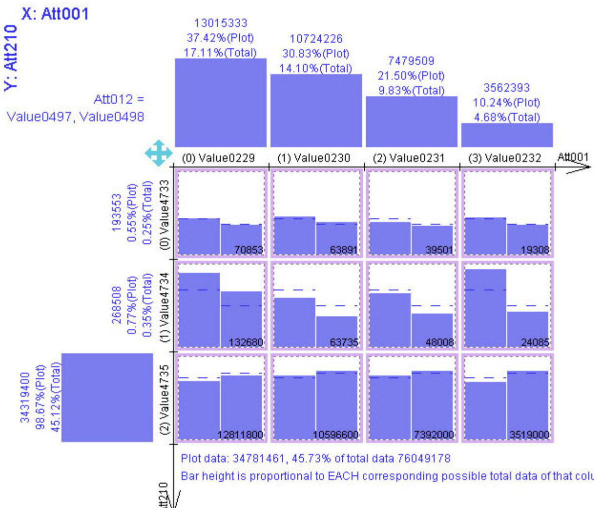


Fig. 11. Detailed visualization of one 3-dimensional rule cube for comparative study

within their corresponding rule cubes or slices, which enables the user to quickly detect interesting knowledge. Our users confirm that the system to allow them to easily analyze their data.

7. CONCLUSIONS

This paper proposed a novel approach to rule analysis to help users find useful knowledge. The approach has four key ideas. The first idea is that the traditional mining paradigm hinders rule analysis. The second idea is that rules need to be analyzed in context. The third idea is that rule analysis can be performed using OLAP operations. The fourth idea is the mining of general impressions. Rule cubes and OLAP provide a general framework for exploration of rules in context to enable the user find useful knowledge. A data mining system, called Opportunity Map, based on the ideas and class association rules have been built. The system has been deployed and is in daily use in Motorola.

REFERENCES

[1] Agrawal, R. and Srikant, R. “Fast algorithms for mining association rules.” VLDB-94, 1994.

[2] Bayardo, R. and Agrawal, R. “Mining the most interesting rules.” KDD-99, 1999.

[3] Bendat J., Persol A. Random data: analysis and measurement procedures. Wiley-Inter science. 2005.

[4] Dong G., Li J. Interestingness of discovered association rules in terms of neighborhood-based unexpectedness. PAKDD-98.

[5] Han J, Cercone N. "RuleViz: A model for visualizing knowledge discovery process". KDD-00, 2000.

[6] Han J., Fu, Y., Wang W., Koperski, K. and Zaiane, O. “DMQL: a data mining query language for relational databases.” SIGMOD Workshop on DMKD, 1996.

[7] Han, J and Kamber, M. Data mining: concepts and techniques. Morgan Kaufmann, 2001.

[8] Hussain, F, Liu, H, Suzuki, E., Lu, H. Exception rule mining with a relative interestingness measure. PAKDD-00, 2000.

[9] Hilderman, R., Hamilton, H. “Evaluation of interestingness measures for ranking discovered knowledge.” PAKDD-2001.

[10] Hofmann H, Siebes A., Wilhelm, A. "Visualizing association rules with interactive mosaic plots". KDD-00.

[11] Jaroszewicz, S., and Simovici, D. “Interestingness of frequent itemsets using bayesian networks as background knowledge.” KDD-04, 2004.

[12] Jorge A., Pocas J., Azevedo P. "Post-processing environment for browsing large sets of association rules". PKDD-02 VDM Workshop, 2002.

[13] Keim D. "Information visualization and visual data mining". IEEE Trans. Vis. Comput. Graph, 2002.

[14] Klemetinen, M., Mannila, H., Ronkainen, P., Toivonen, H., and Verkamo, A.I. “Finding interesting rules from large sets of discovered association rules.” CIKM-1994, 1994.

[15] Liu, B. Web Data Mining: exploring hyperlinks, contents, and usage data. A forthcoming book. 2006/2007

[16] Liu B., Hsu W. and Chen S., “Using general impressions to analyze discovered classification rules.” KDD-97, 1997.

[17] Liu B., Hsu W., and Ma Y. “Integrating classification and association rule mining.” KDD-98, 1998.

[18] Liu B., Hsu W., Ma Y. “Mining association rules with multiple minimum supports.” KDD-99, 1999.

[19] Liu, B., Hsu, W., Mun, L., & Lee, H. “Finding interesting patterns using user expectations.” IEEE TKDE, 11(6), 1999.

[20] Ma S., Hellerstein J. "Ordering categorical data to improve visualization. " INFOVIS-99, 1999.

[21] Meo, R. Psaila, G., and Ceri, S. “A new SQL-like operator for mining association rules.” VLDB-96, 1996.

[22] Ong K-H, Ong K-L, Ng W-K, Lim E-P. "CrystalClear: active visualization of association rules". ICDM-02 Workshop on Active Mining (AM-02), 2002.

[23] Padmanabhan, B. and Tuzhilin, “A. knowledge refinement based on the discovery of unexpected patterns in data mining.” Decision Support Systems, 33(3), July 2002.

[24] Piatesky-Shapiro, G., and Matheus, C. “The interestingness of deviations.” KDD-94, 1994.

[25] Quinlan J.R. C4.5: Programs for Machine Learning. 1993.

[26] Silberschatz, A, Tuzhilin, A. What makes patterns interesting in knowledge discovery systems. IEEE TKDE 8(6), 1996.

[27] Suzuki, E. “Autonomous discovery of reliable exception rules.” KDD-97, 1997.

[28] Tan, P-N. & Kumar, V. “Interestingness measures for association patterns: a perspective.” KDD-2000 Workshop on Post-processing in ML and DM, 2000.

[29] Tuzhilin, A. and Adomavicius, G. “Handling very large numbers of association rules in the analysis of microarray data.” KDD-02, 2002.

[30] Tuzhilin, A., and Liu, B. “Querying multiple sets of discovered rules. KDD-02, 2002.

[31] Virmani A., Imielinski, T. “M-SQL: A query language for database mining.” Journal of DMKD, 1999.

[32] Vapnik, V. The nature of statistical learning theory. 1995.

[33] Wang K., Jiang Y., Lakshmanan L. V.S. “Mining unexpected rules by pushing user dynamics.” KDD-03, 2003.

[34] Zhao K. Liu, B., Tirpak, T. and Schaller, A. "V-Miner: using enhanced parallel coordinates to mine product design and test data". KDD-02, 2004.

[35] Zhao K., Liu B., Tirpak T. and Xiao W. “A visual data mining framework for convenient identification of useful knowledge.” ICDM-05. 2005.