Finding Interesting Rules from Large Sets of Discovered Association Rules

Mika Klemettinen Heikki Mannila Pirjo Ronkainen Hannu Toivonen* A. Inkeri Verkamo
Department of Computer Science
University of Helsinki
P.O. Box 26, FIN-00014 University of Helsinki, Finland
mannila@cs.helsinki.fi

Abstract

Association rules, introduced by Agrawal, Imielinski, and Swami, are rules of the form "for 90 % of the rows of the relation, if the row has value 1 in the columns in set W, then it has 1 also in column B". Efficient methods exist for discovering association rules from large collections of data. The number of discovered rules can, however, be so large that browsing the rule set and finding interesting rules from it can be quite difficult for the user. We show how a simple formalism of *rule templates* makes it possible to easily describe the structure of interesting rules. We also give examples of visualization of rules, and show how a visualization tool interfaces with rule templates.

1 Introduction

Data mining (knowledge discovery in databases) is a field of increasing interest combining databases, artificial intelligence, and machine learning. The purpose of data mining is to facilitate understanding large amounts of data by discovering interesting regularities or exceptions (see e.g. [13]).

Today, in their daily operation, enterprises and organizations gather and store huge amounts of data. Understanding the data, or learning about the implicit information in the data, is often important for strategic decision support or for technical applications. Consider, for instance, how a bank or a supermarket could utilize patterns discovered in their customer transactions, e.g. for product decisions or marketing or risk management, or how telecommunications network alarms could be correlated and predicted based on regularities in the alarm data. Finding interesting regularities manually, e.g., with statistical methods, is time consuming. As the amount of data grows fast, automatic data mining methods are needed to promote data understanding.

Association rules, introduced recently by Agrawal, Imielinski, and Swami [1], are a class of simple but powerful regularities in binary data. An association rule of the form $W \Rightarrow B$, where W is a set of attributes and B a single attribute, states that in the rows of the database where the attributes in W have value true, also the attribute B has value true with high probability. However, there can easily be hundreds or even more association rules holding in a data set. Paradoxically, data mining itself can produce such great amounts of data that there is a new knowledge management problem.

In this paper we consider the problem of finding interesting rules from the set of all association rules holding in a data set. (In effect, we address a problem of data mining of second order.) This is a generic problem in data mining (see [12]): while formal statistical criteria for rule strength and (statistical) significance abound, it is much harder to know which of the discovered rules really interest the user. Of course, this problem is quite hard.

The issue of interestingness of discovered knowledge has been discussed in general by Piatetsky-Shapiro [12]. Hoschka and Klösgen [6] have also used templates for defining interesting knowledge, and their ideas have strongly influenced our work. Their approach is based on few fixed statement types and partial ordering of attributes, whereas our approach is closer to regular expressions. Han et al. [5] present an attribute-oriented approach for pruning uninteresting relations. Piatetsky-Shapiro and Matheus discuss the interestingness of deviations [14].

We show that it can be helpful to inquire the user for some simple additional information about the structure of the data. The basic idea is to apply *templates*, a form of pattern expressions, in information retrieval from the set of discovered rules. Templates can be used to describe the form of interesting rules, and also to specify which rules are not interesting. We have implemented templates in a prototype tool for discovering and examining association rules.

The rest of this paper is organized as follows. Association rules are introduced in Section 2, where we also discuss the problems in managing a large set of rules. In Section 3 we describe the use of templates, and present examples and experience from two case studies with real data. Section 4 investigates how visualization can aid in managing rules. Section 5 is a short conclusion.

2 Association rules and their properties

First we introduce some basic concepts, using the formalism presented in [1]. Let $R = \{I_1, I_2, \ldots, I_m\}$ be a set of attributes, also called items, over the binary domain $\{0, 1\}$. The input $r = \{t_1, \ldots, t_n\}$ for the data mining method is a relation over the relation schema $\{I_1, I_2, \ldots, I_m\}$, i.e., a set of binary vectors of size m. Each row can be considered as a set of properties or items (that is, $t[i] = 1 \Leftrightarrow I_i \in t$).

*On leave from Nokia Research Center.

Appeared in the Third International Conference on Information and Knowledge Management, Ed. Nabil R. Adam, Bharat K. Bhargava and Yelena Yesha, 401 – 407. Nov 29–Dec 2, 1994, Gaithersburg, Maryland. ACM Press.

Let $W \subseteq R$ be a set of attributes and $t \in r$ a row of the relation. If $t[A] = 1$ for all $A \in W$, we write $t[W] = \bar{1}$. An *association rule* over r is an expression $W \Rightarrow B$, where $W \subseteq R$ and $B \in R \setminus W$. Given real numbers γ (confidence threshold) and σ (support threshold), we say that r satisfies $W \Rightarrow B$ with respect to γ and σ, if

$$|\{i \mid t_i[WB] = \bar{1}\}| \geq \sigma n$$

and

$$\frac{|\{i \mid t_i[WB] = \bar{1}\}|}{|\{i \mid t_i[W] = \bar{1}\}|} \geq \gamma.$$

That is, at least a fraction σ of the rows of r have 1's in all the attributes of WB, and at least a fraction γ of the rows having a 1 in all attributes of W also have a 1 in B. Given a set of attributes X, we say that X is *covering*[1] (with respect to the database and the given support threshold σ), if

$$|\{i \mid t_i[X] = \bar{1}\}| \geq \sigma n.$$

That is, at least a fraction σ of the rows in the relation have 1's in all the attributes of X.

Example 1 *In a course enrollment database, the association rule*

Introduction to Unix $\Rightarrow$ Programming in C (0.84, 0.34)

states that 84 % of the students that have taken Introduction to Unix, also have taken Programming in C, and that 34 % of all the students actually have taken both courses. □

Given σ and γ, the collection of all association rules that hold in a data set can be computed efficiently [1, 2, 9]. These algorithms work in time proportional to the size of the database and the number and size of covering subsets. Specifically, one can find all covering subsets reasonably fast. Additionally, one can obtain good approximations to the collection of association rules by using reasonably small samples [9]; the data sets of our case studies are representative of reasonable sample sizes.

The set of all association rules holding in a data set provides a wealth of information about the data. For example, for all attributes we obtain all possible rules that explain the presence of the attribute. This can be contrasted to several other machine learning or data mining methods (e.g., decision tree learning) that provide only one explanation or rule (the one chosen by the heuristic methods) for a given target attribute. On the other hand, several of the rule formalisms used in machine learning are much more powerful than association rules.

As the target of discovery is not fixed, association rules can reveal valuable, unexpected information. This strength of association rules has a drawback: there can be quite many rules holding with sufficient strength in the data set.

Example 2 *The data set of one of our case studies is an enrollment database of courses in computer science. The data set consists of registration information of 1066 students who had registered for at least two courses. There is a row per student, containing the courses the student has registered for. On average, a row has 5 courses. The total number of courses is 112.*

The goal in analyzing this data is to obtain accurate and useful information about the interrelationships between enrollments of various courses. Such relationships can be quite complex, but they can be valuable, e.g., in the planning of the curriculum and in allocating resources. Standard statistical techniques and packages seem quite weak in finding some of the unexpected connections in this type of data, although they are very good in analyzing the connections, once they have been pointed out or discovered.

We used this data to find association rules with the algorithm represented in [9]. With support threshold $\sigma = 0.01$ (corresponding to 11 students) there are 2010 association rules (with confidence threshold $\gamma = 0.00$). The largest left-hand side of a rule consists of 4 courses. With confidence threshold $\gamma = 0.7$, there are still 420 rules, and with $\gamma = 0.9$, 99 rules. □

One of the basic problems in data mining is to know what interests the user. The confidence threshold γ and the support threshold σ ensure that the discovered rules have enough positive evidence. However, a given relation may satisfy a large number of such association rules. To be useful, a data mining system must manage the large amount of generated information by offering flexible tools for further selecting rules.

Example 3 *Our other case study is a database of computer equipment orders from a factory to retail sellers. There are 524 orders, each to one customer, with 1 – 11 high level package identifiers, and on average 12.4 modules. On average, the 524 lines contain 15.2 attributes, and the total number of attributes is 894.*

The analysis of this kind of data could be useful for decision support in the factory: e.g. designing new packages of modules, planning and directing marketing actions, or learning more about retail seller profiles.

The data has some special properties. First, there are attributes of different types (customer, package, module). Second, there are strong associations: a package always contains certain modules. For instance, the package 486SX25modelA contains e.g. modules PowerSupplyM, FloppyDiskDriveN, and SystemBoard486SX25O. Whenever the package occurs on a row, also the corresponding modules occur. The rows in the database look like the following:

AcmeInc, 486SX25modelA, PowerSupplyM, FloppyDiskDriveN, SystemBoard486SX25O, MouseP, HardDiskDriveQ, HousingR, DisplayS

With support threshold $\sigma = 0.06$ (32 orders) there are 4817 association rules. Of these rules, 4468 have confidence of 0.7 or more, 3724 have confidence of 0.9 or more, and 3543 rules have confidence of 1.0. The largest rules have 8 attributes on the left-hand side. □

By setting the thresholds σ and γ sufficiently high, it may of course be possible to obtain a rule set with only a few rules. For instance, only 22 rules of the computer equipment database have support of 0.2 or more. The problem with this method is, however, that such pruning often loses important information. In this case the support threshold $\sigma = 0.06$ is justified: rules valid in about 30 orders are still interesting.

On the other hand, not all rules with high confidence and support are interesting. Rules can fail to be interesting for several reasons.

- A rule can correspond to prior knowledge or expectations.

  The course enrollment database, for instance, satisfies a number of rules that correspond to normal process in the studies. Basic courses are taken before graduate courses, so the following kind of rules are expected:

[1] Agrawal et al. [1] use the term *large*.

Design and Analysis of Algorithms (graduate course) $\Rightarrow$ Fundamentals of ADP (basic course) (0.97, 0.03).

Or, the computer equipment database contains a number of strong associations between packages and modules. Discovering and presenting these known associations is a nuisance.

- A rule can refer to uninteresting attributes or attribute combinations. E.g., the rule

  Fundamentals of ADP (basic course) $\Rightarrow$ Programming in Pascal (basic course) (0.95, 0.60)

  is useless, if the user is only interested in graduate courses.

  Or, in the case of the computer equipment database, the user may be uninterested about a certain retail seller, and would like to filter out all rules that make statements about it.

- Rules can be redundant. For example, the latter of the following rules has no additional predictive power over the first one, and could be pruned as redundant:

  Data Communications, Programming in C $\Rightarrow$ Unix Platform (0.14, 0.03)

  and

  Data Communications, Programming in C, Introduction to Unix $\Rightarrow$ Unix Platform (0.14, 0.03).

  Or, if a package identifier is on the right-hand side of a rule induced from the computer equipment database, then there certainly exist similar rules, but with the modules of the package on the right-hand side.

The problem we consider in this paper is essentially how to find the most interesting rules. We suggest that this is done by giving the user a possibility to specify classes of both interesting and uninteresting rules.

Hoschka and Klösgen deal with the problem of redundancy in their Explora system [6]. It uses partial orderings of attributes and attribute sets to avoid presenting several types of redundant knowledge. However, the two parameters of association rules — confidence and support — make it more difficult to define sensible limits and semantics for redundancy.

We also consider the presentation of rules so that it would be easy to find the most interesting ones.

3 Selecting interesting rules

Interesting and uninteresting classes of rules can be specified with *templates*. Templates describe a set of rules by specifying what attributes occur in the antecedent and what attribute is the consequent. First, attributes are classified to a class hierarchy by the user.

Example 4 *In our example domain of computer science course enrollment, the courses are divided into three classes: basic, undergraduate, and graduate courses. In addition, all courses belong to the parent class 'any course'. Thus, we have such generalizations as*

{Fundamentals of ADP, Programming in Pascal, Information Systems} $\subset$ Basic Course $\subset$ Any Course,

{Artificial Intelligence, Programming in C, Data Communications} $\subset$ Undergraduate Course $\subset$ Any Course,

{Design and Analysis of Algorithms, User Interfaces, Neural Networks} $\subset$ Graduate Course $\subset$ Any Course.

□

Now, a template is an expression

$$A_1, \ldots, A_k \Rightarrow A_{k+1},$$

where each A_i is either an attribute name, a class name, or an expression $C+$ or $C*$, where C is a class name. Here $C+$ and $C*$ correspond to one or more and zero or more instances of the class C, respectively. A rule

$$B_1, \ldots, B_h \Rightarrow B_{h+1}$$

matches the pattern, if the rule can be considered to be an instance of the pattern.

Example 5 *The user is interested in the course Design and Analysis of Algorithms. In particular, it would be interesting to know associations where Design and Analysis of Algorithms is on the right-hand side and there is another graduate course on the left-hand side, such as*

Neural Networks $\Rightarrow$ Design and Analysis of Algorithms (0.48, 0.02)

or

Artificial Intelligence, Stringology $\Rightarrow$ Design and Analysis of Algorithms (0.65, 0.01).

Such rules match the pattern

Graduate Course, Any Course$*$ $\Rightarrow$ Design and Analysis of Algorithms.

□

With templates, the user can explicitly specify both what is interesting and what is not. To be interesting, a rule has to match an *inclusive template.* If a rule, however, matches a *restrictive template*, it is considered uninteresting. To be presented to the user, a rule must be interesting — i.e. match one of the inclusive templates — and it must not be uninteresting — i.e. not match with any of the restrictive templates.

Example 6 *Using the inclusive template*

Graduate Course, Any Course$*$ $\Rightarrow$ Design and Analysis of Algorithms.

the user obtains a number of rules. The user may consider some of them uninteresting, such as those including basic courses:

Compilers, Introduction to ADP $\Rightarrow$ Design and Analysis of Algorithms (0.16, 0.01).

By adding a restrictive template

Basic Course, Any Course$*$ $\Rightarrow$ Any Course

the user can filter out all rules with at least a basic course on the left-hand side.

There are 15 rules that match the above specifications and have confidence of 0.2 or more and support of 0.01 or more. The rules mentioned in the previous example are two of them. This sort of rules give an idea what the students do except for taking Design and Analysis of Algorithms, and the discovered rules can be utilized when planning the course. (Remember, the arrow implies no temporal order. But, with the enrollment data, the arrow tends to point backwards in time.) □

Example 7 *On the other hand, what rules follow, if Design and Analysis of Algorithms is on the left-hand side? Using the inclusive template*

> Design and Analysis of Algorithms, Any Course$*$ $\Rightarrow$ Any Course

and the restrictive templates

> Basic Course, Any Course$*$ $\Rightarrow$ Any Course

and

> Any Course$*$ $\Rightarrow$ Basic Course

one obtains rules with Design and Analysis of Algorithms on the left side and without any basic courses.

Again, with confidence threshold of 0.2 and support threshold of 0.01, there are 47 such rules. One of them states that 60 % of students taking Design and Analysis of Algorithms also have taken Artificial Intelligence.

Another rule is

> Design and Analysis of Algorithms $\Rightarrow$ Neural Networks $(0.22, 0.02)$.

Thus, only 22 % of students taking Design and Analysis of Algorithms also take Neural Networks; on the other hand, above it was found that 48 % of students taking Neural Networks also take Design and Analysis of Algorithms. □

A common problem in knowledge discovery is the use of background knowledge. With association rules, restrictive templates can be employed as a mild form of utilization of existing knowledge: they are especially handy for filtering out rules known beforehand.

Example 8 *The restrictive template*

> Package $\Rightarrow$ Module,

in conjunction with the computer equipment database, filters out all dependencies between packages and their modules. This template prunes 136 rules, 108 out of which present normal package configurations and have confidence of 1.0.

With this kind of general template the user may also lose unknown associations between packages and modules that do not belong to the particular package. A more accurate distinction can, of course, be achieved by using a number of more specific restrictive templates. □

A high level classification of attribute types can be useful for selecting certain types of rules.

Example 9 *The user is interested in rules about the customers (retail sellers). The inclusive templates*

> Any $\Rightarrow$ Customer

and

> Customer $\Rightarrow$ Any

select all rules that are about customers and have exactly one attribute on the left-hand side.

In our database we found, for instance, strong biases in selling packages to retail sellers, and a dozen of modules that are sold to only one particular retail seller. □

Example 10 *The inclusive template*

> Module $\Rightarrow$ Mouse

gives 8 rules from the computer equipment database with confidence of 1.0.

Surprisingly, there are certain power supplies and housings always sold with a mouse. These rules may be valuable for future product decisions about package contents. □

Often, the most interesting association rules are those revealing unexpected information. In order to spot the interesting rules, the management of a large set of discovered rules must first be solved. Templates are one way of focusing the search in the potentially large space of rules. In the next chapter, we discuss visualization of rules as a complementary means of managing discovered rules.

4 Visualization of association rules

We now give a short description of "Rule Visualizer", a prototype tool under development for management of discovered association rules with the aid of templates and visualization. Rule Visualizer consists of three components: Rule Selection, Rule Browsing, and Rule Graph. The Rule Selection component is a tool for specifying the criteria for rules to be presented. It also includes the association rule discovery capability (see [9] for the algorithms). The Rule Browsing component is used for presenting rules separately, and the Rule Graph component visualizes a set of rules as a dependency graph.

Figure 1 presents the Rule Selection tool of Rule Visualizer. With it the user can specify e.g. the confidence and support thresholds, and a lower limit for their product — or *commonness* μ. Commonness threshold can be used, e.g., to avoid rules with both low confidence and low support. Rule size and the maximum number of rules to show can be limited. Finally, inclusive and restrictive templates can be specified.

The two other components, Rule Browsing and Rule Graph, present rules visually. It is difficult to visualize association rules without loosing any essential information — although visualization has been studied extensively, there seem to be no results directly applicable to complex many to one relationships with weights.

The intended basic strategy in using Rule Visualizer is the following. First, association rules are discovered from the data, with support threshold as low as is appropriate, and confidence threshold 0. Next, with the Rule Selection tool, the user can refine the thresholds (but only upwards) and enter inclusive and restrictive templates. Now the user can view the first approximation of the set of interesting rules with Rule Browsing and Rule Graph components. With a mouse the user can edit the graph presentation and refine the templates. The user can also modify the thresholds and templates in the Rule Selection tool. Through the iterative process of refinements and experiments, the user can navigate in the space of rules.

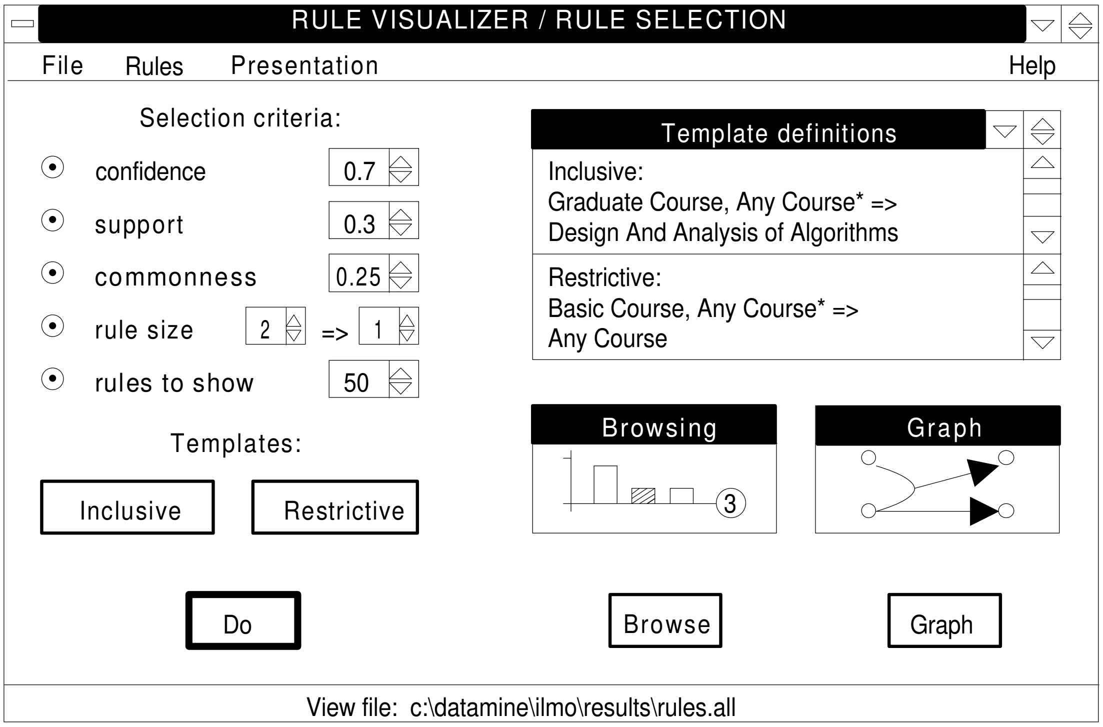

Figure 1: Rule Visualizer / Rule Selection.

4.1 Single rules

We start the discussion of visualization with the problem of showing single association rules to the user so that finding interesting rules is as easy as possible. The trivial method is to use a textual representation. It is obvious, however, that glancing at a long list of rules hardly gives a useful overview of them. Reading even a list of only a few dozen rules is tedious. And, even with reasonable confidence and support thresholds and templates, there can be hundreds or thousands of rules.

For single rules we visualize only confidence and support; the attributes are perhaps best visualized in a network presentation with several other rules. Figure 2 is an example of bar graph presentation. The leftmost and rightmost bars represent the confidence and support of the rule, respectively, while the middle bar represents the commonness. The number of attributes on the left-hand side of the rule is indicated in the circle by the bar graph.

Another way to visualize rule significance is to create rectangles in which one side represents the confidence and the other side the support of a rule. Thus, the area of the rectangle represents the commonness of the two values. Also the Kiviat graphs [7, 8] could be useful for rule visualization.

With the Rule Visualizer, the user can browse the rules in different modes: in textual, graphical, or combined mode (Figure 3). The graphs give a rough idea of the confidence and support even with a quick glance. Also, sometimes the user can find interesting rules simply by looking at the graphical patterns. Often, it is efficient to combine the textual and graphical representations of the rules. The user can first select interesting rules using their graphical patterns — which is a fast way to pick up exceptional cases — and then refine the selection in textual mode.

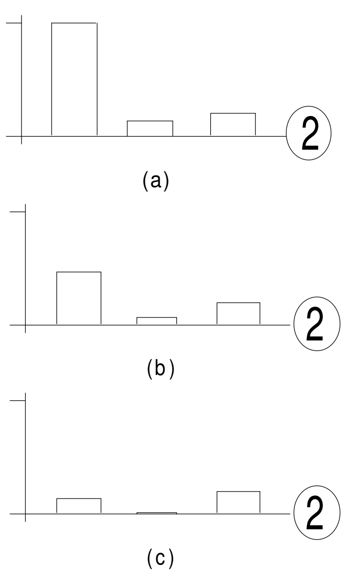

Figure 2: Example bar graphs.

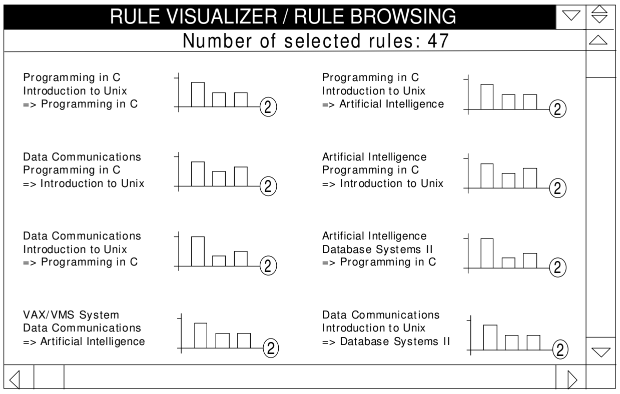

Figure 3: Rule Visualizer / Rule Browsing.

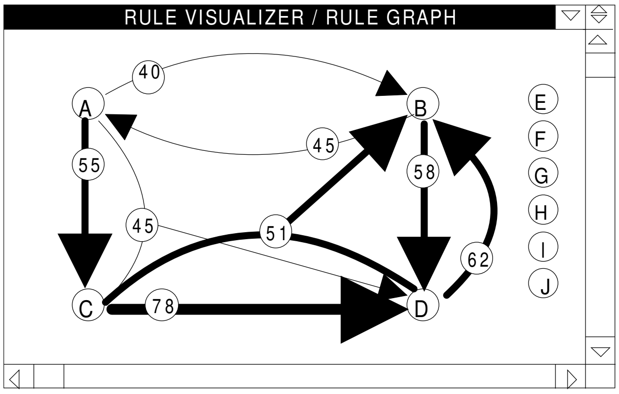

Figure 4: Rule Visualizer / Rule Graph.

Example 11 *Assume we have three rules with the representations in Figure 2. Graphs (a) and (c) represent rules with very high and low confidences, respectively. The advantages of the graphical representation are clear when compared to a textual representation:*

> Distributed Operating Systems, Introduction to Unix $\Rightarrow$ Programming in C (0.96, 0.02)
>
> Database Systems II, OODB $\Rightarrow$ Programming in C (0.53, 0.02)
>
> Data Communications, Introduction to Unix $\Rightarrow$ OODB (0.10, 0.02)

□

4.2 Showing several rules simultaneously

Visualizing a set of association rules is, of course, much harder than showing a single rule. Basically, the problem is to visualize a hypergraph, where an edge consists of all the attributes appearing in a rule, and each edge has a distinguished vertex (the attribute of the right-hand side) and two weights (support and confidence). The wide research in drawing and visualizing ordinary graphs shows that even that problem is by no means easy (see e.g. EDGE [11], DAG [4], or [15]); thus for the much harder problem of visualizing weighted hypergraphs we must be content with fairly simple solutions.

We concentrate on an attribute graph model; simpler graphical alternatives include techniques such as the tree model [10, 16]. The basic idea of the graph model is to represent attributes as nodes and associations as directed arcs (see Figure 4). The thickness of an arc represents the confidence or support of the corresponding rule. Colors can be used for visualizing additional information.

Node placing is a hard problem. Irrespective of the algorithm, even with a relatively small amount of attributes and association rules, the graph easily becomes very dense and hard to graph.

Rule Visualizer offers several ways to reduce the complexity of the graphical representation. The most important of them is the template mechanism, which offers a way of limiting the number of displayed rules. In the Rule Graph tool of Rule Visualizer, the user can manipulate rule templates directly using the graphical representation. Deleting a node from a graph equals to entering two restrictive templates: one with the attribute on the left-hand side and another with the attribute on the right hand side. In Figure 4, attributes E through J have been removed. Similarly, nodes can be marked interesting; this corresponds to the addition of two inclusive templates.

A simple but very powerful and useful way to limit the complexity is to let the user limit the rule sizes. For instance, the user may choose to view only rules with at most two left-hand side attributes.

Joining nodes is a third way to reduce complexity. The user can join nodes — typically a strongly connected cluster of them — to be represented by a single node of combined effect. If a node is joined with another node it is not obvious which of the connections of the joint node are strong enough to be visualized.

One way to reduce complexity is to use a variation of the so-called Spiders technique [3]. This technique allows multiple instances of some nodes. A special case would be to use separate nodes for the left-hand side attributes and the right-hand side attributes. Further on, in these cases, it could be possible to organize the nodes so that the left-hand sides of the rules are on the left and the right-hand sides on the right on the screen. However, the network effect would not be visible. We do not yet have the empirical evidence on the usefulness of such techniques.

5 Concluding remarks

Association rules are a simple and useful form of knowledge that can be efficiently discovered from large data sets. The collection of all association rules that holds in a given relation provides a lot of information about that data. The problem with this data mining technique is that the collection of all rules is typically very large, especially when one is interested also in the behavior of small subgroups of the original data set.

We have considered the problem of finding interesting rules from the collection of all association rules. Our starting observation is that although pruning based on the support and confidence thresholds is effective, it fails to take into account special interests or domain knowledge. We have described how the simple idea of classifying the attributes of the original data set to an inheritance hierarchy, and using templates defined in terms of that hierarchy, can be used to prune the rule sets effectively and according to the user's intuitions.

We have implemented and experimented with the template mechanism, and it has proved useful. We are currently working on a prototype system that also supports visualization of single rules and small rule sets.

A problem that remains is redundancy. Large amounts of rules could potentially be pruned, if there were appropriate ways to remove redundant or nearly redundant rules. Templates could be utilized also in a couple of other ways. To give an approximate order of interestingness to the discovered rules, inclusive templates could be given weights. Or, template-like high level rules could be used to summarize rules or to present more general knowledge. Automatic formation of clusters of strongly connected components would be a useful feature, especially for reducing the complexity of rule graphs.

References

[1] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 International Conference on Management of Data (SIGMOD 93)*, pages 207 – 216, May 1993.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB '94*, September 1994.

[3] G.H. Collier. Thot-II: Hypertext with explicit semantics. In *ACM Hypertext '87 Proceedings*, pages 269 – 289, Chapel Hill, North Carolina, November 1987.

[4] E. R. Gansner, S. C. North, and K. P. Vo. DAG — a program that draws directed graphs. *Software—Practice and Experience*, 18(11):1047 – 1062, November 1988.

[5] Jiawei Han, Yandong Cai, and Nick Cercone. Knowledge discovery in databases: an attribute-oriented approach. In *Proceedings of the 18th International Conference on Very Large Databases (VLDB)*, pages 547 – 559, August 1992.

[6] Peter Hoschka and Willi Klösgen. A support system for interpreting statistical data. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*, pages 325 – 345. AAAI Press / The MIT Press, Menlo Park, CA, 1991.

[7] K. W. Kolence. The software empiricist. *Performance Evaluation Review*, 2(2):31 – 36, 1973.

[8] K. W. Kolence and P. J. Kiviat. Software unit profiles and Kiviat figures. *Performance Evaluation Review*, 2(3):2 – 12, 1973.

[9] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Efficient algorithms for discovering association rules. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *AAAI Workshop on Knowledge Discovery in Databases*, pages 181 – 192, Seattle, Washington, July 1994.

[10] S. Moen. Drawing dynamic trees. *IEEE Software*, 7(4):21 – 28, July 1990.

[11] F. N. Paulisch and W. F. Tichy. EDGE: An extendible graph editor. *Software—Practice and Experience*, 20(S1):63 – 88, June 1990.

[12] Gregory Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*, pages 229 – 248. AAAI Press / The MIT Press, Menlo Park, CA, 1991.

[13] Gregory Piatetsky-Shapiro and William J. Frawley, editors. *Knowledge Discovery in Databases*. AAAI Press / The MIT Press, Menlo Park, CA, 1991.

[14] Gregory Piatetsky-Shapiro and Christopher J. Matheus. The interestingness of deviations. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *AAAI Workshop on Knowledge Discovery in Databases*, pages 25 – 36, Seattle, Washington, July 1994.

[15] L. A. Rowe, M. Davis, E. Messinger, C. Meyer, C. Spirakis, and A. Tuan. A browser for directed graphs. *Software—Practice and Experience*, 17(1):61 – 76, January 1987.

[16] J. Q. Walker II. A node-positioning algorithm for general trees. *Software—Practice and Experience*, 20(7):685 – 705, July 1990.