Data Mining with Optimized Two-Dimensional Association Rules

TAKESHI FUKUDA AND YASUHIKO MORIMOTO
IBM Tokyo Research Laboratory
SHINICHI MORISHITA
University of Tokyo
and
TAKESHI TOKUYAMA
IBM Tokyo Research Laboratory

We discuss data mining based on association rules for two numeric attributes and one Boolean attribute. For example, in a database of bank customers, "*Age*" and "*Balance*" are two numeric attributes, and "*CardLoan*" is a Boolean attribute. Taking the pair (*Age, Balance*) as a point in two-dimensional space, we consider an association rule of the form

$$((Age, Balance) \in P) \Rightarrow (CardLoan = Yes),$$

which implies that bank customers whose ages and balances fall within a planar region P tend to take out credit card loans with a high probability. We consider two classes of regions, rectangles and *admissible* (i.e., connected and x-monotone) regions. For each class, we propose efficient algorithms for computing the regions that give optimal association rules for *gain*, *support*, and *confidence*, respectively. We have implemented the algorithms for admissible regions as well as several advanced functions based on them in our data mining system named SONAR (System for Optimized Numeric Association Rules), where the rules are visualized by using a graphic user interface to make it easy for users to gain an intuitive understanding of rules.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems–*geometrical problems and computations*; G.2.1 [**Discrete Mathematics**]: Combinatorics–*combinatorial algorithms*; H.2.4 [**Database Management**]:

This work is based on the extended abstract of "Data Mining using Two-Dimensional Optimized Association Rules: Scheme, Algorithms, and Visualization" in *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data* (Montreal, Que., Canada, June 4–6), ACM, New York, 1996, pp. 13–23.
This work was partially supported by the Advanced Software Enrichment Project of the Information Technology Promotion Agency of Japan.
Authors' addresses: T. Fukuda, Y. Morimoto, and T. Tokuyama, IBM Tokyo Research Laboratory, 1623-14, Shimo-tsuruma, Yamato City, Kanagawa 242, Japan, e-mail: {fukudat; morimoto; ttoku}@trl.ibm.co.jp; S. Morishita, Department of Computer Science, Office 301, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan, e-mail: ShinichiMorishita@acm.org.

Systems–*query processing*; H.3.1 [**Information Storage and Retrieval**]: Information Search Retrieval–*search process*; I.2.6 [**Artificial Intelligence**]: Learning–*knowledge acquisition*

General Terms: Algorithms, Management

Additional Key Words and Phrases: Association rules, convex hull searching, data mining, image segmentation, matrix searching

1. INTRODUCTION

Recent progress in technologies for data input through such media as bar-coded labels, credit cards, optical character readers (OCRs), and cash dispensers have made it easier for finance and retail organizations to collect massive amounts of data and to store them on disk at a low cost. Such organizations are interested in extracting from these huge databases unknown information that inspires new marketing strategies. Current database systems are their primary means of realizing this aim, but in the database and AI communities, there has been a growing interest in efficient discovery of interesting rules, which is beyond the power of current database functions [Agrawal et al. 1992; 1993a; Breiman et al. 1984; Han et al. 1992; Ng and Han 1994; Piatetsky-Shapiro and Frawley 1991; Quinlan 1986; 1993; Stonebraker et al. 1993].

1.1 Association Rules

Given a database universal relation, we consider the association rule that if a tuple meets a condition C_1, then it also satisfies another condition C_2 with a certain probability (called a *confidence* in this paper). We denote such an association rule (or rule, for short) between the presumptive condition C_1 and the objective condition C_2 by $C_1 \Rightarrow C_2$.[1]

Agrawal et al. [1993b] investigated how to find all rules whose confidences are greater than a specified minimum threshold, such as 50%. They focused on rules with conditions that are conjunctions of $(A = yes)$, where A is a Boolean attribute, and presented an efficient algorithm, which they have applied to basket-data-type retail transactions in order to derive interesting associations between items, such as

$$(Pizza = yes) \wedge (Coke = yes) \Rightarrow (Potato = yes).$$

Improved versions of the algorithm have also been reported [Agrawal and Srikant 1994; Park et al. 1995].

In addition to Boolean attributes, databases in the real world usually have numeric attributes; in a database of bank customers, for instance, these might be age and current account balance. Thus, it is also important to find association rules for numeric attributes. In a companion paper [Fukuda et al. 1996b], we considered the problem of finding simple rules of the form

$$(Balance \in [v_1, v_2]) \Rightarrow (CardLoan = yes).$$

[1]We use the symbol "$\Rightarrow$" in order to distinguish the relationship from logical implication, which is usually denoted by "$\rightarrow$".

This example indicates that customers whose balances fall in the range between v_1 and v_2 are likely to take out credit card loans. If the confidence of this rule exceeds a given threshold θ (say, 10%), the range (i.e., interval) $[v_1, v_2]$ is called a *confident range* of the numeric attribute value "*Balance*" with respect to the Boolean attribute "*CardLoan.*" If there are many confident ranges, we want to obtain one with high *support* (the number of tuples in the range). For this purpose, Fukuda et al. [1996b] introduced some optimization criteria that effectively represent the relation between a numeric attribute and a Boolean one.

1.2 Two-Dimensional Association Rules

In the real world, binary associations between two attributes are usually not enough to describe the characteristics of a data set, and therefore, we often want to find a rule for more than two attributes.

The main aim of this paper is to generate rules (which we call *two-dimensional association rules*) that represent the dependence on a pair of numeric attributes of the probability that an objective condition (corresponding to a Boolean attribute) will be met.

For each tuple t, let $t[A]$ and $t[B]$ be its values for two numeric attributes; for example, $t[A]$ = "Age of a customer t" and $t[B]$ = "Balance of t". Then, t is mapped to a point $(t[A], t[B])$ in an Euclidean plane E^2. For a region P in E^2, we say a tuple t meets condition "$(Age, Balance) \in P$" if t is mapped to a point in P. We want to find a rule of the form $((A, B) \in P) \Rightarrow C$, such as

$$((Age, Balance) \in P) \Rightarrow (CardLoan = yes).$$

In practice, we consider a huge database containing millions of tuples, and hence we have to handle millions of points, which may occupy much more space than the main memory. To avoid dealing with such a large number of points, we discretize the problem; that is, we distribute the values for each numeric attribute into N equal-sized buckets. We divide the Euclidean plane into $N \times N$ pixels (unit squares), and we map each tuple t to the pixel containing the point $(t[A], t[B])$. We use a union of pixels as the region of a two-dimensional association rule.

The probability that the tuples in a pixel (respectively, a region) satisfy the objective condition C is called the *confidence* of the pixel (region). We would like to find a *confident* region whose confidence is above some threshold.

The shape of region P is important for obtaining a good association rule. For instance, if we gather all the pixels whose confidence is above some threshold, and define P to be the union of these pixels, then P is a confident region with (usually) a high support. A query system of this type is proposed by Keim et al. [1994]. However, such a region P may consist of many connected components, creating an association rule that is very difficult to characterize, and whose validity is hence hard to see. Therefore, in order to obtain a rule that can be stated briefly or characterized through visualization, it is required that P should belong to a class of regions that have nice geometric properties.

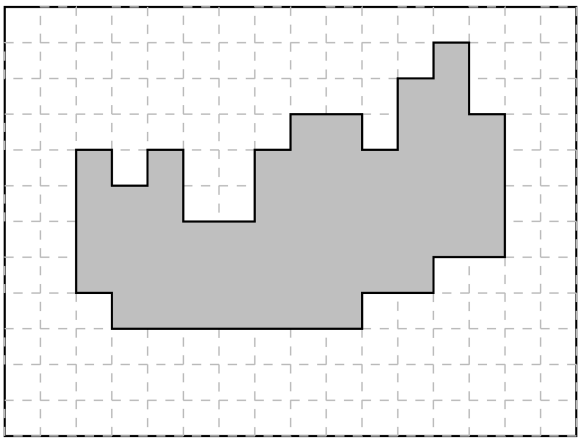

Fig. 1. Admissible region.

1.3 Main Results

The problem of extracting a confident region resembles that of image segmentation in computer vision. Although there are many known algorithms for image segmentation, we need one that outputs an image region capable of generating a good (or, to be precise, an optimized) association rule.

We consider two classes of geometric regions: rectangular regions and *admissible regions*, which are connected *x-monotone regions*. A region is called x-monotone if its intersection with any vertical line is undivided. Figure 1 shows an instance of an admissible region.

We generalize an algorithm introduced by Asano et al. [1996], which segments an object image from the background in a gray image, to our color-image case, and obtain a linear-time algorithm for computing the *focused region*, which is the x-monotone region that maximizes the *gain* (see Section 2 for a definition). Although this algorithm uses sophisticated dynamic programming with fast matrix-searching [Aggarwal et al. 1987], we have confirmed experimentally that our implementation is fast not only in theory but also in practice. If we consider rectangular regions instead of x-monotone regions, the computation time for the optimal gain rule is increased to $O(n^{1.5})$, where n is the number of pixels in the grid.

Next, we give efficient approximation algorithms for generating optimized-support rules and optimized-confidence rules (defined in later sections), through the use of focused regions.

1.4 Visualization

It is convenient to visualize the region of a two-dimensional association rule $((A, B) \in P) \Rightarrow C$. Let us regard the region P as a color image on a grid G as follows: Let $u_{i,j}$ and $v_{i,j}$ denote the total number of tuples and the number of tuples satisfying the objective condition C in the (i, j)th pixel $G(i, j)$, respectively. The pixel $G(i, j)$ has a color vector $(v_{i,j}, u_{i,j} - v_{i,j}, 0)$ in RGB space. This means that its red level is $v_{i,j}$, its green level is $u_{i,j} - v_{i,j}$, and its blue level is 0.

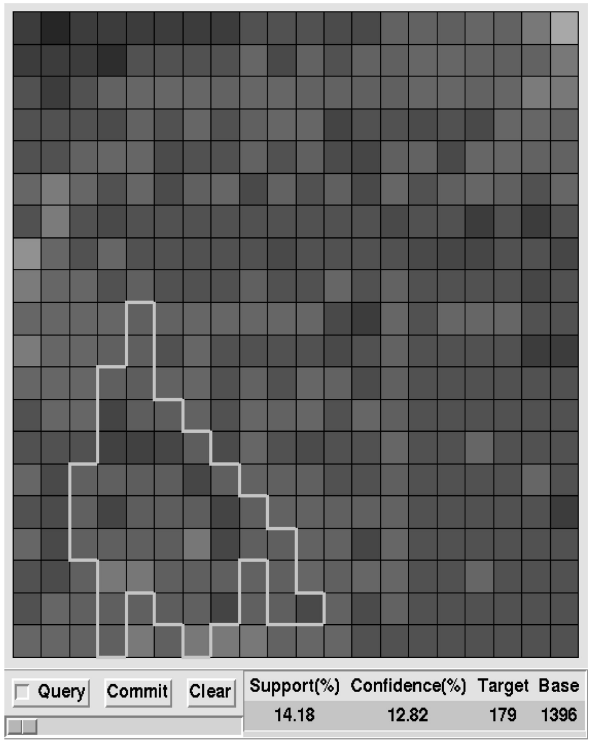

Fig. 2. Visualization system.

Hence, the brightness level is $u_{i,j}$, which represents the number of tuples that fall within the pixel. The confidence of each pixel is represented by its color; thus, a redder pixel has a higher confidence.

We construct a visualization system for our two-dimensional association rules by reforming the color image introduced above, transforming it to make the rules easier for humans to grasp visually. We have tested our system, together with the functions given in our companion paper [Fukuda et al. 1996b] on a test database, and discovered several simple new rules, some of which seem to be of potential value in strategy planning.

The database has about 30 numeric attributes and about 100 Boolean attributes. Suppose that we select two numerical attributes, "*Age*" and "*Balance*," and also a Boolean attribute, "*CardLoanDelay*." These attributes represent the ages of customers, the balances of their accounts, and whether they have delayed payment of their credit card charges, respectively. In order to characterize unreliable customers who have delayed payment, we consider the rule

$$((Age,\ Balance) \in P) \Rightarrow (CardLoanDelay = yes),$$

and find an optimized region of P. We divide data into sets of 20×20 pixels. Our visualization system shows an optimized region in those pixels, as in Figure 2. The data has an average confidence of 3.51%, which means that 3.51% of customers have delayed their credit card payments at one time or another.

By pushing the Query button, we can display a focused region enclosed in thick lines. The confidence and the support (the percentage of tuples in the region) can be found at the bottom of the window. By moving the slider, we can control the trade-off between the support and the confidence: if we move the slider to the left, the support increases while the confidence decreases. In response to such a movement, the visualization system recomputes the region together with its support and confidence. Our system is so fast that one can continuously move the slider and see how the region changes, as if one were watching a motion picture.

In Figure 2, we can find a region with 14.1% of support and 12.8% of confidence, which is much higher than the average confidence. The shape of the region tells us the characteristics of unreliable customers. As a result, we can say, "Unreliable customers, who have delayed payment of their card charges, can be characterized as relatively young people whose balance is low."

1.5 Advanced Functions

In our data mining system SONAR, we also designed several advanced functions based on the optimized association rules. One important function computes the optimized *rectilinear convex region rule* with the aim of outputting more intuitive and learnable rules. Other major functions are decision making functions to compute decision trees and regression trees handling the combined effect of more than two conditional attributes. By using modified versions of our two-dimensional association rules as the branching rules, we can obtain accurate decision trees and regression trees.

2. RELATED WORK

Interrelation among paired numeric attributes constitutes a major research topic in statistics; for example, covariance and line estimators are well-known tools for representing interrelations. However, these tools only show interrelations in an entire data set, and thus cannot extract a subset of data in which a strong interrelation holds.

In order to extract strong local interrelations, several heuristics using geometric clustering techniques have been introduced [Ng and Han 1994].

When we compute a two-dimensional association rule, we could regard the Boolean attribute in the objective condition as a special numeric attribute, and apply three-dimensional clustering methods to extract some interrelations. However, this approach has serious defects. The three attributes we have discussed do not have equivalent roles: the Boolean attribute corresponds to the objective condition, whereas the others give presumptive conditions. Therefore, a clustering method with respect to a three-dimensional proximity criterion does not find a good rule. The other defect is that clustering algorithms in three-dimensional space are often time-consuming. For example, if we want to compute the three-dimensional unit ball that contains the maximum number of points in a given set of n points, it will take $O(n^3)$ time if we use a standard computational geometric algorithm.

Some other studies have also handled numeric attributes and tried to derive rules. Piatetsky-Shapiro [1991] investigated how to sort the values of a numeric attribute, divide the sorted values into approximately equal-sized ranges, and use only those fixed ranges to derive rules whose confidences are almost 100%. His framework does not take account of any ranges other than the fixed ones. Our method is not only capable of outputting optimized ranges, but is also more handy than Piatetsky-Shapiro's method, since we do not need to create candidate ranges beforehand. Recently, Srikant and Agrawal [1996] improved Piatetsky-Shapiro's method by adding a way of combining consecutive ranges into a single range. The combined range could be the whole range of the numeric attributes, in which case the generated rule is trivial. To avoid this, Srikant and Agrawal [1996] presented an efficient way of computing a combined range whose size is at most a threshold given by the user. This method can discover association rules with more than two numeric attributes, while our method can handle at most two. On the other hand, their method uses only axis-parallel rectangles (hypercubes in high-dimensional cases).

Some techniques, related to but not directly applicable to finding optimized association rules, have been developed for handling numeric attributes in the context of deriving decision trees that are used for classifying data into distinct groups. ID3 [Quinlan 1986; 1993], CART [Breiman et al. 1984], $\mathcal{CDP}$ [Agrawal et al. 1993b], and SLIQ [Mehta et al. 1996] perform binary partitioning of numeric attributes repeatedly until each range contains data of one specific group with high probability, while $\mathcal{IC}$ [Agrawal et al. 1992] uses k decomposition. Since both partitionings tend to yield large decision trees, methods such as pruning some branches and linking some ranges together have also been proposed, to reduce the size of the trees.

Considerable work has been done on the segmentation of intensity images, and five different approaches have mainly been used: threshold techniques, edge-based methods, region-based methods, hybrid techniques, and connectivity-preserving relaxation methods (e.g., Zucker [1976], Haralick and Shapiro [1985], Besl and Jain [1988], Sahoo et al. [1988], and Kass et al. [1988]). Some of these methods, especially region-based methods and connectivity-preserving relaxation methods, can be used for finding two-dimensional association rules. One of the disadvantages of the above methods is that they search heuristically for regions that "look good to humans," while our method can find the optimal region rules with respect to natural objective criteria.

3. PRELIMINARIES

3.1 Association Rules

In this paper, we consider association rules, which are stochastic relations between some conditions on tuples in a database. We consider some primitive conditions, and use them to describe more complicated conditions. For a Boolean attribute A, $A = yes$ and $A = no$ are primitive conditions. Our typical primitive conditions on numeric attributes A and B are $A = v$, $A \in I$, and $(A, B) \in P$,

where v is a value and I is a range in the domain of A, and P is a region of the product space of domains of A and B.

Let C_1 and C_2 be conditions on tuples. An *association rule* (or a *rule* for short) has the form $C_1 \Rightarrow C_2$. C_2 is called an *objective condition*, and C_1 is called a *presumptive condition*.

The *support* of condition C is defined as the percentage of tuples that meet condition C, and is denoted by support(C). In this paper, we normally denote the "number of tuples" that meet condition C by support(C) rather than "percentage," since we do not want to declare the base of the percentage each time. The *confidence* of a rule $C_1 \Rightarrow C_2$ is defined as support($C_1 \wedge C_2$)/support(C_1), which is denoted by conf($C_1 \Rightarrow C_2$).

There are several types of association rules for attributes. Our first example is a rule where the presumptive condition is a primitive condition on a Boolean attribute.

Example 3.1. Consider a relation of basket-type retail transactions. Each attribute of the relation is a Boolean one whose domain is {*yes, no*}, and represents an item, such as *Coke* or *Pizza*.

$$(Coke = yes) \Rightarrow (Pizza = yes)$$

is an association rule.

The second example is what we call a *one-dimensional association rule*, where the presumptive condition is on a numerical attribute.

Example 3.2. Consider a bank's data on customers. Each tuple contains a customer's balance and the services (credit card loan or automatic withdrawal, say) for which the customer is eligible. Suppose that 50% of customers whose balance is between 10^4 and 10^5 use credit card loans; then, we have the following rule, whose confidence is 50%:

$$(Balance \in [10^4, 10^5]) \Rightarrow (CardLoan = yes).$$

The third example is a rule where the presumptive condition is a condition on two numerical attributes:

Example 3.3. In a bank's data on customers, we consider another numeric attribute, *Age*. Consider the following rule:

$$\{(Balance \in [10^4, 10^5]) \wedge (Age \in [40, 50])\} \Rightarrow (CardLoan = yes).$$

The presumptive condition of the above association rule can be rewritten as a primitive condition:

$$(Balance, Age) \in P,$$

where P is a rectangular region $[10^4, 10^5] \times [40, 50]$ in the plane, which is the product space of the domains of Balance and Age. In rules of this type, which we call *two-dimensional association rules*, P is not necessarily a rectangle.

3.2 One-Dimensional Association Rules

It is important to mine association rules whose confidence is high, and whose support is sufficiently large. In this subsection, we define optimization criteria for the range I in a one-dimensional association rule $(A \in I) \Rightarrow C$, where A is a fixed numeric attribute and C is a fixed objective condition.

For a tuple t, we define $t[A]$ to be the value of A at t. A tuple t is called a *success* tuple if the condition C holds for it. For simplicity, we write support(I) for support($A \in I$), and hit(I) for support($(A \in I) \wedge C$); they are called the *support* and *hit support* of I, respectively. support(I) is the number of tuples in I and hit(I) is the number of success tuples in I. We write conf(I) for hit(I)/support(I) and call it the *confidence* of I.

Since there are often too many tuples in a large database to be accommodated in main memory, we usually compress the data into N buckets $B_1, B_2, \ldots, B_N$, by using an ordered disjoint bucketing so that for each $t \in B_i$ and $t' \in B_j$ where $i < j$, $t[A] \leq t'[A]$. Let u_i denote the number of tuples in B_i, and v_i denote the number of success tuples in B_i (i.e., $u_i = \text{support}(B_i)$ and $v_i = \text{hit}(B_i)$). It is desirable that the buckets should be (almost) equal-sized; that is to say, tuples should be uniformly distributed into buckets. A fast method of performing such a bucketing is given in Fukuda et al. [1996b].

From now on, we assume that we have equal-sized bucketing, and we consider a range only if it corresponds to a union of contiguously indexed buckets. For simplicity, we denote the range corresponding to $B_s \cup B_{s+1} \cup \cdots \cup B_t$ by $[s, t]$. For a range $I = [s, t]$, we define

$$\text{support}(I) = \sum_{i=s}^{t} u_i \qquad \text{and} \qquad \text{hit}(I) = \sum_{i=s}^{t} v_i.$$

From all ranges, we want to compute characteristic ones that optimize some criteria. Among them, let us consider the *optimized-gain*, *optimized-support*, and *optimized-confidence* ranges.

Definition 3.1. Given a confidence threshold θ, we define the gain of a range I as

$$\text{gain}_\theta(I) = \text{hit}(I) - \theta \times \text{support}(I).$$

The *optimized-gain* range is the range I whose gain $\text{gain}_\theta(I)$ is maximized among all ranges. We often omit the subscript θ if we fix the threshold.

Definition 3.2. Given a confidence threshold c, a range I is *confident* if its confidence conf(I) is not less than c. The *optimized-support* range is the confident range that maximizes support.

Definition 3.3. Given a support threshold s, a range I is *ample* if its support support(I) is not less than s. The *optimized-confidence* range is the ample range that maximizes confidence.

For example, let us consider the rule

$$(Balance \in I) \Rightarrow (CardLoan = yes)$$

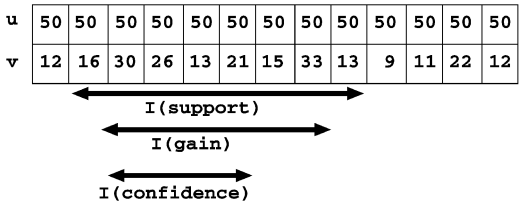

Fig. 3. Optimized ranges.

and suppose that the credit card loan system pays if $100 \times \theta\%$ of customers use credit card loans. Then, the optimized-gain range is the range of customers (with respect to the balance) that maximizes the bank's profit from the credit card loan system. On the other hand, the optimized-support range is the largest range of customers for which the bank does not lose money on the credit card loan system. When there is a fixed number of customers who can be targeted for a promotion, the optimized-confidence range captures the segment of customers whose response rate is maximized.

In Figure 3, $I(gain)$ and $I(support)$ are the optimized-gain and the optimized-support ranges in which the confidence threshold is 0.4. $I(confidence)$ is the optimized-confidence range in which the support threshold is 200.

Each of the optimized-gain, -support, and -confidence ranges can be computed in $O(N)$ time. In the "Programming Pearls column of *Communications of the ACM*, Bentley [1984] presented Kadane's algorithm for computing the optimized-gain ranges. Kadane's algorithm can be found in the appendix of this paper. Fukuda et al. [1996b] gave linear-time algorithms for computing the optimized-support ranges and optimized-confidence ranges.

THEOREM 3.1. *Each of the optimized-gain, -support, and -confidence ranges can be computed in $O(N)$ time.*

3.3 Two-Dimensional Association Rules

In practice, since we consider a huge database containing millions of tuples, we have to handle millions of points, which may occupy much more space than the main memory. To avoid this, we discretize the problem. Let us consider two numeric attributes A and B, and an objective condition C. We distribute the values of A and B into N_A and N_B equal-sized buckets, respectively. Let us consider a two-dimensional $N_A \times N_B$ pixel-grid G, consisting of $N_A \times N_B$ unit squares called *pixels*. $G(i, j)$ is the (i, j)th pixel, where i and j are called the *row number* and *column number*, respectively. The jth *column* $G(*, j)$ of G is its subset consisting of all pixels whose column numbers are j. Geometrically, a column is a vertical stripe. We use the notation $n = N_A \times N_B$. In our typical applications, the ranges of N_A and N_B are from 20 to 500, and thus n is between 400 and 250,000. For simplicity, we assume from now on that $N_A = N_B = N$, although this assumption is not essential. For a set of pixels, the union of pixels in it forms a planar region, which we call a *pixel region*. A pixel region is x-monotone if its intersection with each column is undivided (and thus a vertical

range) or empty. A connected and x-monotone region is called an *admissible region*.

For each tuple t, $t[A]$ and $t[B]$ are values of the numeric attributes A and B at t. If $t[A]$ is in the ith bucket and $t[B]$ is in the jth bucket in the respective bucketings, we define $f(t) = G(i, j)$. Then, we have a mapping f from the set of all tuples to the grid G.

For each pixel $G(i, j)$, let $u_{i,j}$ be the number of tuples mapped to $G(i, j)$. As in the case of one-dimensional rules, a tuple is called a *success* tuple if the objective condition C holds for it. Let $v_{i,j}$ denote the number of success tuples mapped to $G(i, j)$. Given a region P, we define

$$\text{support}(P) = \sum_{G(i,j)\in P} u_{i,j}, \qquad \text{and} \qquad \text{hit}(P) = \sum_{G(i,j)\in P} v_{i,j}.$$

We write $\text{conf}(P)$ for $\text{hit}(P)/\text{support}(P)$ and call it the confidence of P. Given a threshold θ, we define

$$\begin{aligned} g(\theta)_{i,j} &= v_{i,j} - \theta \times u_{i,j} \\ \text{gain}(P) &= \text{hit}(P) - \theta \times \text{support}(P) = \sum_{G(i,j)\in P} g(\theta)_{i,j}. \end{aligned}$$

We want to find the *two-dimensional optimized-gain rule*

$$(f(t) \in P) \Rightarrow C,$$

where P is the admissible region that maximizes $\text{gain}(P)$. The region is called an *optimized-gain admissible region*.

As in the case of ranges, we call a region *confident* (respectively, *ample*) if its confidence (respectively, support) is not less than a given threshold value. Similarly, we define an *optimized-support admissible region*, which is the confident and admissible region that maximizes support, and an *optimized-confidence admissible region*, which is the ample and admissible region that maximizes confidence.

Similarly, the rectangular subregion W of G that maximizes $\text{gain}(W)$ (respectively, $\text{support}(W)$, $\text{conf}(W)$) is called the *optimized-gain* (respectively, *-support, -confidence*) rectangle.

Note that, if we want to find the connected pixel region with the maximum gain, rather than an admissible region or rectangle, the problem becomes NP-hard, in line with the argument used by Garey and Johnson [1977] for the grid Steiner tree problem.

4. ALGORITHMS FOR COMPUTING OPTIMIZED REGIONS

4.1 Optimized Rectangles

There are $O(N^4)$ rectangular subregions of G. Thus, a naive algorithm computes the gain of each of these $O(N^4)$ rectangles and outputs the one with the maximum gain. The time complexity of this algorithm is $O(N^5) = O(n^{2.5})$, which is too expensive. It can be easily reduced to to $O(n^{1.5})$ by transforming the problem into the computation of optimized ranges. We choose a pair $r < r'$

of rows in G, and consider only rectangles whose horizontal edges are on these rows. For each column index j, we define

$$u_j = \sum_{i=r}^{r'} u_{i,j} \qquad \text{and} \qquad v_j = \sum_{i=r}^{r'} v_{i,j}$$

Consider buckets $B_1, B_2, \ldots, B_N$ such that the number of tuples in B_j is u_j and the number of success tuples in B_j is v_j. From Theorem 3.1, we can compute the optimized-gain, -support, and -confidence ranges in $O(N)$ time. Since there are $O(N^2)$ candidate pairs of rows, the optimize gain, support, and confidence rectangles can be computed in $O(N^3) = O(n^{1.5})$.

THEOREM 4.1. *Each of the optimized-gain, -support, and -confidence rectangles can be computed in* $O(n^{1.5})$ *time.*

For the case of the optimized-gain rectangles, the same time complexity can be found in Fischer et al. [1993]. Further improvement of this time complexity is given as a research problem in Programming Pearls [Bentley 1984].

This time complexity is a little higher than that for the optimized admissible regions, and thus a data-mining system using rectangles is considerably slower than one using admissible regions. Furthermore, in our experience, the use of non-rectangular regions often yields useful rules. For example, let us consider "*Age*" and "*Salary*" as the numeric attributes, and "*GoldCard*" as the objective condition. Here, we would expect to find a rule that, among people on the same salary, younger ones are more likely to pay an annual fee for premium credit cards. This expectation is confirmed if we find a two-dimensional association rule whose region resembles a triangle, which is an admissible region, but (of course) not a rectangle. Consequently, we believe that admissible regions are better than rectangles for our class of regions in two-dimensional association rules.

4.2 Optimized-gain Admissible Regions

Since the intersection of an admissible region with a column is an interval, it would appear that if we compute the maximum gain range in each column and compute the union of all those ranges, we can compute the optimized-gain admissible region. Unfortunately, this region is often disconnected, although the connectivity of a region is very important for creating a good rule. The following algorithm is essentially the same as that given by Asano et al. [1996] for solving an image segmentation problem. However, to the best of our knowledge, this is the first case in which an algorithm using "fast matrix searching" [Aggarwal 1987] routines has been implemented in a database system.

For each $m = 1, 2, \ldots, N$, we precompute the indices $bottom_m(s)$ and $top_m(s)$ for all $1 \le s \le N$, where $bottom_m(s)$ and $top_m(s)$ are defined so that

$$\sum_{i=bottom_m(s)}^{s} g(\theta)_{i,m} \qquad \text{and} \qquad \sum_{i=s}^{top_m(s)} g(\theta)_{i,m}$$

are maximized, respectively.

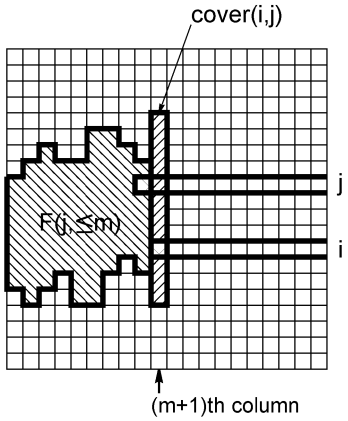

Fig. 4. $F(i, \leq m+1)$ and the associated region.

LEMMA 4.1. *The indices $top_m(s)$ and $bottom_m(s)$ for all $s = 1, 2, \ldots, N$ can be computed in $O(N)$ time.*

PROOF. Define $Sum_m[<j] = \sum_{i=1}^{j-1} g(\theta)_{i,m}$ for $2 \leq j \leq N$, and $Sum_m[<1] = 0$. $bottom_m(s)$ maximizes

$$\sum_{i=bottom_m(s)}^{s} g(\theta)_{i,m} = Sum_m[<(s+1)] - Sum_m[<bottom_m(s)],$$

which is equivalent to the property that $bottom_m(s)$ minimizes $Sum_m[<bottom_m(s)]$. Define $bottom_m(1) = 1$. For each $s = 2, \ldots, N$, compute $bottom_m(s)$ from $bottom_m(s-1)$ in the following manner:

–If $Sum_m[<s] < Sum_m[<bottom_m(s-1)]$, $Sum_m[<bottom_m(s)]$ is minimum when $bottom_m(s) = s$.

–Otherwise, $Sum_m[<bottom_m(s)]$ is minimum when $bottom_m(s) = bottom_m(s-1)$.

During the above step, we only need to scan $g(\theta)_{i,m}$ once for each $i = 1, \ldots, N$.

The computation of $top_m(s)$ is analogous. □

For two indices s and s', we define $cover_m(s, s')$ as follows:

$$cover_m(s, s') = \begin{cases} \sum_{i=bottom_m(s)}^{top_m(s')} g(\theta)_{i,m} & \text{if} \quad s \leq s' \\ \sum_{i=bottom_m(s')}^{top_m(s)} g(\theta)_{i,m} & \text{if} \quad s > s'. \end{cases}$$

Let $G(*, \leq m)$ be the part of the grid on the left of the mth column, including $G(*, m)$. We define $F(i, \leq m)$ to be the maximum gain of admissible regions that contain the pixel $G(i, m)$ and are contained in the region $G(*, \leq m)$. Then, we have the following formula:

$$F(i, \leq m+1) = \max_{1 \leq j \leq N} \{\max(F(j, \leq m), 0) + cover_{m+1}(i, j)\}. \tag{1}$$

When $F(j, \leq m)$ is negative, we do not connect the mth and $(m+1)$th columns. Figure 4 illustrates this formula and the associated region. By using

Formula (1), we can compute $\max_m\{\max_i F(i, \leq m)\}$ and the associated region, which must be the optimized-gain region.

LEMMA 4.2. *Suppose that $F(j, \leq m)$ is given for $j = 1, 2, \ldots, N$. We can compute $F(i, \leq m+1)$ for all $i = 1, 2, \ldots, N$ in $O(N)$ time.*

PROOF. Define $D(i, j) = F(j, \leq m) + cover_m(i, j)$. We can see that the upper and lower triangle parts (D^+ and D^-, respectively) of the matrix D are *totally monotone matrices*.

A matrix M is called *totally monotone* if

$$M(i, j) + M(i+1, j+1) \geq M(i, j+1) + M(i+1, j)$$

for every $1 \leq i < j+1 \leq N$. It is well known [Aggarwal 1987] that all locations of the row maxima of this matrix can be computed in $O(N)$ time (of course, we cannot afford to construct the matrix in order to obtain this time complexity).

Thus, in $O(N)$ time, we can compute all the row maxima of D^+ and D^-, and consequently, of D. By definition, $F(i, \leq m+1)$ is the ith row maximum of D. □

THEOREM 4.2. *The optimized-gain admissible region for a threshold θ can be computed in $O(n)$ time.*

PROOF. We solve Formula (1) for $m = 1, 2, \ldots, N$. This requires $O(N^2) = O(n)$ time. □

4.3 Other Optimized Admissible Regions

Unfortunately, if we consider admissible regions, the optimized-support region and the optimized-confidence region are difficult to compute. Observe that the number of all admissible regions is greater than N^N. Let M denote the total number of tuples. For each type of optimized region, we know only an $O(n^{1.5}M)$-time algorithm (see the Appendix). Moreover, it can be shown that no algorithm running in polynomial time with respect to n and $\log M$ exists unless $P = NP$ (see the Appendix).

In this section, we substitute new optimization criteria for the optimized-support and optimized-confidence criteria. We compute admissible regions that closely approximate the optimized-support/-confidence ones. For each admissible region P, we define a *stamp* point (support(P), hit(P)). We make the following convexity assumption:

> *Convexity Assumption.* Given three admissible regions P_1, P_2, and P_3, let (x_i, y_i) be the stamp point of P_i for $i = 1, 2, 3$. If $x_1 \leq x_2 \leq x_3$ and $y_2 \leq y_1 + (y_3 - y_1)(x_2 - x_1)/(x_3 - x_1)$, we can substitute P_1 or P_3 for P_2 to create a useful association rule. See Figure 5.

In other words, if the stamp point (x_2, y_2) lies below or on the line through stamp points (x_1, y_1) and (x_3, y_3), we do not use the region P_2 to create a rule. The convexity assumption cannot be theoretically confirmed, since the usefulness of an association rule is not a mathematical concept. In practice, however, we have a huge number of stamp points that are fairly densely scattered on the upper convex hull of all stamp points in the Euclidean plane (see Section 6

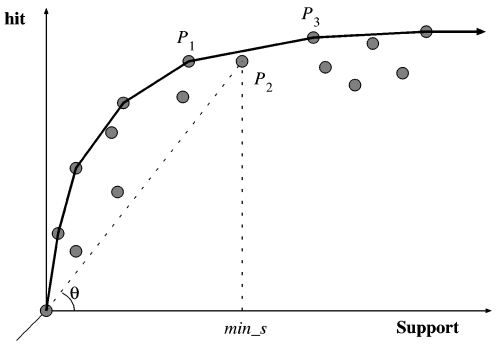

Fig. 5. Convexity assumption.

for an example), as we usually handle more than 20×20 pixels and a large number of data, and hence for any P_2 there exist points P_1 and P_3 fairly close to P_2. Thus, we believe that it is reasonable to use the convexity assumption for computing approximate solutions of optimized rules in a practical data-mining system.

Let us characterize the set of admissible regions that cannot be replaced by other regions according to the convex assumption. We call them *focused regions*, the name used by Asano et al. [1996] in the field of computer vision.

LEMMA 4.3. *An admissible region is focused if and only if it is an optimized-gain admissible region with respect to some confidence threshold.*

PROOF. We consider the set S of all stamp points associated with admissible regions. Because of the convexity assumption, a stamp point associated with a focused admissible region must be a point on the upper convex hull of S. Hence, there exists a tangent line to the convex hull of S at this point. Suppose that the tangent line has a slope τ. Then, this point maximizes $y - \tau x$ for the set of the stamp points. Accordingly, the corresponding region P maximizes $\text{hit}(P) - \tau \times \text{support}(P)$, and hence the optimized-gain admissible region with respect to the confidence threshold τ. □

For a given confidence threshold, the *focused optimized-support admissible region* is defined as the support-maximizing confident admissible region that is focused. For instance, in Figure 5, P_2 is the optimized-support admissible region for a confidence threshold θ. Since P_2 is hidden inside the convex hull and is not focused, P_1, the focused optimized-support admissible region, is substituted for it.

For a given support confidence Z, the *focused optimized-confidence region* is defined as the confidence-maximizing ample region that is focused. For instance, in Figure 5, P_2 is the optimized-confidence admissible region for a support threshold min_s, and P_2 is replaced by P_3, the focused optimized-confidence admissible region.

LEMMA 4.4. *The focused optimized-support region and the focused optimized-confidence region can be computed in $O(n \log M)$ time, where M is the support of the whole grid G.*

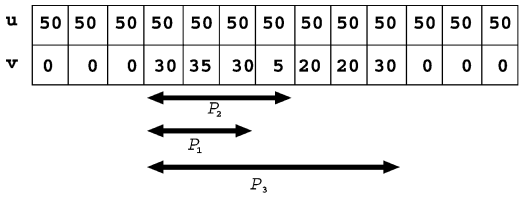

Fig. 6. Optimized support range and its approximation by focused ranges.

PROOF. For a tangent line with slope τ, we can compute the optimized-gain admissible region P for the confidence threshold τ in $O(n)$. Note that when τ increases, the confidence of P increases, and the support of P decreases monotonically. Thus, in order to search for the focused optimized-confidence/-support admissible region, we perform a binary search; that is, we (1) compute the region P_0 for $\tau = 0$ and the region P_1 for $\tau = 1$, (2) compute P_2 for the slope of the line P_0P_1, and (3) repeat this process until we find P' and P'' such that P' or P'' is the region for the slope of $P'P''$. Observe that no focused regions exist between P' and P''.

The above binary search seems to look for whole real numbers. However, since a stamp point has integer coordinate values, each slope τ is a rational number whose denominator and numerator are positive integers in $[1, M]$, and the difference of two such distinct rational numbers is at least $1/M^2$. Thus, we can stop the search if the width of the search range is reduced to $1/M^2$. Hence, the binary search terminates in $O(\log M)$ search steps. Since a focused region for a given threshold can be computed in $O(n)$ time, the time complexity is $O(n \log M)$. □

Let P_2 be the optimized-support admissible region for the confidence threshold θ. Suppose that P_2 is not focused. Then, let P_1 be the focused optimized-support admissible region for θ, and let P_3 be the focused optimized-confidence admissible region for the support of P_2. Figure 5 illustrates this situation. From the definition of the focused optimized-confidence/-support admissible region, P_1 and P_3 are the optimized-gain admissible regions associated with the slope of the line P_1P_3. Thus, the binary search for P_2, which is given in Lemma 4.4, computes P_1 and P_3 in the final step, and therefore we have the following:

LEMMA 4.5. *From the convexity assumption, P_1 and P_3 can replace P_2.*

The case when P_2 is the optimized-confidence region can be handled similarly.

Now we are interested in how close P_1 and P_3 are to P_2. A typical case is that in which $P_1 \subset P_2 \subset P_3$, as shown in Figure 6 (for the 1-dimensional case). If the confidence threshold is 0.5 (i.e., 50%), P_2 is the optimized-support range, which has support 200 and confidence 0.5. However, P_2 is not a focused range (i.e., a maximum-gain range), and instead the pair P_1 (support 150 and confidence 0.63) and P_3 (support 350 and confidence 0.485) are found as substitutes for P_2. Both P_1 and P_3 are maximum-gain ranges associated with a threshold $0.375 = (170 - 95)/(350 - 150)$, where the gain is 38.75. It can be observed that it is reasonable to choose P_1 or P_3 to make an association rule instead of P_2.

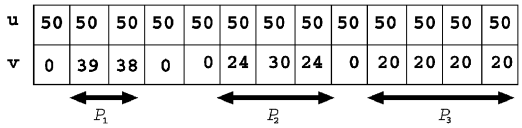

Fig. 7. A bad example.

Unfortunately, it can happen even in the 1-dimensional case that P_2 does not resemble P_1 or P_3. If the confidence threshold is 0.5 in the data of Figure 7, there is no intersection between P_2 and P_1 nor P_3. To overcome such an abnormal case, we could make P_1 a nonfocused region by using a heuristic to decrease the values of v for the pixels in P_1, and find a new focused region that gives a better approximation of P_2. This approach resembles the "cutting plane" method used in operations research [Nemhauser et al. 1989] to find a solution in the interior of a feasible region. However, we have not yet implemented this heuristic, since we have yet to encounter an optimized region that generates an important association rule in practical data.

5. VISUALIZATION

Our scheme for two-dimensional data mining transforms a set of tuples into a color image in a pixel grid G. It thus provides an immediate method for visualizing our association rules. Unfortunately, if we naively use $(v_{i,j}, u_{i,j} - v_{i,j}, 0)$ for the color vector of the (i, j)th pixel, it does not always give what humans would regard as a good image: the result is often too dark, and if the confidence threshold is low, the red level is too low for the difference in confidence to be distinguished. We must therefore give transformations that make our rules more visible. Since these transformations should depend on the display system, we do not have a universal formula. However, we have experimentally implemented a transformation method specialized for our demonstration system, so that users can actually see our rules.

In the "interactive mode" of our demonstration system, the user chooses attributes (from about 30 numeric attributes and 100 Boolean attributes), indicates either gain, support, or confidence in order to select a feature of the rule, and inputs a threshold. The system outputs the corresponding optimized region. If the rule is a known one and the user wants to find a secondary rule, the system can remove the obtained optimized region and find a secondary optimized region by applying the same algorithm to the rest of the pixel grid. We also have the "animated mode," in which the user can control the threshold (almost) continuously and see the changes in the rule. For this purpose, we must compute focused regions for many different thresholds on the fly. The efficiency of the algorithm for computing a focused region makes this approach practical.

6. PERFORMANCE

The algorithms in Section 4 have been written in C++, and implemented as functions in our DataBase SONAR (System for Optimized Numeric Association Rules) prototype. Although we have also tested our system on real databases, we used synthetic data to evaluate its performance. Our experiments were

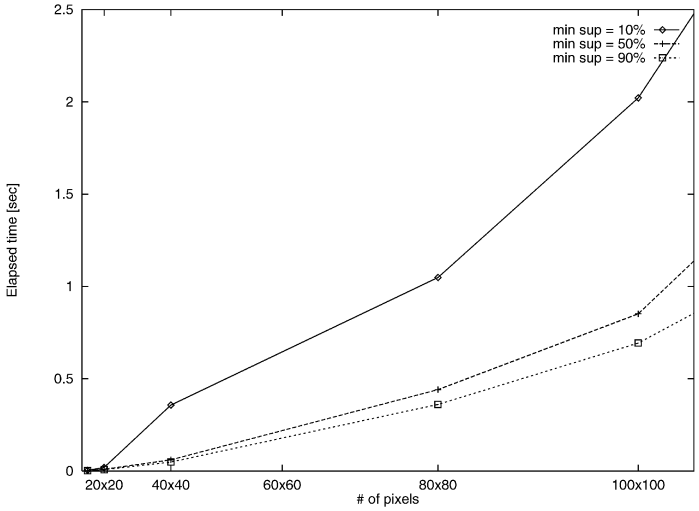

Fig. 8. Finding an optimized-confidence rectangle.

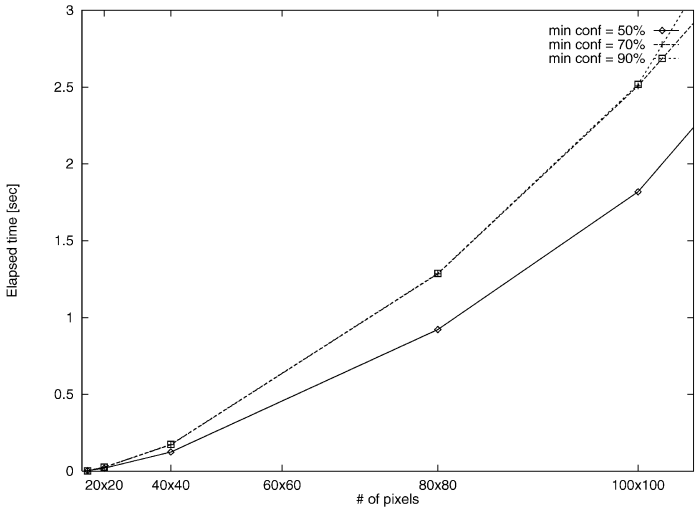

Fig. 9. Finding an optimized-support rectangle.

conducted on an IBM PC Power Series 850 with a CPU clock rate of 133 MHz and 96 MB of main memory, running under AIX 4.1.

We obtained our test data as follows: We first generated random numbers uniformly distributed in $[N^2, 2N^2]$ and assigned them to $u_{(i,j)}$. We then assigned $1, \ldots, N^2$ to $v_{(i,j)}$ from a cell in a corner to the central cell, like a spiral staircase.

6.1 Optimized Rectangles

Figures 8 and 9 respectively show the execution times needed to find an optimized-confidence rectangle with minimum a support of 10%, 50%, and 90%,

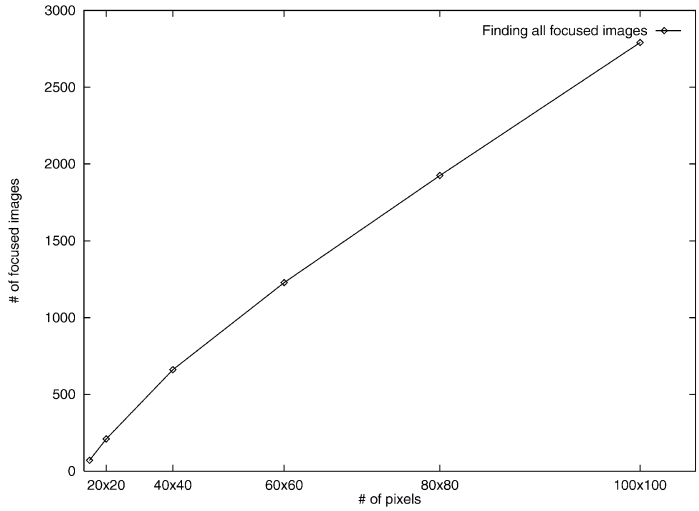

Fig. 10. Number of focused regions.

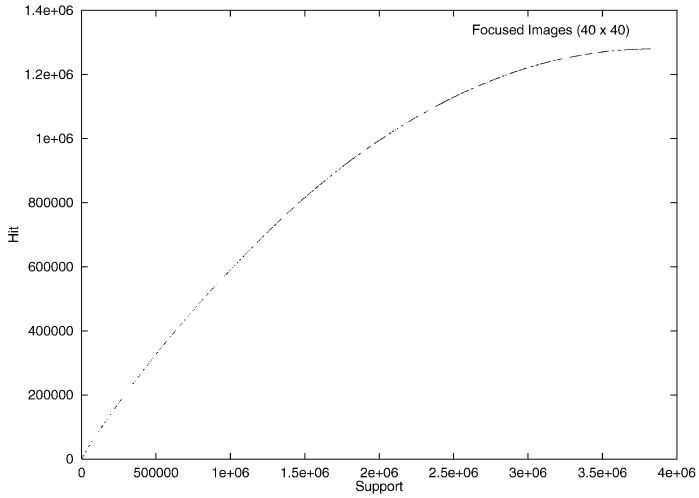

Fig. 11. Focused regions on the upper convex hull.

and an optimized-support rectangle with minimum confidence of 50%, 70%, and 90% for numbers of pixels ranging from 20×20 to 100×100. In both cases, the execution time increases almost linearly in $O(n^{1.5})$.

6.2 Optimized Admissible Regions

To determine the features of our test data, we counted the focused regions in the data (Figure 10). The number increases almost linearly in proportion to the square root of the number of pixels. We also did experiments on a few real data sets with sizes of up to 80×80 in a financial application, and observed that the number of focused regions increases sublinearly to the square root of the number of pixels.

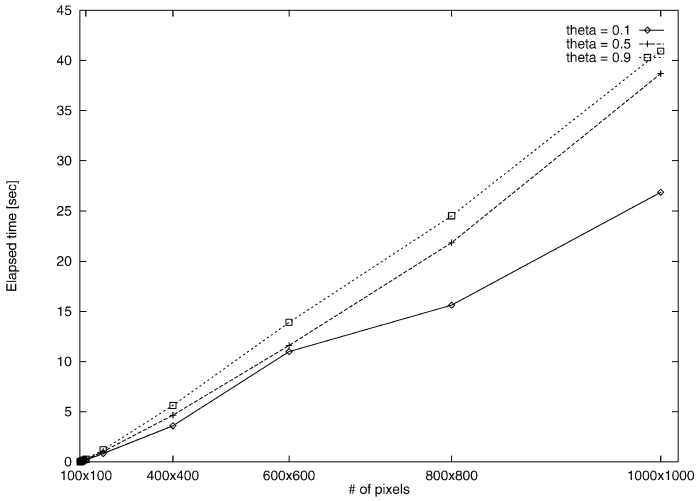

Fig. 12. Finding an optimized-gain region.

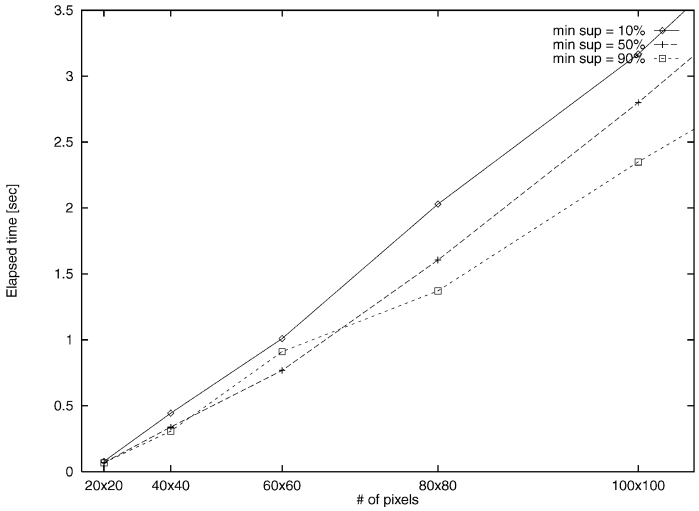

Fig. 13. Finding a focused optimized-confidence admissible region.

Figure 11 shows the focused regions (stamp points) on the upper convex hull of all stamp points in the Euclidean plane when the number of pixels is 40×40. Observe that those stamp points are fairly densely scattered on the upper hull.

Figure 12 shows the execution time needed to find an optimized-gain region (i.e., a focused region) with $\theta = 0.1, 0.5, 0.9$ for the range of data sizes from 20×20 to $1{,}000 \times 1{,}000$. The result of this experiment was what we had expected–the execution time needed to find a focused region increases linearly in proportion to the number of pixels.

Figures 13 and 14, respectively, show the execution times needed to find a focused optimized-confidence region with minimum support of 10%, 50%, and 90%, and a focused optimized-support region with minimum confidence of 50%,

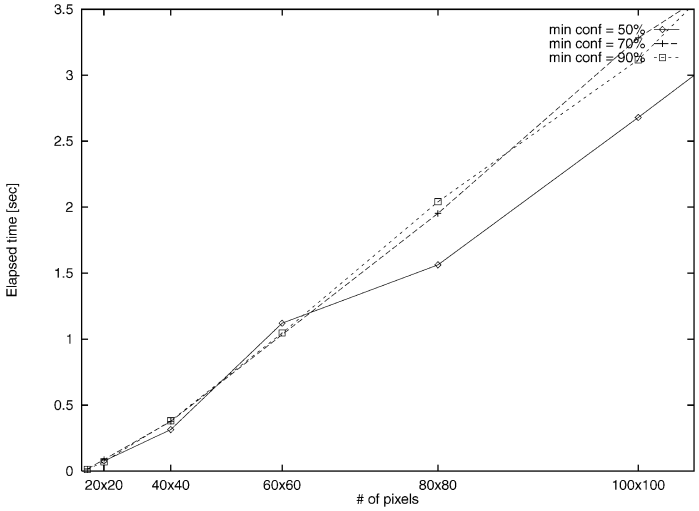

Fig. 14. Finding a focused optimized-support admissible region.

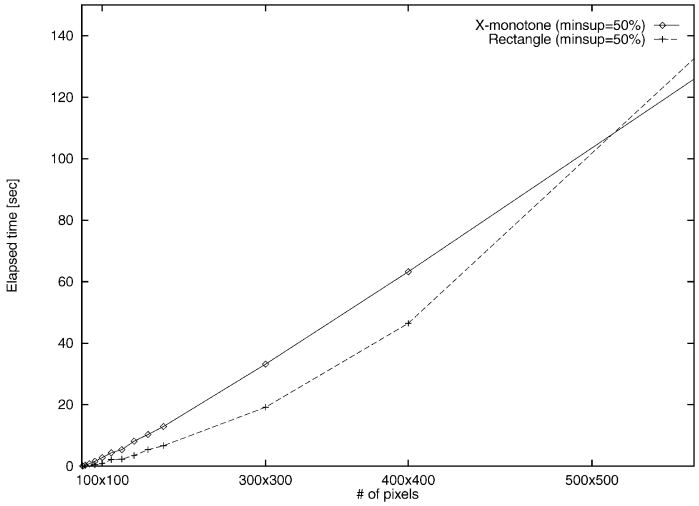

Fig. 15. Performance comparison of optimized-confidence regions.

70%, and 90% for numbers of pixels ranging from 20×20 to 100×100. In both cases, the execution time increases almost linearly in proportion to the number of pixels.

Figure 15 shows the execution times needed to find an optimized-confidence rectangle and a focused optimized-confidence admissible region with a minimum support of 50%. Figure 16 shows the execution times needed to find an optimized-support rectangle and a focused optimized-support admissible region with a minimum confidence of 70%. In each case, focused optimized admissible regions are less expensive to compute than optimized rectangles when the number of pixels is large.

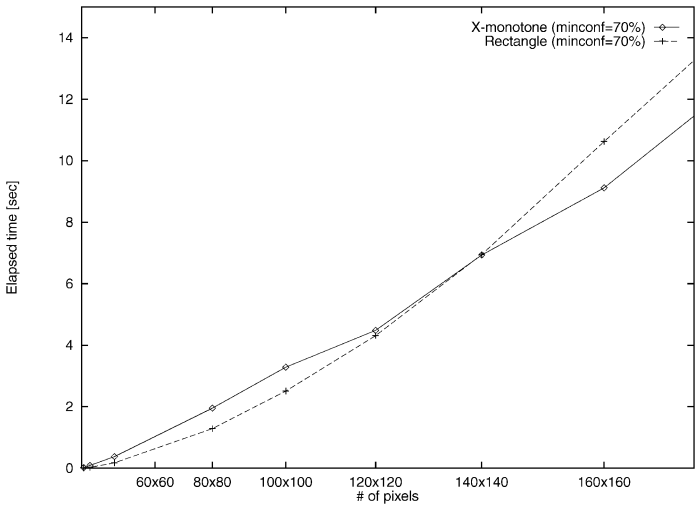

Fig. 16. Performance comparison of optimized-support regions.

7. ADVANCED FUNCTIONS IN SONAR SYSTEM AND APPLICATIONS

We have constructed a database mining system called SONAR (System with Optimized Numeric Association Rules) [Fukuda et al. 1996c] that discovers the optimized confidence rules and support rules. We also designed several advanced functions based on our optimized rules, adding several modifications to our optimization criteria to adapt to the requirements. One is the *rectilinear convex region rules*, and others are decision-making functions such as decision trees and regression trees. We give a brief outline of these functions in this section. Readers are referred to Yoda et al. [1997], Fukuda et al. [1996a], and Morimoto et al. [1997a; 1997b; 1998] for details.

7.1 Rectilinear Convex Region Rules

We have mainly discussed x-monotone regions. However, one defect that was pointed out by several people who have observed our output region rules is that although the output-admissible region usually gives an intuitive idea on the association between attributes, it sometimes happens that the region is wildly notched and that it is difficult to speculate on the meaning of the rule. Since speculation through visualization is very important for users who require decision-support knowledge, this is a serious problem. Moreover, if we use a very fine bucketing, the shape of the region tends to be very sensitive to the sampling if we use a sample subset of the database to construct the region [Yoda et al. 1997]. This dependency on the sample should be avoided, since we want to make the rule applicable not only to data in the database but also to unknown data in general; however, it is often tedious to tune the size of bucketings.

In order to resolve these defects, one idea is to design a tool to smooth the shape of grid regions, and apply the tool to the admissible region output by our algorithm; however, this may result in a loss of optimality, and we would like to avoid this kind of heuristic as far as possible. Our solution keeps the

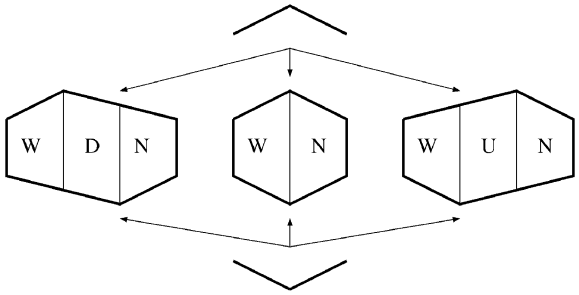

Fig. 17. Partition of a rectilinear region into monotone parts.

optimality formulations but replaces the family of admissible regions by the family of *rectilinear convex regions*. A region is rectilinear convex if it is both x-monotone and y-monotone. We show below that the optimized gain region in this family can be computed in $O(n^{1.5}) = O(N^3)$ time. Therefore, we can compute the focused optimized-confidence region and the focused optimized-support region for the family of rectilinear convex region in $O(n^{1.5} \log M)$ time. This gives a more intuitive region and also it is stable for the data sampling even if we use a fine bucketing (reported in [Yoda et al. 1997]). As a trade-off, it is more expensive to compute the optimized rectilinear convex regions than admissible regions; therefore, we should provide both functions so that the user can flexibly choose whichever is better suited to the application and data.

Let P be a rectilinear convex region. Let m_1 and m_2, respectively, denote the indices of the first and last columns. Let $s(i)$ and $t(i)$ denote the indices of the bottom and the top pixels of the ith column. Since P is a rectilinear convex region, the sequence $(s(m_1), s(m_1+1), \ldots, s(m_2-1), s(m_2))$ of indices of top pixels from left to right increases monotonically up to some column and then decreases monotonically (possibly one of these two monotone parts is empty). Similarly, the sequence of indices of bottom pixels decreases monotonically up to some column and then increases monotonically.

Therefore, we can cut a rectilinear convex region vertically into at most three *monotone* pieces each of which is among the following four types:

- A region that gets wider from left to right: This type of region is named W.
- Regions that slant upward and downward: These types of regions are named U and D, respectively.
- A region that gets narrower from left to right: This type of region is named N.

THEOREM 7.1. *The optimized-gain rectilinear convex region for a threshold θ can be computed in $O(N^3)$ time.*

PROOF. There are some possible combinations as shown in Figure 17, but we consider only the optimal *WUN*-type region that starts with a *W*-type part, continues to a *U*-type part, and ends with an *N*-type part, since other cases can be handled similarly. However, we also use W and WU types to explain our algorithm.

Our algorithm is based on the dynamic programming paradigm. Let $g_{i,j}$ denote the gain of the pixel $G(i, j)$, which is $v_{i,j} - \tau u_{i,j}$. Before running the dynamic programming program, we pre-compute $\sum_{j \in [s,t]} g_{m,j}$, which will be denoted by $g_{m,[s,t]}$, for $m = 1, \ldots, N$ and $1 \le s \le t \le N$. This computation takes $O(N^3)$ time.

Now, let $R_W(m, [s, t])$ (respectively, $R_{WU}(m, [s, t])$, $R_{WUN}(m, [s, t])$) be the rectilinear convex regions that maximize the gain among all W-type (respectively, WU-type, WUN-type) rectilinear convex regions whose last column is the mth one, and their intersections with the mth column range from the sth pixel to the tth pixel. Let $f_W(m, [s, t])$, $f_{WU}(m, [s, t])$ and $f_{WUN}(m, [s, t])$ be their gains, respectively.

First, we consider W-type regions. For $m = 1$, $f_W(1, [s, t]) = g_{1,[s,t]}$. For $m > 1$, if $s = t$, $f_W(m, [s, s]) = \max\{g_{m,s}, f_W(m-1, [s, s]) + g_{m,s}\}$. Consider the case $s < t$. Since the region is of type W, the $(m-1)$th column of $R_W(m, [s, t])$ must be a subinterval $[s, t]$. If it is a subinterval of $[s, t-1]$ (respectively, $[s+1, t]$), we can observe that the region must contain $R_W(m, [s, t-1])$ (respectively, $R_W(m, [s+1, t])$. Hence, if $R_W(m, [s, t])$ contains neither $R_W(m, [s+1, t])$ nor $R_W(m, [s, t-1])$, its $(m-1)$th column must be the interval $[s, t]$.

Hence, the following recurrence holds:

$$f_W(m, [s, t]) = \max \begin{pmatrix} f_W(m-1, [s, t]) + g_{m,[s,t]} \\ f_W(m, [s+1, t]) + g_{m,s} \\ f_W(m, [s, t-1]) + g_{m,t} \end{pmatrix}.$$

On the basis of this recursion formula, we can compute $f_W(m, I)$ for all m and I in $O(N^3)$ time.

Next, consider $f_{WU}(m, [s, t])$. For $m = 1$, $f_{WU}(1, [s, t]) = g_{1,[s,t]}$. For $m > 1$, we pre-compute $\max_{i \le s} f_W(m-1, [i, t])$ and $\max_{i \le s} f_{WU}(m-1, [i, t])$ for all $s \le t$ in $O(N^2)$ time. Since the region is of type U, the $(m-1)$th column of $R_{WU}(m, [s, t])$ should be an interval $[i, j]$ for $i \le s$ and $j \le t$. If $j \ne t$, $j \le t-1$ holds, and we can observe that the region must contain $R_W(m, [s, t-1])$. Hence, we have the following recurrence, which leads us to an $O(N^3)$-time dynamic programming algorithm:

$$f_{WU}(m, [s, t]) = \max \begin{pmatrix} \max_{i \le s} f_W(m-1, [i, t]) + g_{m,[s,t]} \\ \max_{i \le s} f_{WU}(m-1, [i, t]) + g_{m,[s,t]} \\ f_{WU}(m, [s, t-1]) + g_{m,t} \end{pmatrix}.$$

Finally, let $f_{WUN}(m, [s, t])$ denote the gain of the rectilinear convex region that maximizes the gain among all rectilinear convex regions such that their type is WUN, their last column is the mth one, and their intersections with the mth column range from the sth pixel to the tth pixel. For $m = 1$, $f_{WUN}(1, [s, t]) = g_{1,[s,t]}$. For $m > 1$, we have the following recurrence (a mirror formula to that for the W-type):

$$f_{WUN}(m, [s, t]) = \max \begin{pmatrix} f_{WU}(m-1, [s, t]) + g_{m,[s,t]} \\ f_{WUN}(m-1, [s, t]) + g_{m,[s,t]} \\ f_{WUN}(m, [s-1, t]) - g_{m,s-1} \\ f_{WUN}(m, [s, t+1]) - g_{m,t+1} \end{pmatrix}.$$

In this case, we need to compute $f_{WUN}(m, [s, t])$ by using $f_{WUN}(m, [s-1, t])$ and $f_{WUN}(m, [s, t+1])$. Thus, we first compute $f_{WUN}(m, [1, N]) = \max\{f_{WUN}(m-1, [1, N]) + g_{m,[1,N]}, g_{m,[1,N]}\}$, and run the dynamic programming to compute the values of $f_{WUN}(m, I)$ from larger intervals I to smaller ones.

Consequently these recursion formulas provide a dynamic programming method of obtaining an $O(N^3)$-time solution for computing $f_{WUN}(m, [s, t])$ for all m and $s \leq t$. We select the one with the maximum gain. The space complexity is naively $O(N^3)$, but can be easily reduced to $O(N^2)$. □

7.2 Decision Tree Function

7.2.1 *Decision Tree Based on Optimized Rules.* Although we have focused on optimized rules with one or two dimensions (i.e., conditional attributes), a database usually has a lot of attributes. Indeed, we need an automatic decision system to assist users in making decisions by analyzing a large-scale database in which many attributes have a combined effect on the target property. It is not practical for users to make decisions manually by using a set of two-dimensional rules; Neither is it a good idea to extend our two-dimensional method directly to a higher-dimensional case by defining high-dimensional admissible regions, since the computation time will increase explosively. Therefore, we construct decision trees based on our two-dimensional optimized rules in order to analyze the combined effect of many attributes. This idea has been implemented in our SONAR system, and through experiments on many benchmark data and real databases we have shown [Morimoto et al. 1997a] that the system performs very well compared with other decision systems in terms of both accuracy and ease of understanding. For this purpose, we need to replace our threshold-based optimality criteria (optimized confidence and optimized support) with a nonlinear optimization criterion.

A decision tree T is a binary tree for guessing the probability that the target Boolean attribute value $t[W]$ of an unknown new item of data t is "yes," on the hypothesis that this new data item is picked from a distribution space containing the database records as samples. At each node, we give an association rule (called a *branching rule*) to split the data set into two subsets. To be precise, we choose a pair of conditional attributes, compute the optimal admissible region, and divide the data set into two parts, one inside and the other outside the region.

We adopt the greedy heuristics to construct the tree in top-down fashion. That is, we select the "best" association rule at the root node of the tree, and recursively construct a subtree for each of two subsets of data split by the rule. We continue the splitting process until the dataset becomes *pure* for a given threshold (say, 90%): that is, at least 90% of data have the same (yes or no) value of W.

Unfortunately, neither the optimized confident rule nor optimized support rule is suitable as a criterion for selecting the best rule, since the threshold parameter θ is heavily dependent on the pair of conditional attributes and hence difficult to tune. Instead, we use the entropy function of the data splitting as the objective function, and regard the splitting with the minimum entropy as

the best rule. The idea of using an entropy function to construct a decision tree is quite popular in the literature [Quinlan 1993] for databases with categorical attributes; however, it has been pointed out that the method does not give an accurate decision tree for a database with many numeric attributes if we only use one-dimensional association rules as branching rules.

The entropy value $Ent(S)$ (with respect to the objective attribute W) is defined as

$$Ent(S) = -(p \log p + (1 - p) \log(1 - p)),$$

where p is the probability with which W takes the value 0 in the set S. For example, if $p = 0, 0.5$, and 1, then $Ent(S) = 0, 1$, and 0, respectively. Indeed, a purer dataset with respect to W has a lower entropy value. For a splitting of the data set into two subsets S_1 and S_2 with n_1 and n_2 data, respectively, the entropy of the splitting is defined by

$$Ent(S_1; S_2) = \frac{n_1}{n} Ent(S_1) + \frac{n_2}{n} Ent(S_2).$$

A splitting that gives a small entropy value is a good one from the viewpoint of information theory.

A key fact is that the entropy is a concave function, and hence the admissible region P that minimizes the entropy value of the splitting must be a focused region, which we can compute by listing up all focused regions (there are at most M focused regions), and selecting the one with the maximum entropy-gain. This algorithm runs in $O(Mn)$ time for each pair of conditional attributes, but we can improve it by using a branch-and-bound approach so that the expected running time is reduced to $O(n \log M)$ [Morimoto et al. 1997a]. Hence, if there are h attributes in the database, we can select the best rule in $O(h^2 n \log M)$ time.

We can use any other concave function (e.g., the gini-index function or variance) for our objective function to apply the above framework (see Morimoto et al. [1997a] for details). Morimoto et al. [1997a] give experimental results showing that the above construction method is superior to the conventional methods [Quinlan 1993; Mehta et al. 1996] based on one-dimensional binary partition rules (to be precise, a branching rule gives a threshold value of one conditional attribute, so that the data is compared with the threshold and divided into two subsets) with respect to both accuracy and tree size. It has been reported [Morimoto et al. 1997a] that we should use rectilinear convex region rules rather than admissible region rules to obtain an accurate tree. In the following section, we give an example of application of our decision tree function to a financial problem.

7.2.2 *Decision Tree in Credit Risk Analysis.* The following table shows parts of the financial statements of some Japanese companies from 1992 to 1996. It contains 69 numeric attributes such as "`ID`" (ID of a company), "`Net Income/Sales`," and "`Equity Ratio`." It also contains the attribute "`Default`," which shows whether the company defaulted within a year. In order to discover rules about the value of "`Default`," we collected 1036 samples of defaulting companies and another 1036 of nondefaulting companies.

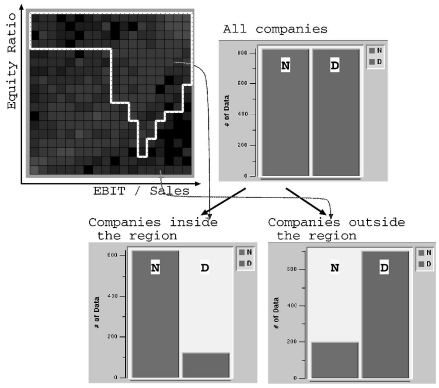

Fig. 18. Rules for analyzing credit risk.

ID	Net Income/ Sales (%)	Equity Ratio (%)	...	Default
xxx	5.119	25.876	...	N
xxx	1.248	3.847	...	N
xxx	0.355	8.941	...	D
xxx	1.235	38.886	...	N
...	...	...	...	...
xxx	−0.096	4.111	...	D

Figure 18 shows the most important two-dimensional rectilinear convex region rule that was found from the table. Note that there are 69 attributes in the table and more than 2,300 permutations (pairs). We examined each pair to find the optimal region, and chose the most important one on the basis of the gini-index value. The region on the plane whose x-axis is "EBIT/Sales" and whose y-axis is "Equity Ratio" divides the original data "S" into two subsets "S_1" (inside the region or below the curve) and "S_2" (outside the region or above the curve) as follows:

	No. of companies	No. of defaulting	No. of nondefaulting
S: All data	2072	1036	1036
S_1: Inside	969	797	172
S_2: Outside	1103	239	864

We divided the data recursively by using two-dimensional rules, and constructed a decision tree to evaluate the credit risk of each company. Figure 19 shows the decision tree constructed from the example. In each leaf node (square node), the ratio of defaulting companies is large (or small) enough to judge the companies in that node. Each company whose credit risk is unknown can be classified into one of the leaf nodes in the tree, and we can estimate the credit risk according to the ratio of defaulting companies in the leaf node.

Compared with the conventional decision tree based on one-dimensional binary partition rules, our tree improves the default prediction error in this instance by a ratio of nearly 10%. Such a difference has a large impact in this application.

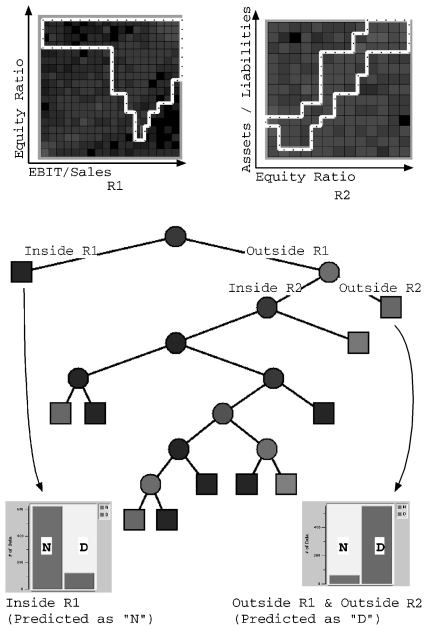

Fig. 19. Decision tree for analyzing credit risk.

7.3 Regression Splitting and Regression Tree

7.3.1 *Handling Numeric Target Attributes.* So far, we have focused on optimized rules with Boolean target attributes. However, users of our initial prototype [Fukuda et al. 1996c] commented that it would be very useful if they could handle numeric target attributes as well. One naive solution to this is to transform the numeric target attribute into a Boolean attribute by a binary partition. Unfortunately, this solution would cause much information to be lost, thus nullifying the effect of optimization. Therefore, we consider the following formulation in Morimoto et al. [1997b]. For a data set, we extend the notion of the entropy with respect to a Boolean objective attribute to the entropy with respect to a numeric objective attribute, since the entropy is usually defined on data distribution. Accordingly, we define the entropy (or variance) of splitting with respect to the numeric objective attribute. Then, we consider the two-dimensional rule minimizing this objective function. This is quite a novel idea for handling numeric objective attributes. The algorithm is almost analogous to that in the previous sections, and a regression tree is obtained as a counterpart of the decision tree. Experiments on this method are given in Morimoto et al. [1997a; 1997b], and we give an example of application to a financial problem.

7.3.2 *Regression Splitting and Tree in Market Analysis.* The following table shows the relations collected from the New York currency markets every Monday from the last Monday of 1985 through the first Monday of May 1993. It contains 384 records and 10 numeric attributes: "W" (week), "M" (month), "Y" (year), "BPS" (British pounds sterling, i.e., US$/pound), "GDM" (deutschmark, i.e.,

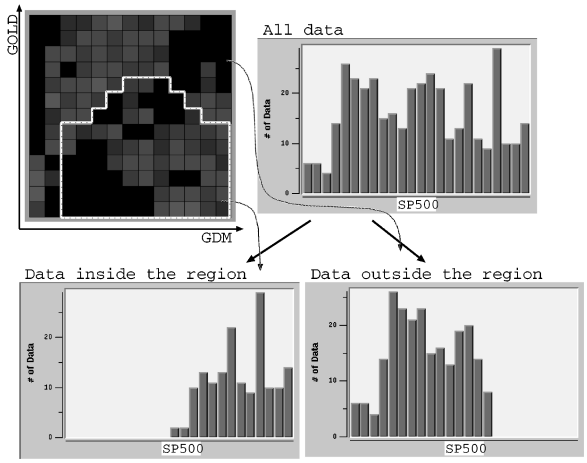

Fig. 20. Rules for analyzing the market.

US$/mark), "YEN" (Japanese yen, i.e., US$/yen), "TB3M" (3-month Treasury bill yields), "TB30Y" (30-years Treasury bill yields), "GOLD" (US$/ounce), and "SP500" (Standard and Poor's index). Suppose we are interested in the "SP500," which is one of the indices for measuring the overall stock market performance.

BPS	GDM	YEN	TB3M	...	GOLD	SP500
1.443	.4074	.00498	7.02	...	326.00	210.88
1.446	.4080	.00495	7.04	...	339.45	205.96
1.437	.4048	.00494	7.13	...	357.25	208.43
1.404	.4087	.00498	7.17	...	355.25	206.43
...	...	...	...	...	...	...
1.568	.6338	.00907	2.91	...	357.50	442.31

Figure 20 shows the most important two-dimensional rectilinear convex region rule that can be found from the table. The region on the plane whose x-axis is "GDM" and whose y-axis is "GOLD" divides the data "S" into "S_1" and "S_2" so that the variance of the "SP500" is minimized. The number of data, the mean, and the variance are as follows:

	No. of data	SP500 mean	SP500 variance
All Data S	384	324	4431
Inside Region S_1	157	391	1118
Outside Region S_2	227	278	1489

We divided the data recursively by using two-dimensional rules, and constructed a regression tree to predict the "SP500." Figure 21 is the regression tree. The variance of the "SP500" value in each leaf node (square node) is small enough for us to be able to predict the value of the "SP500."

Compared with the regression tree constructed by using a conventional method based on one-dimensional binary partition rules, our tree in this instance reduces the prediction error (measured by using a variance function) by 25%, which is very significant in this application.

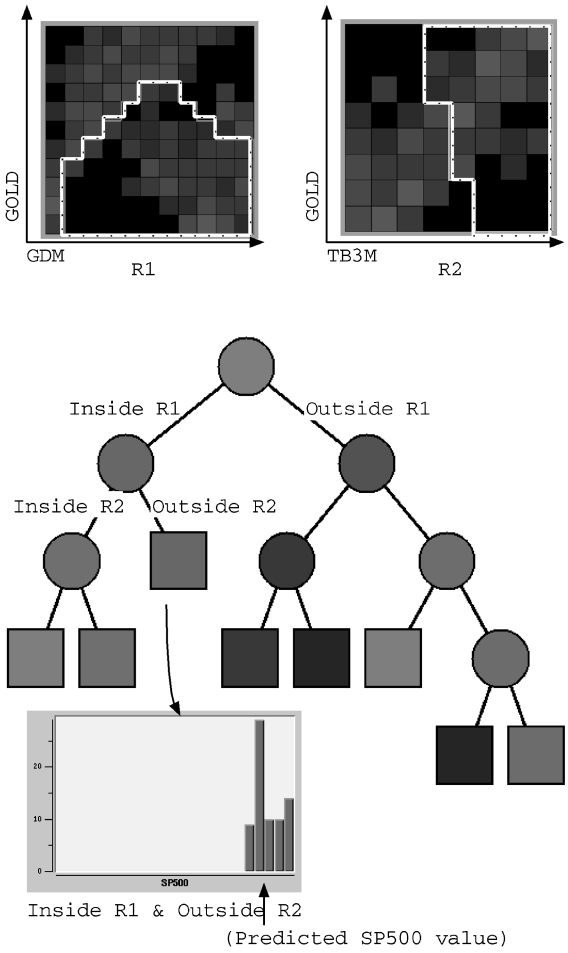

Fig. 21. Regression tree for analyzing the market.

APPENDIX

A.1. FROM "PROGRAMMING PEARLS"

In the "Programming Pearls" column of *Communications of the ACM* Bentley [1984] presented a problem for demonstrating the importance of efficient algorithms in program design. The problem is:

> "Given a list X of N real numbers, compute the maximum sum found in any contiguous subvector of it."

This problem is equivalent to our problem of computing the optimized-gain range. Given a confidence threshold θ, we define a list X so that $X(i) = v_i - \theta \times u_i$. Then, $X[s,t]$ is the solution of the problem if and only if $[s,t]$ is the optimized-gain range.

Bentley introduced four algorithms, whose time complexities are $O(N^3)$, $O(N^2)$, $O(N \log N)$, and $O(N)$, respectively. The linear time algorithms (Kadane's algorithm) scans the data with respect to the array index. For each i,

$$Max(i) = \max_{s \le i} X([s, i]), \text{ and}$$
$$Max(\le i) = \max_{s \le t \le i} X([s, t]).$$

Then, $Max(\le N)$ is the answer. The following two relations are easy to see:

$$Max(i+1) = \max\{0, Max(i)\} + X(i+1), \text{ and}$$
$$Max(\le i+1) = \max\{Max(i+1), Max(\le i)\}.$$

Thus, a simple dynamic programming gives an $O(N)$ time solution.

Therefore, we have the following:

THEOREM A.1. *The optimized-gain range can be found in $O(N)$ time.*

A.2. COMPUTING OPTIMIZED-SUPPORT/-CONFIDENCE ADMISSIBLE REGIONS EXACTLY

Unfortunately, if we consider admissible regions, the optimized-support region and the optimized-confidence region are difficult to compute. For each of them, we only know an $O(n^{1.5}M)$ time algorithm, where M is the total number of tuples. Moreover, it can be shown that no algorithm running in polynomial time with respect to n and $\log M$ exists unless P = NP.

THEOREM A.2. *Let M denote the total number of tuples in the given database. Suppose that we have $n (= N \times N)$ pixels. For each pixel $G(i, j)$, let $u_{i,j}$ denote the number of tuples mapped to $G(i, j)$, and let $v_{i,j}$ denote the number of success tuples mapped to $G(i, j)$. We have the following properties:*

(1) *The optimized-confidence region or the optimized-support region can be computed in $O(n^{1.5}M)$ time.*
(2) *There exists no algorithm that can compute the optimized-confidence (or -support) region in polynomial time with respect to n and $\log M$ unless $P = NP$.*

PROOF. We begin by proving the first property. Consider the set of admissible regions each of which satisfies the conditions that its support is k, all of its pixels are to the left of the mth column, and its intersection with the mth column is the subsequence of pixels ranging from $G(s, m)$ to $G(t, m)$. If the set is nonempty, let $f(k, m, [s, t])$ denote the maximum hit of all regions in the set. Otherwise, define $f(k, m, [s, t]) = -\infty$. Suppose that we have computed $f(k, m, [s, t])$ for all $k = 1, \ldots, M$ and $m, s, t = 1, \ldots, N$. By scanning $f(k, m, [s, t])$ once, we can obtain the obtimized-confidence (or -support) region in $O(n^{1.5}M)$ time.

First, we consider the basic step when $m = 1$. For all pairs (s, t) of $1 \le s < t \le N$ and for each $k = 1, \ldots, M$, $f(k, 1, [s, t])$ can be obtained as follows:

$$f(k, 1, [s, t]) := \sum_{i=s}^{t} v_{1,i} \qquad \text{if} \quad k = \sum_{i=s}^{t} u_{1,i}$$
$$:= -\infty \qquad \text{otherwise.}$$

Next, we present the inductive step when $m > 1$. To compute $f(k, m, [s, t])$, we use an auxiliary function $h(k, m, [s, t])$ that is the maximum hit of admissible regions each of which meets the conditions that its support is k, all of its pixels are to the left of the mth column, and its intersection with the mth column is a sequence that *includes* the subsequence of pixels ranging from $G(s, m)$ to $G(t, m)$. We later show how to compute $h(k, m, [s, t])$ from $f(k, m, [s, t])$. In the following algorithm, we assume $h(k, m-1, [s, t])$ for computing $f(k, m, [s, t])$.

for each $k = 1, \ldots, M$
  $f(k, m, [s, s]) := v_{m,s} + h(k - u_{m,s}, m - 1, [s, s])$;
for each $len = 2, \ldots, N$
  for each $s = 1, \ldots, N - len + 1$
    $t = s + len - 1$;
    for each $k = 1, \ldots, M$,
      $f(k, m, [s, t]) := max\{f(k - u_{m,t}, m, [s, t-1]) + v_{s,t},$
        $h(k - \sum_{i=s}^{t} u_{m,i}, m - 1, [t, t]) + \sum_{i=s}^{t} v_{m,i}\}$;

The above procedure requires $O(nM)$ time. In this procedure, we use $h(k, m-1, [t, t])$ only, instead of $h(k, m-1, [s, t])$ for all $[s, t]$. However, we need $h(k, m, [s, t])$ for all $[s, t]$ in order to compute $h(k, m, [t, t])$ in $O(nM)$ time, and we give the following algorithm:

for each $k = 1, \ldots, M$ $h(k, m, [1, N]) := f(k, m, [1, N])$;
for each $len = N - 1, \ldots, 1$
  for each $s = 1, \ldots, N - len + 1$
  $t := s + len - 1$;
  for each $k = 1, \ldots, M$,
    $h(k, m, [s, t]) := max\{h(k, m, [s-1, t]), h(k, m, [s, t+1]), f(k, m, [s, t])$;

where $h(k, m, [s-1, t]) = -\infty$ when $s - 1 < 1$, and $h(k, m, [s, t+1]) = -\infty$ when $t + 1 > N$. We perform the above $O(nM)$-time procedures for $m = 2, \ldots, N$, and therefore we have given an algorithm for computing the optimized-confidence (or -support) region in $O(n^{1.5}M)$ time.

Next we will prove the second property. Consider the pixels have the following properties:

- $u_{1,j} = v_{1,j} > 0$ for each $j = 1, \ldots, N$
- $u_{2,j} > 0$ and $v_{2,j} = 0$ for each $j = 1, \ldots, N$
- $u_{i,j} = v_{i,j} = 0$ if $i \ge 3$

Suppose that K is the minimum support threshold such that $\sum_{j=1}^{N} u_{1,j} < K \le M$. Observe that the optimized-confidence admissible region for the minimum support K must contain all of $G(1, j_1)$ and some of $G(2, j_2)$. To compute the optimized region, we need to find a subset S of $\{1, \ldots, N\}$ that minimizes

$\sum_{j_2 \in S} u_{2,j_2}$ under the condition that $\sum_{j_2 \in S} u_{2,j_2} \geq K - \sum_{j_1=1}^{N} u_{1,j_1}$. If the optimized-confidence admissible region can be computed in a time polynomial to n and $\log M$, in the same time complexity we can also answer whether or not there exists S such that $\sum_{j_2 \in S} u_{2,j_2} = K - \sum_{j_1=1}^{N} u_{1,j_1}$, which is equivalent to the NP-complete subset sum problem [Karp 1972]. Consequently, unless P = NP, no algorithm exists for computing the optimized-confidence region in polynomial time with respect to n and $\log M$.

The same argument can be carried over to the case of the optimized-support admissible region. Suppose that θ is the minimum support threshold such that

$$\frac{\sum_{j_1=1}^{N} u_{1,j_1}}{\sum_{j_1=1}^{N} u_{1,j_1} + \sum_{j_2=1}^{N} u_{2,j_2}} < \theta \leq 1.$$

Then, the optimized-support admissible region for θ must be the set of all of $G(1, j_1)$ and $G(2, j_2)$ for $j_2 \in S$ such that $S(\subseteq \{1, \ldots, N\})$ maximizes $\sum_{j_2 \in S} u_{2,j_2}$ under the condition that the confidence of the optimized region is at least θ; that is,

$$\frac{\sum_{j_1=1}^{N} u_{1,j_1}}{\sum_{j_1=1}^{N} u_{1,j_1} + \sum_{j_2 \in S} u_{2,j_2}} > \theta,$$

which is equivalent to

$$\left(\frac{1}{\theta} - 1\right) \sum_{j_1=1}^{N} u_{1,j_1} \geq \sum_{j_2 \in S} u_{2,j_2}.$$

If the optimized-support admissible region can be computed in time polynomial to n and $\log M$, in the same time complexity we can answer whether or not there exists S such that

$$\left(\frac{1}{\theta} - 1\right) \sum_{j_1=1}^{N} u_{1,j_1} = \sum_{j_2 \in S} u_{2,j_2},$$

which is equivalent to the NP-complete subset sum problem. □

ACKNOWLEDGMENTS

We are very grateful to Hirofumi Matsuzawa, Yoshihiro Ohta, and Kunikazu Yoda for implementing and experimenting our algorithms as members of IBM Tokyo Research Laboratory's data mining project. We also thank Tetsuo Asano for providing us with the algorithm given in Lemma 4.1, and Naoki Katoh, Rakesh Agrawal, and Ramakrishnan Srikant for fruitful discussions.

REFERENCES

AGRAWAL, R. AND SRIKANT, R. 1994. Fast algorithms for mining association rules. In *Proceedings of the 20th VLDB Conference.* pp. 487–499.

AGRAWAL, R., GHOSH, S., IMIELINSKI, T., IYER, B., AND SWAMI, A. 1992. An interval classifier for database mining applications. In *Proceedings of the 18th VLDB Conference.* pp. 560–573.

AGRAWAL, R., IMICLINSKI, T., AND SWAMI, A. 1993a. Database mining: A performance perspective. *IEEE Trans. Knowl. Data Eng. 5*, 6 (Dec.), 914–925.

AGRAWAL, R., IMIELINSKI, T., AND SWAMI, A. 1993b. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD Conference on Management of Data* (Washington, D.C., May 26–28). ACM, New York, pp. 207–216.

AGGARWAL, A., KLAWE, M., MORAN, S., SHOR, P., AND WILBUR, R. 1987. Geometric applications of a matrix-searching algorithm. *Algorithmica 2*, 209–233.

ASANO, T., CHEN, D., KATOH, N., AND TOKUYAMA, T. 1996. Polynomial-time solutions to image segmentations. In *Proceedings of the 7th ACM–SIAM Symposium on Discrete Algorithms*, ACM, New York, pp. 104–113.

BENTLEY, J. 1984. Programming pearls. *Commun. ACM 27*, 7, (Sept.), 865–871.

BESL, P. J. AND JAIN, R. C. 1988. Segmentation through variable-order surface fitting. *IEEE Trans. Patt. Anal. Mach. Intell. 9*, 2, 167–192.

BREIMAN, L., FRIEDMAN, J. H., OLSHEN, R. A., AND STONE, C. J. 1984. *Classification and Regression Trees*. Wadsworth.

FISCHER, P., HOFFGEN, K.-U., LEFMANN, H., AND LUCZAK, T. 1993. Approximations with axis-aligned rectangles. In *Proceedings of the 9th International Conference on Fundamentals of Computation Theory.* Springer-Verlag, New York.

FUKUDA, T., MORIMOTO, Y., MORISHITA, S., AND TOKUYAMA, T. 1996a. Constructing efficient decision trees by using optimized association rules. In *Proceedings of the 22nd VLDB Conference.* pp. 146–155.

FUKUDA, T., MORIMOTO, Y., MORISHITA, S., AND TOKUYAMA, T. 1996b. Mining optimized association rules for numeric attributes. In *Proceedings of the 15th Annual ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (Montreal, Que, Canada, June 3–5). ACM, New York, pp. 182–191.

FUKUDA, T., MORIMOTO, Y., MORISHITA, S., AND TOKUYAMA, T. 1996c. SONAR: System for optimized numeric association rules. In *Proceedings of the 1996 ACM SIGMOD Conference on Management of Data* (Montreal, Que, Canada, June 4–6). ACM, New York, p. 553.

GAREY, M. R. AND JOHNSON, D. S. 1977. The rectilinear Steiner tree problem is NP-complete. *SIAM J. Appl. Math 32*, 836–834.

HAN, J., CAI, Y., AND CERCONE, N. 1992. Knowledge discovery in databases: An attribute-oriented approach. In *Proceedings of the 18th VLDB Conference.* pp. 547–559.

HARALICK, R. M. AND SHAPIRO, L. G. 1985. Image processing techniques. *ComputerVision Graph. Image Process. 29*, 1, 100–132.

KARP, R. M. 1972 Reducibility among combinatorial problems. *Complex. Comput. Computat.* 85–103.

KEIM, D., KRIEGEL, H., AND SEIDL, T. 1994. Supporting data mining of large database by visual feedback queries. In *Proceedings of the 10th Symposium on Data Engineering.* pp. 302–313.

KASS, M., WITKIN, A., AND TERZOPOULOS, D. 1988. Snakes: Active contour models. *Int. J. Comput. Vision 1*, 321–331.

MEHTA, M., AGRAWAL, R., AND RISSANEN, J. 1996. Sliq: A fast scalable classifier for data mining. In *Proceedings of the 5th International Conference on Extending Database Technology*.

MORIMOTO, Y., FUKUDA, T., MATSUZAWA, H., TOKUYAMA, T., AND YODA, K. 1998. Multivariate rules in data mining: A key to handling correlations in financial data. In *Proceedings of the KDD '98 Workshop on Data Mining in Finance*. Springer-Verlag, New York,

MORIMOTO, Y., FUKUDA, T., MORISHITA, S., AND TOKUYAMA, T. 1997a. Implementation and evaluation of decision trees with range and region splitting. *Constraints, an International Journal 2*, 401–427.

MORIMOTO, Y., ISHII, H., AND MORISHITA, S. 1997b. Efficient construction of regression trees with range and region splitting. In *Proceedings of the 23rd International Conference on Very Large Data Bases* (Aug. 25–29), pp. 166–175.

NEMHAUSER, G. L., RINNOY KAN, A. H. G., AND TODD, M. J. 1989. *Optimization: Handbooks in Operations Research and Management Science Vol.1*. North-Holland, Amsterdam, The Netherlands.

NG, R. T. AND HAN, J. 1994. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th VLDB Conference.* pp. 144–155.

PARK, J. S., CHEN, M.-S., AND YU, P. S. 1995. An effective hash-based algorithm for mining association rules. In *Proceedings of the 1995 ACM SIGMOD Conference on Management of Data* (San Jose, Calif, May 22–25). ACM, New York, pp. 175–186.

PIATETSKY-SHAPIRO, G. 1991. Discovery, analysis, and presentation of strong rules. In *Knowledge Discovery in Databases.* pp. 229–248.

PIATETSKY-SHAPIRO, G. AND FRAWLEY, W. J. EDS. 1991. *Knowledge Discovery in Databases*. AAAI Press, Reston, Va.

QUINLAN, J. R. 1993. *C4.5: Programs for Machine Learning*. Morgan-Kaufmann, San Mateo, Calif.

QUINLAN, J. R. 1986. Induction of decision trees. *Mach. Learn. 1*, 81–106.

SAHOO, P. K., SOLTANI, S., AND WANG, A. K. C. 1988. A survey of thresholding techniques. *Computer Vision, Graph, Image Process. 41*, 2, 233–260.

SRIKANT, R. AND AGRAWAL, R. 1996. Mining quantitative association rules in large relational tables. In *Proceedings of the ACM SIGMOD Conference on Management of Data* (Montreal, Que., Canada, June 4–6). ACM, New York, pp. 1–12.

STONEBRAKER, M., AGRAWAL, R., DAYAL, U., NEUHOLD, E. J., AND REUTER, A. 1993. DBMS research at a crossroads: The Vienna update. In *Proceedings of the 19th VLDB Conference.* pp. 688–692.

YODA, K., FUKUDA, T., MORIMOTO, Y., MORISHITA, S., AND TOKUYAMA, T. 1997. Computing optimized rectilinear regions for association rules. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining.* AAAI Press, Reston, Va., pp. 96–103.

ZUCKER, S. W. 1976. Region growing: Childhood and adolescence. *Comput. Graph. Image Process. 5*, 3, 382–399.

Received November 1996; revised April 1999; accepted January 2001