Data Mining Using Two-Dimensional Optimized Association Rules: Scheme, Algorithms, and Visualization

Takeshi Fukuda
IBM Tokyo Research Laboratory
fukudat@trl.ibm.co.jp

Yasuhiko Morimoto
IBM Tokyo Research Laboratory
morimoto@trl.ibm.co.jp

Shinichi Morishita
IBM Tokyo Research Laboratory
morisita@trl.ibm.co.jp

Takeshi Tokuyama
IBM Tokyo Research Laboratory
ttoku@trl.ibm.co.jp

Abstract

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 in a planar region P tend to use card loan 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, and constructed a system for visualizing the rules.

1 Introduction

Recent progress in technologies for data input through such media as bar-coded labels, credit cards, 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 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 [AGI+92, AIS93a, AIS93b, AS94, BFOS84, HCC92, NH94a, PCY95, PS91, PSF91, Qui86, Qui93, SAD+93].

SIGMOD '96 6/96 Montreal, Canada

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 probability (called *confidence* in this paper). We will denote such an association rule (or a rule, for short) between the presumptive condition C_1 and the objective condition C_2 by $C_1 \Rightarrow C_2$ [1].

Agrawal, Imielinski, and Swami [AIS93b] investigated how to find all rules whose confidences are greater than a specified minimum threshold, such as 50%. They focus on rules with conditions that are conjunctions of $(A = yes)$, where A is a Boolean attribute, and present an efficient algorithm. They have applied the algorithm 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 [AS94, PCY95].

In addition to Boolean attributes, databases in the real world usually have numeric attributes such as age and the balance of account in a database of bank customers. Thus, it is also important to find association rules for numeric attributes. In a companion paper [FMMT96a], we considered the problem of finding simple rules of the form

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

which expresses that customers whose balances fall in the range between v_1 and v_2 are likely to use card loan. 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." Among a lot of confident ranges, we want to obtain one with high *support* (the number of tuples in the range). Fukuda et al. [FMMT96a] introduced some optimization criteria for finding optimized ranges, that

[1] We use the symbol "⇒" in order to distinguish the relationship from logical implication, which is usually denoted by "⟶"

effectively represent interrelation between a numeric attribute and a Boolean one.

Two-Dimensional Association Rules

In the real world, binary associations between two attributes are 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. [KKS94]. However, such a region P may consist of many connected components, and often create an association rule that is very difficult to characterize, and hence hard to see the validity. Therefore, in order to obtain a rule that can be stated briefly or characterized via visualization, it is required that P should belong to a class of regions that have nice geometric properties.

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 that generates a good (or, to be precise, an optimized) association rule.

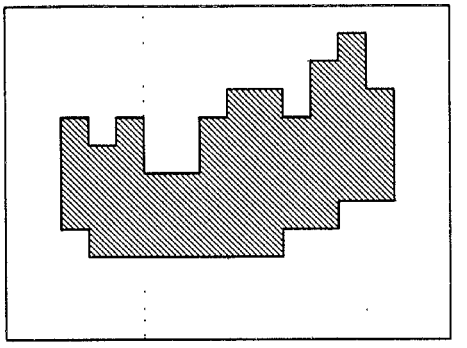

Figure 1: Admissible Region

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 admissible regions.

We generalize an algorithm introduced by Asano et al. [ACKT96], 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 [AKM+87], 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.

Visualization

It is natural 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 numbers of tuples and 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. 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,

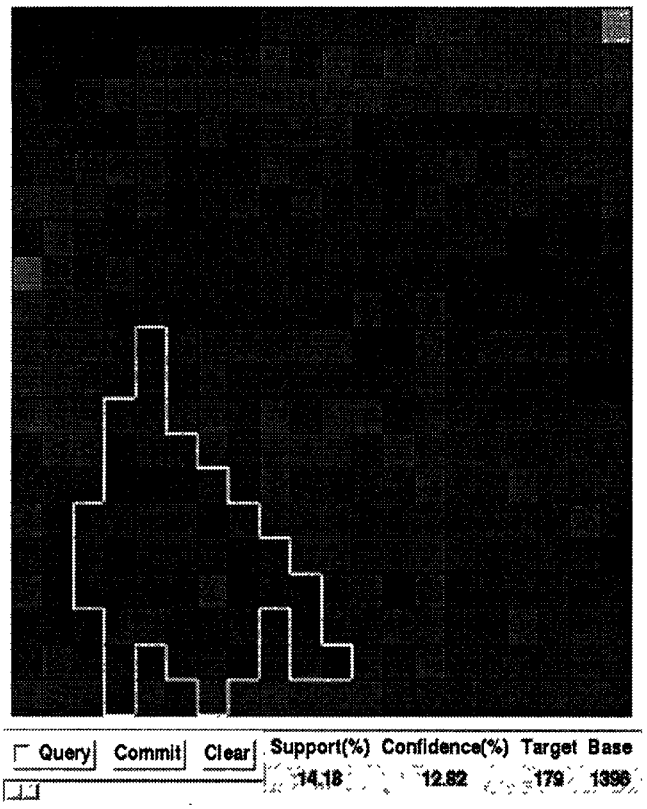

Figure 2: Visualization system

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 [FMMT96a], on some test database, and discovered several new simple rules, some of which seem to be of potential value to users in strategy planning.

The database has about thirty numeric attributes and about one hundred Boolean attributes. Suppose that we select two numerical attributes, "Age" and "Balance," and also a Boolean attribute, "Card Loan Delay," for example. 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 see 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."

2 Preliminaries

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 2.1 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 a rule, which we call a *one-dimensional association rule*, where the presumptive condition is on a numerical attribute.

Example 2.2 Consider a bank's data on customers. Each tuple contains the balance of a customer at the bank and services,(card loan or automatic withdrawal, say) for the customer. 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 2.3 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. We call rules of this type *two-dimensional association rules*, in which P is not necessarily a rectangle (we will discuss this point later in detail).

3 One-Dimensional Association Rules

Optimized Ranges

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

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 $B_1, B_2, .., B_N$, by using an ordered bucketing so that for each $t \in B_i$ and $t' \in B_j$ where $i < j$, $t[A] \leq t'[A]$. The number of tuples in B_i is denoted u_i, and the number of success tuples in B_i is v_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 our companion paper [FMMT96a].

From now on, we assume that we have equal-sized bucketing, and we only consider ranges each of which corresponds to a union of contiguously indexed buckets. For simplicity, we denote the range corresponding to $B_s \cup B_{s+1} \cup, ..., \cup B_t$ by $[s, t]$. For a range $I = [s, t]$, $support(I) = \sum_{i=s}^{t} u_i$ and $hit(I) = \sum_{i=s}^{t} v_i$ by definition. A rule is *confident* if its confidence is not less than a given confidence threshold θ. A range generating a confident rule is called a *confident range*. A rule is *ample* if its support is not less than a given support threshold Z. A range generating an ample rule is called an *ample range*.

There can be many confident ranges, and we want to compute characteristic ones that optimize some criteria. Among them, let us consider the *optimized gain* range, which maximizes $gain_\theta(I) = hit(I) - \theta \times support(I)$, and the *optimized support* range, which is the confident range that maximizes support. We often omit subscript θ if we fix the threshold.

For example, let us consider the rule

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

and suppose that the credit card loan system pays if $100\theta\%$ 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.

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 called the *optimized confidence* range, which is the ample range that maximizes the confidence (in this example, the support threshold is 200).

In this paper, we consider the optimized gain range, give a linear ($O(N)$) time algorithm for computing it, and generalize it to two-dimensional cases.

3.1 From "Programming Pearls"

In the "Programming Pearls [Ben84]" column of CACM, J. Bentley presented a problem for demonstrating importance of efficient algorithms in program design. The problem is:

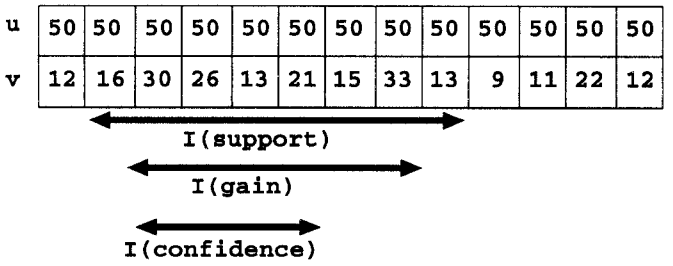

Figure 3: Optimized Ranges

> "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 algorithm (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 3.1 *The optimized gain range can be found in $O(N)$ time.*

4 Two-Dimensional Association Rules

Problem formulation

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, which consists 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 j-th *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 250000. For simplicity, we assume that $N_A = N_B = N$ from now on, 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 (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 i-th bucket and $t[B]$ is in the j-th 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 $G(i,j)$, $u_{i,j}$ is the number of tuples mapped to $G(i,j)$, and $v_{i,j}$ is the number of success tuples mapped to $G(i,j)$. Given a region P, define $support(P) = \sum_{G(i,j) \in P} u_{i,j}$. and $hit(P) = \sum_{G(i,j) \in P} v_{i,j}$. Given a threshold θ, we define $g(\theta)_{i,j} = v_{i,j} - \theta u_{i,j}$, and $gain(P) = hit(P) - \theta \times support(P) = \sum_{G(i,j) \in P} g(\theta)_{i,j}$. We want to find the *two-dimensional optimized gain rule* $(f(t) \in P) \Rightarrow C$, where P is the admissible region that maximizes $gain(P)$. The region is called an *optimized gain admissible region*.

Note that, there may be more than one optimized gain admissible regions associated with a given threshold θ. In such a case, we compute both the optimized gain admissible regions with the maximum support and the minimum support.

As in the case of ranges, we call a region *confident* (resp. *ample*) if its confidence (resp. support) is at least a given threshold value. Similarly, we define an *optimized support admissible region*, which is the confident and admissible region that maximizes support, and *optimized confidence admissible region*, which is the ample and admissible region that maximizes confidence.

Instead of admissible regions, the rectangular subregion W of G that maximizes $gain(W)$ is called the *optimized gain (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 a similar argument to that by Garey and Johnson [GJ77] for the grid Steiner tree problem.

Algorithms for computing 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 using Theorem 3.1, as follows:

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 compute the sum

$X(j) = \sum_{i=r}^{r'} g(\theta)_{i,j}$, and compute the maximum sum in any contiguous subvector of the list $X()$ (the problem in Programming Pearls). Thus, we obtain the rectangle with the maximum gain for this special case in $O(N)$ time. Since there are $O(N^2)$ candidate pairs of rows, the time complexity of the algorithm is $O(N^3) = O(n^{1.5})$. The same time complexity can be found in Fischer et al. [FHLL93].

Further improvement of this time complexity is given as a research problem in Programming Pearls [Ben84]. Similarly, by using linear time algorithms of Fukuda et al. [FMMT96a] for computing the optimized support ranges and optimized confidence ranges, we can compute the optimized support rectangle and the optimized confidence rectangle in $O(n^{1.5})$ time.

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

This time complexity is a little more expensive than that for the optimized gain admissible regions (see the next subsection), 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.

Algorithm for computing an optimized gain admissible region

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 their union, 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. [ACKT96] 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" [AKM+87] routines has been implemented in a database system.

For each $m = 1, 2, .., N$, we pre-compute 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 $\sum_{i=bottom_m(s)}^{s} g(\theta)_{i,m}$ and $\sum_{i=s}^{top_m(s)} g(\theta)_{i,m}$ are maximized, respectively.

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

Proof: Let us define $Sum_m(I) = \sum_{i \in I} g(\theta)_{i,m}$ for each integral subinterval I of $[1, N]$. Then, an inequality $Sum_m(I) + Sum_m(I') \ge Sum_m(I \cap I') + Sum_m(I \cup I')$ holds if $I \cap I' \neq \emptyset$ (indeed, equality holds). This inequality is often called the *Monge inequality*. Define an $N \times N$ matrix M whose (k, l)-th entry is $Sum_m([k, l])$ if $k \le l$, and negative infinity otherwise. Because of the Monge inequality, the matrix becomes a *totally monotone matrix*, that satisfies

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

for every $1 \le i < j+1 \le N$. It is well known [AKM+87] 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). By definition, the location of the maximum entry of the s-th row is $top_m(s)$. Thus, we can compute all $top_m(s)$ $(s = 1, 2, .., N)$ in $O(N)$ time. The computation of $bottom_m(s)$ is analogous. ∎

For two indices s and s', we define $cover_m(s, s')$ to be

$Sum[bottom_m(s), top_m(s')]$ if $s \le s'$, and
$Sum[bottom_m(s'), top_m(s)]$ if $s > s'$.

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

$$F(i, \le m+1) = \max_{1 \le j \le N} \{F(j, \le m) + cover_{m+1}(i, j)\} \quad (**)$$

or 0 if the above value is negative. From this formula, we can compute $\max_m \{\max_i F(i, \le m)\}$ and the associated region, which must be the optimized gain region.

Lemma 4.2 *If $F(j, \le m)$ for $j = 1, 2, .., N$ are given, we can compute $F(i, \le m + 1)$ for all $i = 1, 2, ..N$ in $O(N)$ time.*

Proof: We can see that the upper and lower triangle parts (D^+ and D^-, respectively) of the matrix D defined by $D(i, j) = F(j, \le m) + cover_m(i, j)$ are totally monotone matrices. Thus, in $O(N)$ time, we can compute all the row maxima of D^+ and D^-, and consequently, of D. By definition, $F(i, \le m + 1)$ is the i-th row maximum of D (or 0, if the maximum is negative). ∎

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

Proof: We solve the formula $(**)$ for $m = 1, 2, .., N$. This requires $O(N^2) = O(n)$ time. ∎

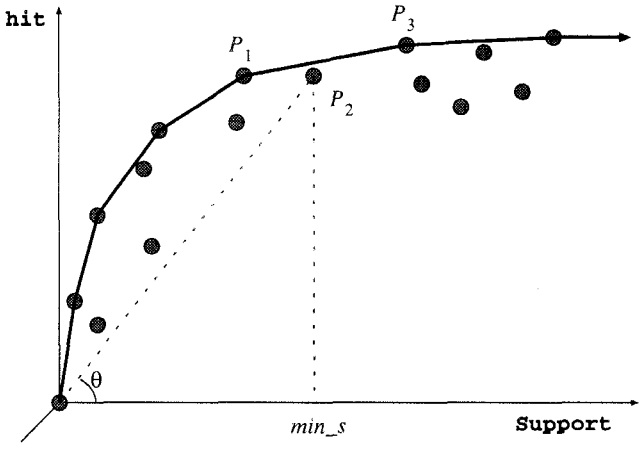

Figure 4: Convexity Assumption

Approximation algorithms for other optimized admissible regions

Unfortunately, if we consider admissible regions, the optimized support region and the optimized confidence region are difficult to compute, and 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 $P = NP$ [FMMT96b].

In this section, we substitute new optimization criteria for the optimized support and optimized confidence criteria. We will 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 4.

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 sample points that are dense in the Euclidean plane, 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. [ACKT96] 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 $hit(P) - \tau \times support(P)$, 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 4, 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 a focused region. For instance, in Figure 4, 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.*

Proof: For the 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'' that are the regions for the slope of the line connecting themselves. 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 a difference of two such different 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 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 4 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 5 (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.

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 6, 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 non-focused 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 [NKT89] to find a solution in the interior of a

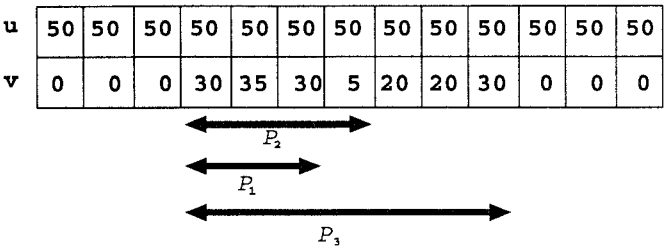

Figure 5: Optimized support range and its approximation by focused ranges.

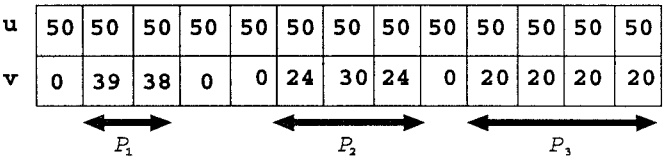

Figure 6: A bad example

feasible region. However, we have not yet implemented this heuristic, since we have encountered no such an optimized region generating an important association rule in practical data so far.

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 one of gain, support, and confidence for selecting 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 for 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 algorithm for computing focused regions has been written in C++, and implemented as a function in our DataBase SONAR (System for Optimized Numeric Association Rules) prototype. Although we have also tested our system on real databases, we evaluates its performance on synthetic test data. We performed experiments 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.

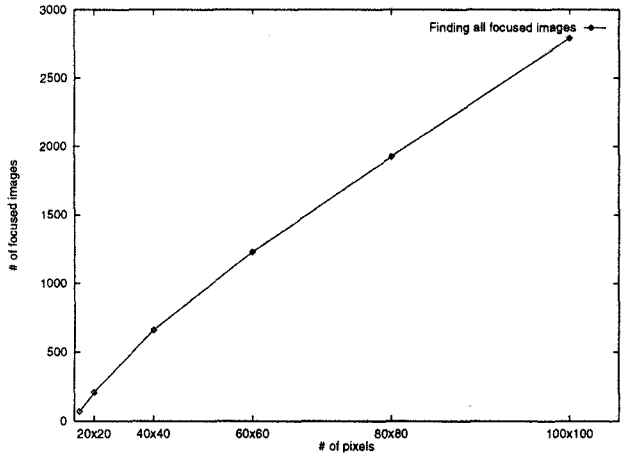

Figure 7: Number of focused regions

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 at a corner to the central cell, like spiral stairs. To determine the features of our test data, we counted how many focused regions exist in the test data (Figure 7). The number of focused regions increases almost linearly in proportion to the square root of the number of pixels. We have also done experiment on a few real data of size 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.

Figure 8 shows the execution time needed to find 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 9 and 10 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

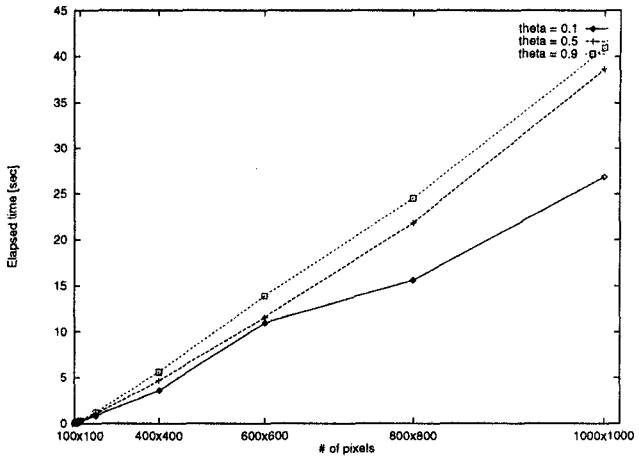

Figure 8: Finding a single focused region

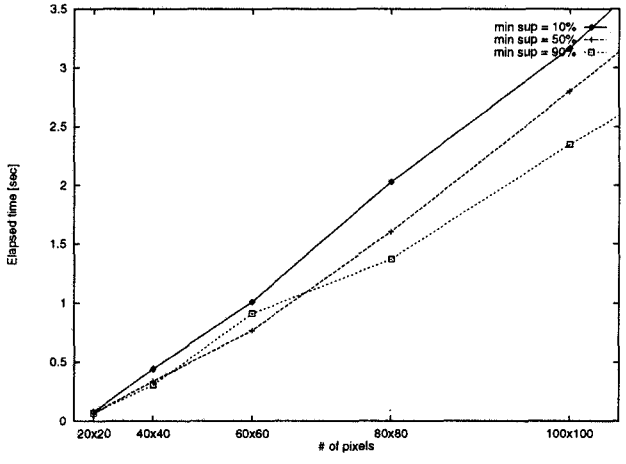

Figure 9: Finding a focused optimized confidence region

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 proportion to the number of pixels.

7 Related Works

Interrelation between paired numeric attributes is a major research topic in statistics; for example, covariance and line estimators are well-known tools for representing interrelations. However, these tools only show interrelation 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 [NH94b].

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 interrelation. However, this approach has

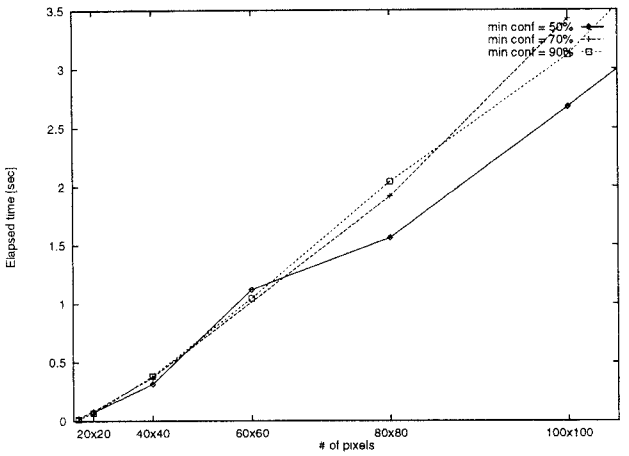

Figure 10: Finding a focused optimized support region

serious defects. These three attributes do not have equivalent roles; the Boolean attribute corresponds to the objective condition, whereas the other attributes 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 works also handle numeric attributes and try to derive rules. Piatetsky-Shapiro [PS91] studies 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%. Other ranges except for the fixed ones are not considered in his framework. Our method is not only capable of outputting optimized ranges but is also more handy than Piatetsky-Shapiro's method, since we need not make candidate ranges beforehand. Recently Srikant and Agrawal [SA96] has improved Piatetsky-Shapiro's method by adding a way of combining some consecutive ranges into one range. The combined range could be the whole range the numeric attribute, which produces just a trivial rule. To avoid this, Srikant and Agrawal present an efficient way of computing a combined range whose size is at most a threshold given by the user.

Some techniques, related 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 [Qui86, Qui93], CART [BFOS84], $\mathcal{CDP}$ [AIS93b], and SLIQ [MAR96] perform binary partitioning of numeric attributes repeatedly until each range contains data of one specific group with high probability, while $\mathcal{IC}$ [AGI+92] uses k decomposition. Since both partitionings tend to yield large decision trees, in order to reduce the size of the trees, methods such as pruning some branches and linking some ranges together have also been proposed.

Acknowledgment

The authors thank T. Asano, N. Katoh, R. Agrawal, and R. Srikant for fruitful discussions.

References

[ACKT96] Tetsuo Asano, Danny Chen, Naoki Katoh, and Takeshi Tokuyama. Polynomial-time solutions to image segmentations. In *Proc. 7th ACM-SIAM Symposium on Discrete Algorithms*, pages 104–113, 1996.

[AGI+92] R. Agrawal, S. Ghosh, T. Imielinski, B. Iyer, and A. Swami. An interval classifier for database mining applications. In *Proceedings of the 18th VLDB Conference*, pages 560–573, 1992.

[AIS93a] Rakesh Agrawal, Tako Imielinski, and Arum Swami. Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):914–925, December 1993.

[AIS93b] Rakesh Agrawal, Tako Imielinski, and Arum Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 207–216, May 1993.

[AKM+87] A. Aggarwal, M. Klawe, S. Moran, P. Shor, and R. Wilbur. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2:209–233, 1987.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th VLDB Conference*, pages 487–499, 1994.

[Ben84] Jon Bentley. Programming pearls. *Communications of the ACM*, 27(27):865–871, September 1984.

[BFOS84] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, 1984.

[FHLL93] Paul Fischer, Klaus-U Hoffgen, Hanno Lefmann, and Tomasz Luczak. Approximations with axis-aligned rectangles. In *Proceedings of the 9th International Conference on Fundamentals of Computation Theory.* Springer-Verlag, August 1993.

[FMMT96a] Takeshi Fukuda, Yasuhiko Morimoto, Shinichi Morishita, and Takeshi Tokuyama. Mining optimized association rules for numeric attributes. In *Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, June 1996.

[FMMT96b] Takeshi Fukuda, Yasuhiko Morimoto, Shinichi Morishita, and Takeshi Tokuyama. Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization. In *Technical Report, IBM Tokyo Research Laboratory*, 1996.

[GJ77] M. R. Garey and D. S. Johnson. The rectilinear steiner tree problem is np complete. *SIAM J. Appl. Math*, 32:836–834, 1977.

[HCC92] Jiawei Han, Yandong Cai, and Nick Cercone. Knowledge discovery in databases: An attribute-oriented approach. In *Proceedings of the 18th VLDB Conference*, pages 547–559, 1992.

[KKS94] D. Keim, H. Kriegel, and T. Seidl. Supporting data mining of large database by visual feedback queries. In *Proc. 10th Data Enginieering*, pages 302–313, 1994.

[MAR96] Manish Mehta, Rakesh Agrawal, and Jorma Rissanen. Sliq: A fast scalable classifier for data mining. In *Proceedings of the Fifth International Conference on Extending Database Technology*, 1996.

[NH94a] Raymond T. Ng and Jiawei Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th VLDB Conference*, pages 144–155, 1994.

[NH94b] Raymond T. Ng and Jiawei Han. Efficient and effective clustering methods for spatial data mining. In *Proc. 20th VLDB Conference*, pages 144–155, 1994.

[NKT89] G. L. Nemhauser, A. H. G. Rinnoy Kan, , and M. J. Todd. *Optimization: Handbooks in Operations Research and Management Science Vol.1.* North-Holland, 1989.

[PCY95] Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. An effective hash-based algorithm for mining association rules. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 175–186, May 1995.

[PS91] G. Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In *Knowledge Discovery in Databases*, pages 229–248, 1991.

[PSF91] G. Piatetsky-Shapiro and W. J. Frawley, editors. *Knowledge Discovery in Databases.* AAAI Press, 1991.

[Qui86] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[SA96] Ramakrishnan Srikant and Rakesh Agrawal. Mining quantitative association rules in large relational tables. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, June 1996.

[SAD+93] Michael Stonebraker, Rakesh Agrawal, Umeshwar Dayal, Erich J. Neuhold, and Andreas Reuter. DBMS research at a crossroads: The vienna update. In *Proceedings of the 19th VLDB Conference*, pages 688–692, 1993.