Computing Optimized Rectilinear Regions for Association Rules *

Kunikazu Yoda Takeshi Fukuda Yasuhiko Morimoto

Shinichi Morishita Takeshi Tokuyama

IBM Tokyo Research Laboratory
{yoda, fukudat, morimoto, morisita, ttoku}@trl.ibm.co.jp

Abstract

We address the problem of finding useful regions for two-dimensional association rules and decision trees. In a previous paper we presented efficient algorithms for computing optimized *x-monotone* regions, whose intersections with any vertical line are always undivided. In practice, however, the quality of x-monotone regions is not ideal, because the boundary of an x-monotone region tends to be notchy, and the region is likely to overfit a training dataset too much to give a good prediction for an unseen test dataset.

In this paper we instead propose the use of a *rectilinear region* whose intersection with any vertical line and whose intersection with any horizontal line are both undivided, so that the boundary of any rectilinear region is never notchy. This property is studied from a theoretical viewpoint. Experimental tests confirm that the rectilinear region less overfits a training database and thefore provides a better prediction for unseen test data. We also present a novel efficient algorithm for computing optimized rectilinear regions.

1 Introduction

Recent progress in technologies for data input 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 inspire new marketing strategies. 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.

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 will 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]. Ideally we expect to have rules with 100% confidence, but in the case of business databases, rules whose confidences are less than 100% but greater than a specified threshold, say 50%, are significant. We call such rules *confident*[2].

In their pioneering work (Agrawal, Imielinski, & Swami 1993), Agrawal, Imielinski, and Swami investigated how to find all confident rules. They focused on rules with conditions that are conjunctions of $(A = yes)$, where A is a Boolean attribute, and presented an efficient algorithm. They have applied the algorithm to basket-data-type retail transactions to derive associations between items, such as $(Pizza = yes) \wedge (Coke = yes) \Rightarrow (Potato = yes)$. Improved versions of their algorithm have also been reported (Agrawal & Srikant 1994; Park, Chen, & Yu 1995).

One-Dimensional Association Rules

In addition to Boolean attributes, databases in the real world usually have numeric attributes, such as age and balance of account in the case of a bank database. Thus, it is also important to find association rules for numeric attributes. We previously (Fukuda *et al.* 1996a) considered the problem of finding simple rules of the form $(Balance \in [v_1, v_2]) \Rightarrow (CardLoan = yes)$, which expresses the fact that customers whose balances fall within a range between v_1 and v_2 are likely to take out credit card loans. If the confidence of the range exceeds a given threshold, the range is called *confident*. Let the *support* of the range be the number of tuples in the range, and let the range be called *ample*[3] if the support of the range exceeds a given threshold.

We want to compute a rule associated with a range that is both confident and ample. Since there could exist many confident and ample ranges, we are interested in finding the confident range with the maximum support (called the *optimized-support* range), and the ample range with the maximum confidence (called the *optimized-confidence* range). Fukuda et al. (Fukuda *et al.* 1996a) presented efficient algorithms for computing those optimized ranges that effectively represent the relationship between a numeric attribute and a Boolean

*Copyright 1997, American Association for Artificial Intelligence (www.aaai.org). All rights reserved. This research is supported by the Advanced Software Enrichment Project of the Information-Technology Promotion Agency.

[1]The confidence of a rule $C_1 \Rightarrow C_2$ is the conditional probability $Pr(C_2|C_1)$.

[2] *"high-confidence"* in another word.

[3] *"well-supported"* in another word.

attribute.

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 we therefore often want to find a rule for more than two attributes. We previously (Fukuda *et al.* 1996b) considered the problem of finding *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 that a tuple t meets the 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 must consider huge databases containing millions of tuples, and hence we have to handle millions of points, which may occupy much more space than the available 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 map each tuple t to the pixel containing the point $(t[A], t[B])$. Let M denote the total number of records in the given database. We expect that the average number of records mapped to one pixel is neither too small nor too large. In practice, we assume that $\sqrt[3]{M} \leq N \leq \sqrt{M}$, thereby ensuring that the average number of records mapped to one pixel ranges from 1 to $\sqrt[3]{M}$. We use a union of pixels as the region of a two-dimensional association rule. Confident regions, ample regions, and optimized-confidence(-support) regions can be naturally defined as in the case of one-dimensional association rules.

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. (Keim, Kriegel, & Seidl 1994). However, such a region P may consist of many connected components, often creating an association rule that is very difficult to characterize, and whose validity is consequently 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 with nice geometric properties.

X-monotone Regions

In a previous paper (Fukuda *et al.* 1996b), we considered two classes of geometric regions: rectangular regions, and x-monotone regions. Apte et al. (Apte *et al.* 1995) also report the use of ranges and rectangular regions in rules for prediction. A region is called *x-monotone* if its intersection with any vertical line is undivided. Optimized x-monotone regions cannot be computed in polynomial time of N and $\log M$ unless $P = NP$; however, we gave an $O(N^2 \log M)$-time approximation algorithm (Fukuda *et al.* 1996b).

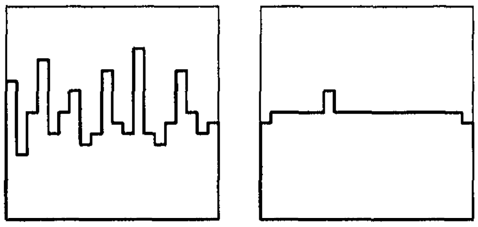
Figure 1: Optimized-Confidence X-monotone (Left) and Rectilinear (Right) Regions

Assuming that $N \leq \sqrt{M}$, the time complexity of the algorithm is $O(M \log M)$. We also presented an $O(N^3)$-time algorithm for computing the optimized-confidence (support) rectangles.

We introduced x-monotone regions to show that various shapes of region can be represented by x-monotone regions, and that the confidence (support) of the optimized-confidence (-support) x-monotone region is much greater than that of the optimized-confidence (-support) rectangle, since any rectangle is an instance of an x-monotone region.

For various reasons, however, x-monotone regions are not ideal. One reason is that the boundary of an x-monotone region tends to be notchy; for instance, Figure 1 (left) shows an instance of an optimized-confidence x-monotone region. In Section 4, we give details of this instance, and present a theoretical analysis of the shape of the boundary of the optimized-confidence x-monotone region.

Another problem is that an optimized x-monotone region is likely to overfit the training dataset seriously, and therefore generally fails to give a good prediction for an unseen dataset. To be more precise, even if we create the optimized-confidence x-monotone region from a training dataset, the confidence of the region against an unseen dataset tends to be worse than the original confidence against the training dataset. With regard to this problem, we present some experimental results in Section 5.

Main Results

We are looking for appropriate classes of regions between two extreme classes, rectangles and x-monotone regions; in this paper we propose the use of *rectilinear regions*. A connected region is called *rectilinear* if both its intersection with any vertical line and its intersection with any horizontal line are always undivided. From this definition, the boundary of a rectilinear region is never notchy; for instance, Figure 1 (right) shows the optimized-confidence rectilinear region generated from the same grid as that used for the x-monotone region in Figure 1 (left). In Section 4 we present a theoretical analysis of this subject. Furthermore, compared with the case of x-monotone regions, an optimized rectilinear region less overfits a training dataset and provides a better prediction for an unseen dataset, which is confirmed by some experiments in Section 5.

We present $O(N^3 \log M)$-time approximation algorithms in Section 3. Assuming that $N \leq \sqrt{M}$, the time complexity of these algorithms is $O(M^{3/2} \log M)$, which is feasible in practice.

An interesting application of rectilinear regions is decision-tree construction. Previously (Fukuda *et al.* 1996c), we proposed the construction of decision trees in which at each node, to split data effectively into two classes, we use the x-monotone region that minimizes Quinlan's entropy function. It is an interesting question whether or not the use of rectilinear regions in place of x-monotone regions might increase the classification accuracy of decision trees with region splitting, and we will report some experimental results in a forth comming paper (Fukuda *et al.* 1997).

Related Work

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 the interrelation in an entire data set, not in a subset of the data in which a strong interrelation holds. Several heuristics using geometric clustering techniques have been introduced for this purpose (Ng & Han 1994; Zhang, Ramakrishnan, & Linvy 1996).

Some other studies deal with numeric attributes and try to derive rules. Piatetsky-Shapiro (Piatetsky-Shapiro 1991) investigates 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. Recently Srikant and Agrawal (Srikant & Agrawal 1996) have improved Piatetsky-Shapiro's method by adding a way of combining several consecutive ranges into a single range. The combined range could be the whole range of the numeric attribute, which produces 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. Their approach can generate hypercubes.

Some techniques have been developed for handling numeric attributes in the context of deriving decision trees. ID3 (Quinlan 1986; 1993), CART (Breiman *et al.* 1984), $\mathcal{CDP}$ (Agrawal, Imielinski, & Swami 1993), and SLIQ (Mehta, Agrawal, & Rissanen 1996) subject a numeric attribute to a binary partitioning, called a *guillotine cut*, that maximizes Quinlan's entropy function (Quinlan 1986). To the best of our knowledge, the idea of splitting a region to maximize the entropy function has never been seriously exploited except by the present authors (Fukuda *et al.* 1996c).

2 Two-Dimensional Association Rules

Pixel Regions

Definition 2.1 Let us consider two numeric attributes A and B. 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 250,000. For simplicity, we assume that $N_A = N_B = N$ from now on, although this assumption is not essential.

Definition 2.2 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 *rectangular* if it is a rectangle. A pixel region is *x-monotone* if it is connected and its intersection with each column is undivided (thus, a vertical range) or empty. A pixel region is *rectilinear* if it is connected, its intersection with each column (vertical line) is undivided or empty, and its intersection with each row (horizontal line) is undivided or empty. ▮

Optimized Two-Dimensional Association Rules

The following is the formal definition of optimized two-dimensional association rules.

Definition 2.3 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.

Let C denote an objective condition. A tuple t is a *success* tuple if t satisfies C. 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, the number of tuples mapped to a pixel in P, $\sum_{G(i,j)\in P} u_{i,j}$, is called the *support* of P, or *support*(P). ▮

In this paper, the support of P denotes a number of tuples rather than a percentage, since we do not want to declare the base of the percentage each time.

Definition 2.4 The number of success tuples mapped to a pixel in P, $\sum_{G(i,j)\in P} v_{i,j}$, is called the *hit* of P, or *hit*(P). The ratio of the number of success tuples to the number of tuples mapped to P, that is, *hit*(P)/*support*(P), is called the *confidence* of P, or *conf*(P). ▮

In order to express an association rule that tuples mapped to region P also meet condition C with a probability of *conf*(P), we use the following notation:

$$(A,B) \in P \Rightarrow C,$$

which is called a *two-dimensional* association rule.

Definition 2.5 A region is called *confident* (resp. *ample*) if the confidence (the *support*) is at least a given threshold. We want to find a rule associated with a region that is both confident and ample, but there could be many such regions. An *optimized-confidence* region is the ample region that maximizes confidence. An *optimized-support* region is the confident region that maximizes support. ∎

We have the following property similar to the case for x-monotone regions.

Theorem 2.1 *An $O(N^3M)$ time algorithm exists for computing the optimized-support (-confidence) rectilinear region, but no algorithm running in polynomial time with respect to N and log M exists unless $P = NP$.*

Proof: See (Yoda *et al.* 1997). ∎

In the next section, we present an $O(N^3 \log M)$-time approximation algorithm for computing optimized-support (-confidence) rectilinear regions.

3 Approximation Algorithms for Optimized Rectilinear Regions

Hand-Probing Technique

In this section, we briefly review the "hand-probing" technique, which Asano et al. (Asano *et al.* 1996) invented for image segmentation and Fukuda et al. (Fukuda *et al.* 1996b) modified for extraction of optimized x-monotone regions. Here we also tailor the algorithm to compute the optimized rectilinear regions.

Definition 3.1 We map each rectilinear region P to a *stamp* point $(support(P), hit(P))$ in an Euclidean plane. ∎

Since there are more than 2^N rectilinear regions, we cannot afford to generate all stamp points; instead, we just imagine them. Consider the upper convex hull of all of the stamp points. A stamp point, say P_1 in Figure 2, associated with the optimized-confidence (-support) rectilinear region does not always exist on the hull. In practice, however, a huge number of points are fairly densely scattered over the upper hull, and it is reasonable to assume that we can find a point, say P_2 in Figure 2, on the hull that is pretty close to P_1. This *convexity assumption*, as we call it, motivates us to create an approximation algorithm for computing optimized rectilinear regions.

The next question is how to compute a point on the upper hull. To this end, we employ the *hand-probing technique* given by Asano et al. (Asano *et al.* 1996). For each stamp point on the upper hull, there must exist a line with a slope τ that is tangential to the hull at this point. This point maximizes $y - \tau x$ for the set of stamp points. Accordingly, the corresponding rectilinear region P maximizes $hit(P) - \tau \times support(P)$, which is $\sum_{G(i,j)\in P} v_{i,j} - \tau u_{i,j}$.

Definition 3.2 Let us call $v_{i,j} - \tau u_{i,j}$ the *gain* of pixel $G(i,j)$, and $hit(P) - \tau \times support(P)$ the *gain* of P, respectively. Among all the rectilinear regions, the optimized-*gain* rectilinear region P maximizes the gain of P. ∎

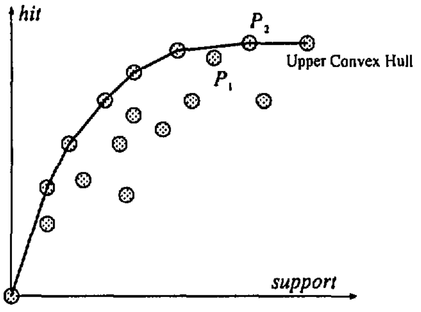

Figure 2: Convexity Assumption

Using the hand-probing technique, we need to scan stamp points on the upper convex hull efficiently to find the optimal point, and we have shown that this can be done by using $O(\log M)$ tangent lines (Fukuda *et al.* 1996b).

In the next subsection, we present an $O(N^3)$-time algorithm for computing the optimized-gain rectilinear region in response to a tangent line with slope τ.

Optimized-Gain Rectilinear Regions

Let P be a rectilinear 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 i-th column. Since P is a rectilinear region, the sequence of top pixels from left to right, $G(i,t(i))$ for $i = m_1, \ldots, m_2$, increases monotonically ($t(i) \leq t(j)$ for $m_1 \leq i < j \leq m_2$), decreases monotonically ($t(i) \geq t(j)$ for $m_1 \leq i < j \leq m_2$), or increases monotonically up to some column and then decreases monotonically. Similarly, the sequence of bottom pixels, $G(i,s(i))$ for $i = m_1, \ldots, m_2$, increases monotonically, decreases monotonically, or decreases monotonically up to some column and then increases monotonically.

The top picture in Figure 3 illustrates the case in which the sequence of the top pixels (the *top sequence*, for short) and the sequence of the bottom pixels (the *bottom sequence*, for short) increase monotonically or decrease monotonically.

Definition 3.3 We have four types of rectilinear region. A region that gets wider from left to right is named W. A region that slants upward (downward, resp.) is named U (D). A region that gets narrower from left to right is named N. ∎

The middle left part of Figure 3 shows the case in which the top sequence increases monotonically up to some column and then decreases monotonically, while the bottom sequence increases monotonically or decreases monotonically. We have two types of rectilinear regions: one is a combination of a W-type sub-rectilinear region and a D-type sub-rectilinear region, which will be referred as a WD-type region. The other is a combination of a U-type sub-rectilinear region and an N-type sub-rectilinear region, which will be called a UN-type region. The middle right part of Figure 3 shows the case in which the top sequence increases or decreases, while the bottom sequence decreases up to some column and then decreases. We have two types of rectilinear region: one is WU-type, and the other is DN-type. The bottom part of Figure 3 shows the case in which the top sequence increases up to some column and then decreases, while the bottom sequence decreases until some column and then increases. We have three types: WDN, WN, and WUN.

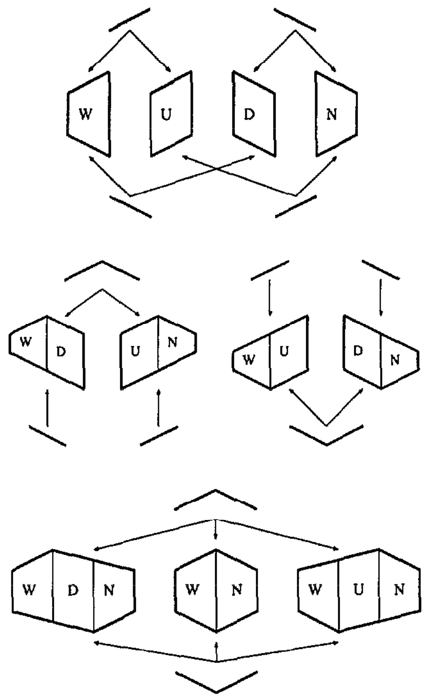

Figure 3: Rectilinear Regions

Theorem 3.1 *Optimized-gain rectilinear regions can be computed in $O(N^3)$ time.*

Proof: Let $f_W(m,[s,t])$ denote the gain of the rectilinear region that maximizes the gain among all W-type rectilinear regions whose last column is the m-th one and whose intersection with the m-th column ranges from the s-th pixel to the t-th pixel. Let $g_{i,j}$ denote the gain of the pixel $G(i,j)$, which is $v_{i,j} - ru_{i,j}$. Before computing $f_W(m,[s,t])$, 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. For $m = 1$, $f_W(1,[s,t]) = g_{1,[s,t]}$. For $m > 1$, if $s = t$, $f_W(m,[s,s]) = \max\{g_{m,s}, f_W(m-1,[s,s]) + g_{m,s}\}$. Otherwise ($s < t$), the following recurrence holds:

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

Next, let $f_U(m,[s,t])$ denote the gain of the rectilinear region that maximizes the gain among all U or WU rectilinear regions whose last column is the m-th one and whose intersection with the m-th column ranges from the s-th pixel to the t-th pixel. For $m = 1$, $f_U(1,[s,t]) = g_{1,[s,t]}$. For $m > 1$, we pre-compute $\max_{i\le s} f_W(m-1,[i,t])$ and $\max_{i\le s} f_U(m-1,[i,t])$ for all $s \le t$, which can be done, in $O(N^2)$ time. We have the following recurrence:

$$f_U(m,[s,t]) = \max\begin{pmatrix} \max_{i\le s} f_W(m-1,[i,t]) + g_{m,[s,t]} \\ \max_{i\le s} f_U(m-1,[i,t]) + g_{m,[s,t]} \\ f_U(m,[s,t-1]) + g_{m,t} \\ (\text{or } g_{m,t} \text{ when } s = t) \end{pmatrix}$$

Next, let $f_D(m,[s,t])$ denote the gain of the rectilinear region that maximizes the gain among all D or WD rectilinear regions whose last column is the m-th one and whose intersection with the m-th column ranges from the s-th pixel to the t-th pixel. For $m = 1$, $f_D(1,[s,t]) = g_{1,[s,t]}$. For $m > 1$, we pre-compute $\max_{t\le i} f_W(m-1,[s,i])$ and $\max_{t\le i} f_U(m-1,[s,i])$, and then obtain the recurrence:

$$f_D(m,[s,t]) = \max\begin{pmatrix} \max_{t\le i} f_W(m-1,[s,i]) + g_{m,[s,t]} \\ \max_{t\le i} f_D(m-1,[s,i]) + g_{m,[s,t]} \\ f_D(m,[s+1,t]) + g_{m,s} \\ (\text{or } g_{m,s} \text{ when } s = t) \end{pmatrix}$$

Finally, let $f_N(m,[s,t])$ denote the gain of the rectilinear region that maximizes the gain among all rectilinear regions whose type is N, UN, DN, WDN, WN, or WUN, whose last column is the m-th one, and whose intersection with the m-th column ranges from the s-th pixel and the t-th pixel. For $m = 1$, $f_N(1,[s,t]) = g_{1,[s,t]}$. For $m > 1$, we have the following recurrence:

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

In this case, we need to compute $f_N(m,[s,t])$ by using $f_N(m,[s-1,t])$ and $f_N(m,[s,t+1])$, which have been computed for longer ranges $[s-1,t]$ or $[s,t+1]$. Thus we first compute $f_N(m,[1,N]) = \max\{f_N(m-1,[1,N]) + g_{m,[1,N]}, g_{m,[1,N]}\}$.

Consequently a simple dynamic programming gives us an $O(N^3)$ time solution for computing $f_W(m,[s,t])$, $f_U(m,[s,t])$, $f_D(m,[s,t])$, and $f_N(m,[s,t])$ for all m and $s \le t$. ∎

4 Mathematical Analysis of a Model

In this section we present a model to explain the advantage of rectilinear regions over x-monotone regions. Although the model may look very specialized and artificial, a similar situation was frequently observed in which the optimal x-monotone regions appeared to be strange.

Consider the sample grid G shown in Figure 4. Suppose that the support of each pixel is 1 (which means thatthe total number of tuples in the grid is N^2) and that the confidence of each pixel is 0 or 1. We denote a pixel whose confidence is 1 by a black box. We denote the lower half part of the grid by R, which consists of the rows below (and including) the $N/2$-th row in Figure 4. The row index is counted from the bottom. We fix a constant K ($\leq N/2$), denote the region below and including the $(N/2-K)$-th row by R', and let $R'' = R - R'$. The grid in Figure 4 is created according to the following rules:

1. Each pixel in R' has the confidence 1.
2. Each pixel in the $N/2-i$-th row has the confidence 1 with a probability of $(i+1)/K$, for $i < K$.
3. Each pixel in $G - R$ has the confidence 1 with a probability of $1/N$.

The intuition behind the above rules is that the confidence of each grid decreases gradually from lower rows to higher in the presence of a low level of noise. Figure 4 shows an instance in which $N = 20$ and $K = 5$.

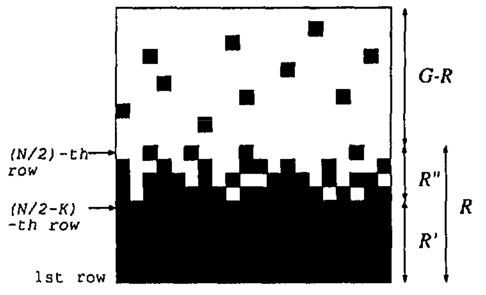

Figure 4: Sample Grid

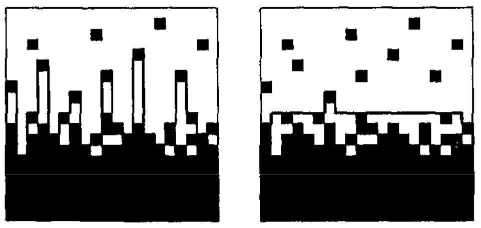

Figure 5: Optimized-Confidence X-monotone (Left) and Rectilinear (Right) Region

Consider the set of all possible grids generated according to the above rules, which we call the *universe*. The universe contains the grid in Figure 4 as an instance. Let us use $0.5N^2$ as the minimum support threshold. For each grid in the universe, let us consider how to generate the optimized-confidence x-monotone region $R_{monotone}$ and the optimized-confidence rectilinear region R_{recti}. Considering the way of constructing grids in the universe, we expect that the shape of the optimized-confidence region is fairly close to that of R, and hence we are interested in the support of the set differences $R_{monotone} - R$ and $R_{recti} - R$. We can prove the following result:

Theorem 4.1 *The expected support of $R_{monotone} - R$ is at least $\sqrt{2K} \cdot N$, while the support of $R_{recti} - R$ is bounded by $2K \log^2 K + O(K \log N)$, which is better than the x-monotone case by a factor of $N/(\sqrt{2K} \log^2 K)$.*

Proof: See (Yoda *et al.* 1997). ▮

Note that the real factor might be much larger than the above theoretical factor. For instance, Figures 5(left) and 5(right) respectively show the optimized-confidence x-monotone region $R_{monotone}$ and the optimized-confidence rectilinear region R_{recti} for the grid in Figure 4. In this particular case, where $N = 20$ and $K = 5$, the support of $R_{monotone} - R$ is 24, and that of $R_{recti} - R$ is 2.

The reader might think that the above particular model favors the family of rectilinear regions over that of x-monotone regions. The advantage of rectilinear regions can also be shown by other typical distributions. For example, if the core R' of the region is an axis-parallel rectangle whose horizontal width is $\Omega(N)$ and whose vertical width is smaller than that, and the boundary region R'' forms K layers whose distribution is the same as in the above example, the same asymptotic analysis can be obtained.

However, it is difficult to give a mathematical analysis without fixing the distribution and rule. We therefore experimentally show the tolerance of rectilinear convex regions in the next section.

5 Experimental Results

Overfitting

In this section, we experimentally show that an optimized x-monotone region is likely to overfit the training dataset seriously, and therefore tends to fail to give a good prediction for an unseen dataset, while we remark that optimized rectilinear regions do not suffer from this overfitting problem so much.

For this experiment we generate synthetic datasets that represent typical cases in practice. Let A and B be numeric attributes such that the domain of both A and B is the interval ranging from -1 to 1, and let C be an objective Boolean attribute. We generate a dataset whose tuples are generated according to the following procedure:

1. Generate random points that are uniformly distributed in $[-1,1] \times [-1,1]$.
2. For each point $(t[A], t[B])$, we determine the value of $t[C]$ as follows, and add tuple t to the dataset.

   Let $p(x,y)$ be a function from $[-1,1] \times [-1,1]$ to $[0,1]$. Set 1 to $t[C]$ with a probability of $p(t[A], t[B])$, and set 0 to $t[C]$ otherwise.

We use the following two functions for $p(x,y)$:

- $p_{linear}(x,y) = \frac{1}{\sqrt{\pi}} \exp(-\frac{(x-y)^2}{2})$, which is the normal distribution $N(0, \sqrt{2}/2)$ with respect to the distance between (x,y) and the diagonal $y = x$.

No. of Pixels	time in second		
	rectangular	x-monotone	rectilinear
8×8	0.001	0.014	0.012
16×16	0.008	0.089	0.087
32×32	0.051	0.378	0.746
64×64	0.364	1.830	6.502
128×128	2.747	7.983	63.018

Table 3: Execution time of computing optimized regions

Performance in Computing Rectilinear Regions

To measure the worst-case performance, we need to produce a synthetic dataset so that we have a large number of stamp points that are densely scattered on the upper convex of all the stamp points (recall Figure 2). To this end, we generated the dataset as follows: we first generated random numbers uniformly distributed in $[N^2, 2N^2]$ and assigned them to $u_{i,j}$, and then we assigned $1, 2, \ldots, N^2$ to $v_{i,j}$ from a cell in the lower-right corner to the central cell in clockwise rotation, like spiral stairs.

The experiments were carried out by using our prototype system, called Database SONAR (System for Optimized Numeric Association Rules). The programs were written in C++ and run on one node of an IBM SP2 workstation, each node of which has a 66-MHz Power2 chip, 2 MB of L2 cache, and 256 MB of main memory, running under the AIX operating system, version 4.1.

Tables 3 shows the execution times required to find the optimized confidence regions for the rectangular, rectilinear, and x-monotone types, respectively, with minimum supports of 50% and for numbers of pixels ranging from 8×8 to 128×128. These experimental tests confirmed that the computation time for each of the optimized rectangular regions, x-monotone regions, and rectilinear regions is respectively $O(n^{1.5})$, $O(n \ln M)$, and $O(n^{1.5} \ln M)$, where n is the number of pixels and M is the number of tuples.

References

Agrawal, R., and Srikant, R. 1994. Fast algorithms for mining association rules. In *Proceedings of the 20th VLDB Conference*, 487–499.

Agrawal, R.; Imielinski, T.; and Swami, A. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, 207–216.

Apte, C.; Hong, S. J.; Prasad, S.; and Rosen, B. 1995. Ramp: Rules abstraction for modeling and prediction. Technical Report RC-20271, IBM Watson Research Center.

Asano, T.; Chen, D.; Katoh, N.; and Tokuyama, T. 1996. Polynomial-time solutions to image segmentations. In *Proc. 7th ACM-SIAM Symposium on Discrete Algorithms*, 104–113.

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

Fukuda, T.; Morimoto, Y.; Morishita, S.; and Tokuyama, T. 1996a. Mining optimized association rules for numeric attributes. In *Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 182–191.

Fukuda, T.; Morimoto, Y.; Morishita, S.; and Tokuyama, T. 1996b. Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, 13–23.

Fukuda, T.; Morimoto, Y.; Morishita, S.; and Tokuyama, T. 1996c. Constructing efficient decision trees by using optimized association rules. In *Proceedings of the 22nd VLDB Conference*, 146–155.

Fukuda, T.; Morimoto, Y.; Morishita, S.; and Tokuyama, T. 1997. Implementation and evaluation of decision trees with range and region splitting. Technical Report RT0202, IBM Tokyo Research Laboratory.

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

Mehta, M.; Agrawal, R.; and Rissanen, J. 1996. Sliq: A fast scalable classifier for data mining. In *Proceedings of the Fifth International Conference on Extending Database Technology.*

Ng, R. T., and Han, J. 1994. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th VLDB Conference*, 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 ACM SIGMOD Conference on Management of Data*, 175–186.

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

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning* 1:81–106.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning.* Morgan Kaufmann.

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.*

Yoda, K.; Fukuda, T.; Morimoto, Y.; Morisita, S.; and Tokuyama, T. 1997. Computing optimized rectilinear regions for association rules. Technical Report RT0201, IBM Tokyo Research Laboratory.

Zhang, T.; Ramakrishnan, R.; and Linvy, M. 1996. Birch: An efficient data clustering method for very large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, 103–114.