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 inspires 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 $C_1 \Rightarrow C_2$. Ideally we expect to have only rules with 100% confidence, but in the case of business databases that reflect our loosely controlled world, rules whose confidences are less than 100% but greater than a specified threshold, say 50%, are significant. We call such rules *confident*.

In their pioneering work [1], 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 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 [2, 14].

One-Dimensional Association Rules

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 [8], we 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* if the support of the range in the rule 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. [8] presented efficient algorithms for computing those optimized ranges that effectively represent relationship 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 we therefore often want to find a rule for more than two attributes. In [7], we 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 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 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 not 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. [10]. 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 [7] we considered two classes of geometric regions: rectangular regions, and x-monotone regions. Apte et al.[3] 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. Computing optimized x-monotone regions is intractable in general, but we gave an $O(N^2 \log M)$-time approximation algorithms for computing optimized x-monotone regions. We also presented an $O(N^3)$-time algorithms for computing the optimized-confidence (support) rectangles. Assuming that $N \leq \sqrt{M}$, the time complexity of the former algorithm is $O(M \log M)$, and that of the latter algorithm is $O(M^{3/2})$.

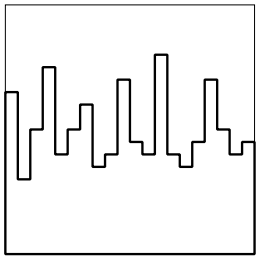
The motivation for the introduction of x-monotone regions was that various shapes of region can be represented by x-monotone regions, and the confidence (support) of the optimized-confidence (-support) x-monotone region is much greater than that of the optimized-confidence (-support) rectangle, since 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(a) illustrates an instance of an optimized-confidence x-monotone region. In Section 4, we give the details of the grid, and we present theoretical analysis on the shape of the boundary of the optimized-confidence x-monotone region.

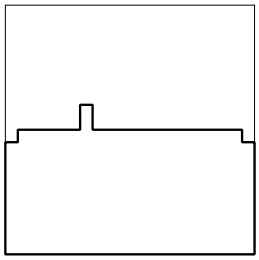
Another problem is that an optimized x-monotone region is likely to overfit the training dataset seriously, and therefore it tends to fail to give a good prediction on an unseen dataset. In more precise, even if we create the optimized-confidence x-monotone region from a training dataset, the confidence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

Main Results

We look for appropriate classes of regions between two extreme classes, rectangles and x-monotone regions, and in this paper we propose to use *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(b) shows the optimized-confidence rectilinear region generated from the same grid for the x-monotone region in Figure 1(a).



(a) Optimized-confidence X-monotone Region



(b) Optimized-confidence Rectilinear Region

Figure 1: Optimized-confidence Regions

In Section 4 we present a theoretical analysis on 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 on an unseen dataset, which is confirmed by some experiments in Section 5.

Analyzing real data, we have often observed that computing the optimized triangular region is meaningful, but unfortunately it becomes time-expensive to find the optimal one in the family of triangular regions. A rectilinear region can approximate a triangular region, which is another reason why we design an efficient algorithm for computing the optimized rectilinear region.

As in the case of x-monotone regions, the problem of computing the optimized-confidence/-support regions is intractable, but 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 making. In [6], we proposed to construct 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 replace of x-monotone regions might increase the classification accuracy of decision trees with region splitting, and we will report some experimental results in the full version of this paper.

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, and thus cannot extract a subset of data in which a strong interrelation holds. Several heuristics using geometric clustering techniques have been introduced for this purpose [13, 19].

Some other studies deal with numeric attributes and try to derive rules. Piatetsky-Shapiro [15] 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 [18] 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. Srikant and Agrawal's approach can generate hypercubes.

Some techniques have been developed for handling numeric attributes in the context of deriving decision trees. ID3 [16, 17], CART [5], $\mathcal{CDP}$ [1], and SLIQ [11] subject a numeric attribute to a binary partitioning, called a *guillotine cut*, that maximizes the Quinlan's entropy function [16]. To the best our knowledge, the idea of splitting a region to maximize the entropy function has never been seriously exploited except by the present authors [6].

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

We are now in a position to formally define 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, which is denoted by $support(P)$. ∎

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, and is denoted by $hit(P)$. 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, and is denoted by $conf(P)$. ∎

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* rectilinear (resp. x-monotone, rectangular) region is the ample rectilinear (x-monotone, rectangular) region that maximizes confidence. An *optimized-support* rectilinear (resp. x-monotone, rectangular) region is the confident rectilinear (x-monotone, rectangular) region that maximizes support. ∎

If we consider rectangular regions, the optimized-support (-confidence) region can be computed in $O(N^3)$

time [7]. In the case of x-monotone regions, the optimized-support region and the optimized-confidence region are difficult to compute. Let M denote the total number of tuples. In [7], Fukuda et al. show that an $O(N^3M)$ time algorithm exists for computing the optimized-support (-confidence) x-monotone region, but no algorithm running in polynomial time with respect to N and $\log M$ exists unless $P = NP$. Computing the optimized rectilinear regions is also difficult, and 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 Appendix A. ∎

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. [4] invented for image segmentation and Fukuda et al. [7] modified for extraction of the 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. [4]. 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 $\sum_{G(i,j) \in P} v_{i,j} - \tau u_{i,j}$.

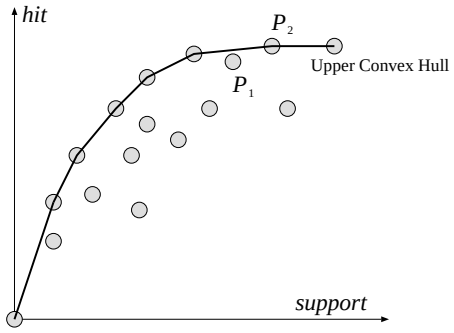


Figure 2: Convexity Assumption

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. ∎

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.

The hand-probing technique allows us to touch one point on the upper hull, but that point is not necessarily the one associated with the optimized-confidence/-support rectilinear region. Thus, using the hand-probing technique, we want to scan the upper convex hull efficiently to find the optimal point, and we show that this can be done by using $O(\log M)$ tangent lines.

For the tangent line with slope τ, we can compute the optimized-gain rectilinear region P. Note that when τ increases, the confidence of P increases, and the support of P decreases monotonically. Thanks to this property, we can perform a binary search for the optimized-confidence/-support rectilinear region as follows:

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.
3. Repeat this process until we find P' and P'' such that P' and P'' themselves are identified by a line whose slope is equal to that of the slope of the line connecting them.

Figure 3 illustrates this process.

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

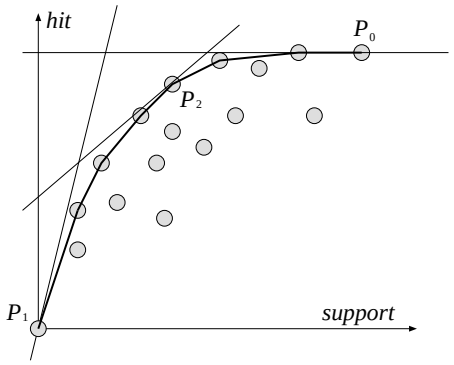


Figure 3: Binary Search on the Upper Hull

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.

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 4 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. The region that gets wider from left to right is named W. The region that slants upward (downward, resp.) is named U (D). The region that gets narrower from left to right is named N. ∎

The middle left part of Figure 4 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 4 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 4 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.

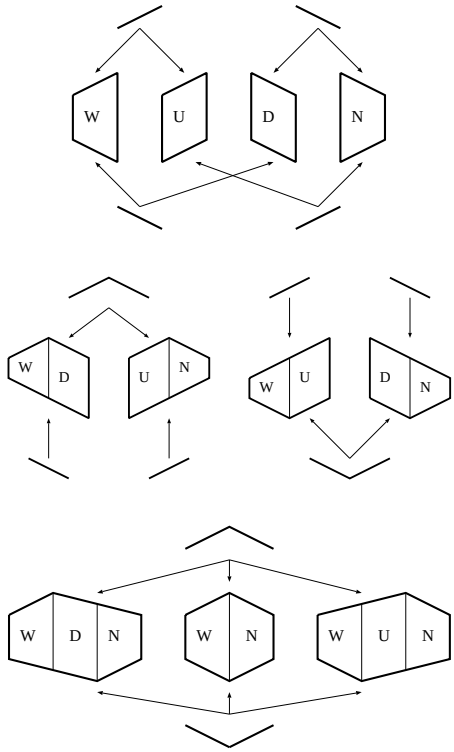


Figure 4: Rectilinear Regions

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 $v_{i,j} - \tau u_{i,j}$. 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 $1 \leq s \leq t \leq N$. 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:

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

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:

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

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:

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

Finally, let $f_N(m,[s,t])$ denote the gain of the rectilinear region that maximizes the gain among all rectilinear regions such that their types are N, UN, DN, WDN, WN, or WUN, and their last columns are the m-th one and whose intersections range 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:

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

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 on 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 when the optimal x-monotone regions appeared to be weird.

Consider the sample grid G shown in Figure 5. Suppose that the support of each pixel is 1, which means the total number of tuples in the grid is N^2, and 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 (or on) the $N/2$-th row in Figure 5. The row index is counted from the bottom. We fix a constant $K(\le N/2)$, denote the region below or on the $(N/2-K)$-th row is denoted by R', and let $R'' = R - R'$. The grid in Figure 5 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 row to higher in the presence of small noise. Figure 5 shows an instance when $N = 20$ and $K = 5$.

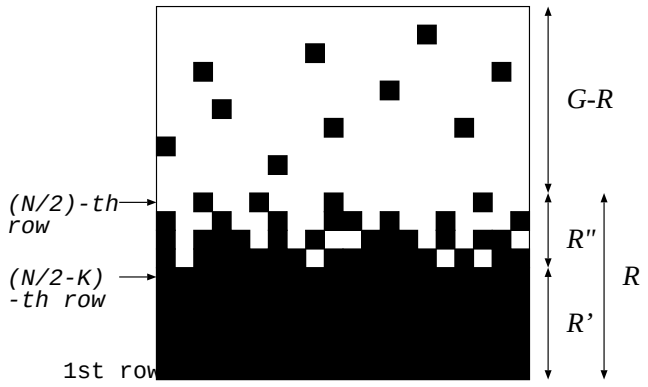


Figure 5: Sample Grid

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 5 as an instance. Let us use $0.5N^2$ as the minimum support threshold. For each grid in the universe, let us consider 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

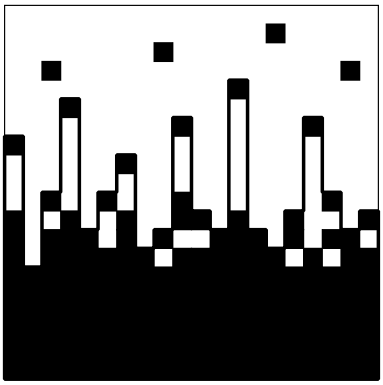


Figure 6: Optimized-confidence X-monotone Region

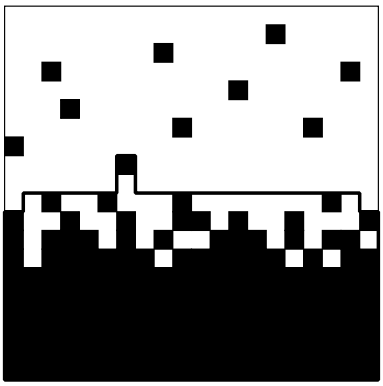


Figure 7: Optimized-confidence Rectilinear Region

in the support of the set difference $R_{monotone} - R$ and $R_{recti} - R$. We can prove the following result.

Theorem 4.1 *The support of $R_{monotone} - R$ is approximately $(\sqrt{2K} - \sqrt{K})N$, 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 Appendix B. ∎

Note that the real factor might be much larger than the above theoretical factor. For instance, Figures 6 and 7 respectively show the optimized-confidence x-monotone region $R_{monotone}$ and the optimized-confidence rectilinear region R_{recti} for the grid in Figure 5. 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.

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

However, it is difficult to give 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 it tends to fail to give a good prediction on 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 $p(x, y)$:

- $p_{linear}(x, y) = \frac{1}{\sqrt{2\pi}} \exp(\frac{-(x-y)^2}{4})$, which means the normal distribution with respect to the distance between (x, y) and the diagonal $y = x$.

- $p_{circular}(x, y) = \frac{1}{2\pi} \exp(\frac{x^2+y^2}{2})$, which is the normal distribution in two dimensional plane.

We generate two datasets both of which consist of 10,000 records. We create one dataset, named D_{linear}, by using $p_{linear}(x, y)$, and the other, named $D_{circular}$, by $p_{circular}(x, y)$.

To compare x-monotone regions and rectilinear regions with respect to overfitting, we perform the following N-fold cross validation;

1. We randomly divide a given dataset, D_{linear} or $D_{circular}$, into N equal-sized subsets.

2. We take the union of $N - 1$ subset and use the union as the training dataset for generating optimized regions; that is, we create the optimized-confidence x-monotone region and rectilinear region for the minimum support threshold, say 50%, from the training data.

No. of Pixels	Training	Test	Test − Training
8 × 8	36.56%	35.22%	-1.34%
16 × 16	37.57%	34.91%	-2.66%
32 × 32	38.97%	34.67%	-4.30%
64 × 64	41.20%	34.31%	-6.89%

(a) Confidence of Optimized X-monotone Regions

No. of Pixels	Training	Test	Test − Training
8 × 8	36.44%	35.24%	-1.20%
16 × 16	37.04%	35.33%	-1.71%
32 × 32	37.49%	35.51%	-1.98%
64 × 64	37.92%	35.30%	-2.62%

(b) Confidence of Optimized Rectilinear Regions

Table 1: Results for D_{linear}

No. of Pixels	Training	Test	Test − Training
8 × 8	14.37%	12.88%	-1.49%
16 × 16	15.54%	13.22%	-2.32%
32 × 32	16.71%	12.95%	-3.76%
64 × 64	18.41%	12.78%	-5.63%

(a) Confidence of Optimized X-monotone Regions

No. of Pixels	Training	Test	Test − Training
8 × 8	14.35%	13.34%	-1.01%
16 × 16	14.90%	13.60%	-1.30%
32 × 32	15.33%	13.65%	-1.68%
64 × 64	15.71%	13.89%	-1.82%

(b) Confidence of Optimized Rectilinear Regions

Table 2: Results for $D_{circular}$

3. We then use the remaining one subset as the test dataset. We computes the ratio of the number of test success tuples in each optimized region to the number of test tuples in the region, which we also call the *confidence* of region *against* the test data.

4. We repeat the above three steps N times, and then we compute the average of all the confidences.

We performed 10-fold cross validation for D_{linear} and $D_{circular}$. Table 1 shows the results for D_{linear}. Table 1(a) shows the confidence of the optimized-confidence x-monotone region for various numbers of pixels ranging from 8×8 to 64×64. The second column shows the confidence of each optimized x-monotone region created from the training dataset, the third column presents the confidence of each optimized region against the test dataset, and the fourth column gives the subtractions of the values in the third column from those in the second. The fourth column therefore tells us how much the optimized x-monotone overfits the training data but fails to predict for the test data. Observe that for the larger number of pixels, the differences in the fourth column increases. Table 1(b) illustrates the confidence of the optimized-confidence rectilinear region, but in this case the subtraction in the fourth column increases with a smaller amount for larger number of pixels. We also observe that the confidence of the optimized rectilinear region against the test dataset (in the second column of Table 1(b)) is always higher than that of the optimized x-monotone region (in the second column of Table 1(a)) for any number of pixels. Thus we conclude that the optimized rectilinear region is less overfit the training data than the optimized x-monotone region does.

Incidentally, the second column of Table 1(a) implies that the confidence of the optimized x-monotone region is rather dependent of the resolution of the grid, while the second column of Table 1(b) means that the confidence of the optimized rectilinear region is almost stable even if the number of pixels increases. We therefore conclude that the confidence of the optimized rectilinear region is less dependent of the choice of the number of pixels.

Table 2 illustrates the results for $D_{circular}$, from which we can also draw conclusions similar to what we have observed for D_{linear}. In general, in real applications, we have also observed the advantage of rectilinear regions over x-monotone regions.

Performance of 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 generate 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 AIX operating system, version 4.1.

Tables 3 (a), (b), and (c) show the execution times required to find the optimized confidence regions for the rectangular, rectilinear, and x-monotone types, respectively, with minimum supports of 30%, 50% and 70% and for numbers of pixels ranging from 8×8 to 256×256. These experimental tests confirmed that the computation time for each of optimized rectangular regions, x-monotone regions, and rectilinear regions is respectively $O(n^{1.5})$, $O(n \ln M)$, and $O(n^{1.5} \ln M)$,

No. of pixels	Minisup		
	30%	50%	70%
8×8	0.004	0.001	0.001
16×16	0.010	0.008	0.007
32×32	0.067	0.051	0.045
64×64	0.479	0.364	0.321
128×128	3.638	2.747	2.413

(a) Optimized rectangular regions

No. of pixels	Minisup		
	30%	50%	70%
8×8	0.016	0.014	0.013
16×16	0.080	0.089	0.086
32×32	0.417	0.378	0.363
64×64	2.029	1.830	1.748
128×128	7.653	7.983	7.181

(b) Optimized x-monotone regions

No. of pixels	Minisup		
	30%	50%	70%
8×8	0.017	0.012	0.010
16×16	0.087	0.087	0.088
32×32	0.735	0.746	0.834
64×64	6.413	6.502	7.170
128×128	57.001	63.018	63.223

(c) Optimized rectilinear regions

Table 3: Execution time of computing optimized rectilinear regions

where n is the number of pixels and M is the number of tuples.

References

[1] R. Agrawal, T. Imielinski, and A. 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.

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

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

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

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

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

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

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

[9] R. M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[10] 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.

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

[12] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

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

[14] J. S. Park, M.-S. Chen, and P. 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.

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

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

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

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

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

Appendix

A Computing optimized rectilinear regions exactly

Theorem 2.1 *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 rectilinear region or the optimized-support rectilinear region can be computed in $O(n^{1.5}M)$ time.*
2. *No algorithm can compute the optimized-confidence (or -support) rectilinear region in polynomial time with respect to n and $\log M$ unless $P = NP$.*

Proof: We begin by proving the first property. Recall the four types of rectilinear region, which are named W, U, D, and N in Section 3. Let us concentrate on W-type rectilinear regions. Consider the set of W-type rectilinear regions each of which satisfies the conditions that its support is k, all of its pixels are to the left of the m-th column, and its intersection with the m-th column consists of pixels ranging from $G(s,m)$ to $G(t,m)$. If the set is non-empty, let $f_W(k,m,[s,t])$ denote the maximum hit of all regions in the set. Otherwise, define $f_W(k,m,[s,t]) = -\infty$. In a similar way, we define $f_U(k,m,[s,t])$ for U or WU regions, $f_D(k,m,[s,t])$ for D or WD regions, and $f_N(k,m,[s,t])$ for N, UN, DN, WDN, WN, or WUN regions. Once we have computed the table of $f_W(k,m,[s,t])$, $f_U(k,m,[s,t])$, $f_D(k,m,[s,t])$, and $f_N(k,m,[s,t])$ for all $k = 1,\ldots,M$ and $m,s,t = 1,\ldots,N$, by scanning the table we can obtain the optimized-confidence (or -support) rectilinear region in $O(n^{1.5}M)$ time.

We will consider the case of computing $f_W(k,m,[s,t])$. The other cases can be solved in a similar way. First we consider the basic step when $m = 1$. For all pairs (s,t) of $1 \le s < t \le N$ and for each $k = 1,\ldots,M$, $f_W(k,1,[s,t])$ can be obtained as follows:

$$\begin{array}{rcll} f_W(k,1,[s,t]) & := & \sum_{i=s}^{t} v_{1,i} & \text{if } k = \sum_{i=s}^{t} u_{1,i} \\ & := & -\infty & \text{otherwise} \end{array}$$

Next, we show the inductive step when $m > 1$. Suppose that $s = t$.

$$f_W(k,m,[s,s]) = \begin{cases} -\infty & \text{if } k < u_{m,s}. \\ v_{m,s} & \text{if } k = u_{m,s}. \\ f_W(k-u_{m,s}, m-1, [s,s]) + v_{m,s} & \text{if } k > u_{m,s}. \end{cases}$$

Next, suppose that $s < t$. If $k < \sum_{i=s}^{t} u_{m,i}$, define

$$f_W(k,m,[s,t]) = -\infty.$$

If $k = \sum_{i=s}^{t} u_{m,i}$, set $\sum_{i=s}^{t} v_{m,i}$ to $f_W(k,m,[s,t])$. Otherwise, compute the following and set the maximum to $f_W(k,m,[s,t])$.

$$\max \left\{ \begin{array}{c} f_W(k - \sum_{i=s}^{t} u_{m,i}, m-1, [s,t]) + \sum_{i=s}^{t} v_{m,i}, \\ f_W(k - u_{m,s}, m, [s+1,t]) + v_{m,s}, \\ f_W(k - u_{m,t}, m, [s,t-1]) + v_{m,t} \end{array} \right\}$$

Thus, a simple dynamic programming gives an $O(N^3 M)$ solution for computing $f_W(k,m,[s,t])$.

Next, we will prove the second property. Consider the pixels $G(i,j)$ that meet the following properties:

- If $i \le j$, $u_{i,j} = v_{i,j} > 0$.
- If $i = j+1$, $u_{i,j} > 0$ and $v_{i,j} = 0$.
- Otherwise, $u_{i,j} = v_{i,j} = 0$.

Suppose that K is the minimum support threshold such that $\sum_{i \le j} u_{i,j} < K \le M$. Observe that the optimized-confidence rectilinear region for the minimum support K must contain all of $G(i,j)$ such that $i \le j$ and some of $G(j+1,j)$. To compute the optimized region, we need to find a subset S of $\{1,\ldots,N-1\}$ that minimizes $\sum_{j \in S} u_{j+1,j}$ under the condition that $\sum_{j \in S} u_{j+1,j} \ge K - \sum_{i \le j} u_{i,j}$. If the optimized-confidence rectilinear region can be computed in a time polynomial to n and $\log M$, in the same time complexity we can also determine whether or not there exists S such that $\sum_{j \in S} u_{j+1,j} = K - \sum_{i \le j} u_{i,j}$, which is equivalent to the NP-complete subset sum problem [9]. Consequently, unless $P = NP$, no algorithm exists for computing the optimized-confidence rectilinear 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 rectilinear region. Suppose that θ is the minimum confidence threshold such that

$$\frac{\sum_{i \le j} u_{i,j}}{\sum_{i \le j} u_{i,j} + \sum_{j}^{N-1} u_{j+1,j}} < \theta \le 1.$$

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

$$\frac{\sum_{i \le j} u_{i,j}}{\sum_{i \le j} u_{i,j} + \sum_{j \in S} u_{j+1,j}} \ge \theta,$$

which is equivalent to

$$(\frac{1}{\theta} - 1) \sum_{i \le j} u_{i,j} \ge \sum_{j \in S} u_{j+1,j}.$$

If the optimized-support rectilinear region can be computed in a time polynomial to n and $\log M$, in the

same time complexity we can determine whether or not there exists S such that

$$(\frac{1}{\theta} - 1) \sum_{i \le j} u_{i,j} = \sum_{j \in S} u_{j+1,j},$$

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

B Mathematical Analysis on a Model

Theorem 4.1 *The support of $R_{monotone} - R$ is approximately $(\sqrt{2K} - \sqrt{K})N$, 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: We first show that the support of the set difference $R_{monotone} - R$ is approximately $(\sqrt{2K} - \sqrt{K})N$.

For the i-th column, let $d(i)$ be the maximum of the index such that its subinterval $[(N/2, i), ...(N/2 - d(i), i)]$ has confidence 0. Then, the expected size of $d(i)$ is $\sqrt{2K} + o(\sqrt{K})$. Removing the above subinterval from each row, from R, we have an x-monotone region Q. Support of $R - Q$ is expectedly $N\sqrt{2K}$.

Thus, $R_{monotone}$ is the union of Q and some noise (positive data) of $G - R$, together with the bridging column interval between the noise and Q.

Since the height of the noise data in $G - R$ in the upper half grid is randomly distributed, approximately $L = \sqrt{2N\sqrt{2K}}$ noise data are included in $R_{monotone}$. If K is large, this region looks much different from R, and gives poor intuition on the rule R which we would like to extract.

In this model, we can also see that using a dense grid is often unsuitable to extract an x-monotone region rule. Suppose that we unite $2i-1$-th column and $2i$-th column into one (and analogously, unite two rows into one), to make an coarser $N/2 \times N/2$ grid, and that the same sample data output the rule region R_{double}. Then, support of the set difference $R_{double} - R$ is decreased to $\sqrt{K}N$ which is approximately $1/\sqrt{2}$ of $R_{monotone} - R$, and the support of the set difference $R_{double} - R_{monotone}$ is approximately $(\sqrt{2K} - \sqrt{K})N$, which we consider too large.

In order to eliminate the ugly side-effect of the noise, we should use a grid as coarse as $M^{1/3} \times M^{1/3}$ grid for a sample data of size M if K is large (say, $\Omega(N)$). Moreover, if the data distribution is skew, it may be necessary to use a coarser grid.

Next we show that the support of $R_{recti} - R$ is bounded by $2K \log^2 K + O(K \log N)$.

$Q_{recti} = R_{recti} \cup R$ must be rectilinear convex, and R_{recti} must be the union of Q_{recti} and a rectangle consisting of some full-columns.

The intersection of the Q_{recti} and the i-th row of R (i.e., $(N/2-i+1)$-th row of the grid) is an interval, which we denote $I(i)$. The complement of $I(i)$ in the row consists of two intervals, each of which has a terminal pixel of the row.

Since the confidence of the i-th row of R is i/K for $i \le K$, the expected number of the positive data in the leftmost t pixels in the row is it/K. Also, the expected number of the largest t' such that the leftmost t' pixels contains at most h positive data is $(h+1)K/i$.

Hence, if $R - Q_{recti}$ contains no positive data, the expected number of pixels in $R - Q_{recti}$ is $2\sum_{i=1}^{K} K/i \le 2K \log K + 2K$. Also, if $R - Q_{recti}$ contains $s \ge 1$ positive data, the expected number of pixels in $R - Q_{recti}$ is less than $2K \log K + (s+2)K$ for $s \le N/K$. If $s > i(i+1)N/2K = N/K + 2N/K + .. + iN/K$ for an i, the expected number is less than $E_Q(s) = 2K \log K + 2K + iN + (s - i(i+1)N/2K)(i+1)/K$. The probability that the number of pixels becomes $(1+c)E_Q(s)$ is at most $e^{-(c \log n)^2/2}$ (this can be proved by using Azuma's inequality [12]).

$R_{recti} \cap G - R$ must also be a rectilinear convex region, indeed, a region below a rectilinear convex chain. Let us consider rectilinear convex chains whose peak is located on a fixed (say, the j-th) column. Let us consider the subregion $P(A)$ of $G - R$ consisting the pixels below the curve $(y - N/2) \le A/|x - j|$, embedding G into the $x - y$ region.

Then, every rectilinear convex region whose peak is at the j-th column is either a subregion of $P(A)$, or contains more than A pixels. The number of pixels in $P(A)$ is at most $2A \log A$. Since the confidence of $G - R$ is $1/N$, the expected number of positive data in $P(A)$ is at most $\mu(A) = 2AN^{-1} \ln A$ (ln is the natural logarithm) .

Moreover, from the Chernoff's bound, the probability that the number of positive data in $P(A)$ exceeds $(1+\gamma)\mu(A)$ for $\gamma > 2e - 1$ ($e = 2.71..$) is at most $2^{-(1+\gamma)\mu(A)/2}$. Since the number of choice of j and A are at most N and N^2, respectively, it happens with very low probability (at most N^{-1}) that a rectilinear convex region with any area A contains more than $4\mu(A)$ positive data if $\mu(A) \ge 2 \log N$.

If $\mu(A) < 2 \log N$, $E_Q(8 \log N) < 2K \log K + 8K \log N$. Hence, the area $R - Q_{recti}$ is $O(K \log N)$ with high probability. We now consider the case $\mu(A) \ge 2 \log N$.

Since $\mu(A) = 2AN^{-1} \log A$, $A = \mu(A)N/2 \log A$. $A = support(R_{recti} - R) \le support(R - Q_{recti})$ must hold if R_{recti} is the maximum confidence region with support threshold 0.5. However, it is not difficult to show that $E_Q(4\mu(A)) \le A$ holds with high probability if $A \ge 2K \log^2 A$.

Thus, the support of $R_{recti} - R$ is bounded by $2K \log^2 K + O(K \log N)$. This is better than the x-monotone case by a factor of $N/(\sqrt{2K} \log^2 K)$. ∎