

Extracting Optimal Association Rules over Numeric Attributes

Alan P. Sprague
Department of Computer and Information Sciences
University of Alabama at Birmingham
Birmingham, Alabama 35294-1170

October 29, 1999

Abstract

Data Mining is the process of finding novel, useful and understandable patterns in massive data. Association rules are a commonly used method to discover these patterns. Until recently, the known techniques for generating association rules required that the data fields (called attributes) be binary. In a series of papers, Fukuda, Yoda, and collaborators have developed an approach to obtain association rules on numeric attributes. They have found optimal association rules for 2-dimensional numeric antecedents, where the shape of the region obtained is a convex region; their algorithm has time complexity $O(n^{1.5})$ on a grid of n pixels. We obtain efficient algorithms to find optimal association rules for 2-dimensional numeric antecedents, where the shape of the region obtained is what we call an anchored convex region or an anchored triangular region. These algorithms have time complexity $O(n)$. However, unless $P = NP$, no polynomial time algorithm finds an anchored convex region or anchored triangular region which is optimal with regard to support or confidence. These two classes of region can find application in situations where it is known from the outset that the data of greatest interest lies close to one edge of the grid, and where the improvement in execution speed from $O(n^{1.5})$ to $O(n)$ is critical.

1 Introduction

Data Mining is the core task in the new research field called Knowledge Discovery in Databases. Data Mining has been defined as the process of finding novel, potentially useful and ultimately understandable patterns in massive data [1]. Association rules are a commonly used method to discover these patterns. In this paper, we assume that we are given a “flat file” — a rectangular array of data. Each row in the array of data is called a *record*, and each column is called an *attribute*.

The *support* of a set S of records (written $\text{support}(S)$) is commonly defined as the number of records in S , divided by the number of records in the entire database.

However, here it will be simpler to define support as the number of records in S .

An association rule is an implication $C_1 \Rightarrow C_2$, where C_1 and C_2 are conditions on records. The *support* of this association rule is defined as the number of records satisfying both C_1 and C_2 , written $\text{support}(C_1 \wedge C_2)$. The *confidence* of the association rule is the proportion of records satisfying C_1 which also satisfy C_2 , so the confidence equals $\text{support}(C_1 \wedge C_2)/\text{support}(C_1)$.

A common objective is to find association rules of high confidence and high support. However, there is a tradeoff between maximizing these two quantities - maximizing one commonly results in a poor value for the other. Let α and β be nonnegative real numbers, with α no greater than the number of records in the database, and $0 \leq \beta \leq 1$. An association rule is called *ample* if its support is at least α , and is called *confident* if its confidence is at least β [3, 2].

Most of the current methods for finding association rules are restricted to binary attributes. The problem of extracting association rules for numeric attributes is the topic of several recent papers [3, 2, 5, 6, 7].

The setting for the papers by Fukuda et al. [2], Yoda et al. [7], and this paper is as follows. Among the attributes are (at least) two real valued attributes; call them x and y . It is likely that the database is huge, so to save space and processing time data values have been discretized, likely into equal sized buckets. In particular, x values have been discretized into n_x buckets, and y values into n_y buckets. This partitions the plane of x values and y values into $n_x n_y$ *pixels*.

A condition C_2 (involving principally attributes other than x and y) is given. We are looking for a condition C_1 on x and y , so that the association rule $C_1 \Rightarrow C_2$ has a large support or confidence or gain. Condition C_1 on x and y may be interpreted geometrically as a region in the x, y plane. (We define: a *region* is a set of pixels.) It is appropriate to restrict our attention to regions which are regular enough to be understandable. The types of region considered here and in [2] will be described below.

For each record t , having values x_t and y_t in the two selected real valued (binned) attributes, t is mapped into pixel (x_t, y_t) . The number of records having values x_t, y_t in these two attributes is denoted by $u(x_t, y_t)$. The number of records which satisfy condition C_2 and have values x_t, y_t in these two attributes is denoted by $v(x_t, y_t)$. Then the confidence of the rule $(x = x_t) \wedge (y = y_t) \Rightarrow C_2$ equals $v(x_t, y_t)/u(x_t, y_t)$.

Fukuda et al. [2] dealt with two types of region in the $n_x \times n_y$ array: a rectangular region, and what they call an “ x -admissible region”. A region is called *x -admissible* if it is connected and every vertical line intersects it in an interval (perhaps empty). Figure 1(a) displays an x -admissible region. Rectangles are very smooth, very constrained regions. By contrast, x -admissible regions are quite unconstrained and can be quite irregular in shape.

Both rectangles and x -admissible regions have weaknesses in Data Mining applications. Rectangles seem to be too constrained a class of regions to express well relationships in data. For example, in a banking application, where bank customers are

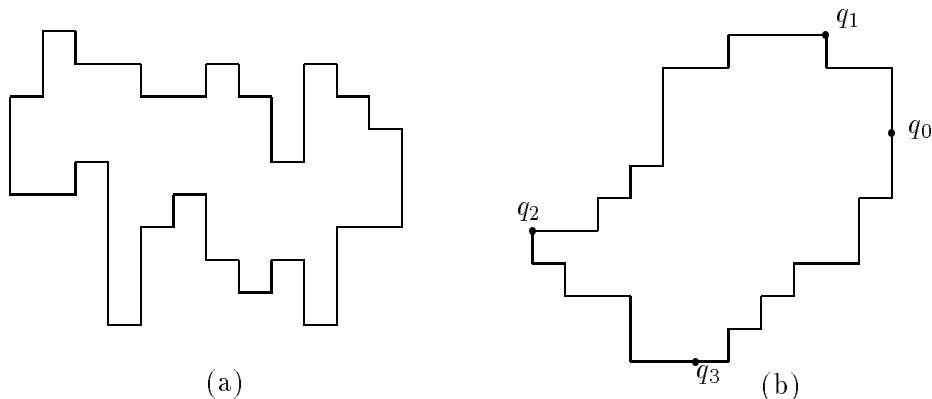


Figure 1: (a) x -admissible region. (b) convex region.

grouped according to age and bank account balance, the region of bank customers most likely to seek a bank loan is not likely to be well modeled as a rectangle. The x -admissible regions seem perhaps to be not constrained enough — they can give rise to strangely shaped regions, and accordingly might not aid in the understanding of the data. Furthermore, as demonstrated in [7], x -admissible regions can overfit data. In Yoda et al. [7], the approach of [2] is extended to a well-positioned intermediate class of regions, called *rectilinear regions* in [7] and called convex regions here. Convex regions hold promise of yielding shapes which simultaneously model the data reasonably well and are understandable, without overfitting to the data.

A set of pixels is *convex* if it is both x -admissible and y -admissible. An equivalent definition of convex region follows. Let q_0 be an arbitrary point on the boundary of the region, whose x coordinate is maximum. Similarly, let q_1 (q_2 , q_3 respectively) be an arbitrary point on the boundary of the region, whose y coordinate is maximum (x coordinate is minimum, y coordinate is minimum). See Figure 1(b). A region is *convex* if a traversal of the boundary from q_0 to q_1 travels only north and west (i.e., toward increasing y coordinate and decreasing x coordinate), and similarly a traversal of the boundary from q_1 to q_2 (q_2 to q_3 , q_3 to q_0 respectively) travels only west and south (south and east, east and north respectively).

Two special classes of convex regions are the following. An *anchored convex region* is a convex region R such that for each pixel (x, y) in R , $(x, y - 1)$ is also in R (unless $y = 1$). Equivalently, a *anchored convex region* is a convex region such that, when points q_0, q_1, q_2, q_3 are defined as in the previous paragraph, q_0, q_2 , and q_3 may all be chosen to be on the x -axis.

An *anchored triangular region* R is a set of pixels such that for any pixel (x, y) in R , $(x - 1, y)$ is also in R (unless $x = 1$), and $(x, y - 1)$ is also in R (unless $y = 1$).

Equivalently, a *anchored triangular region* is a convex region such that, when points q_0, q_1, q_2, q_3 are defined as above, q_0 may be chosen to be on the x axis, q_1 to be on the y axis, and q_2 and q_3 to both be $(0, 0)$. (In this paper we take pixel (i, j) to be the unit square whose lower left corner point has coordinate $(i - 1, j - 1)$ and upper right corner point has coordinate (i, j) . Since the lower left pixel in the grid is pixel $(1, 1)$, the lower left corner point of the grid is $(0, 0)$.)

The *gain* of region R is defined as $v(R) - \theta u(R)$, where θ is some fixed positive quantity. In [2], it is explained how maximization of gain can model maximization of profit in an application. We note that gain is an additive function: a function f over the pixels of a grid is *additive* if for any region R , $f(R) = \sum_{p \in R} f(p)$. The algorithms for gain in [2, 7], and also the algorithms here, are valid for any additive function.

The anchored classes of region introduced here may have application in situations where it is known from the outset that the data of greatest interest lies close to one edge of the grid, and where the improvement in execution speed from $O(n^{1.5})$ to $O(n)$ is critical.

We report three results. In these, n is the number of pixels ($n = n_x n_y$).

- (1) an $O(n)$ algorithm to compute an anchored triangular region which maximizes gain;
- (2) an $O(n)$ algorithm to compute an anchored convex region which maximizes gain;
- (3) the NP-completeness of the decision problem corresponding to the problem of finding an ample anchored triangular region which maximizes confidence, (or a confident anchored triangular region which maximizes support.) The corresponding problem for anchored convex regions is also NP-complete.

Results (1), (2) and (3) are developed in Sections 2, 3, and 4 respectively.

2 Maximizing Gain in an Anchored Triangular Region

In this section we develop an $O(n)$ time algorithm to construct an anchored triangular region of maximum gain in a rectangular array of n pixels.

We are given an array of $n_x \times n_y$ pixels, and $n = n_x n_y$. For each x and y such that $1 \leq x \leq n_x$ and $1 \leq y \leq n_y$, we let $\text{col}(x, y)$ be the set of pixels $\{(x, \hat{y}) : \hat{y} \leq y\}$.

Let $1 \leq x \leq n_x$, $1 \leq y \leq n_y$. Define $\mathcal{R}(x, y)$ to be the set of anchored triangular regions which contain pixel (x, y) , contain no pixel to the right of column x , and contain no pixel above (x, y) in column x . For $x < n_x$ and $y < n_y$ this is equivalent to defining $\mathcal{R}(x, y)$ to be the set of anchored triangular regions which contain (x, y) but not $(x, y + 1)$ or $(x + 1, 1)$.

Define $T(x, y)$ to be the maximum gain of these regions: $T(x, y) = \max\{\text{gain}(R) : R \in \mathcal{R}(x, y)\}$.

Proposition 2.1 For $x > 1$, $T(x, y) = \text{gain}(\text{col}(x, y)) + \max(T(x - 1, \hat{y}) : \hat{y} \geq y)$

Proof. Let $R \in \mathcal{R}(x, y)$. The pixels in R may be partitioned into $\text{col}(x, y)$ and an anchored triangular region in $\mathcal{R}(x - 1, \hat{y})$ for some $\hat{y} \geq y$. Conversely, for each $R \in \mathcal{R}(x - 1, \hat{y})$ for some $\hat{y} \geq y$, $R \cup \text{col}(x, y)$ is in $\mathcal{R}(x, y)$.

Consequently $\max(\text{gain}(R) : R \in \mathcal{R}(x, y)) = \text{gain}(\text{col}(x, y)) + \max(\text{gain}(R) : R \in \mathcal{R}(x - 1, \hat{y}) \text{ for some } \hat{y} \geq y)$. The conclusion follows. \square

The following algorithm computes the gain of an anchored triangular region having maximum gain. Statements 2 and 7 in it will be explained in detail below; here we examine correctness of the algorithm. It is clear that for $x = 1$, $T(x, y)$ should be computed as $\text{gain}(\text{col}(x, y))$, and the algorithm computes $T(1, y)$ correctly. It is also clear that for $x > 1$, Steps 7 and 9 compute $T(x, y)$ in accordance with Proposition 2.1. Hence it is clear that the algorithm computes the maximum gain achievable by an anchored triangular region.

Algorithm to compute maximum gain.

1. For $x = 1$ to n_x :
2. Compute $\text{gain}(\text{col}(x, y))$ for all y .
3. For $y = 1$ to n_y :
4. $T(0, y) = 0$.
5. $\text{maxgain} = 0$.
6. For $x = 1$ to n_x :
7. Compute, for all y , $f(x - 1, y) = \max(T(x - 1, \hat{y}) : \hat{y} \geq y)$
8. For $y = 1$ to n_y :
9. $T(x, y) = \text{gain}(\text{col}(x, y)) + f(x - 1, y)$.
10. $\text{maxgain} = \max(\text{maxgain}, T(x, y))$.

It remains to explain Statements 2 and 7. Let a sequence $h(z)$ be given (for $1 \leq z \leq k$). A function g is the *suffix maxima* of h if $g(z) = \max(h(w) : w \geq z)$. Similarly, g is the *prefix sums* of h if $g(z) = \sum(h(w) : w \leq z)$. These functions are fundamental in the theory of parallel computation; they are also important in developing sequential algorithms, such as here. These functions may be computed in linear time: given $h(z)$ for $1 \leq z \leq k$, if g is defined as the suffix maxima of h then g may be computed by the formulas $g(k) = h(k)$, and $g(z) = \max(h(z), g(z + 1))$ for $z < k$. Similarly, if g is defined as the prefix sums of h then g may be computed by the formulas $g(1) = h(1)$, and $g(z) = h(z) + g(z - 1)$ for $z \leq k$. We emphasize that in both cases g is computable in $O(k)$ time, that is, in linear time.

Proposition 2.2 Let a 2-dimensional array of n pixels be given, and functions u and v be defined on it. An anchored triangular region of maximum gain may be computed in $O(n)$ time.

Proof. We have already noted that the algorithm above computes the maximum gain achievable by an anchored triangular region. As is commonly the case in dynamic

algorithms, it is easy to retain information indicating which region gives rise to the maximum gain, without increasing the computational complexity of the algorithm.

To demonstrate that this algorithm takes linear time, it is sufficient to show that Steps 2 and 7 are computable in $O(n_y)$ time for each fixed x . But for each value of x , Step 2 is computable by a prefix sums calculation, so takes $O(n_y)$ time. Likewise, Step 7 is computable by a suffix maxima calculation, so takes $O(n_y)$ time. \square

3 Maximizing Gain in an Anchored Convex Region

In this section we sketch an $O(n)$ time algorithm to construct an anchored convex region of maximum gain in a rectangular array of n pixels. Numeric examples are presented at the end of the section.

As in Section 2, we are given an array of $n_x \times n_y$ pixels, and $n = n_x n_y$. As in Section 2, for each x and y such that $1 \leq x \leq n_x$ and $1 \leq y \leq n_y$, we let $\text{col}(x, y)$ be the set of pixels $\{(x, \hat{y}) : \hat{y} \leq y\}$.

Let R be a set of pixels. We introduce notation for the maximum y coordinate of a pixel in R :

$$\max_y(R) = \max\{\hat{y} : \text{pixel } (\hat{x}, \hat{y}) \in R\}.$$

Let $1 \leq \hat{x} \leq n_x, 1 \leq b \leq n_y$. Define $\mathcal{R}(\hat{x}, b)$ to be the set of anchored convex regions R such that the set of pixels in R having x coordinate \hat{x} is precisely $\text{col}(\hat{x}, b)$, and which contain no pixel in column $\hat{x} + 1$. $\mathcal{R}(\hat{x}, b)$ is partitioned into two subclasses. $\mathcal{R}_g(\hat{x}, b)$ indicates members of $\mathcal{R}(\hat{x}, b)$ which are “growing” in column \hat{x} : column \hat{x} meets the northwest border of the region. More precisely,

$$\mathcal{R}_g(\hat{x}, b) = \{R \in \mathcal{R}(\hat{x}, b) : \max_y(R) = b\}$$

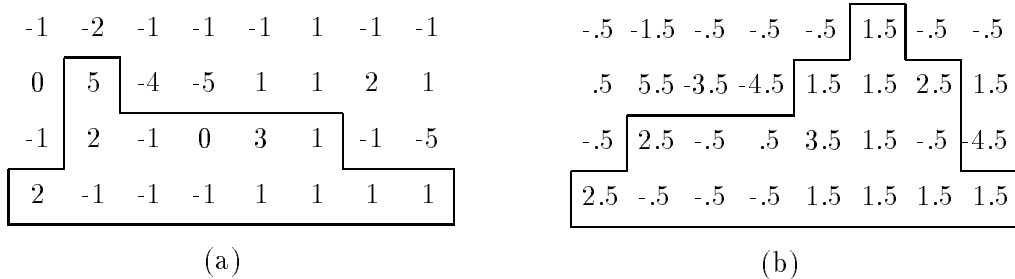
$\mathcal{R}_s(\hat{x}, b)$ indicates members of $\mathcal{R}(\hat{x}, b)$ which are “shrinking” in column \hat{x} , i.e. column \hat{x} meets the northeast border of the region:

$$\mathcal{R}_s(\hat{x}, b) = \{R \in \mathcal{R}(\hat{x}, b) : \max_y(R) > b\}$$

Define $T(x, b) = \max\{\text{gain}(R) : R \in \mathcal{R}(x, b)\}$. The objective is to compute the maximum gain of a nonempty anchored convex region, that is, $\max\{T(x, b) : 1 \leq x \leq n_x, 1 \leq b \leq n_y\}$. For each subscripted class of regions $\mathcal{R}_t(x, b)$ we define $T_t(x, b) = \max\{\text{gain}(R) : R \in \mathcal{R}_t(x, b)\}$.

Proposition 3.1 *For $x > 1$, the following hold.*

- (1) $T_g(x, b) = \text{gain}(\text{col}(x, b)) + \max(0, \max(T_g(x - 1, b') : b' \leq b))$.
- (2) $T_s(x, b) = \text{gain}(\text{col}(x, b)) + \max(\max(T_s(x - 1, b') : b' \geq b), \max(T_g(x - 1, b') : b' > b))$.



.

Figure 2: Two grids. In each, the anchored convex region maximizing gain is indicated.

Proof of this is similar to the proof of Proposition 2.1, and is omitted.

This proposition gives rise to an $O(n)$ algorithm for the anchored convex region of maximum gain, like Proposition 2.1 gave rise to an $O(n)$ algorithm for the anchored triangular region of maximum gain.

Two numeric examples of grids, with the gain of each pixel specified, are displayed in Figure 2. In each, the anchored convex region maximizing gain is shown. We note that the gain of each pixel in Figure 2(b) was obtained by adding .5 to the gain of the corresponding pixel in Figure 2(a).

4 NP-completeness of Maximizing Support or Confidence

In this section we show that (when posed as a decision problem) finding an anchored triangular region which is ample and maximizes confidence is an NP-complete problem. Similarly, the problem of maximizing support of confident anchored triangular regions is NP-complete. These results follow from a simple reduction from PARTITION. The same reduction shows that these problems for convex regions or for anchored convex regions are also NP-complete.

The decision problem that we show NP-complete is the following, which we call ANCHORED TRIANGULAR REGION (ATR). We are given a rectangular grid, with nonnegative integer functions u and v defined on the grid. Numbers α and β are given. Any region R is considered ample if its support is at least α (i.e., $v(R) \geq \alpha$), and any region having confidence at least β is considered confident. The question is, does there exist an anchored triangular region which is both ample and confident?

Note that this decision problem corresponds to the problem of maximizing confidence (among ample regions), and simultaneously corresponds to the problem of max-

imizing support (among confident regions).

In the PARTITION problem we are given a list a_1, a_2, \dots, a_n of positive integers. Let $A = \sum_{i=1}^n a_i$. We are to determine if there is a subset S of $\{1, 2, \dots, n\}$ such that $\sum(a_i : i \in S) = A/2$. This problem is NP-complete [4, p. 47].

Proposition 4.1 *ATR is NP-complete.*

Proof. We reduce PARTITION to this problem. Let a_1, a_2, \dots, a_n be a problem instance of PARTITION. Let $A = \sum_{i=1}^n a_i$.

To construct a problem instance of ATR we divide an $n \times n$ grid into a ‘positive interior’, ‘mixed border’, and ‘exterior’. Pixel coordinates run from 1 to n . The *positive interior* is all pixels (x, y) such that $x + y \leq n$. The *mixed border* is all pixels (x, y) such that $x + y = n + 1$. The remaining pixels constitute the *exterior*. For each pixel p in the positive interior, $u(p) = v(p)$; also, u is defined so that the total support of the positive interior is A . (For example, we can set $u(p)$ to be zero for all positive interior pixels, except $u(1, 1) = A$.) For each pixel $(j, n + 1 - j)$ of the mixed border, set $u(j, n + 1 - j) = 2a_j$ and $v(j, n + 1 - j) = a_j$. For each exterior pixel p , $u(p) = v(p) = 0$.

Choose $\alpha = 1.5A$ and $\beta = .75$. ATR asks if there is an anchored triangular region R of support at least $1.5A$ and confidence at least $.75$. We may restrict consideration to regions R containing every interior pixel, and containing no exterior pixel. Suppose that anchored triangular region R has support at least $1.5A$ and confidence at least $.75$. Let R' be the set of pixels of R which are in the mixed border. We note that $u(R') = 2v(R')$ since for each pixel p in the mixed border $u(p) = 2v(p)$. Denote $v(R')$ by σ , so $u(R') = 2\sigma$.

Since $\text{support}(R) = \sum(v(p) : p \in R) = A + \sum(v(p) : p \in R') = A + \sigma$, then $\sigma \geq A/2$. Since $\text{confidence}(R) = \sum(v(p) : p \in R) / \sum(u(p) : p \in R) = (A + \sigma) / (A + 2\sigma)$, then $\sigma \leq A/2$. Then $v(R') = \sigma = A/2$.

Summarizing, if there is an anchored triangular region which is both ample and confident then there is a set R' of mixed border pixels such that $\text{support}(R') = A/2$. It is easy to show the converse: if there is a set R' of mixed border pixels having support $A/2$ then there is an anchored triangular region which is both ample and confident. The existence of a set R' of mixed border pixels having support $A/2$ is clearly equivalent to the existence of an affirmative solution to the stated problem instance of PARTITION. \square .

It is not difficult to see that the same reduction shows that the decision problem of determining the existence of a convex region (or anchored convex region) which is both ample and confident is also NP-complete.

References

- [1] U.M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge

- discovery: an overview. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 1–34. AAAI Press, 1996.
- [2] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Data mining using two-dimensional optimized association rules: scheme, algorithms, and visualization. In *Proc. ACM SIGMOD Conf. on Management of Data*, pages 13–23. Assoc. for Comput. Mach., 1996.
 - [3] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Mining optimized association rules for numeric attributes. In *Proc. Fifteenth ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Syst.*, pages 182–191. Assoc. for Comput. Mach., 1996.
 - [4] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.
 - [5] R. Miller and Y. Yang. Association rules over interval data. In *Proc. ACM SIGMOD*, 1997.
 - [6] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proc. ACM SIGMOD Conf. Mgmt. Data*, 1996.
 - [7] K. Yoda, T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Computing optimized rectilinear regions for association rules. In *Proc. 3rd Intl. Conf. Knowl. Discovery Data Mining*, pages 96–103. AAAI Press., 1997.