Algorithms for finding attribute value group for binary segmentation of categorical databases

doi: 10.1109/TKDE.2002.1047767

Algorithms for Finding Attribute Value Group for Binary Segmentation of Categorical Databases

Yasuhiko Morimoto, *Member*, *IEEE Computer Society*,
Takeshi Fukuda, *Member*, *IEEE Computer Society*, and Takeshi Tokuyama

Abstract—We consider the problem of finding a set of attribute values that give a high quality binary segmentation of a database. The quality of a segmentation is defined by an objective function suitable for the user's objective, such as "mean squared error," "mutual information," or "χ^2," each of which is defined in terms of the distribution of a given target attribute. Our goal is to find value groups on a given conditional domain that split databases into two segments, optimizing the value of an objective function. Though the problem is intractable for general objective functions, there are feasible algorithms for finding high quality binary segmentations when the objective function is convex, and we prove that the typical criteria mentioned above are all convex. We propose two practical algorithms, based on computational geometry techniques, which find a much better value group than conventional heuristics.

Index Terms—Value groups, binary segmentation, categorical test, decision tree, data reduction, data mining.

1 Introduction

We consider the problem of finding a set of attribute values that give a high quality binary segmentation of a database. The quality of a segmentation is defined by an objective function suitable for the user's objective, such as "mean squared error," "mutual information," and "χ^2," each of which is defined in terms of the distribution of a given target attribute. Our goal is to find value groups on a given conditional domain that split databases into two segments, optimizing the value of an objective function.

1.1 Value Group for Binary Segmentation

Assume that there is a database relation that contains categorical and numerical attributes, and let R be a projected relation from the database. We assume that the attributes in R are all categorical, that is, unordered and discrete. We treat one attribute of the relation as special and call it a *target attribute*. Other attributes of the relation are called *conditional attributes*.

Let A be the target attribute, $\mathrm{dom}(A) = \{a_1, a_2, \cdots, a_k\}$ be the domain of A, and k be the *target domain size*, that is, the number of distinct values. Let $x_i(S)$ denote the number of records in $S \subseteq R$ for which the value of the target attribute A is a_i $(1 \leq i \leq k)$.

Let $C_1, C_2, \cdots, C_M$ be the conditional attributes. We treat these attributes as a single attribute C whose domain is the Cartesian product of their domains, that is,

$$\mathrm{dom}(C) = \mathrm{dom}(C_1) \times \mathrm{dom}(C_2) \times \cdots \times \mathrm{dom}(C_M).$$

If C_i, where $i = 1, 2, \cdots, M$ has n_i distinct values, the *conditional domain size* of C is $n = \prod_i n_i$ for $1 \leq i \leq M$.

To make a binary segmentation of R, we use a set of attribute values, which we call a *value group*, on the conditional attribute C. Let $V \subset \mathrm{dom}(C)$ be a value group that divides the database relation R into two segments $S = \{t \in R \mid t[C] \in V\}$ and $S = \{t \in R \mid t[C] \notin V\}$, where t denotes a record in R and $t[C]$ denotes a value of t for attribute C. We say "V splits R into $(S; \bar{S})$." Our ideal goal is to find, among all possible value groups, a value group V that maximizes (or minimizes) an objective function $f(S; S) = f(x_1(S), \cdots, x_k(S)) = f(\mathbf{x}(S))$.

Note that we do not have to treat all conditional categorical attributes as a single attribute. In many applications, it is better to treat each attribute separately. In such cases, there is a binary segmentation problem for each attribute.

1.2 Motivating Example

Table 1 shows an example of a relation projected from the sales log of a car rental company. In the relation, "Color" and "Size" show the characteristics of each car rented, and "Age" shows the customer's age. Assume that there are five colors ("white," "black," "red," "blue," and "silver") and three sizes ("compact," "medium," and "large") in the relation.

Assume that the company wants to classify customers according to the value of "Age" by using the characteristics of cars. We use "Age" as a target attribute and the other attributes as conditional attributes. If we treat the two conditional attributes as a single attribute, a value group is a set of values like "Color=white and Size=compact" or "Color=red and Size=compact." Finding a high quality value group on the conditional domain gives us clues to understanding what characteristics attracted young, middle-aged, or old customers.

- *Y. Morimoto and T. Fukuda are with IBM Tokyo Research Laboratory, 1623-14, Shimo-tsuruma, Yamato, Kanagawa 242-8502, Japan. E-mail: {morimoto, fukudat}@jp.ibm.com.*
- *T. Tokuyama is with GSIS Tohoku University, Aza-Aramaki, Aoba-ku, Sendai, 980-8579, Japan. E-mail: tokuyama@dais.is.tohoku.ac.jp.*

Manuscript received 12 Nov. 1998; revised 6 Feb. 2001; accepted 9 Apr. 2001. For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 108265.

TABLE 1
Car Rental Sales Log

Color	Size	**Age**
white	compact	**young**
black	compact	**young**
black	medium	**middle-aged**
red	large	**old**
blue	large	**middle-aged**
white	medium	**old**
...	...	...

In general, such value groups are used as part of decision systems such as "splitting tests" for decision trees [4], [22]. Decision trees are easy to compute, and their knowledge representation is easy to comprehend. Moreover, their prediction accuracy is as good as that of other accurate decision systems. Therefore, techniques for fast construction of decision trees from large data sets have been studied intensively in the database community [16], [25], [24], [13]. In recent years, decision trees have often been used as parts of boosting or bagging methods [7], [3], [23], which are known to be among the most accurate prediction methods. A decision tree has two important tasks, class prediction and modeling. Especially for modeling tasks, high quality value groups are significant.

In addition to decision trees, high quality value groups can be applied for some kinds of optimization problems. Moreover, if we apply data segmentations by applying high quality value groups recursively, we can make a value group hierarchy. Such a hierarchy can be used to simplify the conditional domain. We can assign a general value for a value group of each leaf node of the hierarchy. By applying data reduction or value reduction techniques [20], we can reduce the conditional values in a database without losing essential information. Therefore, high quality value groups on conditional domains provide us with essential information for the analysis of databases.

If there are five colors and three sizes, there are at most $15 = 5 \times 3$ values in the conditional domain of the relation. Therefore, we have to examine at most 2^{15-1} possible value groups in this small example to find the optimal one. In general, if the conditional domain size is n, there are $2^{n-1} - 1$ possible value groups. Hence, a naive exhaustive search for the optimal binary segmentation requires $O(2^n)$ time, which is not practical. Therefore, we need feasible heuristics that can find high quality value groups approximating the optimal one with respect to the quality of the binary segmentation.

1.3 Main Results

In this paper, we propose two algorithms, named the Random Enumeration Algorithm and the Probing Algorithm, that we designed by using computational geometry techniques. Both of the algorithms can feasibly compute a high quality value group for cases in which k is a small constant. The domain size n is allowed to be very large.

Each possible value group on a conditional domain can be interpreted as a point in a k-dimensional space, and we can translate the problem of finding the optimal value group into that of finding a point in the k-dimensional space. We proved that the optimal point must be on the convex hull of the point set of all possible value groups. Both of the algorithms compute a point on the convex hull efficiently, that is, in $O(n)$ time, without examining points inside the hull.

The Random Enumeration Algorithm examines points on the convex hull by using random sampling. It outputs the best point of the examined points, which are computed from a small sample. The time complexity is reduced to $O(s^{k-2}n)$, where $s(\ll n)$ is the sample size. We will empirically show that the Random Enumeration Algorithm finds a satisfactory value group with a small sample. This algorithm can run with a small working space and can easily be parallelized.

The Probing Algorithm uses tangential hyperplanes to compute points on the convex hull. If we maintain a list of points that are examined during the search and incrementally construct inscribed and circumscribed convex polygons inside and outside the hull, we can find clues as to the next point to be examined [8], [17], [18]. However, the total cost of the incremental convex polygon maintenance and probing of the convex hull is $O((n+m)|P|)$, if we have m points and $|P|$ facets on the convex hull. The m and P can be asymptotically as large as n^{k-1} and $m^{\lfloor \frac{k}{2} \rfloor}$, respectively, in a pathological input. Though they are known to be much smaller in a normal input, the incremental polygon maintenance is still costly when k becomes large.

The Probing Algorithm maintains only promising facets of the inscribed convex polygon, using some heuristics, and works within a limited working space. It can find a satisfactory value group in an earlier step of the algorithm. At every incremental step of the algorithm, it can report the best solution found so far and, thus, the solution gradually converges to the optimal. In cases where a quick response is required, the Probing Algorithm can return the best value group found in the required time, and it is thus suitable for online applications.

2 Value Group

2.1 Stamp Point

Let $V \subset \mathrm{dom}(C)$ be a value group that divides the database relation R into two segments $S = \{t \in R \mid t[C] \in V\}$ and $S = \{t \in R \mid t[C] \notin V\}$. The value group V can be evaluated by an objective function.

For a segment S, let $x_i(S)$ be the number of data records in S for which the value of the target attribute is a_i. Thus, each segment S of the relation R can be mapped to a point $\mathbf{x}(S) = (x_1(S), x_2(S), \cdots, x_k(S))$ in the k-dimensional Euclidean space, which is referred to as a *stamp point* of S. A stamp point represents the distribution of the target attribute of interest, and our objective functions of binary segmentations, $f(S; \bar{S})$, are defined in terms of $\mathbf{x}(S)$ or $\mathbf{x}(\bar{S})$. If a value group V splits R into $(S; \bar{S})$, we also refer to the stamp point of S as the stamp point of V (i.e., $\mathbf{x}(V) = \mathbf{x}(S)$).

2.2 Criteria for Segmentation

The significance of the discovered rules depends on the user's objective and, hence, there is no universal criterion for measuring the significance of value groups. A useful segmentation should divide data into segments whose target distribution is more skewed than that of the data as a

whole. Therefore, we often use criteria such as the "mean squared error," "mutual information," "χ^2," and so forth, which indicate the extent to which the divided data distributions are skewed and differ from the original data distribution.

In this section, we describe objective criteria and their corresponding functions commonly used for evaluating data segmentations.

2.2.1 Mean Squared Error

The implication of the "gini" criterion [4] is "how much the mean squared error of the target values is decreased by the segmentation." The optimal segmentation according to this criterion minimizes the mean squared error. The error is the sum of the number of misclassified records in each segment, that is, the number of records that are not the same as the majority value in each segment. Let $|S| = \sum_{i=1}^{k} x_i(S)$ and $p_i(S) = x_i(S)/|S|$. The gini criterion is defined as follows:

$$\begin{aligned} Gini(\mathbf{x}(S)) &= Gini(S;\bar{S}) \\ &= \left(1 - \sum_{i=1}^{k} p_i(R)^2\right) \\ &\quad - \frac{|S|}{|R|}\left(1 - \sum_{i=1}^{k} p_i(S)^2\right) - \frac{|\bar{S}|}{|R|}\left(1 - \sum_{i=1}^{k} p_i(S)^2\right). \end{aligned}$$

2.2.2 Mutual Information

The following entropy gain function compares the mutual information gained by a segmentation. It indicates "how much information is given by the segmentation."

$$\begin{aligned} Ent(\mathbf{x}(S)) &= Ent(S;\bar{S}) \\ &= -\sum_{i=1}^{k} p_i(R)\log p_i(R) \\ &\quad + \frac{|S|}{|R|}\sum_{i=1}^{k} p_i(S)\log p_i(S) + \frac{|\bar{S}|}{|R|}\sum_{i=1}^{k} p_i(S)\log p_i(S). \end{aligned}$$

2.2.3 χ^2 (Correlation)

The following χ^2 function indicates how strongly the statistical hypothesis that "S and S are not different from R" is denied.

$$\begin{aligned} Chi(\mathbf{x}(S)) &= Chi(S;\bar{S}) \\ &= \sum_{i=1}^{k} \frac{|S|(p_i(S) - p_i(R))^2 + |\bar{S}|(p_i(\bar{S}) - p_i(R))^2}{p_i(R)}. \end{aligned}$$

Example 2.1. To illustrate the quality of value groups, we consider binary segmentations of the following data. In these data, a target attribute has three values, say a_1, a_2, and a_3, and each value has 40, 30, and 30 records, respectively.

R	a_1	a_2	a_3
100	40	30	30

Let us consider a binary segmentation that divides R into S and $\bar{S}$ as follows:

S	a_1	a_2	a_3
60	40	10	10

S	a_1	a_2	a_3
40	0	20	20

The stamp point for the segmentation is $(40, 10, 10)$ in three-dimensional space. The gini index value, for example, is computed as

$$\begin{aligned} &\left(1 - \left(\frac{40}{100}\right)^2 - \left(\frac{30}{100}\right)^2 - \left(\frac{30}{100}\right)^2\right) \\ &- \frac{60}{100}\left(1 - \left(\frac{40}{60}\right)^2 - \left(\frac{10}{60}\right)^2 - \left(\frac{10}{60}\right)^2\right) \\ &- \frac{40}{100}\left(1 - \left(\frac{0}{60}\right)^2 - \left(\frac{20}{60}\right)^2 - \left(\frac{20}{60}\right)^2\right) \\ &= 0.16. \end{aligned}$$

Let us consider another segmentation:

S	a_1	a_2	a_3
60	20	20	20

$\bar{S}$	a_1	a_2	a_3
40	20	10	10

In this case, the value of the gini function of the stamp point $(20, 20, 20)$ is only 0.01, much smaller than the value of the former segmentation. Notice that mean squared error in this segmentation is larger than that of the former segmentation.

The quality of a value group for a binary segmentation with respect to the gini index can be evaluated by the value of $Gini(\mathbf{x}(S)) = Gini(S;\bar{S})$ if the value group splits data into two segments, say "S" and "$\bar{S}$." The higher the value of the objective function is, the better the quality of the value group is with respect to this criterion. Similarly, we prefer higher values of $Ent(\mathbf{x}(S)) = Ent(S;\bar{S})$ and $Chi(\mathbf{x}(S)) = Chi(S;\bar{S})$.

2.3 Convexity of the Objective Functions

If we consider objective functions that are defined in terms of a stamp point in k-dimensional Euclidean space, the following property can be used in searching for the optimal (or a high quality) point and its corresponding value group:

Definition 2.1 (Convexity). *$f(\mathbf{x})$ is convex if*

$$max\{f(\mathbf{x_1}), f(\mathbf{x_2})\} \geq f((1-\gamma)\mathbf{x_1} + \gamma\mathbf{x_2})$$

for $0 \leq \gamma \leq 1$ and arbitrary points $\mathbf{x_1}$ and $\mathbf{x_2}$ in the domain of f.

Lemma 2.1. *The gini index, entropy gain, and χ^2 are all convex functions on $\mathbf{x}$.*

Proof. For any vector $\Delta \neq 0$, the second derivative of the functions along with Δ is nonnegative.

Let $\mathbf{r} = (r_1, \cdots, r_k)$ be the set of all records to be split, and let $\|\mathbf{r}\| = \sum_{i=1}^{k} r_i$, i.e., $|R|$ and let $\|\mathbf{x}\| = \sum_{i=1}^{k} x_i$, i.e., $|S|$.

Let us first prove inequality for $Gini(\mathbf{x}(S))$. It is transformed as follows:

$$\begin{aligned} Gini(\mathbf{x}(S)) &= Constant + \frac{1}{|R|}G(\mathbf{x}(S)) \\ &\quad + \frac{1}{|R|}G(\mathbf{x}(R) - \mathbf{x}(S)), \end{aligned}$$

where

$$G(\mathbf{x}(S)) = G(\mathbf{x}) = \sum_{i=1}^{k} \frac{x_i^2}{\|\mathbf{x}\|}.$$

Let $\Delta = (\delta_1, \delta_2, \cdots, \delta_k)$ and $Y = (\Delta, \mathbf{x}) = \sum_{i=1}^{k} \delta_i x_i$. The first derivative of $G(\mathbf{x})$ is

$$\begin{aligned} G'(\mathbf{x}) &= \frac{dG(\mathbf{x})}{dY} \\ &= \sum_{i=1}^{k} \frac{1}{\delta_i} \frac{\partial G(\mathbf{x})}{\partial x_i} \\ &= \frac{1}{\|\mathbf{x}\|} \sum_{i=1}^{k} \frac{2x_i}{\delta_i} - \frac{\|\mathbf{t}\|}{\|\mathbf{x}\|^2} \sum_{i=1}^{k} x_i^2, \end{aligned}$$

where $\mathbf{t} = (\delta_1^{-1}, \cdots, \delta_k^{-1})$, and the second derivative is

$$G''(\mathbf{x}) = \frac{2}{\|\mathbf{x}\|} \sum_{i=1}^{k} \left(\frac{\|\mathbf{t}\| x_i}{\|\mathbf{x}\|} - \frac{1}{\delta_i} \right)^2 \geq 0.$$

Therefore, $Gini''(\mathbf{x}(S)) \geq 0$.

We proved the convexity of $Ent(\mathbf{x})$ in [17]. We now show it for $Chi(\mathbf{x}(S))$. The function is transformed as follows:

$$Chi(\mathbf{x}(S)) = C(\mathbf{x}(S)) + C(\mathbf{x}(R) - \mathbf{x}(S)),$$

where

$$C(\mathbf{x}(S)) = C(\mathbf{x}) = \sum_{i=1}^{k} \frac{\left(x_i - r_i \frac{\|\mathbf{x}\|}{\|\mathbf{r}\|} \right)^2}{r_i \frac{\|\mathbf{x}\|}{\|\mathbf{r}\|}}.$$

The first derivative of $C(\mathbf{x})$ is

$$\begin{aligned} C'(\mathbf{x}) &= \frac{dC(\mathbf{x})}{dY} \\ &= \sum_{i=1}^{k} \frac{1}{\delta_i} \frac{\partial C(\mathbf{x})}{\partial x_i} \\ &= \frac{\|\mathbf{r}\|}{\|\mathbf{x}\|} \sum_{i=1}^{k} \frac{2x_i}{\delta_i r_i} - \frac{\|\mathbf{t}\| \|\mathbf{r}\|}{\|\mathbf{x}\|^2} \sum_{i=1}^{k} \frac{x_i^2}{r_i} - \|\mathbf{t}\|, \end{aligned}$$

and the second derivative is

$$\begin{aligned} C''(\mathbf{x}) &= \frac{d^2 C(\mathbf{x})}{dY^2} \\ &= \frac{2\|\mathbf{r}\|}{\|\mathbf{x}\|} \sum_{i=1}^{k} \frac{1}{r_i} \left(\frac{1}{\delta_i} - x_i \frac{\|\mathbf{t}\|}{\|\mathbf{x}\|} \right)^2 \geq 0. \end{aligned}$$

Therefore, $Chi''(x(S)) \geq 0$. □

Thanks to this property, the optimal point that gives the maximum value of an objective criterion must be a vertex of the convex hull of all stamp points in the k-dimensional space. The problem now becomes how to find the optimal vertex of the convex hull efficiently.

2.4 Related Work

For the case in which the target domain size is two, i.e., $k = 2$, we can order the n values so that the optimal value group is one "cut" of the ordered sequence, if we can assume convexity of the objective criterion [4], and all of the mentioned criteria do have this property. Consequently, we have an $O(n \log n)$ algorithm. However, this algorithm is not applicable to cases in which the target domain size is greater than two.

For categorical databases, in which the conditional domain size is large and $k > 2$, there is no practical existing algorithm that can find the optimal value group. Despite the difficulty, however, there are some heuristics for handling the problem [4], [19], [5], [22] that are used in practice for constructing decision trees.

A heuristic, called "two-ing" [4], divides the target domain into two classes, called superclasses, and applies the $O(n \log n)$ algorithm for $k = 2$ to create the optimal subdivision for each of the 2^{k-1} possible divisions into superclasses, and finds the best one among them. This runs in $O(2^{k-1} n \log n)$ time, which is efficient for a small constant k.

Another heuristic [22] greedily merges two conditional values from the conditional domain to reduce the conditional domain size to $n - 1$, so that the objective function is maximized. It repeats this greedy merging process until $n = 2$ and then returns the final two groups. This $O(n^3)$ heuristic can be used even if k is large.

The above heuristics are known to be practical for constructing decision trees. However, neither of them has a guarantee on the optimality of the result.

In [8], [17], [18], we used guided branch-and-bound searching to find the optimal two-dimensional numeric association rule that optimizes entropy. The proposed algorithm efficiently finds *the optimal* vertex of the convex hull of all possible points. However, k may become larger, so that $k > 4$ when we consider the problem for very large databases such as DNA databases and the rental log example. The space limitation does not permit the use of the algorithm on ordinary workstations; therefore, we present feasible algorithms for finding *high quality value groups* for the very large categorical databases which present such a challenging problem.

3 Algorithms

3.1 Preparation

When changing the value group V, we will frequently compute stamp points $\mathbf{x}(V)$, i.e., $\mathbf{x}(S)$. If we scan the database R to find each stamp point, it will always take $O(|R|)$ time. To speed up this process, we preprocess the relation as follows.

Among all possible value groups, we call value groups that consist of only one element *atomic* value groups and denote each such group V_{atom}. We can construct an arbitrary value group for the categorical conditional attribute by making a union of atomic value groups.

We compute a stamp point $\mathbf{x}(V_{\text{atom}}) = (x_1, x_2, \cdots, x_k)$ for each atomic value group beforehand. To find the stamp point of V, we simply need to sum up the points $\mathbf{x}(V) = \sum_{V_{\text{atom}} \in V} \mathbf{x}(V_{\text{atom}})$, which will take $O(n)$ time. We also compute the stamp point of the entire relation $\mathbf{x}(R)$, since the stamp point of $\bar{S} = R \setminus S$, the complement of S, can be easily computed from $\mathbf{x}(\bar{S}) = x(R) - x(S)$.

Example 3.1. Suppose we are given a relation R with categorical attributes A and C. Let A be a target attribute,

A	C	count(*)	
a_1	c_1	26	$= x_1(\{c_1\})$
a_1	c_5	15	$= x_1(\{c_5\})$
a_2	c_2	31	$= x_2(\{c_2\})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
a_k	c_n	17	$= x_k(\{c_n\})$

Fig. 1. Select A, C, count(*) from R group by A, C.

and C be a conditional attribute. The SQL query in Fig. 1 will count the number of records for each distinct value of A and value of C to generate the stamp points of the atomic value groups:

In the rest of the paper, we use P as a set of all stamp points of value groups and $P_{\text{atom}} \subset P$ as a set of all stamp points of atomic value groups. We define $Conv(P)$ to be the convex hull of P.

3.2 Greedy Enumeration Algorithm

Let us consider the case in which the target domain size is two. Any stamp point of P can be characterized as a point in 2-dimensional space, i.e., $\mathbf{x} = (x_1, x_2)$. In this case, there must be an $O(n \log n)$ algorithm for the optimal value group problem if the objective function is convex, as proved in [4]. The following is one such algorithm:

Algorithm 3.1 (Greedy Enumeration)

1. Sort points in P_{atom} in descending order of $y = x_1/(x_1 + x_2)$, taking $O(n \log n)$ time.
2. $\mathbf{p}_0 = (0, 0)$
3. For $i = 1$ to n,
   a. Let $\mathbf{x}_i$ be the ith point of the ordered point sequence.
   b. $\mathbf{p}_i = \mathbf{p}_{i-1} + \mathbf{x}_i$.
   c. Examine the stamp point $\mathbf{p}_i$ by an objective function.

Theorem 3.1. *Algorithm 3.1 scans all points of $Conv(P)$ and finds the optimal answer in $O(n \log n)$ time.*

Proof. We will show that Step 3 scans the upper chain of $Conv(P)$ (see Fig. 2). Let $\{x_1, x_2, \cdots, x_n\}$ be the ordered point sequence of P_{atom} in descending order of $x_1/(x_1 + x_2)$, and let $\mathbf{p}_0$ be the origin $(0, 0)$.

$\mathbf{p}_0$ is on the convex hull. $\mathbf{p}_1 = \mathbf{p}_0 + \mathbf{x}_1$ is also on the convex hull, since, if there is a point above $\mathbf{p}_0\mathbf{p}_1$, there must be a stamp point whose $x_1/(x_1 + x_2)$ is larger than that of $\mathbf{x}_1$. This is a contradiction. Similarly, it is easy to see that if $\mathbf{p}_i$ is on the upper convex hull, $\mathbf{p}_{i+1}$ is also on the convex hull.

We do not have to scan the lower chain of the convex hull since the upper and the lower chains are symmetric. There are n points on the upper chain of $Conv(P)$ and it takes a constant time to examine the objective function for each point. Hence, the $O(n \log n)$ cost for sorting points in P_{atom} dominates the time complexity of this algorithm. □

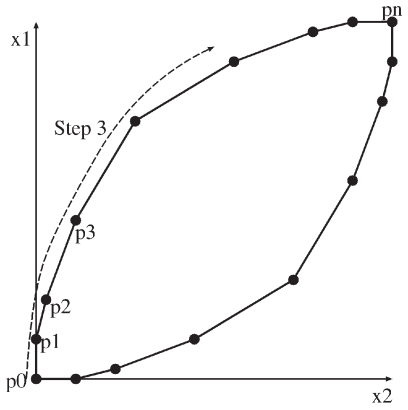

Fig. 2. Convex hull of two-class problem.

Example 3.2. To illustrate this algorithm by using Fig. 2, we assume the stamp points of eight atomic value groups shown in Table 2.

Those stamp points are ordered by $x_1/(x_1 + x_2)$. Let $\mathbf{p}_0$ be the origin and let $\mathbf{p}_i = \mathbf{p}_{i-1} + \mathbf{x}_i$ for $i = 1, \ldots, 8$. The point sequence of $\mathbf{p}_i$ $(i = 1, \ldots, 8)$ is the upper chain of the convex hull of all stamp points.

This algorithm is not applicable to cases in which the target domain size k is greater than two.

3.3 Enumeration Algorithm

If $k > 2$, we project all points of P_{atom} in the k-dimensional space into points $\mathbf{y} = (x_1/\|\mathbf{x}\|, x_2/\|\mathbf{x}\|, \ldots, x_{k-1}/\|\mathbf{x}\|)$ in the $(k-1)$-dimensional space where $\|\mathbf{x}\| = \sum_{i=1}^{k} x_i$. Let H be a $(k-2)$-dimensional) hyperplane that contains at least $k-1$ linearly independent points. H splits the projected space into two halfspaces, i.e., an upper side and a lower side of H. We can define a value group $V(H)$ that corresponds to the union of the projected points in the upper (or lower) halfspace of H. Let $x(V(H))$ be the stamp point of $V(H)$.

Theorem 3.2. *The stamp point $x(V(H))$ must be a point on $Conv(P)$. Conversely, each vertex of $Conv(P)$ can be represented as $x(V(H))$ for a suitable hyperplane H.*

Proof. We first prove the first statement of the theorem. Assume a $(k-2)$-dimensional hyperplane $H : a_1y_1 + a_2y_2 + \cdots + a_{k-1}y_{k-1} = a_k$ in the projected space. Each projected point in the upper halfspace of H must be $a_1y_1 + a_2y_2 + \cdots + a_{k-1}y_{k-1} \geq a_k$. That is equivalent to

TABLE 2
Stamp Points of Eight Atomic Value Groups

P_{atom}	x_1	x_2	$x_1/(x_1 + x_2)$	$\sum x_1$	$\sum x_2$
$\mathbf{x}_1$	3	0	1	3	0
$\mathbf{x}_2$	3	1	.75	6	1
$\mathbf{x}_3$	6	3	.66	12	4
$\mathbf{x}_4$	9	5	.64	21	9
$\mathbf{x}_5$	5	8	.38	26	17
$\mathbf{x}_6$	3	7	.3	29	24
$\mathbf{x}_7$	1	3	.25	30	27
$\mathbf{x}_8$	0	3	0	30	30

TABLE 3
Four 3D Stamp Points

	c_1	c_2	c_3	c_4
$x_1(A = a_1)$	20	20	20	30
$x_2(A = a_2)$	20	10	20	10
$x_3(A = a_3)$	40	10	0	40

$(a_1 - a_k)x_1 + (a_2 - a_k)x_2 + \cdots + (a_{k-1} - a_k)x_{k-1} - a_k x_k \geq 0$ in the original k-dimensional space.

Any point of $Conv(P)$ must have a $(k-1)$-dimensional tangential hyperplane. If the tangential hyperplane has a normal vector Θ, the hyperplane maximizes (or minimizes) the inner product (Θ, x). If a point maximizes (or minimizes) the inner product, it must be the tangential point of the hyperplane and $Conv(P)$.

Let V^+ be the value group that is defined by the upper halfspace of H. And let

$$\Theta_V = (a_1 - a_k, a_2 - a_k, \cdots, a_{k-1} - a_k, -a_k).$$

The stamp point of V^+ maximizes $(\Theta_V, \mathbf{x})$ among P, since if there is a point whose $(\Theta_V, \mathbf{x})$ is larger than V^+, there must be an atomic point whose $(\Theta_V, \mathbf{x})$ is nonnegative except points in V^+ or there must be an atomic point whose $(\Theta_V, \mathbf{x})$ is negative in V^+. It is contradiction. Therefore, the stamp point of V^+ is a point of $Conv(P)$. The second half can be proved in a similar manner. □

Theorem 3.3. *We can enumerate all of the vertices on $Conv(P)$ by examining all combination of $k-1$ atomic points.*

Proof. A set of $k-1$ linearly independent points in the projected space identifies a $(k-2)$-dimensional hyperplane $H : a_1 y_1 + a_2 y_2 + \cdots + a_{k-1} y_{k-1} = a_k$ that corresponds to a $(k-1)$-dimensional hyperplane

$$S : (a_1 - a_k)x_1 + (a_2 - a_k)x_2 + \cdots + (a_{k-1} - a_k)x_{k-1} - a_k x_k = 0$$

which contains the origin. The hyperplane S, which contains $k-1$ linearly independent points, splits P_{atom} into two groups. Note that S and two tangential hyperplanes, each of which contains the stamp point of the corresponding group, have identical normal vector Θ. Since we are focusing on enumerating different value groups, it is enough to examine all combinations of $k-1$ atomic points and corresponding values of Θ. □

Thanks to Theorems 3.2 and 3.3, the following algorithm enumerates all vertices on $Conv(P)$:

Algorithm 3.2 (Enumeration)

1. Project P_{atom} into the $(k-1)$-dimensional) space:

$$\mathbf{x} = (x_1, \cdots, x_k) \mapsto \mathbf{y} = (x_1/\|\mathbf{x}\|, \cdots, x_{k-1}/\|\mathbf{x}\|).$$

2. For each combination of $k-1$ projected points,
   a. If the $k-1$ points are linearly independent,
      i. Define a $(k-2)$-dimensional) hyperplane H containing the $k-1$ points.
      ii. Initialize a stamp point for a value group, $\mathbf{p} = 0$.
      iii. For each atomic point $\mathbf{x}_i$ of the projected points, let $\mathbf{p} = \mathbf{p} + \mathbf{x}_i$ if $\mathbf{x}_i \mapsto \mathbf{y}_i$ is in the upper halfspace associated with H.
      iv. Evaluate $\mathbf{p}$ by means of an objective function.

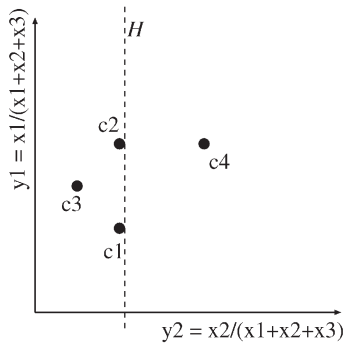

Fig. 3. A segmentation in a projected space.

Lemma 2.1 proves that one of the points on the convex hull gives the optimal value group. Therefore, we can concentrate on enumerating all the combinations of $k-1$ projected points in order to find the optimal value group.

Example 3.3. To explain the Enumeration Algorithm, let us consider the data in Table 3, whose target domain size k is three and conditional domain size n is four.

For each stamp point $\mathbf{x}(\{c_i\}) = (x_1, x_2, x_3)$ of P_{atom} in three-dimensional space, we consider a projected point $\mathbf{y} = (y_1 = x_1/(x_1 + x_2 + x_3), \quad y_2 = x_2/(x_1 + x_2 + x_3))$ in two-dimensional space. Fig. 3 illustrates the two-dimensional space. A straight line H in the projected space, which can be identified by two points, splits the projected points $\mathbf{y}$ into two groups.

The broken line H in Fig. 3, which is $0.0y_1 + 1.0y_2 = 0.25$, splits the atomic points into two group: $V^+ = \{c_1, c_2, c_3\}$ and $V^- = \{c_4\}$. H corresponds to a plane $S : -0.25x_1 + 0.75x_2 - 0.25x_3 = 0$ in the original three-dimensional space. The value group V^+ (respectively, V^-) maximizes (respectively, minimizes) the inner product with the normal vector $\Theta = (-0.25, 0.75, -0.25)$ of S (See also Section 3.5.1).

Let n be a conditional domain size of C. Since one splitting is defined by $k-1$ projected points of n atomic values, there are $O(n^{k-1})$, i.e., nC_{k-1}, different combinations, and it will take $O(n)$ time to obtain the coordinates of a stamp point of a value group. Therefore, the time complexity of the Enumeration Algorithm 3.2 is $O(n^k)$. There is a way to improve this complexity to $O(n^{k-1})$ by using a sophisticated computational geometry algorithm [1]. However, if k and n become large, it is still costly.

3.4 Random Enumeration Algorithm

To reduce the complexity of the Enumeration Algorithm, we 1) take an s-sized random sample from P_{atom}, 2) project

the sample into $(k-1)$-dimensional space, and 3) apply the Enumeration Algorithm to the sample.

Algorithm 3.3 (Random Enumeration)

1. Choose $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_s\}$ atomic values at random, so that each atomic value $\mathbf{x}$ is chosen with probability $\|\mathbf{x}\|s/|\mathbf{x}(R)|$. Recall that $|\mathbf{x}(R)|$ is the total number of records in the database.
2. Project the sample into the $(k-1)$-dimensional space:

$$\mathbf{x} = (x_1, \cdots, x_k) \mapsto \mathbf{y} = (x_1/\|\mathbf{x}\|, \cdots, x_{k-1}/\|\mathbf{x}\|).$$

3. For each combination of $k-1$ points of the s sample points,
   a. If the $k-1$ points are linearly independent,
      i. Define a $(k-2)$-dimensional) hyperplane H containing the $k-1$ points.
      ii. Initialize the stamp point of a value group, $\mathbf{p} = 0$.
      iii. For each atomic value bfx_i of P_{atom}, let $\mathbf{p} = \mathbf{p} + \mathbf{x}_i$ if $\mathbf{x}_i \mapsto \mathbf{y}_i$ is in the upper halfspace associated with H.
      iv. Evaluate $\mathbf{p}$ by means of an objective function.

The time complexity of this randomized algorithm becomes $O(s^{k-1}n)$ because it is proportional to sC_{k-1}, and can be further improved to $O(s^{k-2}n)$. This algorithm needs only a small working space and is easy to parallelize.

3.4.1 Quality Analysis on Sample Size

As we have shown above, the optimal solution is given as a subdivision of atomic stamp points by a hyperplane cut in $(k-1)$-dimensional space. From the PAC learning theory, such a subdivision can be closely approximated by using a small number of samples. Let Y be the set of points in $(k-1)$-dimensional space and Z be a random sample from Y. We say that Z is an ϵ-net for a region family if $|X \cap Y|/|Y| \leq \epsilon$ holds for every region X of the family satisfying $X \cap Z = \emptyset$.

Note that our sample strategy is weighted sampling. First, we consider the unweighted case where $|\mathbf{x}| = 1$ for each x. Suppose that the optimal value group V_{opt} is given by a hyperplane H_{opt}. Let us consider the family of wedges bounded by H_{opt} and another hyperplane. This family of wedges defines at most $O(n^k)$ different value groups of n points, roughly speaking, because the Vapnik-Chervonenkis dimension is k and, hence, it is known [15], [2] that a random sample of size

$$s(\epsilon) = \epsilon^{-1} \max\{6k \log(16k\epsilon^{-1}), 4 \log(2\delta^{-1})\}$$

is an ϵ-net with a probability of at least $1-\delta$. Note that $s(\epsilon)$ is independent of $n = |Y|$.

Let us take a sample that is an ϵ-net for our wedges, and let V_{sample} be the value group obtained by the sample maximizing the objective function (e.g., Gini($\mathbf{x}$)) F. There exists a value group V'_{sample} obtained as the sample such that the edge bounded by H_{opt} and H'_{sample} (the hyperplane associated with V'_{sample}) contains no sample point. Since our sample is an ϵ-net, the set difference between the value groups V'_{sample} and V_{opt} contains at most ϵn points. Fig. 4 shows a sample that is an ϵ-net when $k = 2$. The gray wedge in the figure shows a subdivision bounded by H_{opt} and H'_{sample}.

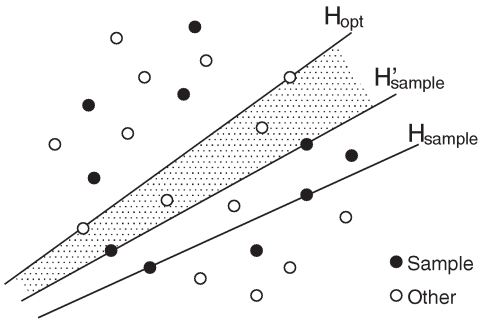

Fig. 4. Sample and subdivision.

For the weighted case, we consider each value group $\mathbf{x}$ as a set of $|\mathbf{x}|$ copies of the point to obtain the result that if we take $s(\epsilon)$ samples, then the set difference between the value groups V'_{sample} and V_{opt} contains points whose total weight is at most $\epsilon|\mathbf{x}(R)|$.

By definition, $F(\mathbf{x}(V_{sample})) \geq F(\mathbf{x}(V'_{sample}))$ and, hence, we have a lower bound of $F(\mathbf{x}(V_{sample}))$. For example, we can show that

$$Gini(\mathbf{x}(V_{opt})) - Gini(x(V_{sample})) \leq 2\epsilon + \alpha\epsilon^2,$$

where $\alpha = |R|(|V_{opt}|^{-1} + |V_{opt}|^{-1})$. Since, we do not want to find a subdivision with a very large α, this is a good approximation if ϵ is small.

We can use the theory of ϵ-approximation for the k-labeled space given by Hasegawa et al. [14] to avoid introducing α into the analysis. Theoretically, if we want to make $\epsilon = 0.01$, $s(0.01) = 600k \log(1600k)$, which is very large. However, the theoretical bound is very pessimistic, and a much smaller sample is sufficient, as we will show by experiment later.

3.5 Probing Algorithm

3.5.1 Hand Probing

The Probing Algorithm also searches for stamp points on $Conv(P)$. Computing a stamp point on $Conv(P)$ and its corresponding value group without knowing the coordinates of the point is called *hand probing* in the field of computational geometry [6]. Geometrically, hand probing in a k-dimensional space means computing a tangential point of a $(k-1)$-dimensional) hyperplane and $Conv(P)$.

Any tangential hyperplane in a k-dimensional space has a normal vector $\Theta = (\theta_1, \theta_2, \cdots, \theta_k)$. By giving a k-dimensional vector Θ, we can compute the tangential point $\mathbf{p}$ of the hyperplane and $Conv(P)$ by maximizing (or minimizing) the inner product $(\Theta, \mathbf{p})$.

Example 3.4. Let us consider once again the Example 3.3. The tangential point of a hyperplane with a normal vector $\Theta = (-0.25, 0.75, -0.25)$ maximizes (or minimizes) the inner product

$$(\Theta, \mathbf{x}) = -0.25x_1 + 0.75x_2 - 0.25x_3.$$

We compute the inner product $(\Theta, \mathbf{x})$ for each stamp point of $P_{\text{atom}} = \{c_1, \cdots, c_4\}$ as shown in Table 4.

TABLE 4
Inner Product of Each Stamp Point

	c_1	c_2	c_3	c_4
$(\Theta, \mathbf{x})$	0	0	10	-10

The value group corresponding to the hand probing is the union of those terms whose inner product is nonnegative (respectively, negative). The coordinates of the tangential point (value group) can be obtained by summing up the coordinates of those terms. In this example, $C = c_1 \vee c_2 \vee c_3$ (respectively, $C = c_4$) is the value group, and the tangential point is $(60, 50, 50)$ (respectively, $(30, 10, 40)$).

The time complexity to compute a stamp point $\mathbf{p}$ maximizing $(\Theta, \mathbf{p})$ for a given Θ is $O(n)$, where n is the conditional domain size.

3.5.2 Convex Hull Searching

We have shown that stamp points on $Conv(P)$ can be computed efficiently by using hand probing. Now, let us consider how to find the optimal points on $Conv(P)$.

Various convex hull algorithms have been studied intensively [21], since many problems, such as optimized numeric association rules [11], [10], [12], two-dimensional association rules [9], [26] in data mining, classification and regression trees [8], [17], [18], can be interpreted as convex hull problems. The Probing Algorithm uses an online convex hull maintenance algorithm called the "beneath-beyond" method.

First of all, we compute k different points, which are linearly independent, by using hand probings with k different random vectors. Consequently, we have $2k$ points. Empirically, a vector Θ_{init} satisfying $(\Theta_{init}, \mathbf{x}(R)) = 0$ determines a satisfactory point as an initial stamp point with respect to a convex criterion. The line containing the origin and $\mathbf{x}(R)$ is the central line of the convex hull, and a hyperplane whose normal vector is Θ_{init} is parallel to the line. Therefore, we include such vectors in the initial set of k vectors. Strictly speaking, we may not be able to find k linearly independent points in a k-dimensional space by any hand probing. In this case, we project all points into a $(k-1)$-dimensional and solve the problem as a $(k-1)$-dimensional one.

We can define an inscribed convex polygon whose vertices are the initial $2k$ points, inside $Conv(P)$. We can also define a circumscribed polygon whose facets are tangential hyperplanes used by hand probings. Fig. 5 shows examples of inscribed and circumscribed polygons of $Conv(P)$ in two-dimensional space, i.e., $k = 2$.

Let $I \subseteq P$ be a set of vertices of an inscribed convex polygon, and let C be a set of intersecting points of tangential hyperplanes of $Conv(P)$ that are vertices of an circumscribed convex polygon. Thanks to Lemma 2.1, we know stamp points inside $Conv(I)$ are not better than the best vertex of I. If we consider a facet of $Conv(I)$, the stamp points between $Conv(C)$ and the facet are not better than the corresponding vertex of $Conv(C)$ and the vertices of the facet. For example, if we consider the facet f_1 in Fig. 5, the stamp points inside the gray triangular region can not be better than $v1$, $v2$, and $v3$. Therefore, the value of a vertex of C gives a lower bound of corresponding points on $Conv(P)$ inside $Conv(C)$ and $Conv(I)$.

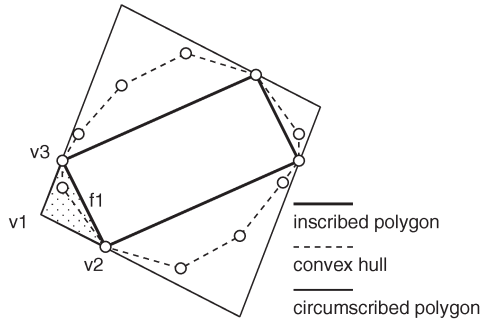

Fig. 5. Inscribed polygon and circumscribed polygon.

The guided branch-and-bound search method [8], [17], [18] recursively refines the inscribed and circumscribed polygons. It efficiently finds the optimal point on $Conv(P)$ by using those lower bounds to order and drop candidate facets. However, there are as many as $(n^{k-1})^{\lfloor \frac{k}{2} \rfloor}$ facets on $Conv(P)$. The method needs to maintain $Conv(I)$ and $Conv(C)$ that may have as many facets as $Conv(P)$ has. Therefore, it is only applicable when k is small enough for the available working space.

In the Probing Algorithm, we predefine the size of the working space that is used to maintain facets of $Conv(I)$ and drop facets so that it works within the limited working space.

The following algorithm is the essential part, which contains refinement procedure of facet queue of the inscribed polygon, of the Probing Algorithm.

Algorithm 3.4 (Probing)

1. Initialize a facet queue Q from I
2. For the first facet f of Q
   a. Compute the normal vector Θ of f.
   b. Compute a tangent point of $Conv(P)$ by using hand probing with Θ.
   c. If a new point $\mathbf{x}$ is found,
      i. Refine $I := I \cup \mathbf{x}$.
      ii. Create new facets by using $\mathbf{x}$.
      iii. Replace the facets in Q.

In each refinement procedure, we can make at least k facets outside of the inscribed convex polygon. Fig. 6 is an example for a three-dimensional case. If we find x by hand probing with the normal vector of the facet $\{1, 2, 3\}$, we add three facets $\{x, 2, 3\}$, $\{1, x, 3\}$, $\{1, 2, x\}$. If k is large, the limited working space may run out after several refinement procedures. We maintain Q as a priority queue. We decide the priority of new facets based on the value of the new point. In the example, the value of x is used for the three new facets.

We empirically find that stamp points for which the value of the objective criterion exceeds the best value tend to lie near the best point or near the points that are closest to the best value. The above heuristics help to find a high quality result in an earlier step of the algorithm.

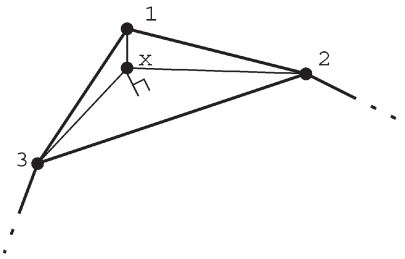

Fig. 6. Inscribed polygon refinement.

Furthermore, in order to prevent too many excessively small expansions of the convex polygon, we use a distance threshold. If the distance between a facet and a new point is smaller than a given threshold, the new facets that contain the new point are pruned away. This heuristic also helps to speed up the searching.

Though the expected running time of the Probing Algorithm is still substantial, it can find a satisfactory point in a period much shorter than the running time, and can return the intermediate value group interactively.

4 Experiments

We implemented the proposed algorithms and performed several experiments to evaluate their performance. All experiments were performed on an IBM RS/6000 workstation with a POWER2 processor running at 66 MHz with 2 MB of L2 cache and 256 MB of real memory.

In this experiment, we generated synthetic data with $n = 1,000$ (conditional domain size) and various values of k (target domain size) to simulate a very large categorical database. Each type of data set has 10,000 records and two categorical attributes, C and A. The conditional attribute C takes $c_1, \cdots, c_n$ distinct values. The target attribute A takes $a_1, \cdots, a_k$ distinct values. For each record of the synthetic data, we randomly and independently assign a value of C and a value of A. For purposes of comparison, we use the "two-ing" which is used in the CART system to compute a binary segmentation. Note that another well-known conventional heuristic used in C4.5 is unsuitable for large n and does not perform well in this condition.

4.1 Random Enumeration Algorithm

Fig. 7 shows the relationship between the execution time for a single run and the improvement in the gini index gained by a value group obtained by the Random Enumeration Algorithm for various sample sizes. The target domain sizes k are 4 and 6. The numbers in the graph represent the sample sizes. Each error-bar (vertical line) indicates the range between the best and worst results of 32 runs for each sample size. The results of the 32 runs are distributed throughout the range, and each point on the range shows the average value of these results.

The "two-ing" method deterministically computes a value group for each problem. Each diamond mark in the figures indicates the time taken to compute the value group and the improvement in the gini index achieved by the value group. We draw a horizontal broken line for each diamond mark so that our algorithms can be easily compared with the "two-ing" method. Though the "two-ing" gives the optimal value group of a certain superclass obtained by grouping k classes, the heuristic is known to find a relatively good approximated value group.

From this experiment, we can see that the Random Enumeration Algorithm generates a result of satisfactory quality within a practical time when the sample size is around 20. Note that the best result of the 32 runs is better than the result of the "two-ing" method, even if we use a small sample size. The CPU time taken for a single run is almost proportional to sC_{k-1}.

When k becomes larger, we have to use a small sample size s so that the algorithm can terminate in a practical amount of time. However, a small sample often gives low-quality results. To overcome this problem, we run this algorithm with a small sample a number of times, and take the best result. Fig. 8 shows the relationship between the execution time for 32 runs and the best improvement in the gini index of the 32 runs when $k = 8$ and 10.

The Random Enumeration Algorithm achieved better or comparable results with a small sample in 32 runs. Those multiple trials can be executed independently. Therefore, we can expect better results if we can use a parallel environment.

4.2 Probing Algorithm

Fig. 9 shows the extent to which the gini index value is improved by the Probing Algorithm, along with the time taken in seconds. Thanks to the heuristics for maintaining the facet queue, we can observe that when the Probing Algorithm found a better result, it often found a still better result within a short period of time. As a result, the Probing Algorithm tends to find a satisfactory value group relatively quickly. Such a value group, which is much better than the

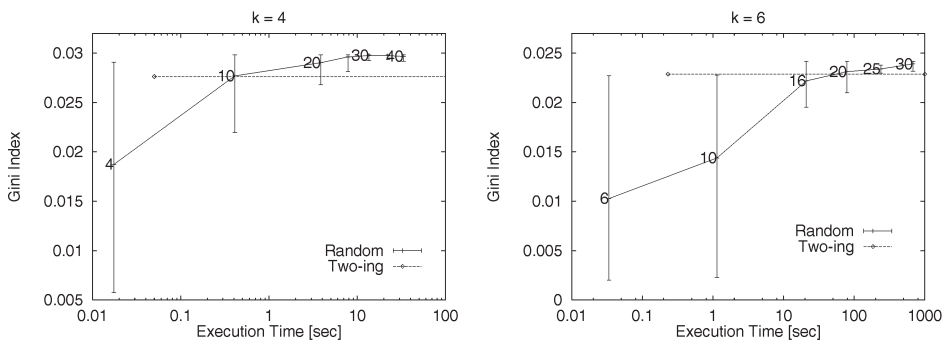

Fig. 7. Performance of the Random Enumeration Algorithm (1).

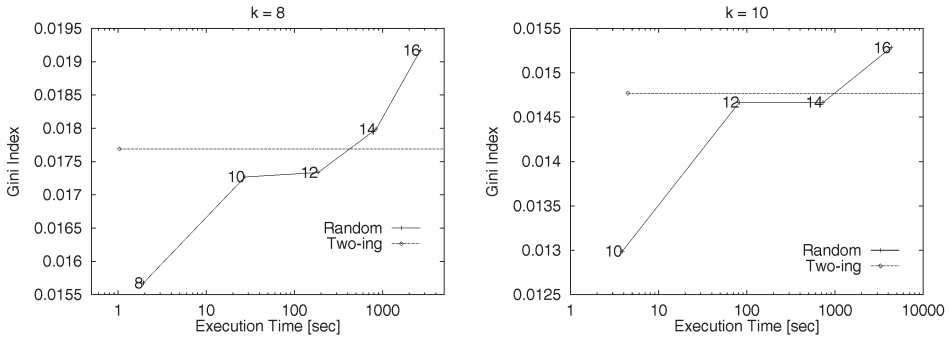

Fig. 8. Performance of the Enumeration Random Algorithm (2).

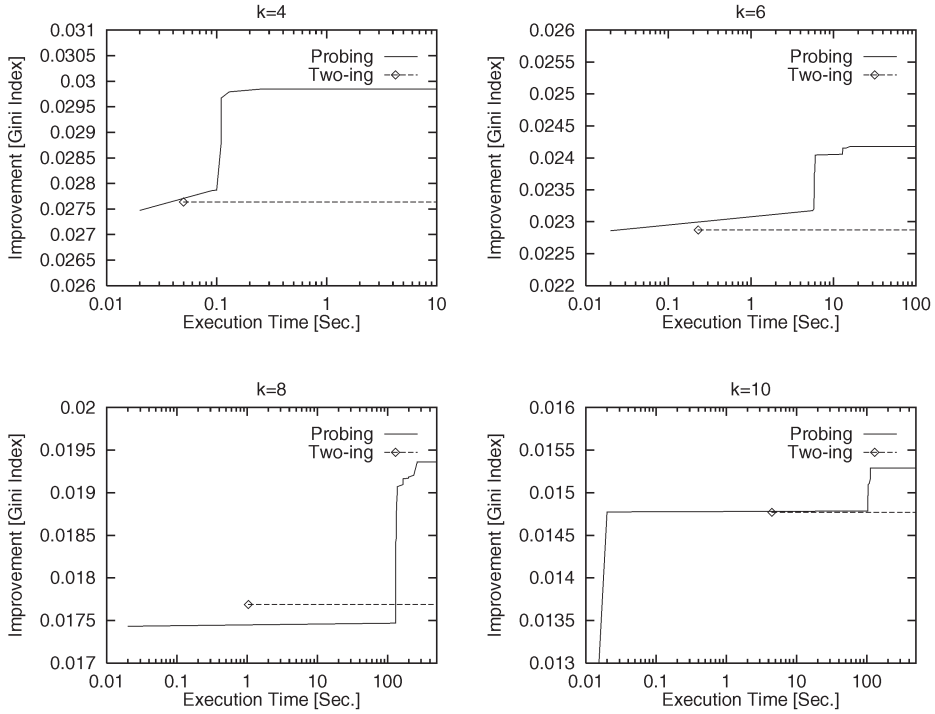

Fig. 9. Performance of the Probing Algorithm.

result given by the "two-ing" method, is satisfactory for most applications. Thus, the Probing Algorithm returns an intermediate result in a practical time.

One defect of the Probing Algorithm is its required working space when k becomes large. However, it can usually find a satisfactory result before its working space becomes too large. In the experiment for $k = 10$, the working space of the Probing Algorithm is less than 64 MB, which is acceptable on most workstations, when the best result in the graph was obtained. In these experiments, we limited the working space of the Probing Algorithm to 130 MB.

These experiments used synthetic databases whose values are randomly assigned. We have tried similar experiments by using other synthetic and real databases whose values are skewed. We observed that the performance of our algorithms is basically the same for such skewed cases.

The results of the experiments show that both the Random Enumeration Algorithm and the Probing Algorithm found a better value group than the "two-ing" method in a practical time, and that both are feasible.

5 Concluding Remarks

We have proposed geometric algorithms for finding value groups that make high quality binary segmentations of a categorical database. We can use the "gini index," "entropy," or "χ^2" as an objective criterion to indicate the extent to which the values of a target attribute are skewed from the original distribution. High quality value groups with respect to these criteria provide us with a clue to understanding the target attribute.

Though the complexities are $O(s^{k-2}n)$ for the Random Enumeration Algorithm and $O((n+m)|P|)$ for the Probing Algorithm, we derived practical implementations by using randomization and strategic facet maintenance. Various experiments confirmed that the algorithms could find satisfactory value groups within reasonable computation times. The quality of the results obtained for various samples by the Random Enumeration Algorithm differs dramatically if we use a small sample size. Therefore, multiple trials are needed to obtain better value groups. Those trials can be executed independently and in parallel; thus, the Random Enumeration Algorithm is

suitable for a parallel environment. On the other hand, the quality of the Probing Algorithm becomes stable after a certain amount of execution time. However, it requires a large working space compared to the Random Enumeration Algorithm. Since the available memory size is increasing, the practical applicability of the Probing Algorithm seems to be promising.

ACKNOWLEDGMENTS

This research was partially supported by the Advanced Software Enrichment Project of the Information-Technology Promotion Agency, Japan. Part of this work is based on the extended abstract of *Algorithms for Mining Association Rules for Binary Segmentation of Huge Categorical Databases* in Proceedings of the 24th VLDB Conference (New York, USA), 1998.

REFERENCES

[1] T. Asano and T. Tokuyama, "Topological Walk Revisited," *Proc. Sixth CCCG,* 1994.
[2] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth, "Learnability and the Vapnik-Chervonenkis Dimension," *J. ACM,* vol. 36, pp. 929-965, 1989.
[3] L. Breiman, "Bagging Predictors," *Machine Learning,* vol. 24, no. 2, pp. 123-140, Aug. 1996.
[4] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, *Classification and Regression Trees.* Wadsworth, 1984.
[5] N.A.R. Center, "Introduction to IND Version 2.1,"GA23-2475-02, 1992.
[6] D. Dobkin, H. Edelsbrunner, and C. Yap, "Probing Convex Polytopes," *Proc. 18th ACM Symp. Theory of Computing,* pp. 387-392, 1986.
[7] Y. Freund and R.E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *J. Computer and System Sciences,* vol. 55, no. 1, pp. 119-139, Aug. 1997.
[8] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Constructing Efficient Decision Trees by Using Optimized Association Rules," *Proc. 22nd VLDB Conf.,* pp. 146-155, 1996.
[9] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Data Mining Using Two-Dimensional Optimized Association Rules: Scheme, Algorithms, and Visualization," *Proc. ACM SIGMOD Conf. Management of Data,* pp. 13-23, June 1996.
[10] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Finding Optimal Intervals Using Computational Geometry," *Proc. Int'l Symp. Algorithm and Computing '96,* pp. 55-64, 1996.
[11] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Mining Optimized Association Rules for Numeric Attributes," *Proc. 15th ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems,* pp. 182-191, 1996.
[12] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Mining Optimized Association Rules for Numeric Attributes," *J. Computer and System Sciences,* 1999.
[13] J. Gehrke, R. Ramakrishnan, and V. Ganti, "RainForest—A Framework for Fast Decision Tree Construction of Large Datasets," *Proc. 24th Int'l Conf. Very Large Data Bases,* pp. 416-427, 1998.
[14] S. Hasegawa, H. Imai, and M. Ishiguro, "ϵ-Approximations of k-Label Spaces," *Theoretical Computer Science,* vol. 137, pp. 145-157, 1995.
[15] D. Haussler and E. Welzl, "Epsilon-Nets and Simplex Range Queries," *Discrete and Computational Geometry,* vol. 2, pp. 127-151, 1987.
[16] M. Mehta, R. Agrawal, and J. Rissanen, "SLIQ: A Fast Scalable Classifier for Data Mining," *Proc. Fifth Int'l Conf. Extending Database Technology,* pp. 18-32, 1996.
[17] Y. Morimoto, T. Fukuda, S. Morishita, and T. Tokuyama, "Implementation and Evaluation of Decision Trees with Range and Region Splitting," *Constraint,* vol. 2, no. 3/4, pp. 401-427, Dec. 1997.
[18] Y. Morimoto, H. Ishii, and S. Morishita, "Efficient Construction of Regression Trees with Range and Region Splitting," *Proc. 23nd VLDB Conf.,* pp. 166-175, 1997.
[19] P.M. Murphy and M.J. Pazzani, "Id2-of-3: Constructive Induction of m-of-n Concepts for Discriminators in Decision Trees," *Proc. Eighth Int'l Workshop Machine Learning,* pp. 183-187, 1991.
[20] Z. Pawlak, J. Grzymala-Busse, R. Slowinski, and W. Ziarko, "Rough Sets," *Comm. ACM,* vol. 38, no. 11, Nov. 1995.
[21] F.P. Preparata and M.I. Shamos, *Computational Geometry, An Introduction.* Springer-Verlag, 1985.
[22] J.R. Quinlan, *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.
[23] J.R. Quinlan, "Bagging, Boosting, and C4.5," *Proc. 13th Nat'l Conf. Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conf.,* pp. 725-730, 1996.
[24] R. Rastogi and K. Shim, "PUBLIC: A Decision Tree Classifier that Integrates Building and Pruning," *Proc. 24th Int'l Conf. Very Large Data Bases,* pp. 404-415, 1998.
[25] J. Shafer, R. Agrawal, and M. Mehta, "SPRINT: A Scalable Parallel Classifier for Data Mining," *Proc. 22nd Int'l Conf. Very Large Data Bases,* pp. 544-555, 1996.
[26] K. Yoda, T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Computing Optimized Rectilinear Regions for Association Rules," *Proc. KDD'97,* pp. 96-103, Aug. 1997.

Yasuhiko Morimoto received the BE, ME , and PhD degree in 1989, 1991, and 2002, respectively, from Hiroshima University. He is currently with the IBM Tokyo Research Laboratory. His research interests include database system, data mining, and knowledge discovery. He is a member of the ACM, the IEEE Computer Society, and the Information Processing Society of Japan.

Takeshi Fukuda received the BE, ME, and PhD degrees from Waseda University in 1989, 1991, and 1999, respectively. He is currently with the IBM Tokyo Research Laboratory, where he has been working on object-oriented databases and data mining. He is a member of the ACM, the IEEE Computer Society, and the Information Processing Society of Japan.

Takeshi Tokuyama received the BS, MS, and PhD degrees in mathematics from the University of Tokyo in 1979, 1981, and 1985, respectively. From 1986 to 1999, he was with the IBM Tokyo Research Laboratory. He was assigned to IBM T. J. Watson Research Center from April 1992 to March 1993. Since October 1999, he has been a professor at Tohoku University. His research interests include computational geometry, algorithm theory, combinatorial optimization, discrete mathematics, and their applications. He is a member of the ACM, the Information Processing Society of Japan, the JSIAM, the Mathematical Society of Japan, and the ORSJ.