Constructing Efficient Decision Trees by Using Optimized Numeric Association Rules

Takeshi Fukuda
fukudat@trl.ibm.co.jp

Yasuhiko Morimoto
morimoto@trl.ibm.co.jp

Shinichi Morishita
morisita@trl.ibm.co.jp

Takeshi Tokuyama
ttoku@trl.ibm.co.jp

IBM Tokyo Research Laboratory
1623-14, Shimo-tsuruma, Yamato City, Kanagawa Pref, 242, JAPAN

Abstract

We propose an extension of an entropy-based heuristic of Quinlan [Q93] for constructing a decision tree from a large database with many numeric attributes. Quinlan pointed out that his original method (as well as other existing methods) may be inefficient if any numeric attributes are strongly correlated. Our approach offers one solution to this problem. For each pair of numeric attributes with strong correlation, we compute a two-dimensional association rule with respect to these attributes and the objective attribute of the decision tree. In particular, we consider a family $\mathcal{R}$ of grid-regions in the plane associated with the pair of attributes. For $R \in \mathcal{R}$, the data can be split into two classes: data inside R and data outside R. We compute the region $R_{opt} \in \mathcal{R}$ that minimizes the entropy of the splitting, and add the splitting associated with R_{opt} (for each pair of strongly correlated attributes) to the set of candidate tests in Quinlan's entropy-based heuristic.

We give efficient algorithms for cases in which $\mathcal{R}$ is (1) x-monotone connected regions, (2) based-monotone regions, (3) rectangles, and (4) rectilinear convex regions. The algorithm for the first case has been implemented as a subsystem of SONAR(System for Optimized Numeric Association Rules) developed by the authors. Tests show that our approach can create small-sized decision trees.

**Proceedings of the 22nd VLDB Conference
Mumbai(Bombay), India, 1996**

1 Introduction

Decision Trees

Constructing an efficient decision tree is a very important problem in database mining [AGL+92, ALS93, BFOS84, MAR96, Q93]. For example, an efficient computer-based diagnostic medical system can be constructed if a small decision tree can be automatically generated for each medical problem from a database of health-check records for a large number of patients.

Let us consider the attributes of tuples in a database. An attribute is called Boolean if its range is $\{0,1\}$, categorical if its range is a discrete set $\{1,..,k\}$ for some natural number k, and numeric if its range is the set of real numbers.

Each data tuple t has $m+1$ attributes A_i, for $i = 0,1,..,m$. We treat one Boolean attribute (say, A_0) as special, denote it by W, and call it the *objective attribute*. The other attributes are called *conditional attributes*.

The decision tree problem is as follows: A set U of tuples is called "positive" (resp. negative) if for a tuple t, the probability that $t[W]$ is 1 (resp. 0) is at least θ_1 (resp. θ_2) in U, for given thresholds θ_1 and θ_2. We would like to classify the set of tuples into positive subsets and negative subsets by using tests with conditional attributes. For a Boolean (conditional) attribute, a test is in the form of "$t[A_i] = 1$?". For a categorical attribute, a traditional test is "$t[A_i] = l$?". For a numeric attribute, a traditional test is "$t[A_i] < Z$?" for a given value Z.

Let us consider a rooted binary tree, each of whose internal nodes is associated with a test that has attributes. We associate with each leaf node the subset (called leaf-cluster) of tuples satisfying all tests on the path from the root to the leaf. If every leaf-cluster is either positive or negative, the tree is called a *decision* *tree.*

For example, assume that we have a database of health-check records for a large number of patients with geriatric diseases. Consider a set of health-check items; say, systolic blood pressure, urine sugar (S), and cholesterol level (C). We would like to decide whether a patient needs a detailed health check for a geriatric disease (say, apoplexy). Suppose that blood-pressure is a numeric attribute, and that urine sugar and cholesterol level are Boolean (+ or −) attributes in the health check database. Figure 1 shows an examples of decision trees corresponding to the table below:

Blood-pressure	Cholesterol +		Cholesterol−	
	Sugar+	Sugar−	Sugar+	Sugar−
*above*110	○ (i.e. geriatric-disease positive)			
90 − 110	○			×
80 − 90	○	×	○	×
*below*80	○	×		

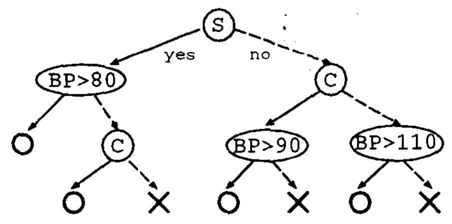


Figure 1: Decision tree

Unfortunately, the problem of constructing a minimum decision tree is known to be NP-hard [HR76, GJ79], if one want to minimize the total sum of the lengths of exterior paths. It is also believed that it is NP-hard if the minimized objective is the "size" (number of nodes) of the tree.

Despite the NP-hardness of the problem, many practical solutions [BFOS84, Q86, QR89, Q93] have been proposed in the literature. Among them, Quinlan's C4.5 program [Q93] applies an entropy heuristic, which greedily constructs a decision tree in a top-down, breadth-first manner according to the "entropy of splitting." At each internal node, the heuristic examines all the candidate tests, and chooses the one for which the associated splitting of the set of tuples attains the minimum entropy value.

If each test attribute is Boolean or categorical, Quinlan's method works well, and SLIQ of Mehta et al. [MAR96] gives an efficient scalable implementation, which can handle a database with 10 million tuples and 400 attributes. SLIQ uses the GINI function instead of entropy.

Handling Numeric Attributes

To handle a numeric attribute, one approach is to make it categorical, by subdividing the range of the attribute into smaller intervals. Another approach is to consider a test of the form $t[A_i] > Z$ or $t[A_i] < Z$, which is called a "guillotine cut", since it creates a "guillotine-cut subdivision" of the Cartesian space of ranges of attributes. Quinlan's C4.5 and SLIQ adopt the latter approach.

However, Quinlan [Q93] himself pointed out that this approach has a serious problem if a pair of attributes are correlated. For example, let us consider two numeric attributes, "height (m)" and "weight (kg)", in the health check database. Obviously, these attributes have a strong correlation. Indeed, the region $0.85*22*height^2 < weight < 1.15*22*height^2$ and its complement provide a popular criterion for separating healthy patients from patients who need dietary cures. In the left chart of Figure 2, the gray region shows the "healthy" region. However, if we construct a decision tree for classifying patients by using guillotine cutting, its subdivision is complicated, and hence, the size of the tree becomes very large (see the right chart of Figure 2).

Therefore, it is very important to propose a better scheme for handling numeric attributes with strong correlations in order to make an efficient diagnostic system based on decision tree.

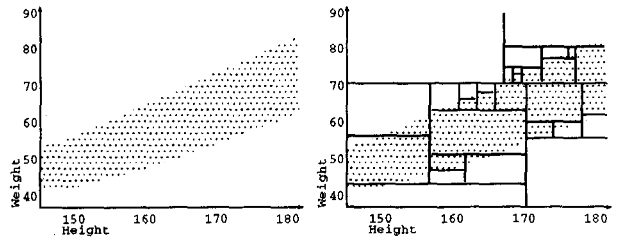


Figure 2: Healthy region, and guillotine-cut subdivision to separate it from data

One popular approach is as follows: Consider each pair of numeric attributes as one two-dimensional attribute. Then, for each such two-dimensional attribute, compute a line partition of the corresponding two-dimensional space so that the corresponding entropy is minimized. One (minor) defect of this method is that it is not cheap to compute the optimal line. Although some work has been done on this problem in computational geometry [AT94, DE93], the worst time complexity remains $O(n^2)$ if there are n tuples. Another (and major) defect is that the decision tree may still be too large even if we use line partition.

Main Results – Splittings with respect to regions

In this paper, we propose the following scheme, applying the two-dimensional association rules (region rules) of Fukuda et al. [FMMT96a, FMMT96b] and an image segmentation algorithm of Asano et al. [ACKT96]. The scheme has been implemented as a subsystem of SONAR (System for Optimized Numeric Association Rules) developed by the authors [FMMT96c].

Let n be the number of tuples in the database. First, for each numeric attribute, we create an equi-depth bucketing so that tuples are uniformly distributed into $N \leq \sqrt{n}$ ordered buckets according to the values of the attribute.

Next, we find all pairs of strongly correlated numeric attributes. For each such a pair A and A', we create an $N \times N$ pixel grid G according to the Cartesian product of the bucketing of each numeric attribute. We consider a family $\mathcal{R}$ of grid regions; in particular, we consider the set $\mathcal{R}(Admi)$ of all *admissible* (i.e. connected and x-monotone) regions and $\mathcal{R}(Base)$ of all *based-monotone* (i.e. bounded by an x-monotone grid curve) regions. Here, a grid region is a union of pixels of G, and it is x-monotone if its intersection with each column of G is either empty or a vertical strip. A grid curve consists of edges of the pixel grid G, and is x-monotone if its intersection with each vertical line is either a point or an interval. Figure 3 shows instances of a based monotone region and an admissible region. A based-monotone region may be disconnected as shown in Figure 3, since the bounding grid curve may contain segments of the upper or lower boundary of G. Note that a connected based-monotone region is an admissible region. We also deal with the family of rectangles and the family of rectilinear convex polygonal regions.

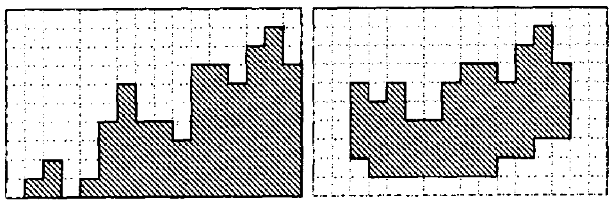


Figure 3: Based Monotone Region (left) and Admissible Region (right)

Regarding the pair of attributes as a two-dimensional attribute, we compute the region R_{opt} in $\mathcal{R}$ that minimizes the entropy function, and consider the decision rule $(t[A], t[A']) \in R_{opt}$. We present algorithms to compute R_{opt} in worst-case times of $O(nN)$ and $O(nN^2)$ for $\mathcal{R}(Base)$ and $\mathcal{R}(Admi)$, respectively. Moreover, in practical instances, our algorithms run in $O(N^2)$ time and $O(N^2 \log n)$ time. Since $N \leq \sqrt{n}$, the time complexities are $O(n)$ and $O(n \log n)$, respectively. For rectangles and rectilinear convex polygonal regions, the time complexity increases to $O(nN^3)$ in the worst case and $O(N^3 \log n)$ in practice.

Now, we add these rules (for all pairs (A, A') of correlated attributes) to Quinlan's original scheme, and construct a decision tree by applying entropy-based heuristic. As a special case of region rules, we also consider rules of the form $(t[A] \in I)$ for an interval I in order to develop our system.

Since the regions separated by guillotine cutting and those separated by line cutting are very special cases of connected based-monotone regions, our method can find decisions that create splittings with smaller entropy values at each step of Quinlan's heuristic. Hence, we can almost always create a smaller tree. In the above example of "height" and "weight", the rule $0.85 * 22 * height^2 < weight < 1.15 * 22 * height^2$ itself defines an admissible region, and hence we can create a nice decision tree of height two (i.e. with the root and two leaves). One defect of our approach is that the decision rule $(t[A], t[A']) \in R$ is sometimes hard to describe. However, we can describe the rule by combining a visualization system and an approximation scheme, using interpolation functions.

We also discuss the generalization of our method to cases in which the objective attribute is categorical.

2 Entropy-Based Data Segmentation for Decision Trees

2.1 Entropy of a splitting

Assume that a data set S contains n tuples. To formalize our definition of entropy of splitting, we consider a more general case in which the objective attribute W is a categorical attribute taking values in $\{1, 2, \ldots, k\}$.

The entropy value $Ent(S)$ (with respect to the objective attribute W) is defined as

$$Ent(S) = - \sum_{j=1,\ldots,k} p_j \log p_j$$

where p_j is the relative frequency with which W takes the value j in the set S.

We now consider the entropy function associated with a splitting of the data. For example, suppose that the objective attribute has three categories, say C_1, C_2, and C_3, and that each category has 40, 30, and 30 data, respectively.

	C_1	C_2	C_3
100	40	30	30

The value of the entropy of the whole data set is

$$-\frac{40}{100}\log\frac{40}{100} - \frac{30}{100}\log\frac{30}{100} - \frac{30}{100}\log\frac{30}{100} = 1.09.$$

Let us consider a splitting of the data set into two subsets, S_1 and S_2, with n_1 and n_2 data, respectively, where $n_1 + n_2 = n$. The entropy of the splitting is defined by

$$Ent(S_1; S_2) = \frac{n_1}{n} Ent(S_1) + \frac{n_2}{n} Ent(S_2).$$

If we assume that the splitting is as follows:

S_1	C_1	C_2	C_3
60	40	10	10

S_2	C_1	C_2	C_3
40	0	20	20

the entropy index value of the dataset after the segmentation is

$$\frac{60}{100}\left(-\frac{40}{60}\log\frac{40}{60} - \frac{10}{60}\log\frac{10}{60} - \frac{10}{60}\log\frac{10}{60}\right)$$
$$+\frac{40}{100}\left(-\frac{20}{40}\log\frac{20}{40} - \frac{20}{40}\log\frac{20}{40}\right) = 0.80.$$

Therefore, the splitting decreases the value of the entropy by 0.29.

Let us consider another splitting:

S_1	C_1	C_2	C_3
60	20	20	20

S_2	C_1	C_2	C_3
40	20	10	10

In this case, the value of the associated entropy is 1.075, a decrease of only 0.015.

Let $f(X) = f(x_1, .., x_k) = \sum_{i=1}^{k} x_i \log(x_i/s(X))$, where $s(X) = \sum_{i=1}^{k} x_i$. We have

$$Ent(S) = -f(p_1, ..., p_k) = -\frac{1}{n} f(x_1, .., x_k),$$

where $n = |S|$ and $x_i = p_i n$. Thus,

$$Ent(S_1; S_2) = -\frac{1}{n}\{f(y_1, ..., y_k) + f(x_1 - y_1, ..., x_k - y_k)\},$$

where x_i (resp. y_i) is the number of tuples t in S (resp. S_1) satisfying $t[W] = i$. We use the following property of f (proof is omitted in this version):

Lemma 2.1 *The function $f(X)$ is convex in the region $X \geq 0$ (i.e. $x_i \geq 0$ for $i = 1, 2, .., k$); that is,*

$$\frac{f(X) + f(X + 2\mathbf{a})}{2} \geq f(X + \mathbf{a})$$

for any vector $\mathbf{a}$ *satisfying $X > 0$ and $X + 2\mathbf{a} > 0$.*

2.2 Splittings with respect to regions

Given a numeric attribute A, Quinlan [Q93] and Mehta et al. [MAR96] considered the following optimized splitting:

1. Let $S(A > Z) = \{t \in S : t[A] > Z\}$ and $S(A \leq Z) = \{t \in S : t[A] \leq Z\}$ for a real number Z. Compute the value Z_{opt} of Z that minimizes $Ent(S(A > Z); S(A \leq Z))$, and consider the splitting of S into $S(A > Z_{opt})$ and $S(A \leq Z_{opt})$.

   By applying the algorithms of Fukuda et al. [FMMT96a], we can extend the above splitting (1) to the following, which is also considered in our decision tree subsystem of SONAR:

2. For an interval I, let $S(A \in I) = \{t \in S : t[A] \in I\}$ and $S(A \in \bar{I}) = \{t \in S : t[A] \notin I\}$. Compute the interval I_{opt} that minimizes $Ent(S(A \in I); S(A \in \bar{I}))$, and consider the associated splitting.

We call the above two kinds of splitting "one-dimensional rules" for short. In this paper, we consider *splittings with respect to grid regions*, which are sometimes called *region rules*.

We specify a number $N \leq \sqrt{n}$, and construct an (almost) equi-depth ordered bucketing of tuples for each numeric attribute A. That is, we construct buckets $B_1^A, .., B_N^A$ each of which contains approximately n/N tuples, satisfying $t[A] \leq t'[A]$ for every $t \in B_i^A, t' \in B_j^A$ and $i < j$. An efficient randomized algorithm for constructing such a bucketing can be found in Fukuda et al. [FMMT96a].

For a pair of numeric attributes A and A', we have a pixel grid G of size $N \times N$ generated as a Cartesian product of bucketings, such that for an (i, j)-th pixel $q(i, j)$, $t \in q(i, j)$ if and only if $t[A] \in B_i^A$ and $t[A'] \in B_j^{A'}$. We denote the pixel containing t as $q(t)$.

We consider a family $\mathcal{R}$ of grid regions of G. For each $R \in \mathcal{R}$, we consider a splitting S into $S(R) = \{t \in S : q(t) \in R\}$ and $S(\bar{R}) = \{t \in S : q(t) \in \bar{R}\}$, where $\bar{R} = G - R$ is the complement of R. Let R_{opt} be the region of $\mathcal{R}$ that minimizes the entropy of the splitting. The region R_{opt} and the associated splitting are called the *optimal region* and the *optimal splitting* (or *region rule*) with respect to $\mathcal{R}$ and the pair of attributes (A, A').

A grid region is called *based-monotone*, if it lies below an x-monotone curve. A grid region is called *admissible* if it is a connected region bounded by a pair of x-monotone grid curves. $\mathcal{R}(Base)$ and $\mathcal{R}(Admi)$ are the sets of all based-monotone and admissible regions of G, respectively.

In Section 3, we present efficient algorithms for computing the optimal splitting with respect to certain families of regions, including $\mathcal{R}(Admi)$ and $\mathcal{R}(Base)$, when the objective attribute W is Boolean.

The construction of a decision tree is top-down, starting from its root in breadth-first fashion. When a new internal node is created, the algorithm first computes all one-dimensional rules for singular attributes, and region rules for correlated pairs of attributes, together with rules associated with Boolean or categorical conditional attributes. Then it chooses the rule that minimizes the entropy. The decision made at the node is associated with the splitting.

2.3 Selecting correlated attributes

Even if A and A' are not strongly correlated, the region rule associated with the pair (A, A') is better with respect to the entropy value than one-dimensional rules on A and A'. However, it does not necessarily give a better system for users, since a region rule is more complicated than a one-dimensional rule. Indeed, some technique (for example, a visualization technique [FMMT96b]) is necessary to explain a region rule.

Hence, it is desirable that a region rule should only be considered for a pair of strongly correlated conditional attributes. We use the entropy value again to decide whether A and A' are strongly correlated.

For simplicity, we assume that $\mathcal{R}(Admi)$ is used as the family of regions. We compute R_{opt} for the pair (A, A') and its entropy value $Ent(S(R_{opt}); S(\overline{R_{opt}}))$. We also compute the optimum intervals I and I' to minimize the entropy of the splitting that corresponds to the rules $A(X) \in I$ and $A'(X) \in I'$, respectively.

We give a threshold $\alpha \geq 1$ to decide A and A' are strongly correlated if and only if

$$\frac{Ent(S) - Ent(S(R_{opt}); S(\overline{R_{opt}}))}{Ent(S) - min\{Ent(S(I); S(\bar{I})),\ Ent(S(I'); S(\bar{I'}))\}} > \alpha$$

The choice of the threshold α depends on the application.

3 Optimization of Splittings

3.1 Naive Hand-Probing Algorithm

From now on, we concentrate on the case in which the objective attribute W is Boolean, although our scheme can be extended to the case in which W is categorical. Therefore, the entropy function is written as

$$Ent(S) = -p \log p - (1-p) \log(1-p),$$

where p is the frequency with which the objective attribute has the value 1 (i.e. "yes") on the set of tuples.

We consider the problem of computing R_{opt} in several families of grid regions of $\mathcal{G}$. Note that it is very expensive to compute R_{opt} by examining all elements of $\mathcal{R}$, since the set $\mathcal{R}(Base)$, for example, has N^N different regions.

Let n_1 and n_2 be the numbers of tuples t of S satisfying $t[W] = 0$ and $t[W] = 1$, respectively. For a region R, let $x(R)$ and $y(R)$ be the number of tuples t located in the pixels in R that satisfy $t[W] = 0$ and $t[W] = 1$, respectively.

Consider the planar point set $P = \{\iota(R) = (x(R), y(R)) : R \in \mathcal{R}\}$, and its convex hull $conv(P)$. Since $x(R)$ and $y(R)$ are nonnegative integers which are at most n, P contains $O(n^2)$ points, and $conv(P)$ has at most $2n$ points on it. We define

$$E(x,y) = -\frac{f(x,y) + f(n_1 - x, n_2 - y)}{n},$$

using the function f defined in the previous section for $X = (x, y)$. Then, the entropy function $Ent(S(R); S(\bar{R}))$ of the splitting is $E(\iota(R)) = E(x(R), y(R))$.

Lemma 3.1 *$\iota(R_{opt})$ must be on $conv(P)$.*

Proof: From Lemma 2.1, $f(x, y)$ is convex, and hence $E(x, y)$ is a concave function. It is well known that the minimum of a concave function over P is taken at an extremal point (that is, a vertex of $conv(P)$). ∎

Hence, naively, it suffices to compute all the vertices of $conv(P)$ and their associated partition curves. Our problem now resemble to global optimization problems [PR90]. In global optimization, extremal points can be computed by using linear programming. However, we know neither the point set P nor the constraint inequalities defining the convex hull; hence we cannot use the linear programming approach in a straightforward manner.

Let $conv^+(P)$ (resp. $conv^-(P)$) be the upper (resp. lower) chain of $conv(P)$; Here, we consider the leftmost (resp. rightmost) vertex of $conv(P)$ belongs to the upper (resp. lower) chain.

Our algorithm is based on the use of what is known in computational geometry as "hand probing" to compute the vertices of a convex polygon [DEY86]. Hand probing is based on the *touching oracle*:

> " Given a slope θ, compute the tangent line with slope θ to the upper (resp. lower) chain of the convex polygon together with the tangent point $v^+(\theta)$ (resp. $v^-(\theta)$). If the slope coincides with the slope of an edge of the polygon, the left vertex of the edge is reported as the tangent point."

Lemma 3.2 *If a touching oracle is given in $O(T)$ time, all vertices of $conv(P)$ can be computed in $O(nT)$ time.*

Proof: We consider an interval $I = [I(left), I(right)]$ of the upper chain of $conv(P)$ between two vertices $I(left)$ and $I(right)$ (see Figure 4). We start with

$\theta = \infty$, find the leftmost vertex p_0 and the rightmost vertex p_1 of $conv(P)$, and set $I(left) = p_0$ and $I(right) = p_1$. Let $\theta(I)$ be the slope of the line through points $I(left)$ and $I(right)$. We perform the touching oracle and find $I(mid) = v^+(\theta_I)$. If $I(mid) = I(left)$, we report that I corresponds to an edge of $conv(P)$, and hence no other vertex exists there. Otherwise, we divide I into $[I(left), I(mid)]$ and $[I(mid), I(right)]$, and process each sub-interval recursively. We find either a new vertex or a new edge by executing the touching oracle in the algorithm. Hence, the time complexity is $O(|P|T)$, where $|P| \leq n$ is the number of vertices of P. ▮

Lemma 3.3 *For a given θ, the touching oracle to $conv(P)$ can be computed in $O(N^2)$ time, if $\mathcal{R} = \mathcal{R}(Admi)$. If preprocessing takes $O(N^2)$ time, it can be computed in $O(N)$ time for $\mathcal{R}(Base)$.*

Proof: It suffices to show how to compute $v^+(\theta)$, since $v^-(\theta)$ can be analogously computed. Let $v^+(\theta) = ((x(R_\theta), y(R_\theta))$, and let the tangent line be $y - \theta x = a$. Then, $y(R_\theta) - \theta x(R_\theta) = a$ and $y(R) - \theta x(R) \leq a$ for any $R \in \mathcal{R}$. Hence, R_θ is the region that maximizes $y(R) - \theta x(R)$. Let $g_{i,j}$ be the number of tuples in the (i,j)-th pixel of G, and let $h_{i,j}$ be the number of tuples satisfying $w(t) = 1$ in the (i,j)-pixel. We write $\Phi_{i,j}(\theta) = h_{i,j} - \theta g_{i,j}$. From our definition, $y(R) - \theta x(R) = \sum_{(i,j)\in R} \Phi_{i,j}(\theta)$.

If $\mathcal{R} = \mathcal{R}(Admi)$, R_θ is the *focused region* defined by Fukuda et al. [FMMT96a], and can be computed in $O(N^2)$ time by using dynamic programming and fast matrix searching (see [FMMT96a, ACKT96]).

Let us consider the case in which $\mathcal{R} = \mathcal{R}(Base)$. Since a based-monotone region R is the region below an x-monotone curve, the intersection of R and the j-th column of the grid G forms a half-column below some row index $top_R(j)$, that is, the set of pixels $(1,j), (2,j), ..., (top_R(j), j)$.

We consider the function $\Psi_{j,m}(\theta) = \sum_{i=1}^{m} \Phi_{i,j}(\theta)$, and the index $m_j(\theta)$, which is the value of m that maximizes $\Psi_{j,m}(\theta)$. Then, we can see that $top_{R_\theta}(j) = m_j(\theta)$; otherwise, we can replace the j-th column of R_θ by $(1,j), ..., (m_j(\theta), j)$ to improve the value of $y(R) - \theta x(R)$.

For each θ, it is easy to compute $m_j(\theta)$ in $O(N)$ time, and hence we can compute R_θ in $O(N^2)$ time.

Moreover, we can compute the piecewise linear function $\max_m \Psi_{j,m}(\theta)$ in $O(N)$ time, considering θ as a parameter. Using this function, we can query $m_j(\theta)$ in $O(\log N)$ time for a given θ. Hence, the time complexity of computing R_θ is $O(N \log N)$ if preprocessing takes $O(N^2)$ time. We can reduce the $O(N \log N)$ computing time to $O(N)$ by applying the fractional cascading data structure [CG86] (omitted in this version of the paper). ▮

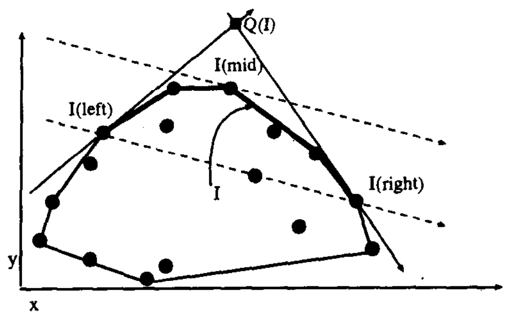


Figure 4: Hand Probe

We have the following similar results for the family of rectangles and the family of rectilinear convex regions, although the time complexity is increased (we omit the proof in this version of the paper).

Lemma 3.4 *The touching oracle to $conv(P)$ can be computed in $O(N^3)$ time, if $\mathcal{R}$ is either the family of all rectangle grid-regions, or the family of all rectilinear convex grid-regions of G.*

Combining Lemmas 3.1, 3.2, 3.3, and 3.4, we have the following theorem:

Theorem 3.1 *R_{opt} can be computed in $O(nN^2)$ time for $\mathcal{R}(Admi)$, $O(nN)$ time for $\mathcal{R}(Base)$, and $O(nN^3)$ time for the family of rectangles and that of rectilinear convex polygons.*

The above time complexity is the worst-case time complexity. In the next section, we further improve the practical time complexity by a factor of $O(n/\log n)$.

3.2 Guided Branch-and-Bound Search

The hand-probing algorithm computes all vertices on the convex hull. However, we only need to compute the vertex corresponding to R_{opt}. Hence, we can improve the performance by pruning unnecessary vertices efficiently. While running the hand-probing algorithm, we maintain the current minimum E_{min} of the entropy values corresponding to the vertices examined so far.

Suppose we have done hand probing with respect to θ_l and θ_r, and next consider the interval $I = [v^+(\theta_l), v^+(\theta_r)] = [I(left), I(right)]$ of $conv^+(P)$. Let $Q(I) = (x_{Q(I)}, y_{Q(I)})$ (see Figure 4) be the point of intersection of the tangent lines whose slopes are θ_l and θ_r. We compute the value $E(Q(I)) = E(x_{Q(I)}, y_{Q(I)})$. If the two tangent lines are parallel, we set $E(Q(I)) = -\infty$.

Lemma 3.5 *For any point $Q' = (x', y')$ inside the triangle $I(left)I(right)Q(I)$,*

$$E(x', y') \geq \min\{E(Q(I)), E_{min}\}.$$

Proof: Immediate from the concavity of $E(x, y)$. ▮

This lemma gives a lower bound for the values of E at the vertices between $I(left)$ and $I(right)$ in $conv^{+}(P)$. Hence, we have the following:

Corollary 3.1 *If $E(Q(I)) \geq E_{min}$, no vertex in the interval I of $conv^{+}(P)$ corresponds to a region whose associated entropy is less than E_{min}.*

On the basis of Corollary 3.1, we can find the optimal region R_{opt} effectively by running the hand-probing algorithm together with the branch-and-bound strategy guided by the values $E(Q(I))$. Indeed, the algorithm examines the subinterval with the minimum value of $E(Q(I))$ first. Moreover, during the process, subintervals satisfying $E(Q(I)) \geq E_{min}$ are pruned away.

We maintain the list $\{E(Q(I)) : I \in \mathcal{I}\}$, using a priority queue. Note that E_{min} is monotonically decreased, while Q_{min} is monotonically increased in the algorithm. Most of subintervals are expected to be pruned away during the execution, and the number of touching oracles in the algorithm is expected to be $O(\log n)$ in practical instances. We have implemented the algorithm as a subsystem of SONAR, and confirmed the expected performance by experiment (as described in Section 4).

Since the touching oracle needs $O(N^2)$ time for $\mathcal{R}(Admi)$, the algorithm MAIN runs experimentally in $O(N^2 \log n)$ time, which is $O(n \log n)$ because $N \leq \sqrt{n}$.

Although we have not yet done enough experiments on other families of regions, we expect that the algorithm behaves similarly, and runs in $O(N^2)$ time for $\mathcal{R}(Base)$, and in $O(N^3 \log n)$ time for the families of rectangles and rectilinear convex polygonal regions.

4 Experimental Results

This section presents detailed performance and efficiency evaluations of SONAR's decision tree function. All performance experiments were performed on IBM RS/6000 530 workstation with 128MB of main memory, running under the AIX 3.2.5 operating system.

4.1 Computing the Optimal Admissible Region

Table 1 shows performance of computing the region R_{opt} minimizing the entropy in $\mathcal{R}(Admi)$. In this experiments, we use an artificial data distributed in an

Size	Time(sec)	# Oracles	$\|conv(P)\|$
20^2	2.86	19	304
40^2	11.8	19	918
60^2	29.5	22	1714
80^2	62.5	26	2675
100^2	86.4	23	3878
120^2	142	26	5151
200^2	422	27	NA
400^2	1633	25	NA
600^2	4368	29	NA
800^2	8299	31	NA

Table 1: Performance for Computing Optimal Admissible Regions

$N \times N$ grid for $20 \leq N \leq 800$, which is the same as that of Fukuda et al [FMMT96b]. The size n of data is larger than N^2 (indeed, we set $n \approx N^4$ in order to create a large number of vertices on $conv(P)$.) The linear performance of a touching oracle with respect to $O(N^2)$ can be found in [FMMT96b], thus omitted in this paper.

The second column of Table 1 shows the CPU time, and the third column shows the number of the touching oracles in the guided branch-and-bound algorithm to find the optimal region. The fourth column shows the number of vertices on $conv(P)$, which is equal to the number of touching oracles done by the naive hand probing algorithm.

It is seen that the number of touching oracles increases very slowly, and the guided branch-and-bound algorithm is much advantageous. Figure 5 confirms that the CPU time follows our $O(N^2 \log n)$ estimation.

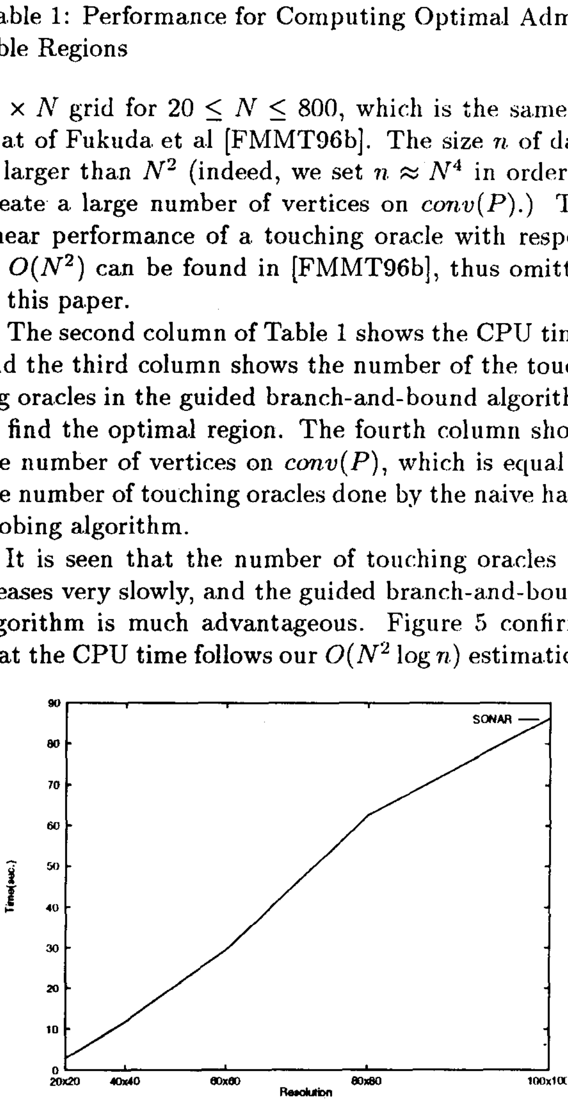


Figure 5: CPU time for Computing the Optimized Region

4.2 Tree Evaluations

Quality evaluation of the decision tree using region rules is presented in this subsection. We use a dataset

Type	# Node	Depth
Conventional Binary Splitting	267	16
Interval Splitting	159	15
Admissible Region Splitting	27	6

Table 2: Comparison of Splitting

in the STATLOG benchmark [MST84]. Since conventional methods cannot handle huge dataset effectively, we choose a relatively small dataset called "diabetes," containing records of female patients (original owners: National Institute of Diabetes and Digestive and Kidney Diseases).

It contains 768 tuples, 8 numerical attributes, and 2 classes representing "positive" and "negative" results for diabetes, respectively.

We constructed trees for the dataset by using three different decomposition methods: conventional binary splitting, as proposed in C4.5 [Q93]; SONAR's interval splitting $t(A) \in I$; and SONAR's admissible region splitting with respect to $\mathcal{R}(Admi)$, where N is set to be 16. We decompose trees repeatedly until all the data in leaf nodes are in the same class. Table 2 shows the number of nodes, and depths of the trees.

Figure 6, 7, and 8 show intermediate (as of depth 3) trees in each method. Each node of the trees is labeled with T (P N), where T is the number of tuples satisfying all the conditions on the path from the root to the node, and P and N respectively show the number of positive and negative tuples. For example, the root nodes contains 768 tuples in which 500 are positive and 268 are negative. At this level, the splittings decrease the entropy value by 0.198, 0.210, and 0.433 for binary splitting, interval splitting, and region splitting, respectively.

For binary splitting and interval splitting, the decision test associated with each node is explicitly written below the node in Figure 6 and 7. One disadvantage for the region split decision tree is that the description of each decision is not simple. In Figure 8, the pair of numeric attributes for the region splitting of each node is presented, but the region itself is not presented. We use a visualization function to describe such a region. Figure 9 is a graphical view of the region used in the root node of the Figure 8. It use two colors (red and blue), and the red (resp. blue) level indicate the number of positive (resp. negative) patient in each pixel. The data in the root node are partitioned based on whether the data is in the admissible region R_{opt} or not. In this example, R_{opt} (the near-triangle region) corresponds to the cluster of patients less likely to be positive for diabetes.

One more important observation is that, attributes such as PedigreeFunc and BP(blood pressure) only appears in Figure 9. For example, BP has strong correlation with Age, and is only found to be crucial by using region rules.

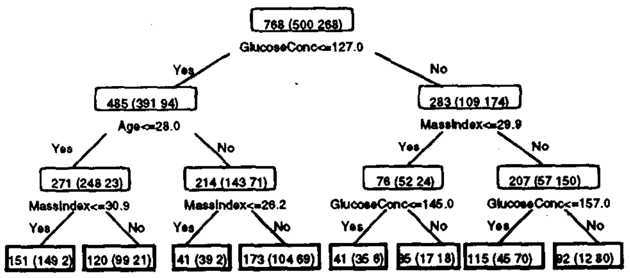


Figure 6: Tree obtained by Binary Splitting

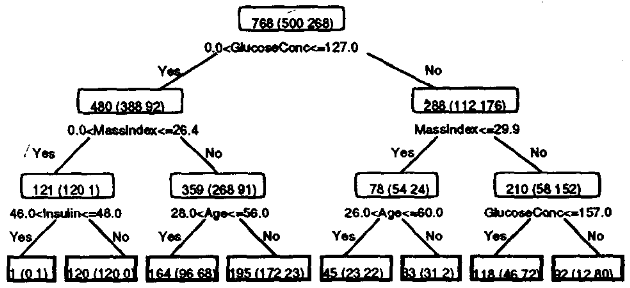


Figure 7: Tree obtained by Interval Splitting

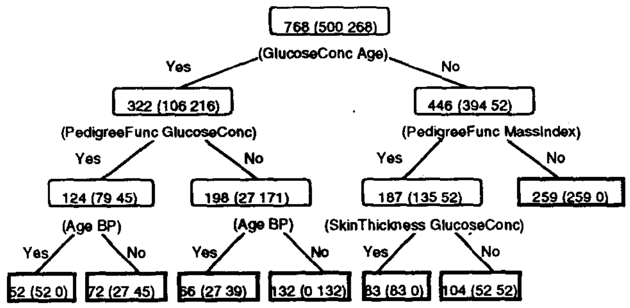
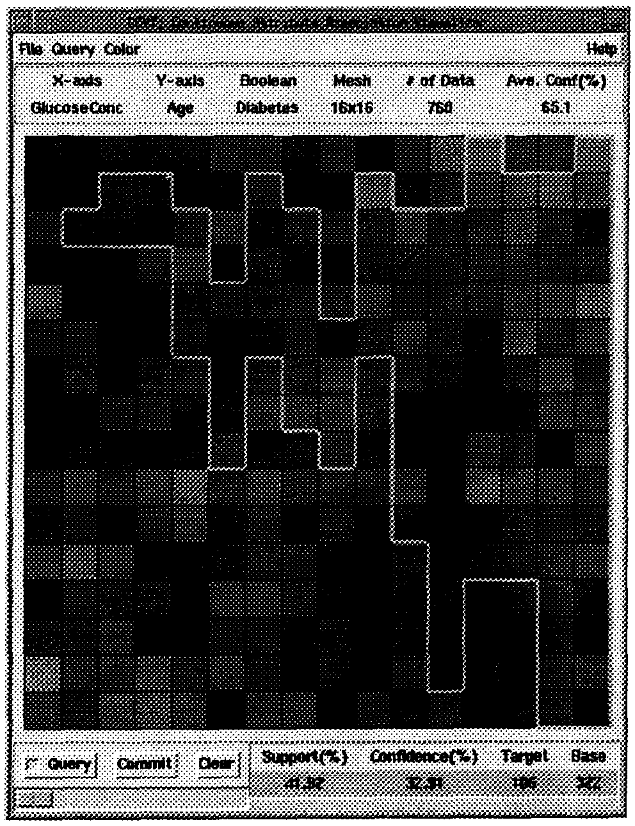


Figure 8: Tree obtained by Admissible Region Splitting

5 Generalizations

5.1 Categorical objective attribute

We can extend our scheme to a categorical objective attribute W with k categories $\{1, 2, .., k\}$, where k is a small integer. For a pair of numeric attributes A and A', and a family $\mathcal{R}$ of regions, we consider a splitting of S into $S(R)$ and $S(\bar{R})$, as before. The region R_{opt} is the region that minimizes the entropy $Ent(S(R); S(\bar{R}))$. The only difference is that it is harder to compute R_{opt} than in the Boolean objective case.



X-axis	Y-axis	Boolean	Mesh	# of Data	Ave. Conf(%)
GlucoseConc	Age	Diabetes	16x16	768	65.1

Support(%)	Confidence(%)	Target	Base
41.37	32.31	106	327

Figure 9: Admissible Region Splitting

For a region R, we define $\iota(R) = (x_1(R), ..., x_k(R))$, where $x_i(R)$ is the number of tuples t in $S(R)$ that satisfy $W(t) = i$. Let $P = \{\iota(R) : R \in \mathcal{R}\}$, and consider its convex hull $conv(P)$ in k-dimensional space.

Then, from Lemma 2.1, we can see that $\iota(R_{opt})$ is on $conv(P)$. Hence, it is enough to examine the vertices of $conv(P)$. We concentrate on the upper part $conv^+(P)$ of $conv(P)$, consisting of facets whose outer normal vector has a positive k-th coordinate, since the lower part $conv^-(P)$ can be treated analogously.

Consider a vector $\Theta = (\theta_1, \theta_2, .., \theta_{k-1}, 1)$, and the hyperplane $H(\Theta)$ tangent to $conv^+(P)$ defined by

$$x_k - \theta_1 x_1 - \theta_2 x_2 - ... - \theta_{k-1} x_{k-1} = c.$$

The *k-dim touching oracle* is to find the point of $H(\Theta)$ tangent to $conv^+(P)$. The touching point $\iota(R(\Theta))$ corresponds to the region $R(\Theta)$ that maximizes

$$F_\Theta(R) = x_k(R) - \theta_1 x_1(R) - ... - \theta_{k-1} x_{k-1}(R).$$

Lemma 5.1 *Given Θ, $R(\Theta)$ can be computed in $O(N^2)$ time for $\mathcal{R}(Admi)$, and in $O(N)$ time with $O(N^2)$ time preprocessing for $\mathcal{R}(Base)$. It can be computed in $O(N^3)$ time for families of rectangles and rectilinear convex regions.*

Proof: Once Θ is given, the value of $F_\Theta(i,j)$ at the (i,j)-th pixel of G can be precomputed. Thus, the region $R(\Theta)$ can be computed analogously to Lemmas 3.3 and 3.4. ▮

Hence, we can generalize the hand-probing algorithm. For simplicity, we give a brief explanation for the case $k = 3$. Suppose that we have used a 3-dimensional touching oracle for Θ, Θ', and Θ''. Next, we consider the plane through the associated three tangent points, and execute the touching oracle with respect to the slope of this plane. Then, we can find either a new vertex or a new face of $conv^+(P)$ by executing the touching oracle. We also compute the intersection point Q of three hyperplanes $H(\Theta)$, $H(\Theta')$, and $H(\Theta'')$, and use the value of the entropy function at Q as a lower bound of the entropy values within the simplex defined by Q and the three tangent points. Thus, we can design a hand-probing algorithm, and implement it by using a guided branch-and-bound strategy.

Naive hand probing requires $O(|P|)$ touching oracles, where $|P|$ is the complexity (total number of faces of all dimensions) of the polygon [DEY86], which is too expensive, since the number of vertices can be $O(n^{d-1})$, and $|P|$ is only bounded as $O(n^{(d-1)\lfloor d/2 \rfloor})$. However, we expect that the number of touching oracles can be reduced to $O(\log |P|)$ in practice if we use branch-and-bound. We have not yet done a detailed analysis carried any experiments to confirm this.

5.2 Rule associated with more than two attributes

We have considered regions in 2-dimensional plane as a way of splitting data so far. In this subsection, we discuss how to use regions in k-dimensional space when $k \geq 3$. Owing to the space limitation, we omit proofs of theorems in this subsection.

We consider a $N^k = N \times N \times ... \times N$ pixel grid G, associated with k numeric attributes A_1,...,A_k. For a (k-dimensional) pixel region R, we can consider a splitting of tuples inside and outside R. Consider a family $\mathcal{R}$ of (k-dimensional) pixel regions of G, and find R_{opt} whose associated splitting minimizes the entropy.

We denote the coordinate system of G by $(z_1, ..., z_k)$. A region is called a ***k*-dimensional based-monotone** region if it lies below a k-dimensional monotone surface. A k-dimensional monotone surface is a *lift up* of a $(k-1)$-dimensional based-monotone region in the grid associated with $z_1, ..., z_{k-1}$; that is, its projection to the hyperplane $z_k = 0$ is a $(k-1)$-dimensional based-monotone region, and the projection is one-to-one. Indeed, if $k = 2$, we can obtain a 2-dimensional monotone surface (curve) as a union of horizontal edges of an x-monotone curve.

Theorem 5.1 *If $\mathcal{R}$ is the family of all k-dimensional based-monotone regions, then R_{opt} can be computed in a worst-case time of $O(\min\{n^2, nN^{k-1}\})$.*

For $k = 3$, we consider another family. A three-dimensional region is an admissible terrain if it lies below a three-dimensional grid surface that is a lift-up of a two-dimensional admissible region.

Theorem 5.2 *If $\mathcal{R}$ is the family of all admissible terrains, then R_{opt} can be computed in a worst case time of $O(\min\{n^2, nN^3\})$.*

In practice, both time complexities can be improved by a factor of $n/\log n$.

6 Concluding Remarks

We have proposed an entropy based greedy construction of decision trees by using region rules. We have confirmed via experiment that the approach generates a small decision tree indeed. Although extensive experiments are necessary to evaluate its effectiveness in practice, we believe that there are many applications in which our decision tree is useful.

Compared to traditional methods, there is a trade-off between size of the tree and description of rules. This trade-off can be controlled by changing the bucket size N and the parameter value α introduced in Section 2.3. It is important to devise criteria for selecting suitable values of them in order to construct a practical automatic decision system.

We can also approximate an x-monotone curve bounding an optimal region by using a low-degree interpolation function. If the degree of the function is given, we can compute such a function by using (for example) method of least-squares. If the entropy value of the splitting associated with the approximate region is not much different from that of the optimal region, we can use the approximate region to make the decision.

References

[ACKT96] T. Asano, D. Chen, N. Katoh, and T. Tokuyama, Polynomial-Time Solutions to Image Segmentations. *Proc. 7th ACM-SIAM Symposium on Discrete Algorithms*, 104–113, 1996.

[AGL+92] R. Agrawal, S. Ghosh, T. Imielinski, B. Iyer, and A. Swami. An Interval Classifier for Database Mining Applications. *Proc. 18th VLDB Conference*, 560–573, 1992.

[ALS93] R. Agrawal, T. Imielinski and A. Swami, Database Mining: A Performance Perspective. *IEEE Transactions on Knowledge and Data Engineering*, 5-6: 914–925, 1993.

[AT94] T. Asano and T. Tokuyama, Partial Construction of an Arrangement of Lines and Its Application to Optimal Partitioning of Bichromatic Point Set. *IEICE Transactions* E-77-A: 595–600, 1994.

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

[CG86] B. Chazelle and L. Guibas, Fractional Cascading: A Data Structuring Technique. *Algorithmica* 1: 133–162, 1986.

[DE93] D. Dobkin and D. Eppstein, Computing the Discrepancy *Proc. 9th ACM Symposium on Computational Geometry,* 47–52, 1993.

[DEY86] D. Dobkin, H. Edelsbrunner, and C. Yap, Probing Convex Polytopes. *Proc. 18th ACM Symposium on Theory of Computing,* 387–392, 1986.

[FMMT96a] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, Mining Optimized Association Rules for Numeric Attributes. *Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database System,* 182–191, 1996.

[FMMT96b] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, Data Mining using Two-dimensional Optimized Association Rules: Scheme, Algorithms, and Visualization. *Proceedings of the ACM SIGMOD Conference on Management of Data,* 13–23, 1996.

[FMMT96c] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, SONAR: System for Optimized Numeric Association Rules. *Proceedings of the ACM SIGMOD Conference on Management of Data,* p.553, 1996.

[GJ79] M. R. Garey and D. S. Johnson, *Computer and Intractability. A Guide to NP-Completeness.* W. H. Freeman, 1979.

[HR76] L. Hyafil and R. L. Rivest, Constructing Optimal Binary Decision Trees is NP-Complete. *Information Processing Letter,* 5: 15–17, 1976.

[MAR96] M. Mehta, R. Agrawal, and J. Rissanen, SLIQ: A Fast Scalable Classifier for Data Mining. *Proceedings of the Fifth International Conference on Extending Database Technology,* 1996.

[MST84] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, *Machine Learning, Neural, and Statistical Classification.* Ellis Horwood, 1984.

[PR90] P. M. Pardalos and J. B. Rosen (ed.), *Annals of Operations Research 25, Computational Methods in Global Optimization* J. C. Baltzer AG, 1990.

[Q86] J. R. Quinlan, Induction of Decision Trees, *Machine Learning,* 1: 81-106, 1986.

[Q93] J. R. Quinlan, *C4.5: Programs for Machine Learning* Morgan Kaufmann, 1993.

[QR89] J. R. Quinlan and R. L. Rivest, Inferring Decision Trees Using Minimum Description Length Principle. *Information and Computation,* 80: 227–248, 1989.