Machine Learning, 8, 87–102 (1992)

Technical Note
On the Handling of Continuous-Valued Attributes in Decision Tree Generation

USAMA M. FAYYAD* FAYYAD@AI-CYCLOPS.JPL.NASA.GOV
KEKI B. IRANI IRANI@CAEN.ENGIN.UMICH.EDU
Artificial Intelligence Laboratory, Electrical Engineering and Computer Science Department, The University of Michigan, Ann Arbor, MI 48109-2110

Editor: Dennis Kibler

Abstract. We present a result applicable to classification learning algorithms that generate decision trees or rules using the information entropy minimization heuristic for discretizing continuous-valued attributes. The result serves to give a better understanding of the entropy measure, to point out that the behavior of the information entropy heuristic possesses desirable properties that justify its usage in a formal sense, and to improve the efficiency of evaluating continuous-valued attributes for cut value selection. Along with the formal proof, we present empirical results that demonstrate the theoretically expected reduction in evaluation effort for training data sets from real-world domains.

Keywords. Induction, empirical concept learning, decision trees, information entropy minimization, discretization, classification

1. Introduction

Empirical learning from examples is receiving considerable attention in terms of research and industrial applications. Programs that learn from pre-classified examples aim at circumventing the knowledge acquisition bottleneck in the development of expert systems. The problem is due to the fact that human experts find it difficult to express their (intuitive) knowledge of a domain in terms of concise, correct situation-action rules. Empirical learning algorithms attempt to discover relations between situations expressed in terms of a set of attributes and actions encoded in terms of a fixed set of classes. By examining large sets of pre-classified data, it is hoped that a learning program may discover the proper conditions under which each action (class) is appropriate.

Learning algorithms typically use heuristics to guide their search through the large space of possible relations between combinations of attribute values and classes. A powerful and popular such heuristic uses the notion of selecting attributes that locally minimize the information entropy of the classes in a data set. This heuristic is used in the ID3 algorithm (Quinlan, 1986) and its extensions, e.g., GID3 (Cheng, et al., 1988), GID3* (Fayyad, 1991) and C4 (Quinlan, 1990), in CART (Breiman, et al., 1984), in CN2 (Clark & Niblett, 1989) and others; see (Lewis, 1968) and (Fayyad, 1991) for a general discussion of the attribute selection problem.

**Present address:* AI Group, M/S 525-3660, Jet Propulsion Lab, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109.

The attributes in a learning problem may be nominal (categorical), or they may be continuous (numerical). The term "continuous" is used in the literature to indicate both real and integer valued attributes. The above mentioned attribute selection process assumes that all attributes are nominal. Continuous-valued attributes must, therefore, be *discretized* prior to attribute selection. This involves choosing a particular discretization among several possible discretizations. In this paper, we focus only on the discretization of continuous-valued attributes. We derive a result about the information entropy minimization heuristic used in discretizing continuous-valued attributes that gives us:

- a better understanding of the heuristic and its behavior,
- formal evidence that supports the usage of the heuristic in this context, and
- a gain in computational efficiency that results in speeding up the evaluation process for continuous-valued attribute discretization.

In Section 2 we discuss the evaluation of continuous-valued attributes, and in Section 3 we motivate the need for further study of this step. Section 4 presents the main result along with its proof. Finally, in Section 5 we empirically evaluate the theoretically predicted speedup on real-world data sets.

2. The discretization criterion for continuous-valued attributes

Concept learning, pattern analysis, and decision tree generation algorithms in general should be capable of handling continuous-valued attributes. A continuous-valued attribute takes on numerical values (integer or real). In general, it is an attribute that has a linearly ordered range of values. A continuous-valued attribute is typically handled by partitioning its range into subranges, i.e., a test is devised that quantizes the continuous range. The discretization should be chosen so as to provide useful classification information with respect to the classes to which the examples in the attribute's range belong. In general, a discretization is simply a *logical* condition, in terms of one or more continuous-valued attributes, that serves to partition the data into at least two subsets.

A continuous-valued attribute is typically discretized during decision tree generation by partitioning its range into two intervals. A threshold value, T, for the continuous-valued attribute A is determined, and the test $A \leq T$ is assigned to the left branch while $A > T$ is assigned to the right branch.[1] We call such a threshold value, T, a *cut point*. This method for selecting a cut point is used in the ID3 (Quinlan, 1986) algorithm and its variants such as GID3 (Cheng, et al., 1988), in the CART algorithm (Breiman, et al., 1984), and others (Gelfand, et al., 1991). This discretization algorithm could be used in general in any algorithm for learning classification trees or rules that handles continuous-valued attributes by quantizing their ranges into intervals. Thresholding against a cut value is not the only way to discretize attributes. One may, for example, consider linear combinations of several attributes and compare the result against a threshold (Breiman, et al., 1984). It may also be possible to avoid thresholding by forming a condition that compares the values of two or more attributes directly, e.g., IF $A < B$, where A and B are two attributes. However, the huge number of such possible expressions makes the space too large to search. Furthermore, most work in the literature adopts the simple scheme of selecting a cut value. We therefore focus on this scheme and study one particular criterion for cut value selection.

Although the result we derive is applicable to continuous-valued attribute evaluation in general, we discuss it in the particular context of top-down decision tree generation (as in the ID3 algorithm). First we describe the use of the entropy minimization heuristic for the selection of an attribute and a cut point for partitioning its range into two regions.

Assume we are to select an attribute for branching at a node having a set S of N examples. For each continuous-valued attribute A we select the "best" cut point T_A from its range of values by evaluating *every candidate cut point* in the range of values. The examples are first sorted by increasing value of the attribute A, and the midpoint between each successive pair of examples in the sorted sequence is evaluated as a potential cut point. Thus, for each continuous-valued attribute, $N - 1$ evaluations will take place (assuming that examples do not have identical attribute values). For each evaluation of a candidate cut point T, the data are partitioned into two sets and the class entropy of the resulting partition is computed. The set S of examples is partitioned into the subsets S_1 and S_2. Let there be k classes $C_1, \ldots, C_k$. Let $P(C_i, S)$ be the proportion of examples in S that have class C_i. The *class entropy* of a subset S is defined as:

$$\text{Ent}(S) = -\sum_{i=1}^{k} P(C_i, S) \log(P(C_i, S))$$

where the logarithm may be to any convenient base. When the base is 2, Ent(S) measures the amount of information needed, in *bits*, to specify the classes in S. This measure gives the degree of "randomness" that the classes appear to exhibit in a set S. The smaller this number, the less even is the class distribution. To evaluate the resulting class entropy after a set S is partitioned into two sets S_1 and S_2, we take the weighted average of their resulting class entropies:

Definition 1: For a set S of examples, an attribute A, and a cut value T. Let $S_1 \subset S$ be the subset of examples in S with A-values not exceeding T and $S_2 = S - S_1$. The *class information entropy of the partition induced by T*, denoted by $E(A, T; S)$, is defined as

$$E(A, T; S) = \frac{|S_1|}{N}\text{Ent}(S_1) + \frac{|S_2|}{N}\text{Ent}(S_2) \tag{1}$$

where $N = |S|$ is the number of examples in the set S.

The cut point T_A for which $E(A, T_A; S)$ is minimal amongst all the candidate cut points is taken as the best cut point. This determines a binary discretization for attribute A.

After all continuous-valued attributes have been discretized, an attribute should be selected for branching out of the node. In algorithms that use information entropy minimization for attribute selection, e.g., (Quinlan, 1986), the attribute A_j, for which $E(A_j, T_A; S)$ is minimum, is the selected attribute among the (now discretized) continuous-valued attributes. When, in turn, attributes are to be selected for partitioning the child nodes, the discretization process must be performed again to rederive a new quantization based on each child node's own examples.

3. Discussion of the cut point selection criterion

It has been demonstrated empirically that this selection criterion constitutes a powerful heuristic when used to guide the search for a good decision tree to classify a set of training examples (Quinlan, 1986). It does not guarantee optimal trees but it is a good method for determining which attributes are relevant to the classification task at hand.

One of the main problems with this selection criterion is that it is relatively expensive. Although it is polynomial in complexity, it must be evaluated $N - 1$ times for each attribute (assuming that the N examples have distinct values). Machine learning programs are designed to work with large sets of training data, so N is typically very large. In the case of nominal (or discretized) attributes, this criterion is not expensive since each attribute needs a single evalution of an r-partition, where r is the number of values of the nominal attribute. Typically, $r \ll N$. Indeed, experience with ID3-like algorithms confirms that they run significantly slower when continuous attributes are present.

The other objection that may be raised is that the algorithm has an inherent weakness in it that will cause it to produce "bad" cut points especially when there are more than two classes in the problem. This objection is based on the fact that the algorithm attempts to minimize the weighted average entropy of the two sets in the candidate binary partition (as shown in Equation 1 above). The cut point may therefore separate examples of one class in an attempt to minimize the average entropy. Figure 1 illustrates this situation. Instead of falling on one of the boundaries B1 or B2, the cut point may fall in between so that the average entropy of both sides is minimized.

However, neither of these objections turns out to be true. We prove below that regardless of how many classes there are, and how they are distributed, the cut point *will always occur on the boundary between two classes* (see Definition 2 for a precise statement of what we mean by a boundary point). This is indeed a desirable property of the heuristic since it shows that the heuristic is "well behaved" in terms of the cut points it favors. It tells us that this heuristic will never select a cut that is considered "bad" from the teleological point of view. In addition, this result will also help us improve the efficiency of the algorithm without changing its function at all. Since the cut point must occur on a boundary, we only need to evaluate boundary points between classes instead of evaluating possibly all $N - 1$ candidate cut points. If there are k classes, then in the best case we only need to

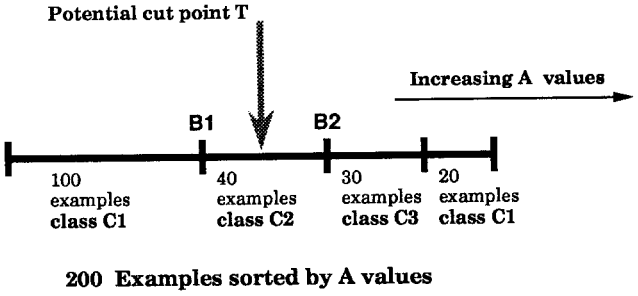

Figure 1. A potential cut point that separates examples from the same class.

evaluate $k - 1$ points. This occurs when the sorted sequence of examples has all examples of the same class adjacent to each other. For the example shown in Figure 1, we only need to evaluate three candidates rather than 189. Of course, in the degenerate case where the sorted sequence of examples forms a sequence in which the class changes from one example to the next, all $N - 1$ points need to be evaluated if all the examples have distinct values for the attribute. However, in practice, the occurrence of this event for all attributes simultaneously is not likely. Savings in computation will most likely still be achieved. We verify this claim empirically in Section 5 by measuring it for a few real-world data sets.

A simple analysis of a "bad" situation can help clarify our intuition about the expected savings in evaluation effort. In a "bad" scenario, all classes will be uniformly distributed, and the values of the attribute will have *no correlation* with the classes. Thus when we place the examples in a sorted sequence of increasing values of the attribute in question[2], the class labels of the sequence may be assumed to be labeled randomly according to a uniform distribution (all classes equally likely). Let $b(n)$ be the *expected number* of boundary points that need to be evaluated in a set of n examples with k classes.

$$b(1) = 0$$

$$b(2) = \frac{k - 1}{k}$$

Now assume that we have $n + 1$ examples with an expected number of boundary points $b(n + 1)$. Take the first n examples in the sorted sequence. We expect to have $b(n)$ boundaries among the first n. Now, the probability that the class of the $n + 1$th example differs from the nth's is $(k - 1)/k$. So the expected number of boundaries is

$$b(n + 1) = b(n) + \left(\frac{k - 1}{k}\right) \cdot 1$$

$$= b(n) + \frac{k - 1}{k}$$

Unfolding the recurrence gives us that

$$b(n) = (n - 1)\frac{(k - 1)}{k}. \tag{2}$$

In a typical training set, we expect $n \gg k$. For example, let us assume we have 20 examples of each class, then $n \approx 20k$. Thus $(n - 1) \approx 20k$. We therefore have:

$$b(n) \approx 20k \cdot \frac{(k - 1)}{k}$$

$$\approx 20(k - 1)$$

Therefore, in this undesirable scenario, when the attribute values show no correlation whatsoever with the classes, we expect to save approximately $20m$ evaluations per node, where m is the number of continuous attributes in the problem. The savings grow as the correlation between classes and attribute values increases. It also grows if the classes are not uniformly distributed—they rarely are.

Another way to view this situation, without necessarily assuming that classes are equally likely is to ask the following question: What is the probability that two consecutive examples in the sequence have the same class? We compute the lower bound of this probability for the "worst case" when there is no correlation between the attribute values and the examples. Assume we have C_i examples of each class, $i = 1 \ldots, k$. Assuming no correlation between values and classes, the probability that two consecutive examples have the same class is approximately

$$\sum_{i=1}^{k} \left(\frac{C_i}{n}\right)^2$$

Note that if the classes are equally likely, $C_i/N \approx 1/k$. Thus

$$\text{Prob}\{\text{same class}\} \geq \sum_{i=1}^{k} \left(\frac{1}{k}\right)^2$$

$$\approx \frac{1}{k}$$

and increases as correlation between attribute values and classes increases.

We shall focus on the problem of discretizing continuous-valued attributes into two intervals. Later in the paper, we briefly mention the generalization of the algorithm to produce two or more intervals. We now turn our attention to modeling the problem and proving the aforementioned result about the discretization criterion.

4. Cut points are always on boundaries

In this section we prove that the value T_A for attribute A that minimizes the average class entropy $E(A, T_A; S)$ for a training set S must always be a value between two examples of different classes in the sequence of sorted examples. We must be careful in defining what we mean by a boundary point between two classes since in the sorted sequence, some examples of different classes may have identical values for the attribute.

Let S be a training set and A be a continuous-valued attribute. Sort the examples in S by increasing value of attribute A. Let T be some candidate cut point in the range of A, that is T is a value between two consecutive examples in the sequence that have different values for A. We introduce the notation $A(e)$ to denote the A-value of an example $e \in S$.

Definition 2: A value T in the range of the attribute A is a *boundary point* iff in the sequence of examples sorted by the value of A, there exist two examples e_1, $e_2 \in S$, having different classes, such that $A(e_1) < T < A(e_2)$; and there exists no other example $e' \in S$ such that $A(e_1) < A(e') < A(e_2)$.

In other words, T is a boundary point if it falls between two consecutive examples that do not belong to the same class. In the special case when a group of two or more examples have the *same value* for attribute A but belong to more than one class, then the cut points on either side of this group of examples are also boundary points. When the attribute is integer-valued rather than real-valued, we are more likely to encounter the special case of repeated attribute values. In such a situation, the examples with identical values cannot be separated from each other using the attribute in question. However, the cut points on either side of the group must be treated as boundary points.

Theorem 1. *If T minimizes the measure $E(A, T; S)$, then T is a boundary point.*

Proof: We shall proceed to show that the minimizing T cannot occur within a group of adjacent examples all of which have the same class.

In general, assume that T occurs somewhere within a sequence of n_j examples of the same class, where $n_j \geq 2$. Without loss of generality, assume that this class is C_k, where k is the number of classes (this will simplify the proof later).

Assume that n_c examples in this sequence of n_j examples of class C_k have an A-value less than T, $0 \leq n_c \leq n_j$.

Figure 2 illustrates the situation. Our sequence consists of n_j examples that have values greater than T_1 and less than T_2 where T_1 and T_2 are boundary points according to Definition 2.

We will show that $E(A, T; S)$ is minimized at $n_c = 0$ or at $n_c = n_j$ thus forcing T to coincide with one of the boundary points T_1 or T_2.

Let there be L examples in S with A-values $< T_1$, and R examples in S with A-values $> T_2$, where $0 \leq L, R \leq N - n_j$. Note that $n_j + L + R = N$ by definition (see Figure 2).

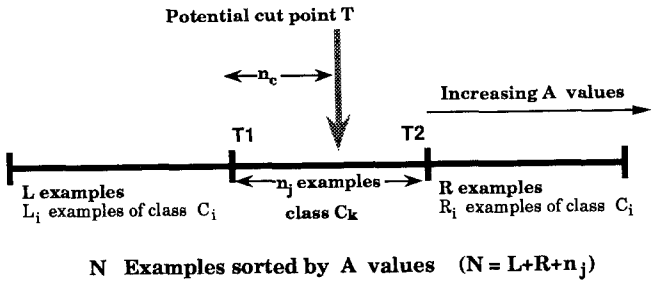

Figure 2. Modeling a possible partition at $A = T$.

For the L examples on the left, let L_i examples be of class C_i, $i = 1, \ldots, k$. Similarly let R_i examples of the R examples to the right be of class C_i, $i = 1, \ldots, k$. Note that $0 \leq R_i \leq R$, $0 \leq L_i \leq L$,

$$\sum_{i=1}^{k} L_i = L \quad \text{and} \quad \sum_{i=1}^{k} R_i = R$$

We need to show that $E(A, T; S)$ is minimized at $n_c = 0$ or at $n_c = n_j$ thus making T a boundary point.

To simplify the notation, let $n_{jc} = n_j - n_c$. From Equation 1:

$$E(A, T; S) = \frac{|S_1|}{N}\text{Ent}(S_1) + \frac{|S_2|}{N}\text{Ent}(S_2)$$

$$= -\frac{L + n_c}{N}\left[\sum_{i=1}^{k-1}\frac{L_i}{L + n_c}\log\left(\frac{L_i}{L + n_c}\right) + \frac{n_c + L_k}{L + n_c}\log\left(\frac{n_c + L_k}{L + n_c}\right)\right]$$

$$-\frac{R + n_{jc}}{N}\left[\sum_{i=1}^{k-1}\frac{R_i}{R + n_{jc}}\log\left(\frac{R_i}{R + n_{jc}}\right) + \frac{R_k + n_{jc}}{R + n_{jc}}\log\left(\frac{R_k + n_{jc}}{R + n_{jc}}\right)\right]$$

$$E(A, T; S) = -\frac{1}{N}\left[\sum_{i=1}^{k-1}(L_i\log(L_i) - L_i\log(L + n_c)) + (n_c + L_k)\log(n_c + L_k) - \right.$$

$$(n_c + L_k)\log(L + n_c) + \sum_{i=1}^{k-1}(R_i\log(R_i) - R_i\log(R + n_{jc})) +$$

$$\left.(R_k + n_{jc})\log(R_k + n_{jc}) - (R_k + n_{jc})\log(R + n_{jc})\right]$$

Now let $I(x) = x\log(x)$, perform the substitution, multiply both sides by N, and group some terms into the summations.

$$N \cdot E(A, T; S) = -\sum_{i=1}^{k-1} I(L_i) + \sum_{i=1}^{k} L_i\log(L + n_c) - I(n_c + L_k) + n_c\log(L + n_c)$$

$$-\sum_{i=1}^{k-1} I(R_i) + \sum_{i=1}^{k} R_i\log(R + n_{jc}) - I(R_k + n_{jc}) + n_{jc}\log(R + n_{jc})$$

Note that

$$\sum_{i=1}^{k} L_i \log(L + n_c) = L \log(L + n_c)$$

and similarly for R_i. Thus,

$$N \cdot E(A, T; S) = -\sum_{i=1}^{k-1} I(L_i) + L \log(L + n_c) - I(n_c + L_k) + n_c \log(L + n_c)$$

$$-\sum_{i=1}^{k-1} I(R_i) + R \log(R + n_{jc}) - I(R_k + n_{jc}) + n_{jc} \log(R + n_{jc}) \tag{3}$$

We now take the first and second derivatives of this expression with respect to n_c. Note that $d/dx\ I(a + x) = \log(a + x) + 1$ and $d/dx\ I(a - x) = -\log(a - x) - 1$. Also, recall that $n_{jc} = n_j - n_c$.

$$\frac{d}{dn_c}(N \cdot E(A, T; S)) = \frac{L}{L + n_c} - \log(n_c + L_k) - 1 + \frac{n_c}{L + n_c} + \log(L + n_c)$$

$$-\frac{R}{R + n_j - n_c} - \log(R + n_j - n_c) - \frac{n_j - n_c}{R + n_j - n_c} + \log(R_k + n_j - n_c) + 1$$

$$= \log(L + n_c) - \log(L_k + n_c) + \log(R_k + n_j - n_c) - \log(R + n_j - n_c)$$

Taking the second derivative with respect to n_c gives:

$$\frac{d^2}{dn_c^2} E(A, T; S) = \frac{1}{N}\left[\frac{1}{L + n_c} - \frac{1}{L_k + n_c} - \frac{1}{R_k + n_j - n_c} + \frac{1}{R + n_j - n_c}\right]$$

However,

$$\frac{1}{L - n_c} < \frac{1}{L_k + n_c} \text{ since } L_k < L \text{ by definition, and}$$

$$\frac{1}{R + n_j - n_c} < \frac{1}{R_k + n_j - n_c} \text{ since } R_k < R \text{ by definition.}$$

We therefore conclude that

$$\frac{d^2}{dn_c^2} E(A, T; S) < 0 \text{ for all } n_c \text{ in the range } 0 \leq n_c \leq n_j$$

This tells us that in the range $0 \leq n_c \leq n_j$, $E(A, T; S)$ is *convex downwards*. Hence its minimum must be at one of the extremes of the interval (Luenberger, 1973).

The minimum value of $E(A, T; S)$ must therefore occur at one of the two boundaries $n_c = 0$ or $n_c = n_j$.

This proves that T must be a boundary point. □

Note that even though the numbers in the model of Figure 2, including n_c are integers, it is consistent to treat them as continuous and perform the differentiation. Fortunately the final result indicates that $n_c = 0$ or $n_c = n_j$ and these are both integers; thus our assumption is consistent with the situation.

Corollary 1. *The algorithm used by ID3 for finding a binary partition for a continuous attribute will always partition the data on a boundary point in the sequence of the examples ordered by the value of that attribute.*

Proof. Follows immediately from Theorem 1 and definitions. □

4.1. Support for the entropy minimization heuristic

The first implication of Corollary 1 is that it serves to support the usage of the entropy minimization heuristic in the context of discretization. We use the information entropy heuristic because we know, intuitively, that it possesses some of the properties that a discrimination measure should, in principle, possess. However, that in itself does not rule out possibly undesirable situations, such as that depicted in Figure 1. The Corollary states that "obviously bad" cuts are never favored by the heuristic. This result serves as further formal support for using the heuristic in the context of discretization, since it tells us that the heuristic is well-behaved from the teleological point of view. In other words, it possesses one more desirable property: it implicitly excludes bad cuts of the type shown in Figure 1.

Corollary 1 also provides support for extending the algorithm to extract multiple intervals, rather than just two, in a single discretization pass. The motivation for doing this is that "better" trees are obtained[3] (Fayyad & Irani, 1991); see (Fayyad, 1991) for details. The training set is sorted once, then the algorithm is applied recursively, always selecting the best cut point. A criterion is applied to decide when to refrain from applying further binary partitioning to a given interval. The details of the stopping criterion are not within the scope of this paper. The fact that only boundary points are considered makes the top-down derivation of intervals feasible (since the algorithm never commits to a "bad" cut at the top) and reduces the computational effort as described below.

4.2. Improving the efficiency of the algorithm

Besides showing that the information entropy minimization selection criterion behaves well and can never favor a "bad" cut point, Corollary 1 can be used to increase the efficiency of the algorithm without changing its effects at all. After sorting the examples by the value of the attribute A, the algorithm need only examine the b boundary points rather than all $N - 1$ candidates. Note that

$$k - 1 \leq b \leq N - 1$$

Since typically $k \ll N$ we expect significant computational savings to result in general. We demonstrate speedups in terms of the number of potential cut points evaluated in the next section.

As a sidelight, we present a corollary of the proof of Theorem 1 that allows us to optimize the algorithm further. Consider the new algorithm that only evaluates boundary points. It computes the entropy of one boundary point then jumps to the nearest neighboring boundary point and evaluates its entropy. Assume that the algorithm just evaluated $E(A, T_1; S)$ and that the next range of examples consists of n_j examples of class C_j with the boundary cut point value being T_2.

Corollary 2. *If the entropy at a boundary cut point T_1 is $E(A, T_1; S)$, and the next boundary cut point is at T_2, separated by n_j examples of class C_j from T_1, then*

$$E(A, T_2; S) = E(A, T_1; S) + \frac{F(n_j, L, R, L_j, R_j)}{N} \tag{4}$$

where $F(n_j, L, R, L_j, R_j) =$
$[I(R) - I(L) + I(L_j) + I(L + n_j) - I(L_j + n_j) - I(R_j) - I(R + n_j) + I(R_j + n_j)]$

Proof. Follows directly from the expression 3 in the proof of Theorem 1, which can be simplified to

$$E(A, T; S) = \frac{1}{N}\left[-\sum_{i=1}^{k-1} I(L_i) + I(L + n_c) - I(\mathrm{n_c} + L_j) \right.$$
$$\left. - \sum_{i=1}^{k-1} I(R_i) + I(R + n_j - n_c) - I(R_j + n_j - n_c) \right]$$

by setting

$$F(n_j, L, R, L_j, R_j) = N \cdot [E(A, T_2; S) - E(A, T_1; S)]$$

□

We therefore have an efficient incremental procedure for evaluating the boundaries without dealing with partitioning the data. Note that the summations no longer need to be evaluated.

Thus, rather than evaluating $2k$ divide-log-multiply-add floating point operations for each candidate cut point, we only need to evaluate a constant number (eight) of log-multiply-add. This is advantageous when there are more then 3 classes in the data.

The algorithm for evaluating the boundary points starts by setting $L = 0$, $R = N$, $L_i = 0$, and $R_i = |C_i|$, $1 \leq i \leq k$, where $|C_i|$ is the number of examples in S that have class C_i. Each time, the next n_j examples of class C_j are traversed. The values of L, R, L_j, and R_j are updated, and the new entropy difference is evaluated according to equation (4) above.

5. Empirical evaluation

In this Section we describe results obtained by running ID3 on several data sets with continuous attributes to get an estimate of the amount of savings achieved for real-world data sets. We measure speedup in terms of the number of evaluations of potential cut points. The speedup ratio is defined to be the number of potential evaluations performed by the old algorithm divided by the number of evaluations performed by the new algorithm that utilizes Corollary 1. Experiments with randomly generated artificial data sets resulted in speedups that ranged from 2 to 30 times. However, speedup on real-world data is of course more important. We collected reasonably sized data sets that are used in real-world applications or that have been reported in the machine learning literature. We restrict attention to medium sized data sets (100 to 300 examples) since these are the sizes for typical applications.

We need to point out two important facts regarding the results reported in this paper.

1. Our speedup metric measures speedup in the attribute discretization evaluation effort and does not measure the overhead associated with sorting the examples by increasing values for each attribute, generating new nodes in the decision tree, partitioning examples, and selecting an attribute once all attributes have been discretized.
2. The results reported are for the ID3 algorithm. ID3 partitions the range of a continuous-valued attribute into two intervals. Algorithms that extract multiple intervals using a generalization of this procedure achieve higher speedups (see (Fayyad, 1991) for details). Algorithms that search for rules rather than decision trees also spend more effort on discretization.

The generalized algorithm in (Fayyad, 1991), described briefly in Section 4.1, extracts possibly more than two intervals for a given attribute at any evaluation pass. The training data are sorted once, then the algorithm is applied recursively; always selecting the best cut point. Thus, the time spent on evaluating cut points increases significantly while other overhead, such as sorting, remains unchanged. In this case, the results derived in this paper play an even bigger role in speeding up the run time of the algorithm.

To evaluate speedup, we used eight data sets that have a majority of continuous attributes. The data sets are described in Table 1. Some of these were obtained from the U.C. Irvine Machine Learning Repository and others came from our own industrial applications of machine learning described in (Irani, et al., 1990).

Table 1. Details of the data sets used

Data Set	Name	Examples	Attributes	Classes
Faulty operation data from the Jet Propulsion Laboratory Deep Space Network antenna controller	DSN	258	12	5
Problems in a reactive ion etching process (RIE) in semiconductor manufacturing from Hughes Aircraft Company	SRC1	94	8	4
The waveform domain described in (Breiman et al., 1984)	WVFRM	150	21	3
Data obtained from a response surface of multiple response variables in a set of wafer etching experiments conducted at Hughes	RSM1	300	3	35
Same as RSM1 except that classes are mapped to only two values: "good" and "bad"	RSM2	300	3	2
Publicly available heart disease medical data from an institute in Cleveland	HEART	303	13	2
The glass types data from the USA Forensic Science Service	GLASS	214	9	6
The famous iris classification data used by R.A. Fisher (1936)	IRIS	150	4	3
The echocardiogram data of heart diseases from the Reed Institute of Miami	ECG	132	9	2

The data sets represent a mixture of characteristics ranging from few classes with many attributes to many classes with few attributes. We intuitively expected smaller speedups for data sets with a large number of classes. This is generally the case. The RSM1 and RSM2 entries, for example, show that the speedup increases dramatically when a data set with many classes is transformed to a two-class problem. The larger the number of classes, the higher the lower bound on the number of boundaries to be evaluated. The simplified analysis of Section 3 resulting in Equation 2 also predicts this trend as the number of classes grows. Of course, the actual number of boundaries is dependent on the distribution of the values of a particular attribute over the classes. The evaluation effort measure is the total number of candidate cut points evaluated during the generation of the decision tree. The graph on the left in Figure 3 shows the actual speedup in evaluation in terms of cut point evaluations performed by ID3 to the necessary subset of evaluations (labeled ID3-S) determined by Corollary 1. The graph on the right shows the corresponding speedup ratios

$$\frac{\text{number of cut values evaluated by ID3}}{\text{number of necessary evaluations}}.$$

Note that the results shown in Figure 3 are only applicable to the evaluation of continuous-valued attributes. The presence of nominal attributes in the data would slow down the observed speedup in terms of total time to generate the decision tree. As mentioned earlier, our measure of speedup focuses only on reduction in number of evaluations and ignores

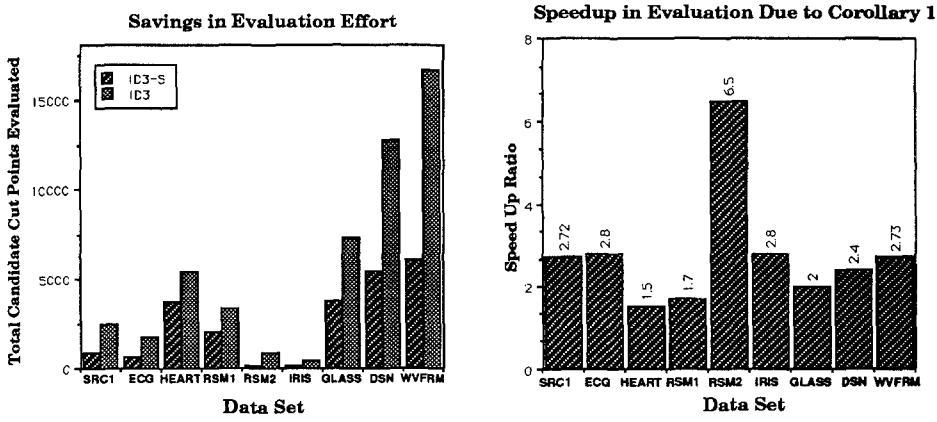

Figure 3. Empirical results demonstrating actual speedup.

the effort spent by the algorithm on nominal attribute selection and other algorithm overhead. The reason we chose it is that it is machine- and implementation-independent. However, to get a feel for the speedup in tree generation time, we did collect timing measurements. The timings are obviously machine- and implementation-dependent. We used a Macintosh IIcx computer and the program is implemented in the C programming language. The corresponding run times are depicted in Figure 4. With such a measure of performance, the speedup would obviously be reduced significantly if the majority of attributes are not continuous or if the implementation is not particularly efficient. In any case, the computational improvement is a side benefit of the results presented in this paper. The major theme of this paper is to provide further evidence in support of the usage of the information entropy heuristic.

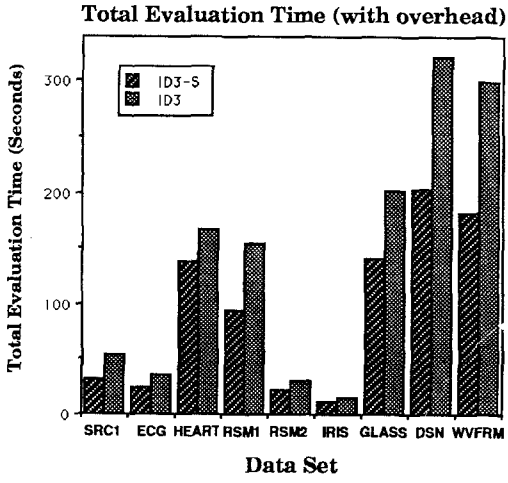

Figure 4. Run times for the various data sets.

6. Conclusion

We have presented a result regarding continuous-valued attribute discretization using the information entropy minimization heuristic. The result points out desirable behavior on the part of the heuristic which in turn serves as further theoretical support for the merit of the information entropy heuristic. In addition, the efficiency of the cut point selection heuristic can be increased without changing the final outcome of the algorithm in any way. Classification learning algorithms that use the information entropy minimization heuristic for selecting cut points can benefit from this result. It also serves as a basis for generalizing the algorithm to multiple interval discretization, as in (Fayyad, 1991). Empirical evaluation showed that speedup was indeed obtained in terms of significant reduction in the number of potential cut points evaluated.

Acknowledgments

The authors wish to thank the *Machine Learning* reviewers for their thorough reading, insightful comments, and many suggested improvements. Special thanks to professors J.R. Quinlan and D. Kibler for many suggestions that improved the presentation and corrected some errors in the earlier drafts. Discussions with Moncef Maiza were also helpful. This work was supported in part by a DeVlieg Industrial Fellowship and by a Hughes Unrestricted Grant.

Notes

1. The test $A > T$ stands for: "the value of A is greater than T."
2. For the purposes of this simplified discussion, assume all examples have distinct values for the attribute. See next section for implications of repeated attribute values.
3. One tree being "better" that another in this context means that it is smaller in size and that its (empirically estimated) error rate is lower. In (Fayyad & Irani, 1990) we address the meaning of "better" more formally.

References

Breiman, L., Friedman, J.H., Olshen, R.A., & Stone, C.J. (1984). *Classification and regression trees*. Monterey, CA: Wadsworth & Brooks.

Cheng, J., Fayyad, U.M., Irani, K.B., & Qian, Z. (1988). Improved decision trees: A generalized version of ID3. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 100–108). San Mateo, CA: Morgan Kaufmann.

Clark, P. & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning, 3*, 261–284.

Fayyad, U.M. & Irani, K.B. (1990). What should be minimized in a decision tree? *Proceedings of the Eighth National Conference on Artificial Intelligence AAAI-90* (pp. 749–754). Cambridge, MA: MIT Press.

Fayyad, U.M. & Irani, K.B. (1991). A machine learning algorithm (GID3*) for automated knowledge acquisition: Improvements and extensions. (*General Motors Research Report CS-634*). Warren, MI: GM Research Labs.

Fayyad, U.M. (1991). *On the induction of decision trees for multiple concept learning*. Doctoral dissertation, EECS Department, The University of Michigan.

Fisher, R.A. (1936). The use of multiple measurements in taxonomic problems. *Annual Eugenics, 7,* Part II, 179–188.

Gelfand, S., Ravishankar, C. & Delp, E. (1991). An iterative growing and pruning algorithm for classification tree design. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 13:2, 163–174.

Irani, K.B., Cheng, J., Fayyad, U.M., & Qian, Z. (1990). Applications of machine learning techniques in semiconductor manufacturing. *Proceedings of The S.P.I.E. Conference on Applications of Artificial Intelligence VIII* (pp. 956–965). Bellingham, WA: SPIE: The International Society for Optical Engineers.

Lewis, P.M. (1962). The characteristic selection problem in recognition systems. *IRE Transactions on Information Theory, IT-8,* 171–178.

Luenberger, D.G. (1973). *Introduction to linear and nonlinear programming*. Reading, MA: Addison-Wesley.

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

Quinlan, J.R. (1990). Probabilistic decision trees. In Y. Kodratoff & R. Michalski (Eds.), *Machine learning: An artificial intelligence approach, volume III.* San Mateo, CA: Morgan Kaufmann.