Data Mining and Knowledge Discovery 3, 197–217 (1999)

Partitioning Nominal Attributes in Decision Trees

DON COPPERSMITH copper@watson.ibm.com
SE JUNE HONG hong@watson.ibm.com
JONATHAN R.M. HOSKING hosking@watson.ibm.com
IBM Research Division, T.J. Watson Research Center, Yorktown Heights, NY 10598, USA

Editor: Heikki Mannila

Abstract. To find the optimal branching of a nominal attribute at a node in an L-ary decision tree, one is often forced to search over all possible L-ary partitions for the one that yields the minimum impurity measure. For binary trees ($L = 2$) when there are just two classes a short-cut search is possible that is linear in n, the number of distinct values of the attribute. For the general case in which the number of classes, k, may be greater than two, Burshtein et al. have shown that the optimal partition satisfies a condition that involves the existence of $\binom{L}{2}$ hyperplanes in the class probability space. We derive a property of the optimal partition for concave impurity measures (including in particular the Gini and entropy impurity measures) in terms of the existence of L vectors in the dual of the class probability space, which implies the earlier condition.

Unfortunately, these insights still do not offer a practical search method when n and k are large, even for binary trees. We therefore present a new heuristic search algorithm to find a good partition. It is based on ordering the attribute's values according to their principal component scores in the class probability space, and is linear in n. We demonstrate the effectiveness of the new method through Monte Carlo simulation experiments and compare its performance against other heuristic methods.

Keywords: binary decision tree, classification, data mining, entropy, Gini index, impurity, optimal splitting

1. Introduction

Decision trees are among the models most often used in data mining applications. When generating a decision tree, the main step at each node is to determine which attribute, if any, is to be tested and which of the many possible tests of the attribute's values should be performed. Each possible test yields a partition of the set of examples; the aim is to find the partition that minimizes the weighted sum of the class impurities of each branch, or, equivalently, the partition that maximizes the impurity improvement or "gain". In practice one first determines for each attribute the best test and its corresponding impurity, and then finds the attribute that gives the best among these "best tests".

We consider the problem of finding the best test for a nominal attribute with n values in a k-class L-ary decision tree. We are particularly concerned with cases in which n or k, or both, are quite large; this situation arises in many pattern recognition problems and in some large real data mining applications.

The problem is to find the optimal partition of n elements into L bins. A partition of the n distinct values of the attribute induces a partition of the examples: each example is put into the branch corresponding to the value that the given attribute takes for that example. The

class impurity of the examples in each branch is computed, weighted, summed and assigned to the given partition. An n by k contingency matrix, computed at the start of the procedure, can be used to compute the impurity measure for each partition that is considered. The number of distinct partitions of n elements is exponential in n: for example, if $L = 2$, a binary tree, there are $2^{n-1} - 1$ two-way partitions.

There are a variety of impurity measures. The two most often used are the Gini index, used in CART (Breiman et al., 1984) and its variants, and entropy, used in C4.5 (Quinlan, 1993) and its variants. Let $\mathbf{n} = (n_1, \ldots, n_k)$ be a vector of nonnegative real numbers representing the number of examples in each class, and let $N = \sum_i n_i$ be the total number of examples at a given node. The *Gini index* is defined as

$$g(\mathbf{n}) = 1 - \sum_i n_i^2 / N^2. \tag{1}$$

The frequency-weighted Gini index is defined as

$$G(\mathbf{n}) = N g(\mathbf{n}) = \sum_{i \neq j} n_i n_j / N. \tag{2}$$

We similarly define *entropy*

$$h(\mathbf{n}) = -\sum_i \frac{n_i}{N} \log \frac{n_i}{N} \tag{3}$$

and the frequency-weighted entropy measure

$$H(\mathbf{n}) = N h(\mathbf{n}) = N \log N - \sum_i n_i \log n_i. \tag{4}$$

For mathematical convenience, $h(\mathbf{n})$ and $H(\mathbf{n})$ have been defined using the natural logarithm $\log x$, and therefore differ from the usual binary entropy by a factor of $\log 2$; the difference is of no practical importance. We use $I(\mathbf{n})$ to denote a frequency-weighted impurity measure standing for either G or H.

Let A be an index set representing the values of the given attribute. To each attribute value $\alpha \in A$ there corresponds a vector of class frequencies,

$$\mathbf{n}^\alpha = \left(n_1^\alpha, \ldots, n_k^\alpha\right),$$

where n_i^α is the number of examples in class i for which the given attribute takes the value α. An L-way partition $\Pi = \{B_1, \ldots, B_L\}$ of A,

$$B_1 \cup B_2 \cup \cdots \cup B_L = A, \quad B_i \cap B_j = \emptyset,$$

partitions the N examples such that the class frequency of each bin is

$$\mathbf{N}^\ell = \sum_{\alpha \in B_\ell} \mathbf{n}^\alpha = \left(N_1^\ell, \ldots, N_k^\ell\right), \quad \ell = 1, \ldots, L.$$

The *total impurity* of the partition is $\sum_\ell I(\mathbf{N}^\ell)$, which is to be minimized.

To each attribute value $\alpha \in A$ there also corresponds a vector of class probabilities

$$\mathbf{p}^\alpha = \left(p_1^\alpha, \ldots, p_k^\alpha\right),$$

where

$$p_i^\alpha = n_i^\alpha \Big/ \sum_j n_j^\alpha.$$

The points $\mathbf{p}^\alpha$ lie in the *class probability space*, which is the $(k-1)$-simplex

$$\sigma_{k-1} = \left\{(x_1, \ldots, x_k) : x_i \geq 0, \sum x_i = 1\right\}. \tag{5}$$

Any partition of σ_{k-1} into $S_1, \ldots, S_L$ induces a partition of A into $B_1, \ldots, B_L$, where $B_\ell = \{\alpha : \mathbf{p}^\alpha \in S_\ell\}$, $\ell = 1, \ldots, L$.

In Section 2 we derive a property of the optimal partition, and we show that it can be regarded as a partition of σ_{k-1} into L convex polyhedra. In Section 3 we contrast the new property with those of Chou (1988, 1991) and Burshtein et al. (1989, 1992) and discuss the important case of binary trees. In Section 4 we present a new heuristic search method, and in Section 5 we compare its performance to that of some other heuristic techniques. Section 6 contains two examples and Section 7 some concluding remarks.

2. A necessary condition for a partition to be optimal

We first define the class of impurity measures to which our results apply. We use the $(k-1)$-simplex σ_{k-1} defined in (5) and the normalization map π that maps nonnegative k-vectors to σ_{k-1}:

$$\pi(n_1, \ldots, n_k) = \left(\frac{n_1}{\sum_j n_j}, \ldots, \frac{n_k}{\sum_j n_j}\right) \in \sigma_{k-1}.$$

Thus π maps a frequency vector $\mathbf{n}$ into a vector $\mathbf{p} = \pi(\mathbf{n})$ in the class probability space. Two frequency vectors $\mathbf{n}^\alpha$ and $\mathbf{n}^\beta$ are proportional if and only if $\pi(\mathbf{n}^\alpha) = \pi(\mathbf{n}^\beta)$, i.e., if $\mathbf{p}^\alpha = \mathbf{p}^\beta$.

Definition 1. A *concave impurity measure* is a real-valued function i, defined on σ_{k-1}, that satisfies

(i) For any $\lambda \in (0, 1)$ and any $\mathbf{p}, \mathbf{q} \in \sigma_{k-1}$,

$$i(\lambda\mathbf{p} + (1-\lambda)\mathbf{q}) \geq \lambda i(\mathbf{p}) + (1-\lambda) i(\mathbf{q}),$$

with equality if and only if $\mathbf{p} = \mathbf{q}$.

(ii) Writing $\mathbf{p} = (p_1, \ldots, p_k)$, $i(\mathbf{p}) = 0$ if $p_i = 1$ for some i.

The corresponding *frequency-weighted impurity measure* is a real-valued function I, defined on the space of nonnegative vectors $\mathbf{n} = (n_1, \ldots, n_k)$, that satisfies $I(\mathbf{n}) = N\, i(\pi(\mathbf{n}))$ or

$$I(n_1, \ldots, n_k) = N\, i\left(\frac{n_1}{N}, \ldots, \frac{n_k}{N}\right),$$

where $N = \sum_{j=1}^{k} n_j$.

Other natural properties of an impurity measure, e.g., that $i(\mathbf{p}) \geq 0$ with equality if and only if $p_i = 1$ for some i, follow straightforwardly from (i) and (ii). We do not require that the impurity measure be symmetric, i.e., that $i(\mathbf{p})$ is invariant to permutation of the elements of $\mathbf{p}$. Similar definitions of concave impurity were used by Breiman et al. (1984, p. 126) and Burshtein et al. (1992, p. 1642).

Concave impurity measures include those of the form $i(p_1, \ldots, p_k) = \sum_j f(p_j)$ where f is a strictly concave function with $f(0) = f(1) = 0$. Both the Gini index and entropy have this representation, with $f(p) = p - p^2$ and $f(p) = -p \log p$ respectively.

The following lemma states some properties of frequency-weighted impurity measures that we shall use in our main result, Theorem 1 below.

Lemma 1. *A frequency-weighted impurity measure I based on a concave impurity measure has the following properties*:

(i) (*homogeneity*) *For any nonnegative vector* $\mathbf{n}$ *and any* $\lambda > 0$,

$$I(\lambda\mathbf{n}) = \lambda\, I(\mathbf{n}).$$

(ii) (*strict concavity*) *For any nonnegative vectors* $\mathbf{m}$ *and* $\mathbf{n}$,

$$I(\mathbf{m} + \mathbf{n}) \geq I(\mathbf{m}) + I(\mathbf{n}),$$

with equality if and only if $\mathbf{m}$ *and* $\mathbf{n}$ *are proportional.*

Proof: For (i), let $\mathbf{n} = (n_1, \ldots, n_k)$ and $N = \sum_j n_j$. The proof of (i) is immediate, since both $I(\lambda\mathbf{n})$ and $\lambda\, I(\mathbf{n})$ are equal to $\lambda N i(\pi(\mathbf{n}))$.

For (ii), let $\mathbf{m} = (m_1, \ldots, m_k)$, $\mathbf{n} = (n_1, \ldots, n_k)$, $M = \sum_j m_j$, $N = \sum_j m_j$; further, let $\mathbf{p} = \pi(\mathbf{m})$, $\mathbf{q} = \pi(\mathbf{n})$, and $\lambda = M/(M + N)$. Then

$$\pi(\mathbf{m} + \mathbf{n}) = \frac{\mathbf{m} + \mathbf{n}}{M + N} = \lambda\mathbf{p} + (1 - \lambda)\mathbf{q}.$$

Thus

$$\begin{aligned} I(\mathbf{m} + \mathbf{n}) &= (M + N)\, i(\lambda\mathbf{p} + (1 - \lambda)\mathbf{q}) \\ &\geq (M + N)\{\lambda\, i(\mathbf{p}) + (1 - \lambda)\, i(\mathbf{q})\} \\ &= M\, i(\mathbf{p}) + N\, i(\mathbf{q}) \\ &= I(\mathbf{m}) + I(\mathbf{n}), \end{aligned}$$

with equality if and only if $\mathbf{p} = \mathbf{q}$, i.e., if and only if $\mathbf{m}$ and $\mathbf{n}$ are proportional. □

Remark. Property (ii) is a well known property of many impurity measures: the total impurity of a node is never less than the total impurity of the branch nodes regardless of how the examples are partitioned into the branch nodes, i.e., the impurity gain is always nonnegative.

We now state a necessary condition for a partition to be optimal.

Theorem 1. *Consider a finite collection of nonnegative vectors*

$$\mathbf{n}^\alpha = \left(n_1^\alpha, \ldots, n_k^\alpha\right), \quad \alpha \in A,$$

with at least L distinct images $\pi(\mathbf{n}^\alpha) \in \sigma_{k-1}$. For a partition $\Pi = \{B_1, \ldots, B_L\}$ of A, such that

$$B_1 \cup B_2 \cup \cdots \cup B_L = A, \quad B_i \cap B_j = \emptyset,$$

let

$$\mathbf{N}^\ell = \sum_{\alpha \in B_\ell} \mathbf{n}^\alpha = \left(N_1^\ell, \ldots, N_k^\ell\right), \quad \ell = 1, \ldots, L.$$

Let I be the frequency-weighted impurity measure corresponding to a concave impurity measure. Then for any partition that minimizes the total impurity $\sum_\ell I(\mathbf{N}^\ell)$, there is a collection of L vectors $\mathbf{v}^1, \ldots, \mathbf{v}^L$ such that

$$\alpha \in B_\ell \Leftrightarrow (\forall j \neq \ell)\mathbf{v}^\ell \cdot \mathbf{n}^\alpha < \mathbf{v}^j \cdot \mathbf{n}^\alpha.$$

This inequality is strict. Thus an optimal partition corresponds to a partition of σ_{k-1} into L convex polyhedra.

Proof: Let $\Pi = \{B_1, \ldots, B_L\}$ be an optimal partition. Suppose first that $\mathbf{N}^j$ and $\mathbf{N}^\ell$ are proportional. By (ii) of Lemma 1 we could combine bins B_j and B_ℓ into one bin without changing the total impurity, since $I(\mathbf{N}^j + \mathbf{N}^\ell) = I(\mathbf{N}^j) + I(\mathbf{N}^\ell)$. This would leave only $L - 1$ bins. By assumption $\pi(\mathbf{n}^\alpha)$ takes on at least L distinct values, so one bin B_a contains two vectors $\mathbf{n}^\alpha, \mathbf{n}^\beta$ with $\pi(\mathbf{n}^\alpha) \neq \pi(\mathbf{n}^\beta)$, and one of these, say $\pi(\mathbf{n}^\alpha)$, differs from $\pi(\mathbf{N}^a)$. Split B_a into $\{\mathbf{n}^\alpha\}$ and $B_a - \{\mathbf{n}^\alpha\}$ to restore L bins. The total impurity strictly decreases, by (ii) of Lemma 1:

$$I(\mathbf{N}^a) > I(\mathbf{N}^a - \mathbf{n}^\alpha) + I(\mathbf{n}^\alpha),$$

contradicting optimality of Π. By this contradiction we see that no two bins are proportional.

Define vectors

$$\mathbf{v}^\ell = \left(v_1^\ell, \ldots, v_k^\ell\right), \quad \ell = 1, \ldots, L,$$

with

$$v_i^\ell = \frac{\partial}{\partial N_i} I(N_1, \ldots, N_k)\Big|_{\mathbf{N}^\ell}.$$

For a trial vector $\mathbf{n} = (n_1, \ldots, n_k)$, the inner product $\mathbf{v}^\ell \cdot \mathbf{n}$ represents the instantaneous change of the total impurity as a small multiple of $\mathbf{n}$ is added to $\mathbf{N}^\ell$:

$$\mathbf{v}^\ell \cdot \mathbf{n} = \frac{d}{dt}\left[I(\mathbf{N}^\ell + t\mathbf{n}) + \sum_{j \neq \ell} I(\mathbf{N}^j) \right].$$

Suppose, in contradiction to the conclusion of the theorem, that $\alpha \in B_\ell$ but $\mathbf{v}^\ell \cdot \mathbf{n}^\alpha \geq \mathbf{v}^j \cdot \mathbf{n}^\alpha$ for some $j \neq \ell$. Set

$$F(t) = I(\mathbf{N}^j + t\mathbf{n}^\alpha) + I(\mathbf{N}^\ell - t\mathbf{n}^\alpha), \quad 0 \leq t \leq 1$$

(note that $\mathbf{N}^\ell - t\mathbf{n}^\alpha$ is non-negative for $0 \leq t \leq 1$, so $F(t)$ is well defined). Then we have

$$\frac{dF}{dt}\bigg|_{t=0} = \mathbf{v}^j \cdot \mathbf{n}^\alpha - \mathbf{v}^\ell \cdot \mathbf{n}^\alpha \leq 0.$$

Lemma 2. *Assume that* $\mathbf{N}^j$ *and* $\mathbf{N}^\ell$ *are not proportional. Then for* $0 < t < u \leq 1$ *we have* $\{F(t) - F(0)\}/t > \{F(u) - F(0)\}/u$.

Proof: Using the results proved in Lemma 1,

$$\begin{aligned} I(\mathbf{N}^j + t\mathbf{n}^\alpha) &\geq I((1 - t/u)\mathbf{N}^j) + I((t/u)(\mathbf{N}^j + u\mathbf{n}^\alpha)) \\ &= (1 - t/u) I(\mathbf{N}^j) + (t/u) I(\mathbf{N}^j + u\mathbf{n}^\alpha) \end{aligned}$$

and similarly

$$I(\mathbf{N}^\ell - t\mathbf{n}^\alpha) \geq (1 - t/u) I(\mathbf{N}^\ell) + (t/u) I(\mathbf{N}^\ell - u\mathbf{n}^\alpha).$$

Because the three quantities $\mathbf{N}^j$, $\mathbf{N}^\ell$, and $\mathbf{n}^\alpha$ are not all proportional, at least one of these inequalities is strict. Summing, we find that

$$\begin{aligned} I(\mathbf{N}^j + t\mathbf{n}^\alpha) + I(\mathbf{N}^\ell - t\mathbf{n}^\alpha) > {} & (1 - t/u) I(\mathbf{N}^j) + (t/u) I(\mathbf{N}^j + u\mathbf{n}^\alpha) \\ & + (1 - t/u) I(\mathbf{N}^\ell) + (t/u) I(\mathbf{N}^\ell - u\mathbf{n}^\alpha), \end{aligned}$$

i.e., that

$$F(t) > (1 - t/u) F(0) + (t/u) F(u).$$

Rearranging the terms and dividing by t gives $\{F(t) - F(0)\}/t > \{F(u) - F(0)\}/u$, completing the proof of Lemma 2.

Now continuing the proof of the theorem: this monotonicity implies that

$$0 \geq \left.\frac{dF}{dt}\right|_{t=0} = \lim_{t\to 0+} \frac{F(t)-F(0)}{t} > \frac{F(1)-F(0)}{1}.$$

Thus

$$F(0) > F(1),$$

i.e.,

$$I(\mathbf{N}^j) + I(\mathbf{N}^\ell) > I(\mathbf{N}^j + \mathbf{n}^\alpha) + I(\mathbf{N}^\ell - \mathbf{n}^\alpha),$$

and so

$$I(\mathbf{N}^j) + I(\mathbf{N}^\ell) + \sum_{i\neq j,\ell} I(\mathbf{N}^i) > I(\mathbf{N}^j + \mathbf{n}^\alpha) + I(\mathbf{N}^\ell - \mathbf{n}^\alpha) + \sum_{i\neq j,\ell} I(\mathbf{N}^i).$$

That is, the partition would be strictly improved by moving α from B_ℓ to B_j, contradicting the optimality of Π. We conclude that in an optimal partition, if $\alpha \in B_\ell$ then $\mathbf{v}^\ell \cdot \mathbf{n}^\alpha < \mathbf{v}^j \cdot \mathbf{n}^\alpha$, with strict inequality, for all $j \neq \ell$.

The partition of σ_{k-1} into L convex polyhedra P_ℓ follows:

$$\mathbf{x} \in \text{Interior}(P_\ell) \Leftrightarrow \mathbf{v}^\ell \cdot \mathbf{x} < \mathbf{v}^j \cdot \mathbf{x}, \quad \forall j \neq \ell;$$

by homogeneity,

$$\mathbf{v}^\ell \cdot \mathbf{n} < \mathbf{v}^j \cdot \mathbf{n} \Leftrightarrow \mathbf{v}^\ell \cdot \pi(\mathbf{n}) < \mathbf{v}^j \cdot \pi(\mathbf{n}). \qquad \square$$

Remark. An optimal partition for the Gini index need not be an optimal partition for entropy, and there need be no relation between their respective partitioning vectors $\mathbf{v}^\ell$.

3. Other conditions and the case of the binary tree

For a two-class binary tree, $k = 2$ and $L = 2$, Breiman et al. (1984, Section 9.4) have shown that the optimal split under an arbitrary concave impurity measure, which includes the two measures we treat here, satisfies the condition that there exists a threshold p' for either class, say class 1, such that all attribute values α for which $p_1^\alpha \leq p'$ belong in one bin and the other attribute values belong in the other bin. Hence, the search for the optimal split requires just $n-1$ computations of the total impurity; it is sufficient to consider partitions in which the set B_1 contains the α values corresponding to the j smallest values of p_1^α for $j = 1, \ldots, n-1$.

Chou proved for entropy (Chou, 1988) and later for a general class of impurity measures (Chou, 1991) a necessary condition for the optimal L-ary partition: "its bins satisfy a nearest neighbor condition with their centroids, where the 'distance' measure used for

computing both the nearest neighbors and the centroids is the divergence corresponding to the given impurity measure". See Chou (1991) for the definition of divergence, which is closely related to the total impurity. For binary trees with an arbitrary number of classes, this condition reduces to the existence of a hyperplane in the class probability space that separates the bins. Chou suggested a heuristic search for a good, though not necessarily optimal, partition based on a K-means clustering algorithm of the points $\mathbf{p}^\alpha, \alpha \in A$, in the class probability space. Each iteration of the clustering algorithm requires $O(nkL)$ operations and the number of iterations necessary is not known. Analysis of the closeness to optimality of this method has not been reported.

Burshtein et al. (1989, 1992) gave a different condition satisfied by the optimal partition for a general class of concave impurity measures: there exist $\binom{L}{2}$ hyperplanes in the class probability space that separate all pairs of bins. The condition which we have given in Section 2 applies to the Gini and entropy impurity measures and is simpler to state. It also implies Burshtein et al.'s condition: in the notation of Theorem 1, bins ℓ and j can be separated by the hyperplane $(\mathbf{v}^\ell - \mathbf{v}^j) \cdot \mathbf{n} = 0$. We do not yet know whether this condition could lead to a simpler search algorithm for the optimal partition. Burshtein et al. gave an algorithm to enumerate all linearly separable partitions. The algorithm is based on linear programming and its complexity is polynomial in n. In the case of the binary tree, both Burshtein et al.'s condition and ours reduce to the separation of the two bins by a single hyperplane in the class probability space, which in turn reduces for two-class binary trees to the simple condition given in Breiman et al. (1984) and mentioned above.

The binary tree deserves special attention because it is so frequently used. The conditions satisfied by the optimal partition suggest that to find the optimal partition one should search over the partitions of attribute values induced by linearly separable partitions of the class probability space. In the worst case the points $\mathbf{p}^\alpha$ are in general position in the $(k-1)$-dimensional simplex σ_{k-1}, i.e., no set of k points lies in a hyperplane of dimension $k-2$, and the number of partitions that needs to be examined is

$$\sum_{i=1}^{k-1} \binom{n-1}{i}.$$

This follows from a result of Cover (1965, Theorem 1 and Table 1), who showed that the number of linearly separable dichotomies of n points in general position in d dimensions is

$$C(n, d+1) = 2 \sum_{i=0}^{d} \binom{n-1}{i};$$

Cover's count is for ordered bins and includes partitions in which one bin is empty. While it is of interest to be able to enumerate the partitions in an efficient manner, to search for the optimal partition by exhaustive enumeration of linearly separable partitions is not practical for large n and k.

The "Flip-Flop" heuristic of Nadas et al. (1991) uses ideas from the special case of two-class binary trees discussed earlier to restrict the set of partitions that is searched. Nadas et al. seem to prefer their "modified algorithm", which is as follows.

1. Start with a randomly chosen binary partition of the attribute values.
2. By reversing the role of attribute and class, obtain a two-"class" problem in which the two "classes" are the two sets into which the attribute values have been partitioned. Order the (original) classes according to their "class" probabilities for the new problem. Consider the $k-1$ partitions that preserve this ordering of the classes. Choose from among these partitions the one that minimizes the total impurity for the full n-"class" problem in which every attribute value is treated as a class.
3. Reverse the role of attribute and class again, obtaining a two-class problem in which the two classes are the two sets into which the class values have been partitioned. Order the attribute values according to their class probabilities for the new problem. Consider the $n-1$ partitions that preserve this ordering of the attribute values. Choose from among these partitions the one that minimizes the total impurity for the full k-class problem.
4. Repeat steps 2 and 3 iteratively until the resulting partition does not change.

Nadas et al. do not prove any optimality properties for this procedure, but they report that the process converges quickly and that the solution is "surprisingly close to the best possible". The number of evaluations of the total impurity is of $O(n+k)$ assuming that the number of iterations does not depend on n and k.

Another heuristic search algorithm is used in the SLIQ tree generation package (Mehta et al., 1996). Mehta et al. (1996) cite IND (NASA, 1992) as the initial source where this method was proposed. For $n \leq 10$, SLIQ searches over all $2^{n-1}-1$ binary partitions. For larger n, it uses a "greedy" algorithm that starts with an empty bin and a full bin. It moves one element from the second bin to the first such that the move results in the best split. The process is iterated until there is no improvement in the splits. Mehta et al. (1996) claim that this heuristic finds the optimal partition if n is small and also performs well for larger values of n. The computational complexity of the SLIQ heuristic is $O(n^2)$.

An obvious extension of the SLIQ procedure is to allow the transfer of elements from the second to the first bin to continue until the second bin is empty and to use the best partition among all that were considered during the iterations. This extension uses only slightly more computation than the original procedure, since the early iterations involve more impurity comparisons than the later ones. In Section 5 we shall see that this extension of the SLIQ procedure gives significant improvements in the closeness to optimality of the partition finally chosen.

4. A new heuristic search for binary trees

We seek a binary partition of attribute values that is quick to compute yet close to the optimal partition in terms of the achieved reduction in class impurity. From optimality considerations, we search for a partition based on a separating hyperplane in the class probability space. For the sake of speed, we consider partitions that arise from assigning a scalar value to each attribute value and forming partitions according to whether scalar lies above certain thresholds. To achieve both objectives, we first choose a particular direction in the class probability space and then consider separating hyperplanes that are perpendicular to this direction. We choose the direction to be that of the first principal component of

the n points $\mathbf{p}^\alpha, \alpha \in A$, with each point weighted according to the number of examples for which the given attribute takes the value α. This should be a good choice because the first principal component captures as much as possible of the variation in the points in the class probability space.

Assume without loss of generality that attribute values that have equal class probability vectors have been combined into a single attribute value. Let $\mathbf{N}$ denote the n by k contingency matrix whose rows are $\mathbf{n}^\alpha, \alpha \in A$, and $\mathbf{P}$ the corresponding class probability matrix whose rows are $\mathbf{p}^\alpha$. Let $N^\alpha = \sum_{i=1}^k n_i^\alpha$ be the number of examples that have attribute value α. The vector of mean class probabilities is then

$$\bar{\mathbf{p}} = \frac{1}{N} \sum_{\alpha \in A} N^\alpha \mathbf{p}^\alpha = \frac{1}{N} \sum_{\alpha \in A} \mathbf{n}^\alpha, \tag{6}$$

and the weighted covariance matrix of the $\mathbf{p}^\alpha$ points is

$$\mathbf{\Sigma} = \frac{1}{N-1} \sum_{\alpha \in A} N^\alpha (\mathbf{p}^\alpha - \bar{\mathbf{p}})(\mathbf{p}^\alpha - \bar{\mathbf{p}})^{\mathrm{T}}. \tag{7}$$

Let $\mathbf{v}$ be the first principal component, i.e., the eigenvector corresponding to the largest eigenvalue of $\mathbf{\Sigma}$. The first principal component score of $\mathbf{p}^\alpha$ is defined as

$$S_\alpha = \mathbf{v} \cdot \mathbf{p}^\alpha.$$

Arrange the scores $S_\alpha, \alpha \in A$, in ascending order as $S_{(1)} \le S_{(2)} \le \cdots \le S_{(n)}$. Consider partitions of A that respect this ordering; these partitions are $\Pi_j = \{B_j, B_j'\}$, $j = 1, \ldots, n-1$, defined by

$$\alpha \in B_j \text{ if } S_\alpha \le S_{(j)}, \quad \alpha \in B_j' \text{ if } S_\alpha > S_{(j)}. \tag{8}$$

The chosen partition is the one that yields the minimum total impurity among the partitions Π_j, $j = 1, \ldots, n-1$.

This procedure requires $n-1$ total impurity evaluations and the principal component computation, for which standard algorithms use $O(k^3)$ operations. It usually finds an optimal or very nearly optimal partition, as will be shown empirically in the next section.

For a still closer approach to optimality a wider range of partitions can be considered. In particular we define the "swap variant" of our procedure, a small extension of the basic procedure in which we consider not only the partitions in (8) but also those partitions obtained by exchanging adjacent elements between the sets B_j and B_j'. The additional partitions are $A = C_j \cup C_j'$, $j = 1, \ldots, n-1$, defined by

$$\alpha \in C_j \text{ if } S_\alpha < S_{(j)} \text{ or } S_\alpha = S_{(j+1)}, \quad \alpha \in C_j' \text{ if } S_\alpha = S_{(j)} \text{ or } S_\alpha > S_{(j+1)}.$$

The total number of impurity evaluations for the extended procedure is $2(n-1)$, and is still linear in n.

5. Comparative experiments

A series of Monte Carlo simulation experiments was run to compare the performance of some of the partition search heuristics described above. Each simulation run considered an attribute taking n distinct values and a class variable taking k distinct values. For each attribute value α, a vector of class frequencies $\mathbf{n}^{\alpha}$ was generated by choosing each of its elements independently from a uniform distribution on the set $\{0, 1, 2, 3, 4, 5, 6, 7\}$. This yields class probability vectors that range over the class probability space and have a substantial probability of lying on the boundary of the space (i.e., some class frequencies are zero), a common occurrence in practical decision tree problems. The search heuristics were applied to the resulting contingency matrix. The procedure was repeated M times ($M = 10{,}000$ in all of the experiments) and counts were kept of the number of times that each search heuristic identified the true optimal partition, and of various statistics designed to measure the amount by which a heuristic fails to identify the optimal partition.

To quantify the amount by which a heuristic fails to identify the optimal partition, we use the difference between the impurity of the optimal partition and that of the partition chosen by the search heuristic, which we call the "excess". "Relative excess" is the excess as a proportion of the impurity of the optimal partition. Thus a relative excess of 0.05 means that the search heuristic chose a partition with impurity (Gini index or entropy) 5% higher than that of the optimal partition. Obviously, a good search heuristic is one that in repeated applications yields partitions whose excess is generally small and frequently zero.

Table 1 compares partition selection methods in terms of their ability to identify the optimal partition. Four methods are considered: SLIQ and SLIQext, the SLIQ "greedy algorithm" and its extension described in Section 3; and PC and PCext, the new heuristic based on principal component scores and its "swap variant" described in Section 4. The results cover a range of n and k values with $n \leq 12$. Larger values of n were not considered, owing to the computational expense of searching over all partitions to find the optimal partition. The results show a consistent pattern. In all cases, the new PC procedure improves on the SLIQ procedure and the extensions PCext and SLIQext provide significant improvements over their counterparts PC and SLIQ. With the Gini index as the impurity criterion, the performance of the methods is consistently ordered from worst to best as SLIQ, SLIQext, PC, PCext. With entropy as the impurity criterion, SLIQext and PCext are best and have generally comparable performance. As n and k increase, all methods become less successful at identifying the optimal partition. We reiterate that for $n \leq 10$ the SLIQ package evaluates all partitions; the "SLIQ method" considered here is the search algorithm used by the SLIQ package for $n > 10$. The pattern of the results is clear enough that we may confidently expect it to hold for larger values of n.

The next set of simulation experiments investigated the amount by which the impurity of the chosen partition exceeds that of the optimal partition. The experiments covered a subset of the (n, k) pairs considered in Table 1, and all of the methods considered there plus FF, the flip-flop heuristic of Nadas et al. (1991). We found that the flip-flop algorithm occasionally failed to converge quickly; in these instances it was terminated after 100 iterations. This occurred with approximately equal frequency using the Gini and entropy measures, 4 times in 10,000 runs for $n = 12$, $k = 3$, 25 times in 10,000 runs for $n = 12$, $k = 9$. Table 2 gives

Table 1. Percentage of simulation runs in which each of the partition selection heuristics failed to identify the true optimal partition. Results based on $M = 10{,}000$ simulation runs. Tabulated values P have standard errors $\sqrt{P(1-P)/M}$, which vary from 0.1 at $P = 1$ to 0.5 at $P = 50$.

		Gini impurity				Entropy			
		k				k			
n	Method	3	5	7	9	3	5	7	9
6	SLIQ	12.0	16.4	17.8	18.1	14.9	19.4	21.7	22.3
	SLIQext	2.4	4.7	6.2	6.0	2.8	5.5	6.6	6.4
	PC	2.0	3.7	4.0	4.8	7.1	11.8	12.5	12.9
	PCext	0.8	1.4	1.5	1.6	3.8	6.0	6.5	6.2
8	SLIQ	21.0	27.1	29.8	32.4	26.0	33.0	35.3	38.9
	SLIQext	5.3	9.3	11.1	12.8	6.3	11.1	13.1	14.6
	PC	4.5	7.3	9.2	9.6	12.0	19.4	20.5	21.6
	PCext	2.5	3.9	5.0	4.7	7.8	12.4	13.0	13.6
10	SLIQ	27.5	35.3	39.3	41.3	33.0	42.3	46.5	49.1
	SLIQext	7.8	13.8	17.2	18.7	9.6	16.8	19.6	21.5
	PC	7.7	12.7	15.5	16.3	17.0	25.7	28.9	29.6
	PCext	4.6	8.0	9.0	9.9	11.8	18.3	20.0	20.8
12	SLIQ	32.2	41.5	46.0	48.0	39.9	48.7	53.1	55.9
	SLIQext	10.6	18.0	22.7	23.6	13.6	21.0	25.4	26.7
	PC	10.2	17.6	21.0	22.2	21.2	32.5	36.2	36.9
	PCext	7.0	11.4	14.0	14.9	15.6	24.5	26.9	28.0

the results. "% Fail" is the percentage of simulation runs for which the selected partition was not optimal, as in Table 1. "Ave." is the average relative excess for those runs in which the excess was nonzero, expressed as a percentage; "Total" is the total of the relative excess values over the 10,000 runs; it is the product of the two preceding values and measures the overall amount by which the methods fall short of optimality. "Max." is the largest observed relative excess in the 10,000 simulation runs, expressed as a percentage; although it conveys some information about worst-case performance it is not very reliable and varies considerably when simulation experiments are repeated using different seeds for the random number generator.

The results in Table 2 generally reinforce those of Table 1. For the average relative excess, the worst-to-best ordering SLIQ, SLIQext, PC, PCext holds throughout. The ranking of these four methods on total relative impurity is the same as on failure percentage, except that PCext outperforms SLIQext for all cases except the combination ($n = 6$, $k = 3$, entropy). The SLIQ algorithm occasionally gives very high values for relative excess. The flip-flop heuristic performs well when $k = 3$, particularly when using entropy as the impurity measure, but is not competitive with SLIQext and PCext for $k = 9$.

The distributions of relative excess impurity from these experiments are shown in figure 1 (using Gini impurity) and figure 2 (using entropy). These illustrate the conclusions drawn

Table 2. Statistics of relative excess impurity for various partition selection heuristics and values of n and k. "% Fail" is the percentage of simulation runs for which the excess was nonzero, i.e., the selected partition was not optimal. "Ave." is the average relative excess for those runs in which the excess was nonzero, expressed as a percentage. "Total" is the total of the relative excess values over the 10,000 runs. "Max." is the largest observed relative excess in the 10,000 simulation runs, expressed as a percentage.

			Gini impurity				Entropy			
n	k	Method	% Fail	Ave.	Total	Max.	% Fail	Ave.	Total	Max.
6	3	FF	7.2	1.06	7.64	7.88	4.0	0.82	3.28	6.83
		SLIQ	12.0	1.26	15.16	13.95	14.9	1.71	25.56	22.22
		SLIQext	2.4	0.88	2.13	5.17	2.8	1.17	3.22	9.00
		PC	2.0	0.51	1.00	3.59	7.1	1.16	8.30	7.82
		PCext	0.8	0.40	0.33	1.85	3.8	1.08	4.15	6.62
6	9	FF	14.3	0.16	2.26	1.26	13.0	0.36	4.53	3.43
		SLIQ	18.1	0.18	3.34	1.11	22.3	0.48	10.67	3.57
		SLIQext	6.0	0.11	0.67	0.73	6.4	0.30	1.92	1.77
		PC	4.8	0.07	0.31	0.37	12.9	0.29	3.79	1.92
		PCext	1.6	0.05	0.08	0.23	6.2	0.27	1.67	1.71
12	3	FF	18.7	0.54	10.16	4.43	15.9	0.57	9.06	5.87
		SLIQ	32.2	1.18	38.08	13.59	39.9	1.55	62.07	12.34
		SLIQext	10.6	0.64	6.80	4.24	13.6	0.91	12.27	8.52
		PC	10.2	0.37	3.82	2.17	21.1	0.77	16.24	6.04
		PCext	7.0	0.36	2.51	2.15	15.6	0.73	11.37	5.40
12	9	FF	38.9	0.10	3.95	0.77	39.1	0.22	8.54	1.71
		SLIQ	48.1	0.16	7.62	1.02	55.9	0.39	22.07	2.27
		SLIQext	23.6	0.09	2.04	0.59	26.7	0.21	5.71	1.64
		PC	22.2	0.05	1.16	0.33	36.9	0.18	6.69	1.67
		PCext	14.9	0.05	0.75	0.33	28.0	0.17	4.81	1.46

from Table 2: using Gini impurity, the PC and PCext methods give the lowest values of relative excess; using entropy, the flip-flop algorithm performs well for $k = 3$ while PCext and SLIQext are best for $k = 9$.

Finally we present some results for many-valued attributes, for which we take $n = 50$. In this case it is impracticable to evaluate all possible partitions, so the best (lowest-impurity) partition found by any of the five methods replaced the optimal partition as the basis from which relative excess was computed. Results are given in Table 3. Their general pattern is similar to that of Table 2, which gives confidence that this pattern holds for a wide range of values of n and k. The flip-flop method performs well for $k = 3$ when entropy is used as the impurity measure, but in other cases is inferior to the PC and PCext methods. We also note that in the $k = 50$ case the flip-flop method occasionally failed to converge; in 10,000 simulations this occurred 414 times using the Gini index and 373 times using entropy. The SLIQext, PC and PCext methods all give consistent improvement over the SLIQ algorithm.

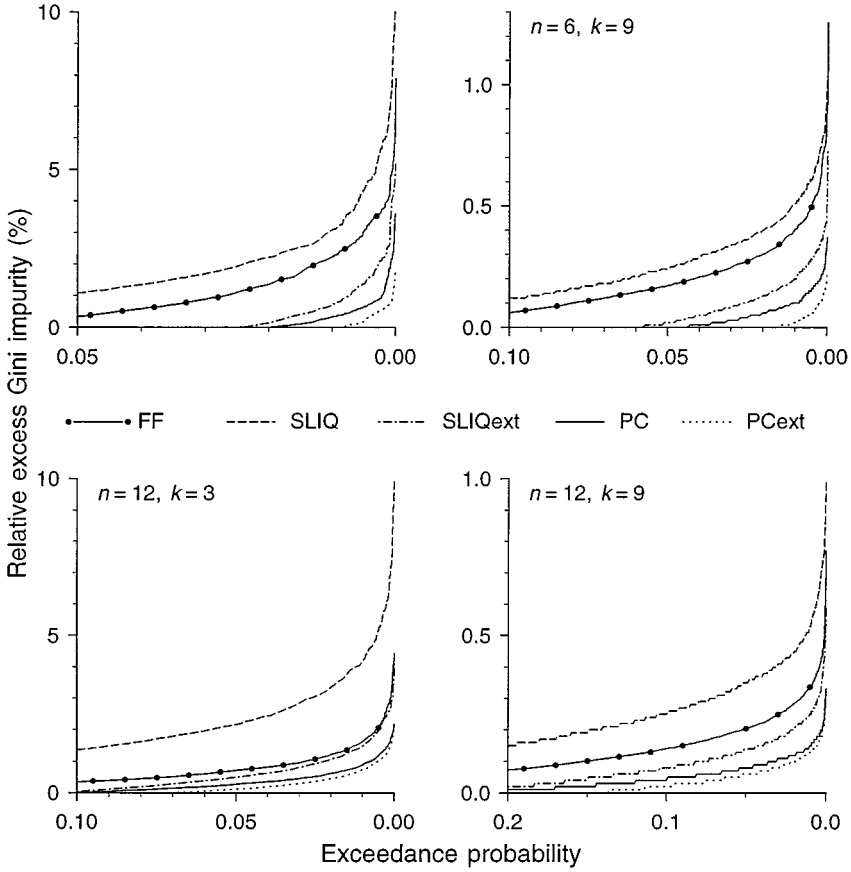

Figure 1. Distribution of relative excess Gini impurity, for various partition selection heuristics and values of n and k. Only the upper part of the distribution is shown, because most of the distribution is concentrated at zero.

SLIQext and PCext have the best overall performance; the typical case is that SLIQext and PCext are both ranked best (i.e., have zero excess) in about 50% of the simulation runs, while PCext has lower average excess and consistently achieves the lowest value of total relative excess.

6. Examples

First we give an artificial example to illustrate the computations in the PC method. The data are for 507 fruits picked late in the season. Attributes include kind of fruit, size, shape, color, etc. The class to be modeled in a binary tree is whether the fruit is in one of the three classes pick-and-ship (PS), no-pick (NP), or pick-and-ripen (PR). There are seven different color values: green, blue, purple, dark orange, light orange, yellow and red. The contingency matrix is shown in Table 4. The relative frequencies for dark orange and light orange are equal, so they can be combined into a single attribute value "Orange"; thus for

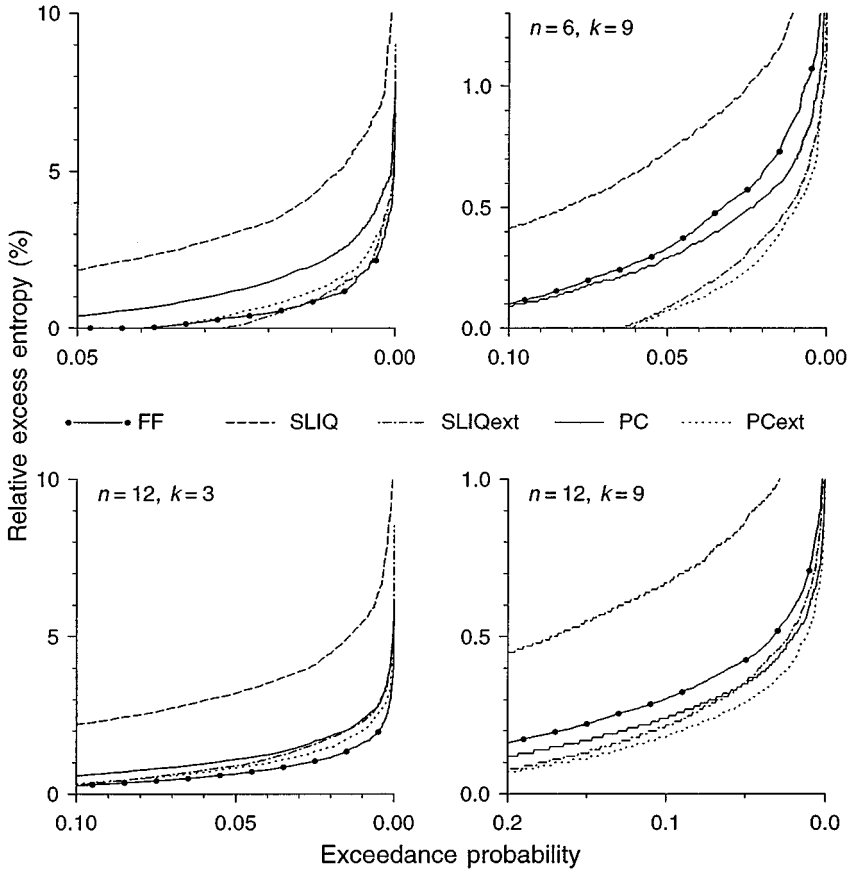

Figure 2. Distribution of relative excess entropy, for various partition selection heuristics and values of n and k. Only the upper part of the distribution is shown, because most of the distribution is concentrated at zero.

this example we have $n = 6$ and $k = 3$. We denote the six attribute values by their initial letters: G, P, B, O, Y, R.

The class probability matrix is

$$\mathbf{P} = \begin{bmatrix} 0.833 & 0.125 & 0.042 \\ 0.760 & 0.160 & 0.080 \\ 0.694 & 0.111 & 0.194 \\ 0.727 & 0.045 & 0.227 \\ 0.952 & 0.019 & 0.029 \\ 0.966 & 0.014 & 0.019 \end{bmatrix}.$$

The vector of mean class probabilities (6) is

$$\bar{\mathbf{p}} = [0.876 \quad 0.041 \quad 0.083].$$

Table 3. Statistics of relative excess impurity for the case $n = 50$. Excess impurity is computed relative to the impurity of the best of the five methods simulated here. "% Fail" is the percentage of simulation runs for which the excess was nonzero, i.e., the method was not the best of the five methods. "Ave." is the average relative excess for those runs in which the excess was nonzero, expressed as a percentage. "Total" is the total of the relative excess values over the 10,000 runs. "Max." is the largest observed relative excess in the 10,000 simulation runs, expressed as a percentage.

			Gini impurity				Entropy			
n	*k*	Method	% Fail	Ave.	Total	Max.	% Fail	Ave.	Total	Max.
50	3	FF	69	0.18	12.58	2.11	44	0.28	12.09	3.42
		SLIQ	64	0.44	27.85	7.20	83	0.75	62.20	7.15
		SLIQext	45	0.23	10.53	2.46	68	0.31	20.96	3.46
		PC	57	0.13	7.15	1.91	81	0.21	17.48	3.53
		PCext	51	0.13	6.47	1.64	78	0.21	16.16	3.53
50	9	FF	92	0.06	5.36	0.40	81	0.12	9.42	0.72
		SLIQ	73	0.07	5.07	0.69	79	0.22	17.66	1.23
		SLIQext	48	0.04	1.99	0.29	53	0.11	5.85	0.68
		PC	67	0.02	1.26	0.16	77	0.08	5.77	0.55
		PCext	49	0.02	1.03	0.16	62	0.08	5.11	0.54
50	50	FF	92	0.0039	0.36	0.0198	91	0.0266	2.42	0.1252
		SLIQ	85	0.0044	0.37	0.0304	91	0.0541	4.94	0.1988
		SLIQext	55	0.0023	0.13	0.0159	54	0.0158	0.86	0.0871
		PC	72	0.0012	0.09	0.0074	74	0.0091	0.67	0.0637
		PCext	50	0.0014	0.07	0.0074	53	0.0102	0.54	0.0637

Table 4. Class frequencies for the fruit classification example.

	Class		
Color	PS	NP	PR
Green	20	3	1
Blue	19	4	2
Purple	25	4	7
Dark orange	64	4	20
Light orange	16	1	5
Yellow	100	2	3
Red	200	3	4

The weighted covariance matrix (7) is

$$\boldsymbol{\Sigma} = \begin{bmatrix} 6.296 & -1.633 & -4.663 \\ -1.633 & 0.899 & 0.735 \\ -4.663 & 0.735 & 3.928 \end{bmatrix}.$$

The eigenvalues of this matrix are 7.486, 0.867, and 0. The eigenvector corresponding to the largest eigenvalue is the first principal component,

$$\mathbf{v} = [0.781 \quad -0.183 \quad -0.597].$$

The first principal component scores S_α, for the attribute values in the original order G, B, P, O, Y, R, are

0.603, 0.516, 0.406, 0.424, 0.723, 0.740.

When these values are put in ascending order the corresponding ordering of the attribute values is P, O, B, G, Y, R.

Table 5 shows the total impurity, for both the Gini and entropy measures, for all $2^5 - 1 = 31$ binary partitions of the six color values. In the table, the "Partition" columns indicate which colors belong to each component of the partition. For instance, line 3 of the table has the partition values 1 0 0 0 1 0 and corresponds to the partition defined by the test "is the color one of {purple, yellow}?"

As can be seen from the table, we need only 5 impurity computations using the basic PC method, or 10 by its "swap variant" PCext, while an exhaustive search requires 31. In this example, the minimum Gini impurity solution is at line 25 and the minimum entropy at line 29, and both are found by the PC method.

As a second example, we illustrate a situation in which an attribute has many values by describing a case with $n = 48$ and $k = 3$. In the US, drought severity is often measured numerically by the Palmer Drought Severity Index (PDSI). As defined by Palmer (1965), a PDSI value of zero corresponds to "normal conditions"; values of -3, -4, and -5 correspond to "drought", "severe drought", and "extreme drought" respectively. Distributions of monthly average values of PDSI at 1035 sites in the continental US are given in the US National Drought Atlas (Willeke et al., 1995; Teqnical Services Inc., 1996). We classified the sites according to their drought susceptibility: a site was assigned to class 5 if the July average PDSI at the site is -5 or less with probability at least 0.02; to class 4 if the July average PDSI is -4 or less with probability at least 0.02 but the site is not in class 5; and to class 3 otherwise. We used the state as an attribute that could be used to classify the sites. The number of sites in a state varies from 3 (Rhode Island) to 47 (California). The "State" attribute has 48 values and yields 43 distinct class probability vectors. The class frequencies are given in Table 6. The class probability vectors are plotted on figure 3, in which the triangular plotting area represents the simplex σ_2, the class probability space. Figure 3 also shows the direction of the first principal component of the plotted points, and the partitions chosen by the SLIQ, PC and FF algorithms using the Gini impurity measure. The partition chosen by PC can of course be obtained by splitting the points by a line perpendicular to the direction of the first principal component. SLIQ, PC and FF choose different partitions. PC's choice has the lowest impurity; it is also chosen by SLIQext and PCext.

In the drought-susceptibility example, the FF method surprisingly fails to find a partition as good as PC's choice. In a three-class problem, the binary partitioning of the classes that occurs in the flip-flop algorithm implies that the components of the partition finally chosen must be separable by a line parallel to one of the bounding lines of the simplex. The partition chosen by PC does satisfy this criterion: it can be generated by the line on which

Table 5. Partitions and their total impurity values for the fruit classification example. Bold type indicates the lowest values of Gini index and entropy.

	Partition						Total impurity		Needed by PC?	Needed by PCext?
	P	O	B	G	Y	R	Gini index	Entropy		
1	1	0	0	0	0	0	111.88	225.84	Yes	Yes
2	1	0	0	0	0	1	111.91	224.64	No	No
3	1	0	0	0	1	0	113.77	230.20	No	No
4	1	0	0	0	1	1	107.78	214.22	No	No
5	1	0	0	1	0	0	112.19	225.51	No	No
6	1	0	0	1	0	1	112.02	224.72	No	No
7	1	0	0	1	1	0	113.71	229.50	No	No
8	1	0	0	1	1	1	107.45	213.90	No	No
9	1	0	1	0	0	0	111.32	223.28	No	Yes
10	1	0	1	0	0	1	112.47	225.89	No	No
11	1	0	1	0	1	0	113.69	229.22	No	No
12	1	0	1	0	1	1	108.46	216.31	No	No
13	1	0	1	1	0	0	111.37	221.85	No	No
14	1	0	1	1	0	1	112.40	224.70	No	No
15	1	0	1	1	1	0	113.49	227.41	No	No
16	1	0	1	1	1	1	107.79	214.50	No	Yes
17	1	1	0	0	0	0	104.90	206.29	Yes	Yes
18	1	1	0	0	0	1	113.17	226.62	No	No
19	1	1	0	0	1	0	110.50	218.90	No	No
20	1	1	0	0	1	1	112.89	225.61	No	No
21	1	1	0	1	0	0	105.39	207.06	No	Yes
22	1	1	0	1	0	1	113.20	227.75	No	No
23	1	1	0	1	1	0	110.39	219.10	No	No
24	1	1	0	1	1	1	113.10	227.51	No	No
25	1	1	1	0	0	0	**104.19**	203.68	Yes	Yes
26	1	1	1	0	0	1	112.84	226.66	No	No
27	1	1	1	0	1	0	109.64	216.60	No	Yes
28	1	1	1	0	1	1	113.56	228.64	No	No
29	1	1	1	1	0	0	104.37	**202.80**	Yes	Yes
30	1	1	1	1	0	1	112.59	226.04	No	Yes
31	1	1	1	1	1	0	109.30	214.94	Yes	Yes

the class 3 probability is 0.4. However, even when started from this partition of the attribute values, the FF algorithm at its Step 2 chooses to partition the classes as $\{\{3, 4\}, \{5\}\}$ rather than $\{\{3\}, \{4, 5\}\}$ and the partition to which the algorithm converges is generated by a line of constant class 5 probability.

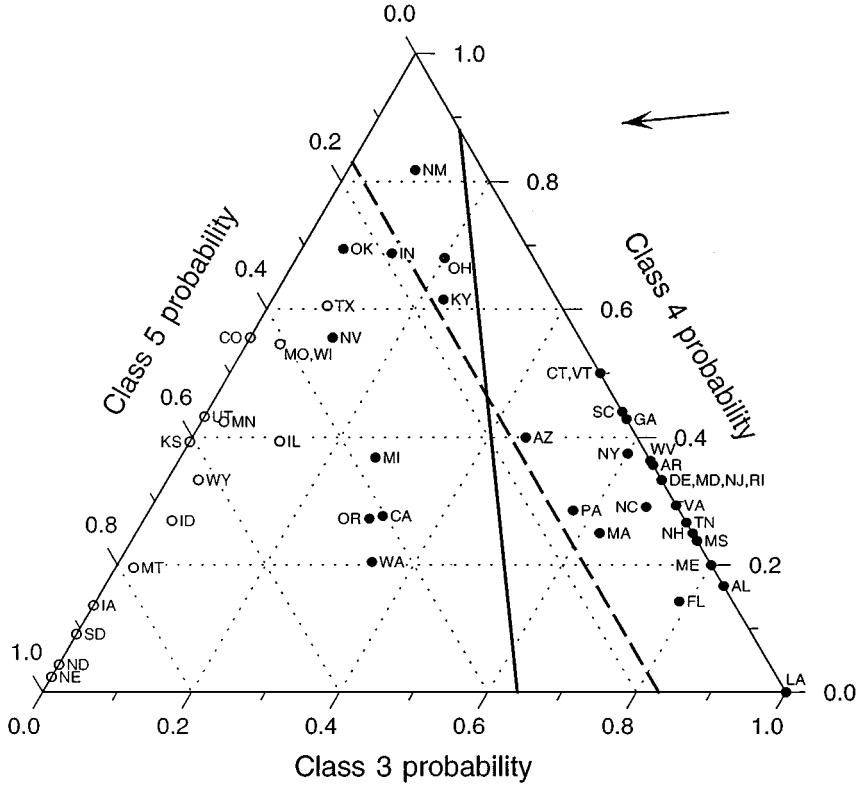

Figure 3. Class probability space for the drought-susceptibility example. Arrow indicates direction of first principal component. Solid line indicates the partition chosen by the PC method. Dashed line indicates the partition chosen by the flip-flop method. Dots and circles indicate the partition chosen by the SLIQ algorithm.

7. Conclusions

We have devised a new procedure for partitioning the examples at the nodes of a binary decision tree when the partition is based on a nominal attribute that takes many distinct values. Compared with two existing procedures, the flip-flop algorithm of Nadas et al. (1991) and the search heuristic used by SLIQ (Mehta et al., 1996), the new "PC" method and its "swap variant" PCext are fast and over a wide range of situations achieve a greater reduction in the impurity of the class variable at the nodes of the decision tree.

A search of all possible partitions requires $2^{n-1} - 1$ evaluations of the impurity measure for the split; the PCext method requires at most $2(n - 1)$ impurity evaluations and has computational complexity $O(n + k^3)$. This reduction of complexity makes it practical to find or closely approximate the optimal partition for many real data mining applications in which a binary decision tree is used as the classification model.

Finally we note that the comparisons in Tables 3–5 all deal with partitioning at a single node. In practice, the nodes are usually recursively partitioned and then pruned. This may reduce the importance of having an optimal or nearly optimal partition at each node.

Table 6. Class frequencies for the drought-susceptibility example.

	Class				Class				Class		
State	3	4	5	State	3	4	5	State	3	4	5
AL	10	2	0	ME	8	2	0	OH	5	17	3
AZ	9	8	3	MD	8	4	0	OK	2	25	9
AR	9	5	0	MA	5	2	1	OR	10	9	14
CA	15	13	19	MI	5	7	7	PA	12	6	3
CO	0	10	8	MN	1	14	18	RI	2	1	0
CT	2	2	0	MS	16	5	0	SC	14	11	0
DE	2	1	0	MO	1	12	9	SD	0	2	20
FL	11	2	1	MT	1	8	32	TN	11	4	0
GA	12	9	0	NE	0	1	41	TX	3	23	12
ID	1	7	18	NV	1	5	3	UT	0	16	21
IL	4	13	16	NH	3	1	0	VT	2	2	0
IN	4	22	6	NJ	6	3	0	VA	12	5	0
IA	0	3	19	NM	2	18	2	WA	15	9	20
KS	0	11	17	NY	24	15	1	WV	7	4	0
KY	3	8	2	NC	16	7	1	WI	1	12	9
LA	12	0	0	ND	0	1	22	WY	1	8	15

Acknowledgments

We are grateful to the referees for comments that improved the structure of the paper, particularly Section 2.

References

Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J. 1984. Classification and Regression Trees. Monterey, CA: Wadsworth International Group.

Burshtein, D., Pietra, V.D., Kanevsky, D., and Nadas, A. 1989. A splitting theorem for tree construction. Research Report RC 14754, IBM Research Division, Yorktown Heights, NY.

Burshtein, D., Pietra, V.D., Kanevsky, D., and Nadas, A. 1992. Minimum impurity partitions. Ann. Statist., 20:1637–1646.

Chou, P.A. 1988. Application of information theory to pattern recognition and the design of decision trees and trellises. Ph.D. Dissertation, Stanford Univ., Stanford, CA.

Chou, P.A. 1991. Optimal partitioning for classification and regression trees. IEEE Trans. PAMI, 13:340–354.

Cover, T.M. 1965. Geometrical and statistical properties of system of linear inequalities with applications in pattern recognition. IEEE Trans. Elec. Comput., EC-14:326–334.

Mehta, M., Agrawal, R., and Rissanen, J. 1996. SLIQ: A fast classifier for data mining. Proc. 5th Int. Conf. on Extending Database Technology, Avignon, France.

Nadas, A., Nahamoo, D., Picheny, M.A., and Powell, J. 1991. An iterative flip-flop approximation of the most informative split in the construction of decision trees. Proc. ICASSP-91, pp. 565–568.

NASA. 1992. Introduction to IND Version 2.1, GA23-2475-02 edition. NASA Ames Research Center.

Palmer, W.C. 1965. Meteorological drought. Research Paper 45, Weather Bureau, Washington, DC.
Quinlan, J.R. 1993. C4.5 Programs for Machine Learning. San Francisco, CA: Morgan Kaufmann.
Teqnical Services Inc. 1996. National Electronic Drought Atlas (CD-ROM). Teqnical Services Inc., New London, CT.
Willeke, G.E., Hosking, J.R.M., Wallis, J.R., and Guttman, N.B. 1995. The National Drought Atlas (draft). IWR Report 94-NDS-4, US Army Corps of Engineers, Fort Belvoir, VA.

Don Coppersmith received his Ph.D. in Mathematics at Harvard in 1977. He is currently a Research Staff Member at the IBM Thomas J. Waston Research Center, Yorktown Heights, N.Y. His research interests include cryptography, combinatorics, and algorithms.

Se June Hong received his B.Sc. degree in Electronic Engineering from Seoul National University and M.S./Ph.D. degrees in Electrical Engineering from the University of Illinois. He joined the IBM Poughkeepsie Laboratory in 1969 working in the areas of fault tolerant computing and design automation. Since 1978 he has been at the IBM T.J. Watson Research Center at Yorktown Heights, New York, where he is currently a Research Staff Member working on data mining technology. He is a foreign member of the national academy of Engineering of Korea, a fellow of IEEE, and a member of ACM, AAAI, KSEA, and Sigma Xi.

Jonathan R.M. Hosking has been with the IBM Research Division, Yorktown Heights, N.Y., since 1986. He holds an M.A. in Mathematics from Cambridge University and a Ph.D. in statistics (time series analysis) from Southampton University (1979). He is the author of over 50 research papers, covering such subjects as feature selection and ranking in classification problems, statistics for summarizing data samples, long-memory time-series models, and estimation of the frequency of extreme environmental events.