Model-Based Classification Trees

Donald Geman * Bruno Jedynak †

March 2000

Abstract

The construction of classification trees is nearly always top-down, locally optimal and data-driven. Such recursive designs are often globally inefficient, for instance in terms of the mean depth necessary to reach a given classification rate. We consider statistical models for which exact global optimization is feasible, and thereby demonstrate that recursive and global procedures may result in very different tree graphs and overall performance.

Keywords. Classification tree, optimal testing strategy, dynamic programming, pattern recognition.

*Department of Mathematics and Statistics, University of Massachusetts, Amherst, MA 01003; Email:geman@math.umass.edu. Supported in part by the NSF under grant DMS-9217655, ONR under contract N00014-97-1-0249, Army Research Office under MURI grant DAAH04-96-1-0445,

†Département de probabilités et statistiques, Université des Sciences et Technologies de Lille, 59655 Villeneuve d'Ascq Cedex; Email:Bruno.Jedynak@univ-lille1.fr

1 Introduction

Most of the literature on classification (or decision) trees is about inducing them from a training set $\mathcal{L}$ of labeled feature vectors in order to classify unlabeled data. Usually a tree T_{loc} is built in a top-down, recursive fashion from a pool of "tests" ("experiments," "questions") which are functions of a single feature. First the root is assigned a test, then each child of the root, and so forth until a stopping rule is enforced. At each internal node, each test in the pool is ranked according to a criterion based on information gain (e.g., entropy reduction) and the best test is assigned to the node; the gains are estimated from $\mathcal{L}$. The construction $\mathcal{L} \rightarrow T_{loc}$ is then data-driven and based on local optimization. Performance is often measured by classification error, and sometimes also by the efficiency of the representation (for example expected depth). Two seminal works are [8] and [25], and applications are numerous in statistics, pattern recognition, machine learning and other fields.

An alternative approach - the one here - begins with a statistical model $\mathcal{M}$ for the joint distribution of the tests and the classes (labels); then a tree T_{glo} is characterized by a global criterion for efficient classification. The construction $\mathcal{M} \rightarrow T_{glo}$ is then model-driven and based on global optimization. The model $\mathcal{M}$ might be estimated from data or derived in a Bayesian sense from a "forward model" for the distribution of the data given the class together with a "prior model" for the marginal distribution of the class variable. The optimality criterion might involve a tradeoff between accuracy (e.g., measured by the average entropy at the leaves or misclassification error) and computation (e.g., the average number of tests performed). A different notion of optimality based on efficient coding is discussed in [26]. Generally, calculating optimal trees is computationally prohibitive, whether model-driven or data-driven, and the literature is correspondingly sparse; see [18], [22] and the approximations in [26].

Our goal is to demonstrate that, in the model-based situation, the performance of tree classifiers based on recursive designs, e.g., stepwise entropy reduction, can be markedly inferior to those based on global designs. (The same is true of data-

driven trees, although this is more difficult to demonstrate as explicitly.) Another analysis of this discrepancy appears in the work of Garey [17] and others in the special case in which the test outcomes are determined by the classes ("constrained twenty questions"). The difference is especially pronounced with skewed priors, i.e., when *a priori* some classes are much more likely than others.

A simple example is given in Figure 4 for a model $\mathcal{M}$ with two classes $\{a, b\}$, one of which (class a) is rare; T_{loc} is on the left and minimizes entropy level-by-level and T_{glo} is on the right and minimizes a criterion based on both accuracy and computation. Both trees have the same error rate, but the *expected* depth of T_{loc} is about twice that of T_{glo}, and the testing strategy in T_{glo} is virtually the "opposite" of the greedy one. The expected depth necessary to reach a given level of accuracy is of particular importance when the tests are costly or when $\mathcal{L}$ is small and hence the estimation of information gains quickly becomes unstable. In another example (see Figure 5) the prior is uniform, both trees have average depth around ten, but the error rate of T_{loc} is many times that of T_{glo}.

Exact computations of optimal strategies, whether by brute force or clever reductions, are scarce, at least apart from the work cited above and a few very special cases in which they can be expressed in closed form, analytically. The emphasis here is on direct computation when the tests are repeatable, conditionally independent given the classes and the cost of a tree is a linear combination of the average terminal entropy and the average depth. Computing T_{glo} is then sometimes feasible, although intensive, because the optimal test to perform at any interior node is determined by the depth of the node and the conditional distribution on classes at the node. In other words, the posterior distribution is a "sufficient statistic" in that it carries all the information in the previous tests which is relevant for deciding how to continue. Optimal trees can then be generated from dynamic programming and variants thereof.

The complexity of an exact computation depends on the number M of distinct tests (in distribution) and the maximum depth D of the tree. We focus on complexity

as a function of D for fixed, relatively modest values of M. In one variant, the complexity is of order D^{2M}, which is feasible, in contrast to M^{2^D}, which is the the total number of possible trees, i.e., the order of a brute force computation without exploiting the independence assumption. Some of these observations can be traced back to DeGroot's 1970 classic text [15], where *fixed-length* optimal trees are discussed under the above assumptions, although none are actually constructed, probably due to a lack of computing resources.

In the following two sections we review the stochastic framework for tree-structured classification and the standard construction by stepwise entropy reduction; we also introduce a cost functional which accounts for both mean depth and mean terminal entropy and describe a simple recursion that characterizes minimal cost trees. A special case in which the test results are determined by the class is considered briefly in Section 4. In Section 5, we specialize to the independent model. We present a simple characterization of the cost-minimizing testing strategy in terms of the posterior distribution as well as analyze the resulting complexity of global optimization; in fact, two algorithms are presented, one top-down and the other bottom-up, for computing minimal cost trees. Bounds on the information gain are given in Section 6 and in Section 7 examples are given which illustrate the superiority of global strategies in several cases. Finally, some concluding remarks are made in Section 8.

2 Tree-Structured Classification

The goal is to assign a class label from a finite set $\mathcal{Y} = \{a, b, c, ...\}$ to a "feature vector" $\xi = (\xi_1, \xi_2, ..., \xi_p)$. Classification is based on a finite tree graph $\mathcal{T}$. The terminal nodes (denoted $\partial\mathcal{T}$) are each labeled by a class. The internal nodes (denoted $\dot{\mathcal{T}}$) are each labeled by a "test" - a discrete function $X(\xi)$ of the feature vector. For simplicity we will use only *binary tests*, for example $X = I_{\{\xi_i > c\}}$, which is the standard form of the tests in CART [8] and other algorithms. We write $\mathbf{X} = \{X_1, X_2, ..., X_M\}$ for the pool

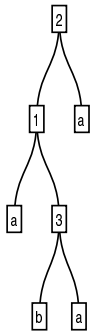

Figure 1: Example of a classification tree.

of available tests. The index of the test assigned to $t \in \dot{\mathcal{T}}$ is denoted $\pi(t) \in \{1, ..., M\}$, the depth of t by $d(t)$ (the depth of the root node is 0) and the set of observations preceding t by Q_t.

We regard the set of tests as random variables (relative to a background probability space) and we assume there is a true class $Y \in \mathcal{Y}$, another random variable. The class Y may or may not be determined by the tests or by the underlying feature vector. Let $\mathcal{M}$ denote the joint probability distribution of Y and $\mathbf{X}$; we will write $p_0(y), y \in \mathcal{Y}$, for the marginal (or "prior") distribution of Y and $g(\mathbf{x}|y)$ for the conditional distribution $P(\mathbf{X} = \mathbf{x}|Y = y), \mathbf{x} \in \{0,1\}^M, y \in \mathcal{Y}$.

Let $T = T(\mathbf{X})$ denote the resulting random variable taking values in $\partial\mathcal{T}$; thus, Q_t is the event $\{T = t\}, t \in \partial\mathcal{T}$, and depends on the outcomes of the tests $X_{\pi(s)}$ at internal nodes s along the branch from the root to t. The classifier is denoted by $\hat{Y}_T = \hat{Y}_T(\mathbf{X})$ and takes values in $\mathcal{Y}$. The class assigned to $t \in \partial\mathcal{T}$ is always the *maximum a posteriori* estimator, i.e., the class y which maximizes $P(Y = y|T = t)$.

Figure 1 is an example of a classification tree with two classes a, b. As indicated, the test performed at the root is X_2. If $\{X_2 = 0\}$ is observed, test X_1 is performed; if $\{X_2 = 0\} \cap \{X_1 = 0\}$ is observed then class a is inferred; and so forth. The history of the terminal node $t \in \partial\mathcal{T}$ labeled b is $Q_t = \{X_2 = 0\} \cap \{X_1 = 1\} \cap \{X_3 = 0\}$.

3 Testing Strategies

For simplicity, we will write $P_t(.)$ for conditional probability $P(.|Q_t)$, and p_t for the posterior distribution of Y given Q_t: $p_t(y) = P(Y = y|Q_t)$. The conditional (Shannon) entropy of Y at node t is

$$H_t(Y) = H(Y|Q_t) = -\sum_y P_t(Y = y) \log_2 P_t(Y = y).$$

If $P(Q_t) = 0$, we set $H_t(Y) = 0$. The entropy at the root of $\mathcal{T}$ is $H(Y)$.

3.1 Local Optimization

If we perform test m at $t \in \dot{\mathcal{T}}$, the average class entropy given this test and the previous outcomes is

$$H_t(Y|X_m) = P_t(X_m = 0)H_{t_0}(Y) + P_t(X_m = 1)H_{t_1}(Y),$$

where t_0 and t_1 are the two descendents of t. (If t is the root node, we will write $H(Y|X_m)$ for $H_t(Y|X_m)$, and if $P(Q_{t_0}) = 0$ or $P(Q_{t_1}) = 0$, we set $H_t(Y|X_m) = H_t(Y)$.) The standard "one step ahead" testing strategy is

$$\pi(t) = \arg \min_{m=1,\ldots,M} H_t(Y|X_m). \tag{1}$$

It can also be characterized as choosing the test X_m which most reduces the *mean* Kullback-Liebler distance between the (random) conditional distributions $p_t(y|X_m)$ and $p_t(y|\mathbf{X})$. Together with a stopping rule, this is the recursive design for building T_{loc}.

3.2 Global Optimization

The aim of a global strategy is to build a tree classifier that balances error and computation. The former is measured by *average terminal entropy*

$$H(Y|T) = \sum_{t \in \partial \mathcal{T}} P(Q_t)H_t(Y)$$

and the later by *expected depth*:

$$Ed(T) = \sum_{t \in \partial \mathcal{T}} P(Q_t) d(t) = \sum_{s \in \dot{\mathcal{T}}} P(Q_s).$$

(The second equality results from writing $d(t) = \sum_{s \in \dot{\mathcal{T}}} I_{\{s<t\}}$, where $s < t$ indicates s precedes t in $\mathcal{T}$, and then interchanging the sums.) One could minimize entropy subject to a bound on expected depth, or vice-versa, but hard constraints are difficult to enforce. Instead we introduce a control parameter $\lambda > 0$ and define the *cost* of T as

$$C(T, \mathcal{M}) = H(Y|T) + \lambda Ed(T) \tag{2}$$

We write $C(T, \mathcal{M})$ to emphasize that the cost depends on the distribution of $(Y, \mathbf{X})$. One global optimization problem is then to minimize $C(T, \mathcal{M})$ over all T. Instead, we will minimize $C(T, \mathcal{M})$ subject to a maximum depth $D = \max_{t \in \partial \mathcal{T}} d(t)$.

The expected depth is of course the expected number of tests performed in order to reach a terminal node. Trees minimizing a cost function based instead on the maximal depth or the total number of tests would be very different. In fact, for a fixed error rate, reducing the expected depth necessitates increasing the allowable maximum depth. Notice also that tests with varying costs can easily be accommodated by replacing $Ed(T)$ by

$$\sum_{t \in \dot{\mathcal{T}}} c(\pi(t)) P(Q_t)$$

where $c(m)$ is the cost of test X_m. We will always assume $c(m) \equiv 1$.

3.3 A Recursion

Let $C^\star(\mathcal{M}, D)$ be the minimum value of $C(T, \mathcal{M})$ over all trees whose maximal depth is bounded by D. Consider a tree with test m at the root and let $\mathcal{T}_0$ and $\mathcal{T}_1$ be the left and right subtrees, respectively. Then clearly

$$\begin{aligned} C(T, \mathcal{M}) &= \lambda + P(X_m = 0) \left(H(Y|T_0, X_m = 0) + \lambda Ed(T_0) \right) \\ &+ P(X_m = 1) \left(H(Y|T_1, X_m = 1) + \lambda Ed(T_1) \right). \end{aligned}$$

It follows that $C^\star(\mathcal{M}, D)$ obeys the following recursion:

Proposition 1 *For $D = 0$,*

$$C^\star(\mathcal{M}, 0) = H(p_0).$$

For $D > 0$,

$$C^\star(\mathcal{M}, D) = \min \left\{ \begin{array}{l} H(p_0); \\ \\ \lambda + \min_{m \in \{1,\ldots,M\}} \left\{ \begin{array}{l} P(X_m = 0) C^\star(\mathcal{M}(.|X_m = 0), D-1) + \\ P(X_m = 1) C^\star(\mathcal{M}(.|X_m = 1), D-1) \end{array} \right. \end{array} \right. \tag{3}$$

The minimal cost $C^\star(\mathcal{M}, D)$ is positive and decreasing in D, and hence converges as $D \to \infty$ to the minimal cost of an unbounded tree. We approximate this cost by $C^\star(\mathcal{M}, D)$ for a large enough value of D. Direct evaluation of $C^\star(\mathcal{M}, D)$ is computationally prohibitive (except for small numbers of tests and small depths). However, if the tests are conditionally independent, then exact computation becomes feasible in non-trivial cases, as we shall see shortly.

4 Constrained Twenty Questions

Perhaps the simplest model is the one underlying the familiar parlor game of "twenty questions": The class is determined by the tests (in particular $M > \log_2 |\mathcal{Y}|$) and the tests are determined by the class (i.e., there is no randomness once Y is known). The model $\mathcal{M}$ is then determined by p_0 and the binary string of M test results for each class. Since doing all the tests determines Y, the natural problem is to find the testing strategy which asks the fewest number of questions on average in order to determine Y, i.e., the tree T_{glo} which minimizes $Ed(T)$ subject to $H(Y|T) = 0$.

Since Y determines $\mathbf{X}$, we have $H_t(Y|X_m) = H_t(Y, X_m) - H_t(X_m) = H_t(Y) - H_t(X_m)$. Hence (1) reduces to

$$\pi(t) = \arg \max_{m=1,\ldots,M} H_t(X_m),$$

which amounts to choosing the test at node t which divides the classes into two groups whose masses (measured by p_t) are as equal as possible.

If there is a test for *every* subset of classes ("complete tests"), then the best global strategy is the Huffman code for p_0 and

$$H(p_0) \leq Ed(T_{glo}) \leq Ed(T_{loc}) \leq H(p_0) + 1.$$

(We omit the proof of the last inequality.) However, the general problem of computing T_{glo} is NP complete [22]. Dynamic programming leads to an algorithm [17] which is exponential in either M or $|\mathcal{Y}|$, and is feasible for "small" values of these parameters. Garey and Graham [18] consider the case in which p_0 is uniform and compare the performance of greedy and optimal strategies over all possible families of tests, showing that the former can perform very poorly depending on this family.

5 Conditionally Independent, Repeatable Tests

In contrast to constrained twenty questions, suppose the tests are (conditionally) non-degenerate, an obviously more realistic case. However, in order to achieve computational feasibility, at least for "small" problems, we add the assumption of "repeatability" This provides a richer framework than constrained twenty questions in which to display the disparity in efficiency between local and global strategies.

Specifically, we suppose from here on that the tests are *conditionally independent* given Y:

$$g(\mathbf{x}|y) = \prod_{m=1}^{M} g_m(x_m|y),$$

where $g_m(x|y) = P(X_m = x|Y = y), x \in \{0,1\}, y \in \mathcal{Y}$. Suppose further that the tests are *infinitely repeatable* in the sense that there are many independent copies of each type of test. We shall continue to write $\mathbf{X} = \{X_1, ..., X_M\}$ for a *generic* set of distinctly-distributed tests. The full family of available tests is then $\{\mathbf{X}_1, \mathbf{X}_2, ...\}$, where $\mathbf{X}_j, j = 1, 2, ...$, are independent, identically distributed copies of $\mathbf{X}$. The model is then determined by $\{p_0, g_m\}$.

Remarks on Repeatability: i) This differs from constrained twenty questions in that the same test (in distribution) may now appear several times along the same branch of $\mathcal{T}$.

ii) This setting (conditional independence and repeatability) is precisely the one in [15]. More generally, it is at the intersection of sequential statistics [12], game theory [6] and adaptive control processes [5]. In these domains, optimal strategies can, in principle, be computed using dynamic programming; still, cases in which they can be expressed in simple analytic terms are uncommon and the emphasis is on asymptotic results (e.g., $Ed(T) \to \infty$) for greedy procedures. See also [13], [20] and [11], in which printed characters are classified with trees based on the assumption the image values are class-conditionally independent.

iii) This paper was motivated by experiments in pattern recognition (see the Note in §7). In most such applications, repeatability is not a realistic assumption, and nor is conditional independence for that matter, at least in strict terms. However, when the original feature vector is varied and high-dimensional (as in image processing), and the number of classes is small, it may often be the case that certain subsets of tests have *nearly* the same conditional distribution and are *nearly* conditionally independent.

5.1 Sufficiency of the Posterior

The key observation is that the evolution of the distribution of $(\mathbf{X}, Y)$ as tests are performed depends only on the evolution of the posterior distribution of Y. More specifically, if Q denotes a history of tests, then the posterior is $p(y|Q) = P(Y = y|Q)$ and

$$\begin{aligned} P(\mathbf{X} = \mathbf{x}, Y = y|Q) &= p(y|Q)P(\mathbf{X} = \mathbf{x}|Q, Y = y) \\ &= p(y|Q)g(\mathbf{x}|y) \end{aligned}$$

Here $\mathbf{X}$ represents a "fresh copy" of tests conditionally independent of those appearing in Q. It follows that, for the independent model, we can just as well index the minimal cost $C^\star$ by the posterior p_t as $\mathcal{M}$.

Updating the posterior based on a new test X_m is very simple:

$$\begin{aligned} p(y|Q, X_m = x) &= \frac{P(X_m = x|Q, Y = y)P(Y = y|Q)P(Q)}{\sum_{y' \in \mathcal{Y}} P(X_m = x|Q, Y = y')P(Y = y'|Q)P(Q)} \\ &= \frac{g_m(x|y)p(y|Q)}{\sum_{y' \in \mathcal{Y}} g_m(x|y')p(y'|Q)} \end{aligned}$$

In particular, at the children t_0 and t_1 of an internal node t, we obtain $p_{t_0}(y)$ and $p_{t_1}(y)$ from $p_t(y)$ by choosing $Q = Q_t, m = \pi(t)$ and $x = 0, 1$, respectively. In a similar manner, we see that

$$H(Y|Q_t, X_m) = \sum_{x=0,1} \sum_{y \in \mathcal{Y}} g_m(x|y)p_t(y)H(Y|Q_t, X_m = x)) \tag{4}$$

where $p(y|Q_t, X_m), y \in \mathcal{Y}$ can be expressed in terms of g_m and p_t as above.

The consequence for the local strategy (1) is that *computing* $H(Y|Q_t, X_m)$ *under the model* $\{p_0, g_m\}$ *is the same as computing* $H(Y|X_m)$ *under the model* $\{p_t, g_m\}$. One implication of this was pointed out in [15]: If there is a dominating test X_{m^*} in the sense that $H(Y|X_{m^*}) \leq \min_m H(Y|X_m)$ *under any prior* p_0, then only this test would appear in both T_{loc} and T_{glo}. Needless to say, such tests never exist in practice.

Turning to global strategies, the test assignment π^* of the optimal tree now has a very simple characterization. Let $\mathcal{P}_0 = \{p_0\}$ and, for $k > 0$, let $\mathcal{P}_k$ denote the set of *all possible posterior distributions after* k *tests*, i.e., all possible distributions $p(.|Q)$ where Q is a conjunction of k test results. In particular, $p_t \in \mathcal{P}_{d(t)}$. Then depending on λ and the model $\{p_0, g_m\}$, there is a sequence of functions

$$\Psi_k : \mathcal{P}_k \to \{1, 2, ..., M\}, \ 0 \leq k \leq D - 1$$

which gives the optimal test at depth k as a function of the posterior after k tests. Here again D is the maximum allowable depth. In other words, at any internal node t of the optimal tree:

$$\pi^*(t) = \Psi_{d(t)}(p_t) \tag{5}$$

Consequently, due to conditional independence, the complexity of computing a global strategy reduces to counting posteriors, which, as we shall see in the following sections, is further simplified by the assumption of repeatability.

5.2 Computational Complexity

If the number of tests M and the maximum depth D are small enough we can compute $C^\star(p_0, D)$ and the corresponding tree $\mathcal{T}_{glo}$ very efficiently. The interest of this cost analysis is that within these constraints one can display comparisons between exact and virtually exact minimal cost trees T_{glo} and the corresponding greedy trees T_{loc} and thereby asses the performance loss as well as the feasibility of alternatives to stepwise entropy reduction.

The important computational issue is the growth of $\mathcal{P}_k$ as k increases and how finely we quantize it if we forgo an exact computation (as in Example 3 in the following section). For example, in the simplest case of just two classes $\{a, b\}$, suppose we quantize $p(a|Q) \in [0, 1]$ into L levels; obviously $p(b|Q) = 1 - p(a|Q)$. The complexity of using dynamic programming in order to compute the minimal cost tree (under the approximation resulting from this quantization) is then only $O(MLD)$. *However*, this *a priori* quantization induces errors and a better approximation to T_{glo} is discussed in §5.2.2.

We can compute the complexity of an exact recursion. In order to determine $C^\star(p_0, D)$ we need $C^\star(p(.|X_m = x), D-1)$ for $x = 0, 1$ and $m = 1, ..., M$. In other words, we need $C^\star(p, D-1)$ for

$$p \in \mathcal{P}_1 = \{p(.|X_i = x), 1 \leq i \leq M, x \in \{0, 1\}\}$$

which in turn requires $C^\star(p, D-2)$ for

$$p \in \mathcal{P}_2 = \{p(.|X_i = x_i, X_j = x_j), 1 \leq i, j \leq M, x_i, x_j \in \{0, 1\}\},$$

and so forth. Hence we need to compute the size of each $\mathcal{P}_k$. This is of course also evident from a backwards induction argument based on the characterization of

Tree Depth	0	1	2	3	4	5	6	7	8	9	10	total
No. Posteriors	1	4	10	20	35	56	84	120	165	220	286	1001

Figure 2: Possible posteriors after k tests, $k = 0 \ldots 10$.

the optimal strategy given by (5); see §5.2.1 below. If k tests are performed, the posterior obviously depends only on the number of events of each "type" (m, x), where $m = 1, 2, ..., M$ and $x \in \{0, 1\}$. The order in which these events occur along the branch is irrelevant. Let η_j be the number of events of type $j = 1, 2, ..., 2M$ relative to some ordering of the $2M$ pairs (m, x). Then of course $0 \leq \eta_j \leq 2M$ and $\sum_j \eta_j = k$. We want the number of distinct sequences $(\eta_1, ..., \eta_{2M})$. But there is a 1-1 correspondence between these and sequences $\alpha_j = \eta_1 + \cdots + \eta_j + j$ for $j = 1, 2, ..., 2M - 1$. Since

$$1 \leq \alpha_1 < \alpha_2 < \cdots < \alpha_{2M-1} \leq k + 2M - 1,$$

we have

$$|\mathcal{P}_k| = \binom{k + 2M - 1}{2M - 1}.$$

Values for $M = 2$ and $k = 0, \ldots, 10$ are given in Figure 2. Notice that $|\mathcal{P}_k|$ grows slowly with k compared with $(2M)^k$, which is the number of possible situations after k tests. This simple argument allows us to compute optimal trees in reasonable time for $10 \leq D \leq 20$ and $2 \leq M \leq 4$.

The computation of T_{glo} can be organized either iteratively and "bottom-up" using standard dynamic programming or recursively and "top-down" using (3). The latter is slower in simple cases but has the advantage that it can be easily modified to yield an *approximation* of the optimal tree when the number of tests gets relatively large. This approximation is different from, and superior to, the one mentioned earlier based on *a priori* quantization of the posterior.

5.2.1 Bottom-up Computation

We start at the terminal nodes and work up:

- Step 0: For each $p \in \mathcal{P}_D$, compute and store $C^\star(p, 0) = H(p)$
- Step 1: For each $p \in \mathcal{P}_{D-1}$, compute and store $C^\star(p, 1)$ using equation (3) and the values stored in Step 0.
- Step $D-1$: For each $p \in \mathcal{P}_1$, compute and store $C^\star(p, D-1)$ using equation (3) and previously stored values.
- Step D: Compute $C^\star(p_0, D)$ using equation (3) and previously stored values.

Hence, the algorithm amounts to filling in a table with $D+1$ rows corresponding to different depths. The entries in row k are a variable length set of vectors - all the distributions in $\mathcal{P}_k$ - and the minimal costs; the first row has only p_0. The first row filled is row D; it has $|\mathcal{P}_D|$ entries. Then row $D-1$ is filled using the entries in row D; it has $|\mathcal{P}_{D-1}|$ entries, and so on. After the table is made it is a simple matter to generate the functions $\{\Psi_k\}$ and hence T_{glo} itself (equivalently, the optimal testing strategy π) by a top-down pass collecting the minimizing tests at each level.

Since at each step there is a loop over posteriors and possible tests, the total complexity as a function of D and M is proportional to

$$M \sum_{k=0}^{D} |\mathcal{P}_k| = M \sum_{k=0}^{D} \binom{k+2M-1}{k} = M \binom{D+2M}{D} \tag{6}$$

The first equality was derived in the previous section and the second one can be found for example in [23], p. 54. Consequently, the complexity in D is bounded by D^{2M}. Notice that this bound is *independent* of the number of classes.

In case of $M = 2$ and $D = 10$ the effective computing time on a 225 Mhz PC is one-tenth of a second. Figure 3 shows the value in (6), in thousands, when the number of tests is $M = 2, 3, 4$ and the maximal depth is $D = 10, 20, 30$.

	2	3	4
10	2	24	175
20	21	691	12,432
30	93	5,843	195,613

Figure 3: Value of équation (6), in thousands, for 2, 3 or 4 test types and maximum depth 10, 20 or 30.

5.2.2 Top-down Computation

The computation can also be organized recursively, but top-down. The algorithm still involves completing the table mentioned above, but the computations are performed in a different order corresponding to a depth-first examination of the M-ary tree associated with (3). Thus the core of the program is a recursive procedure that computes $C^\star(p,k)$. Start with $k = D$ and $p = p_0$; if this value is in the table return it. If not, go to (3) and look for $C^\star(p,k)$ for $k = D-1$ and $p = p(.|X_m = x)$ for $x = 0$ and $m = 1$; p is computed from $\{p_0, g_m\}$ as indicated above. If this entry is not in the table, call the same procedure again for $k = D-2$, each time computing the new posterior and checking to see if it is in the table. At the beginning the procedure is called D times until we simply compute $C^\star(p,0) = H(p)$ for posterior corresponding to the event $Q = \{X_{11} = 0, X_{12} = 0, ..., X_{1D} = 0\}$ where $X_{1j}, 1 \leq j \leq D$ are independent copies of X_1. The main program is a call to this procedure with the parameters $p = p_0$ and $k = D$.

Although this implementation is more demanding than dynamic programming, the amount of computation is much less than it appears. Very quickly most, and then all, of the entries needed to compute $C^\star(p,k)$ are found in the table. Moreover, the recursive method can be easily modified to *approximate* an optimal tree as follows: Instead of looking for an exact match for the posterior, check if the optimal cost has been already computed at the given depth for a distribution *sufficiently close* to the

desired posterior. This provides a much better approximation to the optimal tree than *a priori* quantization of the posterior, which is problematic as the number of classes increases.

6 Bounds on the Information Gain

In this section we consider maximum possible gains and minimum possible costs due to a set of tests. Let $t \in \dot{\mathcal{T}}$. The information gain at node t is $H_t(Y) - H_t(Y|X_{\pi(t)})$, and the information gain due to T is $H(Y) - H(Y|T)$. Note that since the tree $\mathcal{T}$ already provides a binary coding of the values of T, and since the mean code length of a random variable is always larger than its entropy, one always has

$$H(Y) - H(Y|T) \leq H(Y,T) - H(Y|T) = H(T) \leq Ed(T).$$

Proposition 3 provides a better bound in the case of conditionally independent, repeatable tests. It is based on the following identity, the proof of which follows easily by induction on the number of leaves.

Proposition 2

$$H(Y) - H(Y|T) = \sum_{t \in \dot{\mathcal{T}}} P(Q_t)(H_t(Y) - H_t(Y|X_{\pi(t)})) \tag{7}$$

In the case of conditionally independent tests, there is a simple, tractable bound on the information gain of any T. For each $m = 1, ..., M$, define the "channel capacity" [14]

$$c(X_m, Y) = \max_{p_0}[H(Y) - H(Y|X_m)].$$

The maximum is over all possible distributions for Y. Let t be an internal node of $\mathcal{T}$; the information gain $H_t(Y) - H_t(Y|X_{\pi(t)})$ is determined by p_t and $\{g_{\pi(t)}\}$. Hence the information gain at t is bounded by $c(X_{\pi(t)}, Y)$. Substituting this bound into (7)

and using the characterization in §3 of expected depth as a sum over internal nodes, we arrive at the following bound on the total information gain, which may also be interpreted as a coupled constraint on $H(Y|T)$ and $Ed(T)$:

Proposition 3 *For any tree T and any model $\{p_0, g_m\}$,*

$$H(Y) - H(Y|T) \leq Ed(T) \max_{m \in \{1,\ldots,M\}} c(X_m, Y) \tag{8}$$

Since $c(X_m, Y) \leq 1$, this bound is better than the general one given earlier.

7 Experiments

We now give several examples to illustrate the difference in performance between T_{loc} using the recursion (1) and T_{glo} using the cost functional (2). The behavior we exhibit remains the same if Shannon entropy is replaced by another "purity measure"; indeed, changing the splitting criterion does not seem to have a great effect on performance in general ([8],[10]). Moreover, although the examples are based on the independent model of Section 5, we believe the disparity observed might be even greater with a non-trivial, conditional dependency structure among the tests. However, constructing globally optimal trees for general models is not practical.

Example 1: The performance of classification trees made using (1) may degrade considerably if $\max_{y \in \mathcal{Y}} p_0(y)$ is near one. Here is a toy example in which the greedy strategy selects the "wrong" tests at small depths, resulting in an expected depth 1.6 times larger than T_{glo} in order to achieve the same error rate or final entropy.

There are two classes with $p_0(a) = 10^{-4}$ and $p_0(b) = 1 - 10^{-4}$ and two tests with

$$g_1(1|a) = 1 \text{ and } g_1(1|b) = 0.5$$

$$g_2(1|a) = 0.5 \text{ and } g_2(1|b) = 0.$$

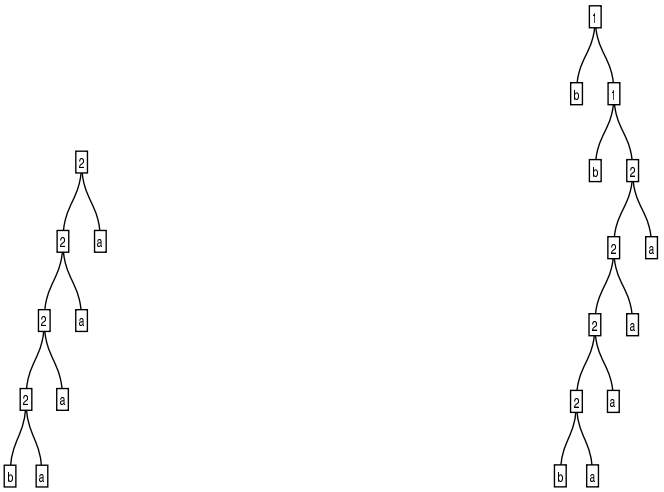

Figure 4: Left: Locally optimal tree. Right: Globally optimal tree. The error rates are the same but the *mean* depth of the global tree is smaller.

In other words, X_1 always answers "yes" on the rare class and answers randomly on the common class, and vice-versa for X_2.

Note: This example was motivated by experiments with learning algorithms for visual selection [3]; the rare class corresponds to an "object" being present at a fixed location in a large scene and the common class to "background." The first test has false negative error zero (i.e., "loses" no objects) but has false positive error 0.5, and vice-versa for X_2. Given such tests are available (and of equal cost) and given a dynamic testing strategy, how does one minimize computation subject to an error constraint?

The cost function for the globally optimal tree is (2) with maximum depth $D = 6$ and $\lambda = 10^{-4}$. The tree which minimizes cost is displayed in Figure 4; it was computed using the exact top-down recursion discussed in Section 5.2.2. The terminal nodes are labeled according to the mode of the posterior distribution. The error rate is 0 when

$Y = b$ and is $\frac{1}{16}$ when $Y = a$, concentrated in the deep node labeled b which is reached with probability approximately $\frac{1}{4}$. The mean depth is small because the probability of reaching the depth one (resp. depth two) terminal is nearly $\frac{1}{2}$ (resp. $\frac{1}{4}$), resulting in $Ed(T_{glo}) \approx \frac{5}{2}$. (The righthand side of (8) is $2.5 \times .32$ which is much larger than the actual information gain because the starting entropy is small: $H(Y) = 0.0015$.)

The locally optimal strategy always prefers test X_2 because

$$H(Y|X_1) \approx H(Y) \text{ and } H(Y|X_2) \approx \frac{1}{2}H(Y).$$

(In contrast, the global strategy puts X_1 at the top even though it provides much less average information about Y.) The depth is determined by matching the error rate of T_{glo} and the resulting tree is shown in Figure 4. The probability of exiting at the deepest terminal nodes is nearly one, which makes $Ed(T_{loc}) \approx 4$.

Example 2: Consider now a less extreme example, still with two classes and two tests. The prior is $p_0(a) = p_0(b) = 0.5$ and

$$g_1(1|a) = 0.9 \text{ and } g_1(1|b) = 0.4$$

$$g_2(1|a) = 0.6 \text{ and } g_2(1|b) = 0.1.$$

The maximum depth for T_{glo} is $D = 30$ and the tree is constructed the same way as in Example 1.

The performance of T_{loc} and T_{glo} in several cases is given in Figure 5. We adjusted the parameter λ to make either $H(Y|T_{glo}) \approx H(Y|T_{loc})$ or $Ed(T_{glo}) \approx Ed(T_{loc})$. Recall, we estimate Y by the most likely class at the leaves, denoted $\hat{Y}_T$. In this case $H(Y) = 1$ and $\max\{c(X_1, Y), c(X_2, Y)\} = 0.21$, which leads to the constraint

$$1 \leq H(Y|T) + 0.21Ed(T).$$

This is consistent with the values in Figure 5.

Again, there is generally a significant difference in performance between T_{loc} and T_{glo}, as well as in the shape of the trees; for instance, T_{glo} is very unbalanced relative

	$P(\hat{Y}_T \neq Y)$	$H(Y\|T)$	$Ed(T)$
T_{loc}	0.014	0.083	10.2
T_{glo}	0.010	0.080	6.6
T_{glo}	0.001	0.012	10.2
T_{loc}	0.051	0.237	5.6
T_{glo}	0.021	0.147	5.4
T_{glo}	0.038	0.228	4.5

Figure 5: Comparing performance of local and global strategies for the model in Example 2 with maximal depth 30.

Class y	a	b	c	d	e	f
$p_0(y)$	0.5	0.1	0.1	0.1	0.1	0.1
$g_1(1\|y)$	0.9	0.1	0.9	0.1	0.1	0.1
$g_2(1\|y)$	0.9	0.1	0.1	0.9	0.1	0.1
$g_3(1\|y)$	0.1	0.9	0.1	0.1	0.9	0.1
$g_4(1\|y)$	0.1	0.9	0.1	0.1	0.1	0.9

Figure 6: The model in Example 3. There are 6 classes and 4 types of tests.

to T_{loc}. It seems that the expected depth with balanced priors needs to be larger than with skewed priors in order to see a very sharp difference. For example, see Figure 5 in the case $Ed(T_{loc}) = Ed(T_{glo}) = 10.2$.

Example 3: Examples with more classes and more tests show the same qualitative behavior. We use the approximation procedure outlined in Section 5.5.2. in order to compute T_{glo} with six classes and four tests; the model is presented in Figure 6 and the results in Figure 7.

	$P(\hat{Y}_T \neq Y)$	$H(Y\|T)$	$Ed(T)$
T_{loc}	0.037	0.190	6.6
T_{glo}	0.023	0.180	5.4
T_{glo}	0.010	0.087	6.5

Figure 7: Comparing performance of local and global strategies for the model in Example 3.

8 Discussion

Classification trees are a popular method for addressing problems arising in non-parametric estimation, especially in domains such as pattern recognition ([4],[19], [21], [28]) in which the data are often high dimensional. Artificial neural networks are more popular, but tree-structured decision-making is easier to interpret; another advantage is the natural way in which "feature selection" is performed during tree construction [9]. As a result, there is a continuing interest in improving methods for constructing tree classifiers, especially in the data-driven case in which trees are "induced" from samples in a training set $\mathcal{L}$, i.e., test statistics and conditional entropies are estimated from $\mathcal{L}$. For example, from time to time new purity measures, splitting rules and pruning recipes are proposed and existing ones are compared ([8],[10], [24],[27]). And recently the dramatic gains from using *multiple trees* have been documented and analyzed from the point of view of randomization, negative correlation and the bias/variance decomposition ([1],[2],[7],[16], [29]).

We have analyzed the limitations of the basic induction method itself, at least in cases in which the greedy designs are likely to lead to very inefficient trees when measured by global criteria such as mean path length. Such cases arise when some classes are very rare and when the training set $\mathcal{L}$ is small; the benefits of choosing good tests are then accentuated since the amount of data available at a node for estimating information gains and class likelihoods is rapidly decreasing with depth of

the node. Indeed, we would argue that the interesting limit in pattern recognition and other applications is $|\mathcal{L}| \to 0$ rather than $|\mathcal{L}| \to \infty$, and that the most effective way to introduce problem-specific knowledge into the design of the classifier is by "hard-wiring" global constraints.

Finally, how might global optimization be relevant for inducing trees either from data or from more complex models, especially in applications to pattern recognition and machine learning where assumptions such as independence and repeatability are usually violated? The natural path would appear to be $\mathcal{L} \to \mathcal{M} \to T_{glo}$: First estimate a model from the data and then calculate an efficient tree from the model. But unless $\mathcal{M}$ is severely restricted *a priori*, it will not be sufficiently elementary to deduce T_{glo}. Yet it is precisely the rich dependency structure in the feature vector which makes the underlying classification problem interesting and challenging. Perhaps globally optimal strategies which are derived from simplified, approximate models (for instance assuming conditional independence but using the actual marginal test statistics) might serve as "blueprints" for recursive tree construction. Given the disparities we have illustrated, the rewards could be significant.

References

[1] Y. Amit, G. Blanchard, and K. Wilder. Multiple randomized classifiers: Mrcl. Technical Report 496, Department of Statistics, University of Chicago, 1999.

[2] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9:1545–1588, 1997.

[3] Y. Amit, D. Geman, and B. Jedynak. Efficient focusing and face detection. In H. Wechsler and J. Phillips, editors, *Face Recognition: From Theory to Applications, NATO ASI Series F*. Springer-Verlag, Berlin, 1998.

[4] Y. Amit, D. Geman, and K. Wilder. Joint induction of shape features and tree classifiers. *IEEE Trans. PAMI*, 19(11), 1997.

[5] R. Bellman. *Adaptive Conrol Process: A Guided Tour.* Princeton University Press, 1961.

[6] D. Blackwell and M. A. Girschick. *Theory of Games and Statistical Decisions.* John Wiley, 1954.

[7] L. Breiman. Arcing classifiers. *Annals of Statistics*, 26:801–878, 1998.

[8] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth, Belmont, CA., 1984.

[9] D. Brown, V. Corruble, and C. L. Pittard. A comparison of decision tree classifiers with backpropagation neural networks for multimodal classification problems. *Pattern Recognition*, 26:953–961, 1993.

[10] W. Buntine and T. Niblett. A further comparison of splitting rules for decision-tree induction. *Machine Learning*, 8:75–85, 1992.

[11] R. G. Casey and G. Nagy. Decision tree design using a probabilistic model. *IEEE Trans. Information Theory*, 30:93–99, 1984.

[12] H. Chernoff. *Sequential Analysis and Optimal Design.* SIAM, 1972.

[13] Y. S. Chow, H. Robbins, and D. Siegmund. *Great Expectations: The Theory of Optimal Stopping.* Houghton Mifflin, 1971.

[14] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley, New York, 1991.

[15] M. H. DeGroot. *Optimal Statistical Decisions.* McGraw-Hill, New York, 1970.

[16] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *J. Artificial Intell. Res.*, 2:263–286, 1995.

[17] M. R. Garey. Optimal binary identification procedures. *SIAM J. Appl. Math.*, 23:173–186, 1972.

[18] M. R. Garey and R. L. Graham. Performance bounds on the splitting algorithm for binary testing. *Acta Informatica*, 3:1974, 1974.

[19] S. B. Gelfand and E. J. Delp. On tree structured classifiers. In I. K. Sethi and A. K. Jain, editors, *Artificial Neural Networks and Statistical Pattern Recognition*, pages 51–70. North Holland, Amsterdam, 1991.

[20] J. C. Gittins. *Multi-armed Bandit Allocation Indices.* John Wiley, 1989.

[21] J. Huang, S. Gutta, and H. Wechsler. Detection of human faces using decision trees. In *Proceedings, Second International Conference on Automatic Face and Gesture Recognition*, pages 248–252. IEEE Computer Society Press, 1996.

[22] L. Hyafil and R. Rivest. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5:15–17, 1976.

[23] D. E. Knuth. *Fundamental Algorithms.* Addison-Wesley, 1973.

[24] M. Miyakawa. Criteria for selecting a variable in the construction of efficient decision trees. *IEEE Trans. Computers*, 38:130–141, 1989.

[25] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[26] J. R. Quinlan and R. L. Rivest. Inferring decision trees using minimum description length principle. *Information and Computation*, 80:227–248, 1989.

[27] I. K. Sethi. Decision tree performance enhancement using an artificial neural network implementation. In I. K. Sethi and A. K. Jain, editors, *Artificial Neural Networks and Statistical Pattern Recognition.* North Holland, Amsterdam, 1991.

[28] L. Spirkovska. Three-dimensional object recognition using similar triangles and decision trees. *Pattern Recognition*, 26:727–732, 1993.

[29] K. Wilder. *Decision tree algorithms for handwritten digit recognition.* PhD thesis, University of Massachusetts, Amherst, Massachusetts, 1998.