Decision Tree Construction in Fixed Dimensions: Being Global is Hard but Local Greed is Good

Michael T. Goodrich*‡ Johns Hopkins goodrich@cs.jhu.edu

Vincent Mirelli US Army Res. Lab vmirelli@nvl.army.mil

Mark Orletsky†‡ Johns Hopkins orletsky@cs.jhu.edu

Jeffery Salowe QuesTech, Inc. jsalowe@nvl.army.mil

Abstract

We study the problem of finding optimal linear decision trees for classifying a set of points in $\mathbb{R}^d$ partitioned into concept classes, where d is a fixed, but arbitrary, constant. We show that optimal decision tree construction is NP-complete, even for 3-dimensional point sets. Nevertheless, we can prove a number of interesting approximation bounds on the use of random sampling for finding optimal splitting hyperplanes in greedy decision tree constructions. We give experimental evidence that, while providing asymptotic guarantees on split quality, this random sampling approach behaves as good in practice as uniform randomization strategies that do not provide such guarantees. Finally, we provide experimental justification for coupling this random sampling strategy with locally-greedy "hill climbing" methods.

1 Introduction

A general framework for machine learning is that one is given a (hopefully representative) sample S of n points taken from some much larger (possibly infinite) set of points $\mathcal{C} \subseteq \mathbb{R}^d$, which is partitioned in an unknown way into a finite number of *concept classes*. Each point p in S is given with its classification—the name of the concept class to which p belongs. The learning problem is to build a classification method from the points in S such that given an arbitrary query point $q \in \mathcal{C}$ one can quickly and accurately assign a concept class to q using this method. This framework differs from some other, more general learning frameworks (e.g., see [1, 3, 2, 12, 14, 15]), but is sufficiently powerful to contain many practical, "real world" instances of the learning problem.

In the *linear decision tree* approach to this learning problem one uses S to build a decision tree T where each decision involves testing if the query point is beneath or beyond a specific d-dimensional hyperplane. A query proceeds down the tree T from the root until arriving at a leaf that is labeled with the name of a concept class, which is then output as the classification for the query point. An example of classified points and an associated decision tree are shown in Figure 1. Research in artificial intelligence is showing linear decision trees to be a powerful, practical method for performing these machine learning classification tasks [4, 5, 8, 13, 18, 19, 20, 21, 22, 23]. Typically, the "real world" problems where such methods are being applied involve points that are taken from $\mathbb{R}^d$, with d being a reasonably-small constant (typically, $d \leq 50$). There has been a fairly extensive empirical study of such instances of the learning problem, but there has not been a great deal of analytic study of linear decision tree learning for such applications. The goal of this paper is to give such an analytic study of fixed-dimensional decision-tree construction while also maintaining a close tie with results of experimental research.

*This research supported in part by the NSF under Grants IRI-9116843 and CCR-9300079.

†This research supported in part by the NSF under Grant IRI-9116843.

‡This work was also supported by the S3 Information Processing Directorate under the auspices of the U.S. Army Research Office Scientific Services Program administered by Battelle (Delivery Order 1330, Contract No. DAAL03-91-C-0034).

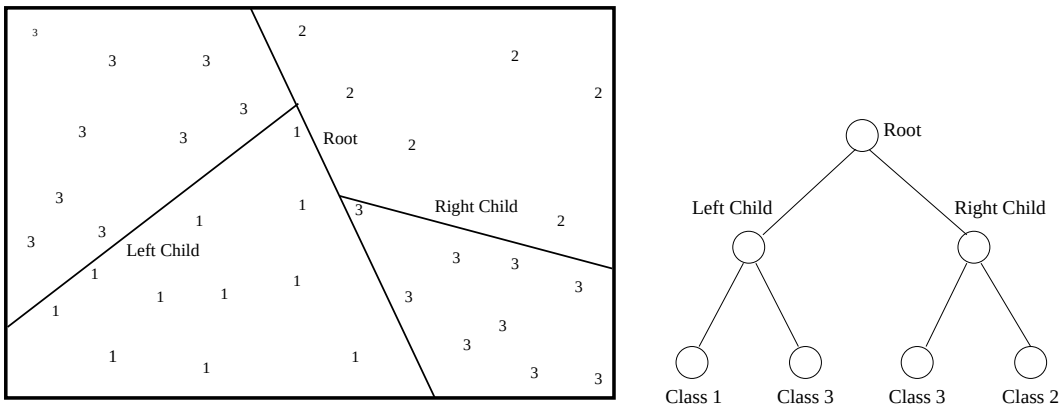

Figure 1: Two dimensional classified data points and an associated decision tree.

1.1 Being Global is Hard

One of the main challenges in using a decision tree for the learning problem is in quickly constructing a tree that is accurate. It is well known, for example, that constructing a best decision tree in general settings [14, 15] or in arbitrary dimensions [7, 17] is NP-complete; hence, it is extremely unlikely that we will ever discover a polynomial-time algorithm for constructing a best decision tree in these contexts. In the context of fixed-dimensional learning, however, each of these NP-completeness proofs fail. Indeed, each of their respective optimization problems are polynomial-time solvable in a fixed-dimensional setting. Thus, one might be tempted to believe that global decision tree optimization might actually be tractable in fixed dimensions.

Unfortunately, this is not the case. For, as we show in this paper, the problem of finding an optimal k-node decision tree classifying n points taken from two classes in $\mathbb{R}^3$ is NP-complete. Even so, one can still produce very good decision trees in practice by relying on the greedy method.

1.2 Local Greed is Good

The greedy approach is to build the tree in a top-down fashion. Given a current tree node v, and a subset $S(v) \subseteq S$ (the learning set for v) in $\mathbb{R}^d$, the goal is to find a splitting hyperplane h that minimizes some *impurity* measure for the points in $S(v)$. One then creates two children for v, which are respectively associated with the points of $S(v)$ beneath h and those beyond h (any points on h can be handled using some "tie-breaking" scheme). Finally, one recurses on the new children of v if their impurity measure is still above a desired threshold. Typically, this tree construction phase is then followed by a pruning phase, which improves the tree even more.

Unfortunately, the greedy approach is perhaps a bit "too greedy" in this context. This is because finding a best splitting hyperplane h appears to require testing all the $\binom{n}{d}$ hyperplanes determined by the n points in $S(v)$. Using a straightforward algorithm for finding h, then, requires $O(n^{d+1})$ time, although this can be reduced to $O(n^d)$ using computational geometry techniques[1]. Even though this is at least a polynomial-time computation when d is constant, it is still effectively intractable for realistic problem sizes. For example, when n is, say, about 1,000 and d is about 10, this computation would take approximately 2^{100} steps (which is more than the current estimate on the number of atoms in the universe). To deal with this combinatorial explosion, of course, requires that we restrict the set of candidate hyperplanes in some way.

[1] A more sophisticated method is to construct the arrangement of hyperplanes dual to the input points and then traverse this arrangement in a depth-first fashion.

Most previous approaches to decision tree construction have been to restrict the set of candidate hyperplanes to the $O(dn)$ that are perpendicular to one of the coordinate axes. These hyperplanes are easy to test (they only require a linear comparison for a single coordinate of a test point), and the best such hyperplane can be found in $O(dn \log n)$ time. Unfortunately, restricting the candidate set of hyperplanes in this way ignores any possible dependencies that may exist between coordinates, which is common in "real world" data sets. This is therefore very possibly much too restrictive.

For this reason, several researchers, including the designers of the well-known and widely-used CART system by Breiman *et al.* [8], have still allowed for non-axis-perpendicular splits, but have restricted the set of candidates to those encountered in a heuristic "hill-climbing" search. In this approach, one starts with an initial candidate hyperplane

$$h_0 : a_1 x_1 + a_2 x_2 + \cdots + a_d x_d = 1,$$

and one iteratively varies a_1, a_2, and so on, each time fixing the value that achieves the lowest impurity measure. This approach can be implemented to run in $O(dn \log n)$ time, and it is guaranteed to find a (possibly non-axis-perpendicular) hyperplane that achieves a local minimum with respect to the impurity measure (and this iterative strategy). The main problem with this approach, however, is that it is highly sensitive to the choice of initial hyperplane h_0 and it is insensitive to the way the input points are distributed in $\mathbb{R}^d$.

We show how to overcome these deficiencies, however, by designing a scheme that is *distribution sensitive*. Our method gives a collection of starting hyperplanes that can be found quickly and which also "cover" the input points, even if there are dense clouds of points packed into an area of small volume. In fact, we are able to provide worst-case bounds on the approximation error between the split our method finds and an optimal split. We give simple randomized methods as well as more sophisticated deterministic procedures based upon computational geometry techniques. We have implemented the randomized strategies in the OC1 decision-tree learning system, which was developed by Murthy *et al.* [18] as an improvement to the well-known CART system [8]. We give empirical results that show that classification accuracies are as good as uniformly-random methods that do not provide worst-case approximation bounds. Moreover, we provide empirical justification for the locally-greedy hill-climbing strategy [8].

2 Global Optimization is Hard even in 3-Dimensions

In this section we show that decision tree optimization is hard even if the points in the learning set S are taken from $\mathbb{R}^3$. In measuring the size of a decision tree we will only count the internal nodes, since this is the measure of the number of splits needed to classify a certain collection of points. Thus, in this nomenclature, a decision tree that contains only one split will have one node, even though it actually has three vertices, two of which are leaves.

The specific problem we address is the following:

> DECISION-TREE. Given a set S of n points in $\mathbb{R}^3$, divided into two concept classes "red" and "blue," is there a decision tree T with at most k nodes that separates the red points from the blue points?

Theorem 2.1: DECISION-TREE *is NP-complete.*

Proof: First, let us observe that DECISION-TREE is in NP. This is because each candidate split in a linear decision tree is determined by 3 points, hence, there are $\Theta(n^3)$ candidate splits. We can therefore guess k splits and a tree structure with one of these splits at each node, and we can then test that this decision tree separates all the red and blue points.

To prove that DECISION-TREE is NP-hard we will reduce the POLYTOPE VERTEX-COVER problem problem to it, which was shown to be NP-complete by Das and Goodrich [11]. In this problem, one is given a convex polyhedron P in $\mathbb{R}^3$ and an integer k and asked if there is a subset of the vertices of P that cover all the edges of P.

For the sake of simplicity, let us allow as input to the DECISION-TREE problem point sets where red points and blue points "overlap". A complete classification of such a pair of points must therefore have a split that passes through this common location in space. (This restriction can be relaxed by forcing such pairs to be separated by an "infinitesimal" amount ϵ.) Our reduction is based upon judiciously placing such pairs of points on the edges of Q, the Poincaré dual to P, i.e., Q is a convex polyhedron whose 1-skeleton is the graph-theoretic planar dual to the 1-skeleton of P. Thus, a *face cover* in Q corresponds immediately to a vertex cover in P. We place two red-blue pairs along each edge of Q so that the only way four such pairs can be co-planar is if they all lie on the same face of Q. Let S denote this set of red and blue points. Note that, since Q is a convex polyhedron, each face of Q contains at least six pairs of points in S. This construction can all be done in polynomial time.

We claim that there is a k-node decision tree for S if and only if there is a k-face face-cover for Q (and, hence, a k-node vertex cover for P). First, note that if there is a k-face face-cover for Q, then there must be k planes that collectively contain all the pairs in S; hence, there is a k-node decision tree for S. For the more difficult direction, suppose there is no k-face face-cover for Q; that is, any face cover requires more than k faces. This implies that any decision tree restricted to splits containing faces of P must have more than k nodes. Note, however, that each such split contains at least six pairs of points in S whereas any other type of split contains at most three pairs of points in S. Therefore, since each pair of points in S must be contained in some split, there must be more then k nodes in any decision tree that completely separates the pairs in S. This completes the proof. □

Thus, it is perhaps too ambitious to try to optimize an entire decision tree at once, even in a fixed-dimensional setting.

3 Efficient Decision Tree Construction

Instead, researchers have concentrated on using an effective heuristic for decision tree construction—the greedy method. Recall that this approach is to build the tree in a top-down fashion, as mentioned in the introduction. Given a current tree node v, and a set of n input points $S(v)$ (the learning set for v) in $\mathbb{R}^d$, the goal is to find a splitting hyperplane h that minimizes some *impurity* measure for the points in $S(v)$.

3.1 The Uniform Strategy

The OC1 decision-tree learning system gives a strategy for performing this task by performing k different hill-climbing searches, each starting from one of a set of k starting hyperplanes $h_1, h_2, \ldots, h_k$. The best split encountered in all these hill-climbs is the one chosen for the current node v, where quality is measured relative to a given impurity function. The method for choosing each h_i in the OC1 system is simple—just pick each of coordinates $a_1, a_2, \ldots, a_d$ at random (from the appropriate range for each feature). This overcomes the effect of a bad initial hyperplane choice to a degree (in that OC1 uniformly beats the CART system), but it does not easily lend itself to guaranteed performance bounds.

The difficulty in providing such a point with a such a uniform strategy is that if the input points are not uniformly distributed, then it is very possible that the uniform strategy will "miss" some dense, and presumably information-rich, region in the space of candidate hyperplanes simply because it has small volume. Even though such a region could contain a large number of candidate hyperplanes, its small volume insures that the uniform strategy will most likely "pass it by". For this reason, our approaches for selecting candidate starting hyperplanes are different—they are distribution-sensitive.

3.2 A Data-Sensitive Strategy

Our method is to choose each of the k candidate hyperplanes h_i to be the hyperplane determined by d of the n input points, selected at random from $S(v)$. While on the surface this may seem to not be that substantial a change from the uniform initialization procedure, it actually is quite different in some fundamental ways. At an intuitive level the main advantage of this method is that is guaranteed to select uniformly from the entire set of candidate hyperplanes—even if most of them are "packed" into a small region in the space of candidate hyperplanes. More importantly, however, we can actually prove something about how "far" one of are candidate hyperplanes can be from the optimal splitter, which is something that is not possible with the uniform initialization procedure.

For any hyperplane h, let $A(h)$ denote the number of points in $S(v)$ that are above h. Also, let h_{opt} denote the optimal splitter, and let h_j denote the best initial hyperplane in our scheme (without even performing the hill-climbing improvement).

Theorem 3.1: *The expected difference between $A(h_{\text{opt}})$ and $A(h_j)$ is only*

$$O(n/[kf(n/k^{1/d})]^{1/d}),$$

where $f(l)$ denotes the expected size of the convex hull of a random subset of size l taken from $S(v)$.

Proof: Dualize the input points in $S(v)$ to hyperplanes. Each point in this dual space corresponds to a candidate hyperplane. Measure the *distance* between two candidates points p and q (corresponding to hyperplanes in the primal) as being the number of (dual) hyperplanes one crosses in moving from p to q. Imagine performing a transformation of coordinates so that the point dual to h_{opt} becomes the origin o. We wish to show that the point closest to o and dual to one of our candidate hyperplanes has expected distance $O(n/[kf(k^{1/d})]^{1/d})$ from d. In [10] Clarkson and Shor show (using a different notation, of course) that the number of points at distance l from o is expected to be $\Theta(l^d f(n/l))$. Thus, the expected number of points at distance l from o that correspond to candidate hyperplanes in our scheme is

$$\left[\frac{k}{\binom{n}{d}}\right] l^d f(n/l).$$

The theorem follows, then, by observing that this expectation is $\Omega(1)$ if l is $O(n/[kf(k^{1/d})]^{1/d})$. □

In other words, we can actually bound in a worst-case sense how close we expect to come to the optimal splitter. Moreover, this bound can be expressed as a function of k, the number of candidate starting hyperplanes. And this doesn't even take into consideration the improvements we achieve by then performing a hill-climbing procedure from each of these candidates.

To make the above theorem concrete, consider a set $S(v)$ of n points that are uniformly distributed inside a d-dimensional sphere. In this case $f(k^{1/d})$ is $\Theta(k^{(d-1)/(d+1)})$; hence, the expected difference between $A(h_{\text{opt}})$ and $A(h_j)$ in this case is $O(n/k^{2/(d+1)})$. Thus, if n is, say, 50,000 and d is, say, 5, then we can expect to come within an additive factor of a third of the input points by considering just 27 random candidate hyperplanes (and this is the worst-case!).

3.3 Deterministic Methods

Interestingly, we can actually achieve the above results deterministically and in a worst-case fashion using computational geometry techniques. One possible method is to dualize the input points in $S(v)$ to hyperplanes and find an $O(k)$-sized $(1/k^{1/d})$-cutting for this set. Recall that a $(1/r)$-cutting of a set of hyperplanes is a simplicial complex χ partitioning $\mathbb{R}^d$ such that the interior of each simplex $\tau \in \chi$ is intersected by at most n/r hyperplanes. Such a complex of size $O(r^d)$ can be found deterministically in

$O(nr^{d-1})$ time [9, 16], which, in terms of k, is a complex of size $O(k)$ that can be found in $O(nk^{(d-1)/d})$ time. Taking the hyperplanes dual to the vertices of this complex will achieve the same bounds quoted in Theorem 3.1.

4 Error Bounds for Common Decision Tree Impurity Measures

The most natural impurity measure included in CART and OC1 is to choose a splitting hyperplane that maximizes the number of correctly classified points, where a region is classified by a plurality of the points. (We call this "sum correctly classified" below.) Breiman et al. [8], however, argue that this impurity measure has some important deficiencies, so they define other impurity measures, for which we give worst-case approximation bounds in this section.

Recall that the overall objective is to minimize classification error at the leaves, so minimizing the classification error at an internal node may not be necessary. Theorem 3.1 implies that there is a starting hyperplane h (i.e., before hill-climbing) for which the cardinality of the set of points in the left halfspace defined by h is approximately the same as the cardinality of the left halfspace defined by h^*, a hyperplane that optimizes a particular impurity measure. In this section, we explore the relationship between Theorem 3.1 and some of the impurity measures.

Recall that $A(h)$ represents the number of points above hyperplane h; let $B(h) = n - A(h)$ be the number of points below (or on) hyperplane h. Suppose there are c classes, labeled 1 through c. Denote the cardinality of the points in class i above h by $A_i(h)$, and denote the similar quantity below h by $B_i(h)$. Let $a(h) = A(h)/n$, $b(h) = B(h)/n$, $a_i(h) = A_i(h)/A(h)$, and so on. Finally, let $g(k,d) = [kf(k^{1/d})]^{1/d}$.

The purpose of the "sum correctly classified" impurity measure is to find a hyperplane h^* that maximizes $s_1(h) = \max_i A_i(h) + \max_j B_j(h)$. A slight extension of Theorem 3.1 implies that there is a hyperplane h such that

$$A_i(h) \in [A_i(h^*) - O(n/g(k,d)), A_i(h^*) + O(n/g(k,d))].$$

Therefore, hyperplane h satisfies

$$s_1(h^*) - O(n/g(k,d)) \leq s_1(h) \leq s_1(h^*) + O(n/g(k,d)).$$

The analysis for the impurity measure "max minimum correctly classified" is similar.

The "twoing criterion" is defined in the following way. It finds a hyperplane h^* that maximizes

$$s_2(h) = a(h)b(h)(\sum_{i=1}^{c} |a_i(h) - b_i(h)|)^2.$$

We must compare $s_2(h)$ with

$$s_2(h^*) = a(h^*)b(h^*)(\sum_{i=1}^{c} |a_i(h^*) - b_i(h^*)|)^2.$$

Let $e = |A(h^*) - A(h)|$. Then $|a(h^*) - a(h)| = |b(h^*) - b(h)| = e/n$. Furthermore, it is possible to show that each term $|a_i(h) - b_i(h)|$ satisfies

$$||a_i(h) - b_ih)| - |a_i(h^*) - a_i(h^*)|| = O\left(\frac{e}{\min\{A(h), B(h)\}}\right).$$

It is apparent that a hyperplane h with $e = O(n/g(k,d))$ satisfies

$$s_2(h^*) - O\left(\frac{cn}{\min\{A(h), B(h)\}g(k,d)}\right) \leq s_2(h) \leq s_2(h^*) + O\left(\frac{cn}{\min\{A(h), B(h)\}g(k,d)}\right).$$

The error bounds for the Gini index were determined in a similar fashion.

A summary of the impurity measures and the error bounds is given in Table 4. All but the Gini index are stated as maximization problems.

Impurity Measure	Definition	Error Bound
Twoing Criterion	$a(h)b(h)(\sum_{i=1}^{c} \lvert a_i(h) - b_i(h) \rvert)^2$	$O(\frac{cn}{\min\{A(h),B(h)\}g(k,d)})$
Gini Index	$a(h)(1 - \sum_{i=1}^{c} a_i(h)^2) + b(h)(1 - \sum_{i=1}^{c} b_i(h)^2)$	$O(\frac{cn}{\min\{A(h),B(h)\}g(k,d)})$
Max Minimum	$\min\{\max_i A_i(h), \max_j B_j(h)\}$	$O(n/g(k,d))$
Sum Correct	$\max_i A_i(h) + \max_j B_j(h)$	$O(n/g(k,d))$

5 Empirical Results

In this section we describe the implementation of the data-sensitive method for choosing a starting hyperplane (prior to hill climbing) and give some comparisons of the accuracies of the resultant decision trees produced by this method relative to those produced by the uniform method. The OC1 decision tree software package [18] was used to produce the decision trees for each of these starting methods. In all cases, the data-sensitive approach produces decision trees that are comparable in terms of match qualities to the uniform approach.

5.1 The Data Sensitive Hyperplane Generation Method

The method of generating a data-sensitive hyperplane used for these experiments is to choose the hyperplane that is defined by d randomly chosen points from the set of current points to be classified. In this implementation, sets of d points would be chosen from the current set until a linearly independent set was obtained. The hyperplane defined by these points was then constructed using Gaussian elimination with backward substitution. If, in choosing the points, a pre-defined number of attempts were made before a linerly independent set was found, the algorithm would revert to the uniform method of generating the hyperplane randomly. This pre-defined limit was 25 for all experiments presented below and this limit was rarely reached.

5.2 The Experimental Setup

Each run of the software consisted of a five-fold cross-validation experiment. In such an experiment, the learning data were randomly partitioned into five equal parts. Four of the five parts would be used to construct a decision tree and the remaining part would be used to assess the quality of the tree produced. In the five-fold cross validation experiment, each one of the five parts would be kept, in turn, as the part to be used for testing and the other four would be used to construct the tree. The accuracies of the five different passes would then be averaged to determine the quality of the run. In the experiments below, the accuracies of 100 such five-fold cross-validation experiments were averaged to obtain each point plotted.

5.3 Experiment 1

Experiment 1 was constructed to determine the extent to which the classification accuracy of the decision trees produced by the data-sensitive approach would compare to (or track with) those produced by the uniform hyperplane generation approach. Three different data sets were used and the relative performances of the two methods were comparable in all cases.

5.3.1 Experiment 1 - Data Set 1

Data Set 1 consists of three hundred three-dimensional points such that one hundred were randomly selected with uniform distribution from the ball of 0.01 radius centered about the point (1, 0, 0), one hundred were similarly selected from an identical ball centered at (0, 1, 0), and the last third were selected

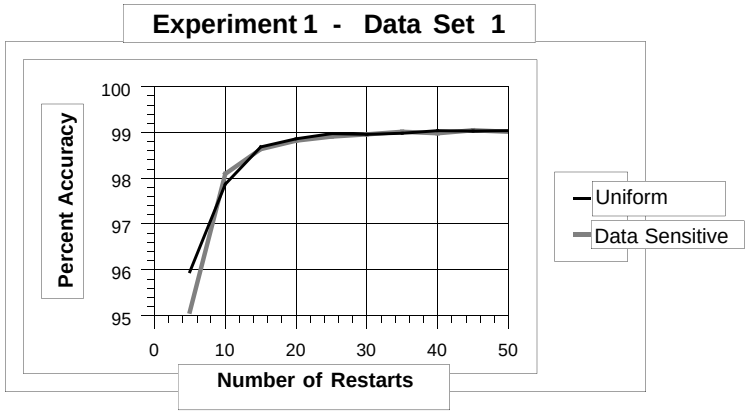

Figure 2: Comparison of accuracies of the data-sensitive starting method to the uniform starting method for Data Set 1.

Figure 3: Comparison of accuracies of the data-sensitive starting method to the uniform starting method for Data Set 2.

from the ball centered at (0, 0, 1). The set of points was then partitioned into two classes by a separating plane perpendicular to the vector (1, 1, 1) that caused half of the points (150) to be below this plane and the other half to be above this plane. Notice that this plane cuts through each of the three balls from which the points were selected. For the points that were randomly generated for this experiment, those selected from the ball on the x-axis were split 48/52 by this separating plane, those from the ball on the y-axis were split 50/50 and those from the ball on the z-axis were split 52/48.

The OC1 decision-tree software was run to classify this point-set using both uniformly selected initial hyperplanes and initial hyperplanes chosen by the data-sensitive method. The number of starting hyperplanes was varied from 5 through 50 in steps of 5. At each such number of starting hyperplanes, one hundred five-fold cross-validation experiments were run using each of the two methods, the accuracies were then averaged and the results are plotted in Figure 2. The overall average of the accuracies of the data-sensative starting method is 98.46 and the overall average of those of the uniform starting method is 98.54 which, in conjunction with the extremely tight point for point tracking of the two methods, indicates that there is no degradation in classification performance for this data set.

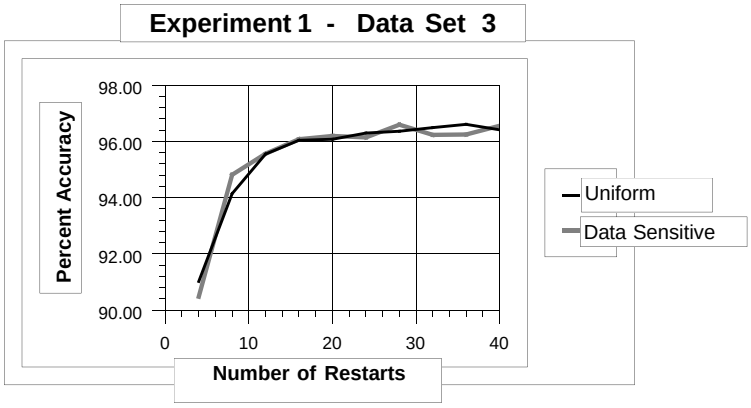

Figure 4: Comparison of accuracies of the data-sensitive starting method to the uniform starting method for Data Set 3.

5.3.2 Experiment 1 - Data Set 2

Data Set 2 consisted of three hundred three-dimensional points that were selected with uniform probability from the set of points that are a distance one from the origin and have positive values in all three dimensions. Thus the points were selected from one eighth of the surface of the described sphere. This data set is a variation of the width-zero uniform annulus described by Bentley [6]. The set of points was then partitioned into two classes by a separating plane perpendicular to the vector (1, 1, 1) that caused two hundred of the points to be below this plane and the other one hundred to be above this plane.

The OC1 software was again used to construct trees using both starting methods. The number of starting hyperplanes was varied from 2 through 20 in steps of 2. At each such number of starting hyperplanes, one hundred five-fold cross-validation experiments were run using each of the two methods, the accuracies were then averaged and the results are plotted in Figure 3. The overall average of the accuracies of the data-sensative starting method is 96.73 and the overall average of those of the uniform starting method is 96.67.

5.3.3 Experiment 1 - Data Set 3

Data Set 3 consists of three hundred points chosen with uniform distribution from the sphere of radius 0.5 centered at (0.5, 0.5, 0.5). The points were then divided into four regions as follows. Three planes perpendicular to the vector (1, 1, 1) were selected so that 75 points fell below all three planes, 75 points fell between the first and second planes, 75 points fell between the second and third and 75 points were above all three planes. The class assigned to the points in the first region was 1, second region 2, third region 1 and fourth region 2.

The OC1 software was again used to construct trees using both starting methods. The number of starting hyperplanes was varied from 4 through 40 in steps of 4. At each such number of starting hyperplanes, one hundred five-fold cross-validation experiments were run using each of the two methods, the accuracies were then averaged and the results are plotted in Figure 4. The overall average of the accuracies of the data-sensative starting method is 95.49 and the overall average of those of the uniform starting method is 95.50.

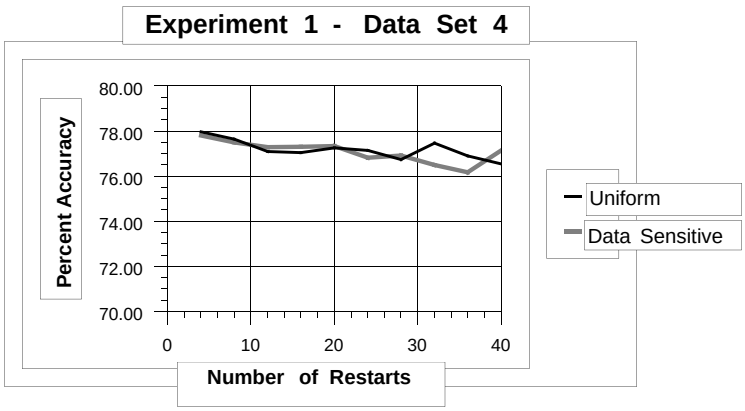

Figure 5: Comparison of accuracies of the data-sensitive starting method to the uniform starting method for Data Set 4.

5.3.4 Experiment 1 - Data Set 4

Data Set 4 is the *auto mpg* data available from the *Machine Learning Database Repository* at the University of California at Irvine (*ftp.ics.uci.edu*). This data is seven-dimensional data in which instances fall into one of three classes. The OC1 software was used to construct trees using both starting methods. The number of starting hyperplanes was varied from 4 through 40 in steps of 4. At each such number of starting hyperplanes, one hundred five-fold cross-validation experiments were run using each of the two methods, the accuracies were then averaged and the results are plotted in Figure 5. The overall average of the accuracies of the data-sensative starting method is 77.07 and the overall average of those of the uniform starting method is 77.18.

5.4 Experiment 2

The process of hill-climbing [8] is a greedy method for decreasing the impurity measure of a particular step in the construction of a decision tree. Experiment 2 was constructed to observe the extent to which the (local) process of hill-climbing effects the quality of the (global) decision trees produced. In experiment 2, the method of uniform starting and the method of data-sensitive starting were used both with and without hill-climbing to produce decision trees. The number of starting hyperplanes was varied from 10 through 100 in increments of 10, and at each of these points the accuracies of 100 five-fold cross-validation experiments were averaged to give the values presented in Figure 6. The overall average of the accuracies of the uniform starting method without hill-climbing is 77.38 and with hill climbing is 98.23. The overall average of the accuracies of the data-sensitive starting method without hill-climbing is 77.45 and with hill climbing is 98.20.

The data set that was used is similar to the Data Set 1 from Experiment 1 with only a few variations. Again, three-dimensional points were uniformly selected from each of three balls centered at point 1 on a positive axis. The balls in this experiment had a radius of 0.1, which is ten times the radius of the balls in Data Set 1, and the total number of points in this data set is 150 (50 from each ball), as opposed to three hundred of Data Set 1. The points were then partitioned into two equal groups by a plane perpendicular to the vector (1, 1, 1). For the points that were generated for this experiment, the partition caused the points from the ball on the x-axis to be split 26/24, those from the ball on the y-axis to be split 22/28 and those from the ball on the z-axis to be split 27/23.

Figure 6: Accuracy of the decision tree for both starting methods with and without hill-climbing.

Acknowledgement

We would like to thank the U.S. Army Research Office for their support. The views, opinions, and/or findings contained in this document are those of the authors and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

References

[1] D. Angluin. Computational learning theory: Survey and selected bibliography. In *Proc. 24th ACM Symp. on Theory of Computing*, pages 351–369, 1992.

[2] E. M. Arkin, H. Meijer, J. S. B. Mitchell, D. Rappaport, and S. S. Skiena. Decision trees for geometric models. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 369–378, 1993.

[3] Esther M. Arkin, Michael T. Goodrich, Joseph S. B. Mitchell, David Mount, Christine D. Piatko, and Steven S. Skiena. Point probe decision trees for geometric concept classes. In *Proc. 3rd Workshop Algorithms Data Struct.*, volume 709 of *Lecture Notes in Computer Science*, pages 95–106. Springer-Verlag, 1993.

[4] Kristin P. Bennett. Decision tree construction via linear programming. In *Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society Conference*, pages 97–101, 1992.

[5] Kristin P. Bennett. Global tree optimization: A non-greedy decision tree algorithm. In *Proceedings of Interface 94: The 26th Symposium on the Interface*, Research Triangle, North Carolina, 1994.

[6] J. L. Bentley. K-d trees for semidynamic point sets. In *Proc. 6th Annu. ACM Sympos. Comput. Geom.*, pages 187–197, 1990.

[7] A. Blum and R. L. Rivest. Training a 3-node neural net is NP-Complete. *Neural Networks*, 5:117–127, 1992.

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

[9] B. Chazelle. Cutting hyperplanes for divide-and-conquer. Manuscript, Princeton Univ., Princeton, NJ, 1991.

[10] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

[11] G. Das and M. T. Goodrich. On the complexity of approximating and illuminating three-dimensional convex polyhedra. Technical Report, Dept. Computer Science, Johns Hopkins Univ., Baltimore, MD, 1995. Also under submission to WADS 95.

[12] Thomas R. Hancock. Learning decision trees on the uniform distribution. In *Sixth ACM Conference on Computational Learning Theory (COLT 93)*, page ??, July 1993.

[13] D. Heath, S. Kasif, and S. Salzberg. Learning oblique decision trees. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1002–1007, Chambery, France, 1993. Morgan Kaufmann.

[14] Hyafil and Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5, 1976.

[15] S. Judd. On the complexity of loading shallow neural networks. *J. of Complexity*, 4:177–192, 1988.

[16] J. Matoušek. Epsilon-nets and computational geometry. In J. Pach, editor, *New Trends in Discrete and Computational Geometry*, volume 10 of *Algorithms and Combinatorics*, pages 69–89. Springer-Verlag, 1993.

[17] N. Megiddo. On the complexity of polyhedral separability. *Discrete Comput. Geom.*, 3:325–337, 1988.

[18] Sreerama K. Murthy, Simon Kasif, and Steven Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–33, August 1994.

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

[20] J. R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221–234, 1987.

[21] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[22] S. Rasoul Safavin and David Landgrebe. A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man and Cybernetics*, 21(3):660–674, May/June 1991.

[23] S. Salzberg, R. Chandar, H. Ford, S. K. Murthy, and R. White. Decision trees for automated identification of cosmic rays in hubble space telescope images. *Publications of the Astronomical Society of the Pacific*, To appear, March 1995.