INFORMS Journal on Computing
Vol. 17, No. 1, Winter 2005, pp. 25–31
ISSN 0899-1499 | EISSN 1526-5528 | 05 | 1701 | 0025

DOI 10.1287/ijoc.1030.0047

Oblique Multicategory Decision Trees Using Nonlinear Programming

W. Nick Street
Management Sciences Department, S232 Pappajohn Business Building, University of Iowa,
Iowa City, Iowa 52242, USA, nick-street@uiowa.edu

Induction of decision trees is a popular and effective method for solving classification problems in data-mining applications. This paper presents a new algorithm for multi-category decision tree induction based on nonlinear programming. This algorithm, termed OC-SEP (Oblique Category SEParation), combines the advantages of several other methods and shows improved generalization performance on a collection of real-world data sets.

Key words: artificial intelligence; programming; nonlinear; applications; machine learning; decision trees
History: Accepted by Amit Basu, Area Editor; received July 2000; revised March 2001, August 2002, June 2003; accepted June 2003.

1. Introduction

As our ability to collect massive amounts of data increases, the fields of machine learning and data mining take on greater importance. We define data mining (Fayyad et al. 1996) as the process of gleaning actionable knowledge and insight from these large data sets. The algorithmic process of building predictive models from the data is an important step in data mining. This step is often performed using techniques from machine learning that allow categorization, prediction, and summarization of the data.

Many problems require classification of examples into one of several categories. For example, we may wish to determine which patients have cancer based on the results of medical tests, which customers are likely to purchase insurance based on demographic data, or which Web documents might interest a certain user. The field of inductive machine learning offers many choices for building these predictive models, such as decision trees, artificial neural networks, and support-vector machines. All of these models generalize the information about cases with known classifications (the *training* examples) to predict the outcome of new cases.

The inductive learning step involves minimization of some error metric to build a model that classifies all or most of the known cases correctly. This underlying optimization problem has lead to the application of mathematical-programming methods in learning. Examples include the use of linear programming in building decision trees (Bennett 1992), quadratic programming for support-vector machines (Vapnik 1995), and conjugate gradient and quasi-Newton methods for artificial neural networks (Hertz et al. 1991). A comprehensive review of the role of mathematical programming in data mining is given in Bradley et al. (1999).

The focus of this work is on the popular decision-tree model. Decision trees can be induced by constructing one or more separating surfaces (typically, hyperplanes) in the feature space of the training examples. This separation is represented as a decision node in the tree. The algorithm then recursively partitions the subsets of data in the various regions formed by the previous step, until all (or nearly all) of the points in a region belong to the same class. For example, the left of Figure 1 shows a simple classification problem with two classes in two dimensions. On the right is the resulting decision tree, using two separating planes. Note that the planes in this example are oblique, that is, not parallel to a coordinate axis. A decision-tree algorithm such as C4.5 that uses only axis-parallel planes would require several more planes, resulting in a larger tree. Decision trees are widely used because of their simplicity, training speed, and consistently good predictive accuracy on a wide variety of problems. Further, the representation of the learned concept as a combination of simple rules allows for relatively easy interpretation by the human decision maker.

This paper presents a new method called OC-SEP for the construction of decision trees based on nonlinear programming. The remainder of the paper is organized as follows. Section 2 reviews the decision-tree literature and presents a set of criteria for our algorithm. Section 3 motivates and presents the nonlinear-programming model. In §4 we present a collection of computational results demonstrating the

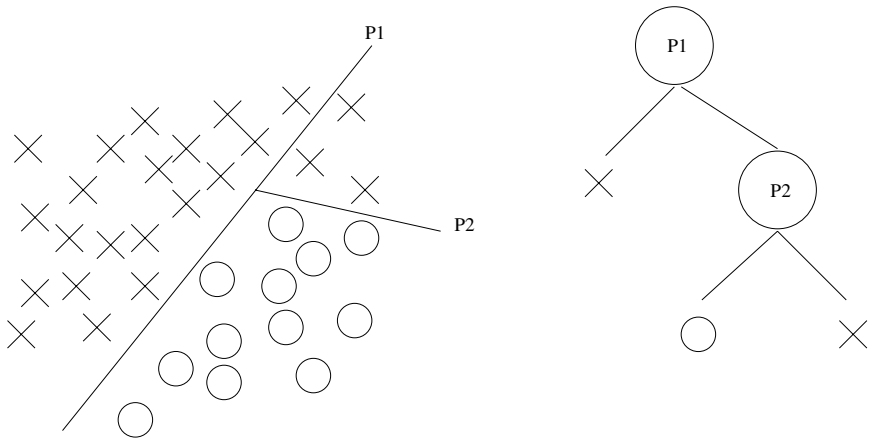

Figure 1 Decision-Tree Example

effectiveness of OC-SEP on real-world data and comparing the run times of the various decision-tree algorithms. Section 5 concludes the paper and discusses future directions for this research.

2. Decision Trees

The seminal work in automatic induction of decision trees comes from two sources. In the artificial intelligence community, the original and still most popular system is Quinlan's ID3 (Quinlan 1986) and its successor C4.5 (Quinlan 1993). In the statistics literature, the CART system (Breiman et al. 1984) is similarly prevalent. Since these systems are very similar in their inductive biases, we discuss the C4.5 approach as representative of their capabilities.

C4.5 examines one feature at each decision node, thereby partitioning the feature space using axis-parallel separating planes. The feature on which to split is chosen by maximizing the *information* gained in the split. Information gain is an entropy measure designed to increase the average class purity of the resulting subsets. Once a feature is chosen, a split is made between all possible values of a features, or at discretization boundaries for continuous features. Therefore, a decision node can have many children. This tends to bias the resulting tree in favor of features with a large number of possible values.

C4.5 and CART are computationally efficient and have proven very successful in practice, comparing favorably to other inductive-learning techniques such as artificial neural networks (Mooney et al. 1989). However, the fact that (as a default) they are limited to constructing axis-parallel separating planes limits their effectiveness in domains where some combination of features is highly predictive of the target concept.

A number of later algorithms address this shortcoming. Representative of these is the OC1 family of classifiers (Murthy et al. 1994), which uses a randomized greedy optimization scheme to construct potentially oblique separating planes. The OC1 system allows a number of different separation objectives, all based on some measure of class purity in the resulting child nodes.

Limitations of purity-based optimization criteria were addressed by Fayyad and Irani (1992). They noted that purity criteria such as information gain are not sensitive to class separation because they measure it only indirectly, often resulting in trees with more nodes and worse generalization performance. They introduced a new class of measures based on the orthogonality of the class vectors in the child nodes. The *class vector* of a set $\mathcal{S}$ of examples is defined as $[c_1/|\mathcal{S}|, c_2/|\mathcal{S}|, \ldots, c_k/|\mathcal{S}|]$, where c_i is the number of examples of class i in $\mathcal{S}$. Intuitively, we would like a decision-tree node to separate some of the classes from the others, while keeping the examples of like classes together. In other words, the class vectors of sibling nodes should be as nearly orthogonal as possible. Based on this observation Fayyad and Irani defined a family of measures called C-SEP (Class SEParation) and demonstrated the effectiveness of one such measure, the cosine of the angle between the two vectors. (In a slight abuse of their terminology, we use the name OC-SEP for a particular objective and the resulting tree-induction algorithm.) The search method used in their algorithm was similar to that of C4.5, limiting concept representations to axis-parallel decision boundaries.

Fayyad (1991) also proved that limiting decision trees to binary splits results in no loss of generality, and showed empirically that such a constraint usually creates trees with fewer leaves and better generalization. We adopt this heuristic for oblique planes as well, since n-ary splits for $n > 2$ may tend to produce unnecessary tree nodes.

The optimal construction of decision boundaries has a long history in the mathematical-optimization

community. Since the 1960s, linear programming has been used to construct piecewise-linear and piecewise-nonlinear surfaces that separate two classes of points (Mangasarian 1965, 1968; Glover 1990). Typically, these systems minimize some measure of the distance of the misclassified points from the separating plane, although more recent techniques (Mangasarian 1994, Chen and Mangasarian 1996, Bennett and Bredensteiner 1997) directly minimize the number of such points. A natural extension of these techniques to decision-tree construction, known as MSM-T, was introduced by Bennett (1992). Direct application of these methods is limited to two-class problems.

The preceding discussion identified a number of dimensions along which the various decision-tree algorithms differ. They can be summarized as follows.

- *Oblique vs. axis-parallel planes*: Oblique decision boundaries offer greater flexibility, resulting in smaller trees, and remove the need for discretization of continuous features.
- *Binary vs. n-ary trees*: Binary trees provide equal representational power and do not suffer from the preference for splitting on many-valued features.
- *Two-category vs. multicategory classification*: A technique designed specifically for binary classification requires some form of ad-hoc extension for a problem with k classes, such as a collection of k two-class trees, each of which separates one class from all the others.
- *Optimization objective*: Using orthogonality as the criterion in the optimization step holds several advantages over purity-based measures for multiclass problems, as described above. In turn, purity tends to construct smaller trees than does direct misclassification minimization, at least when used at nodes near the root (Brodley 1995).
- *Optimization method*: When constructing oblique trees, powerful and computationally efficient techniques of linear and nonlinear programming reduce the problem of local minima.

Table 1 summarizes the above systems along these dimensions but is not meant to be a complete list of the design criteria for decision-tree algorithms. For example, once a full decision tree is built, some of the nodes are usually removed or "pruned" to reduce the possibility of overfitting. Several different methods, such as cost-complexity pruning (Breiman et al. 1984) and pessimistic-error pruning (Quinlan 1993), have been proposed for this task. Further, any algorithm must choose the single feature or subset of features used to split the data at a particular node; see for example (Bradley et al. 1998) for a mathematical-programming formulation that reduces the dimensionality of a hyperplane separator. While important, we consider these issues to be secondary to the basic structure of the separating surface, and the optimization method and objective used to create the surface. Therefore, the goal of the work presented here is to create multicategory binary decision trees with oblique planes, using a nonlinear-programming approach to the optimization of an orthogonality-based objective function.

Table 1 Classification of Decision-Tree Methods Along Relevant Dimensions

	C4.5	OC1	C-SEP	MSM-T
Oblique planes	No	Yes	No	Yes
Binary trees	No	Yes	Yes	Yes
Multicategory	Yes	Yes	Yes	No
Orthogonality objective	No	No	Yes	No
LP or NLP-based optimization	n/a	No	n/a	Yes

3. Oblique Category Separation (OC-SEP)

3.1. Motivation: Separation via Mathematical Programming

Our new approach to multicategory separation is built on the foundation of previous methods based on mathematical programming. In particular, the Robust Linear Program (RLP) (Bennett and Mangasarian 1992, Mangasarian 1993) for two-class separation is summarized here, both as a groundwork for our method and as an introduction to the terminology used.

Consider two disjoint sets $\mathcal{A}_1$ and $\mathcal{A}_2$ that we wish to discriminate. The available training points from $\mathcal{A}_1$ and $\mathcal{A}_2$ can be collected as matrices A_1 and A_2, with each of the n_1 rows of A_1 containing the feature values for one training example from set $\mathcal{A}_1$, and A_2 likewise containing the n_2 points of $\mathcal{A}_2$. The goal is to construct a separating plane $x^T w = \theta$ in the feature space of these examples such that all the points of A_1 lie on one side of the plane (say, $A_1 w > e\theta$, where e is a vector of ones of appropriate dimension) and all the points of A_2 lie on the other ($A_2 w < e\theta$). This will be possible only if A_1 and A_2 are linearly separable, which in general is not the case. We therefore choose to minimize the average distance between the plane and the misclassified points. This is achieved with the following normalized minimization problem:

$$\underset{w,\,\theta}{\text{minimize}} \quad \frac{1}{n_1}\|(-A_1 w + e(\theta+1))_+\|_1 + \frac{1}{n_2}\|(A_2 w - e(\theta-1))_+\|_1, \qquad (1)$$

where $\|\cdot\|_1$ denotes the 1-norm and $(z)_+$ denotes $((z)_+)_i = \max\{z_i, 0\}$, $i = 1, \ldots, m$, for $z \in R^m$.

This objective can be shown to be equivalent to the following linear program:

$$\begin{aligned} \underset{w,\theta,y,z}{\text{minimize}} \quad & \frac{ey}{n_1}+\frac{ez}{n_2} \\ \text{subject to} \quad & A_1w-e\theta+y\geq e \\ & -A_2w+e\theta+z\geq e \\ & y,z\geq 0. \end{aligned} \tag{2}$$

This "robust linear program" has a number of favorable properties, including:

- If A_1 and A_2 are linearly separable, a separating plane is found.
- The null solution $w=0,\ \theta=0$ is never unique, and is obtained only when the centroids of the two point sets are equal.
- Use of the 1-norm error, rather than the more traditional 2-norm error, makes (2) more resistant to the effects of outliers.

We note that this approach closely resembles the popular support-vector machine method (Vapnik 1995). This relationship is explored by Bennett and Bredensteiner (2000) and Bradley et al. (1999).

The linear program (2) can be recursively solved on subsets of the original training set to construct a decision tree, as was done in Bennett (1992). However, there is no clear extension of this approach to the multicategory case, since we do not know a priori which classes should be separated by each plane, and on which side of the plane each class should lie.

3.2. Multicategory Separation

As mentioned, mathematical-programming-based separation methods typically specify in the objective which class should be on each side of the separating plane. A direct extension of this idea to the multicategory case uncovers a combinatorial problem: Which subset of the k classes should be on the "left" of the plane? Clearly it is computationally infeasible to try all possible subsets. Instead, we move from a misclassification-minimization approach to the orthogonality measure.

The foundation of the orthogonality-based separation criterion used in C-SEP is the class vector, which contains the count of elements of each class in the given subset of examples. As in C-SEP, we attempt to maximize the orthogonality of the class vectors induced by a separating plane at a particular node. However, since we wish to use continuous optimization methods, an explicit count of the elements to produce the class vectors will not be available. Therefore, as in previous MP methods, we will approximate these counts with a distance measurement. The value $e^T((A_1w-e\theta)_+)$ is proportional to the total distance from the plane of all points in A_1 that are on the right (greater-than) side of the plane; similarly, $e^T((-A_1w+e\theta)_+)$ gives a distance measure for the points on the left. We can therefore obtain a suitable substitute for class vectors by applying this equation to each A_i. To simplify the following objectives, we use the term v_1 to represent this vector, which has k components $v_{1,i}=e^T((A_iw-e\theta)_+)$. The other class vector is similarly denoted v_2.

Our first attempt at an orthogonality objective is therefore the inner product of the two class vectors formed with these approximations:

$$\begin{aligned} \underset{w,\theta}{\text{minimize}} \quad & v_1^Tv_2 \\ \text{subject to} \quad & e^T((A_iw-e\theta)_+)=v_{1,i}, \quad i=1,\ldots,k \\ & e^T((-A_iw+e\theta)_+)=v_{2,i}, \quad i=1,\ldots,k. \end{aligned} \tag{3}$$

The above objective will obtain a minimum value of zero whenever the two class vectors v_1 and v_2 are orthogonal; this occurs when the points of each class are either entirely on the right or entirely on the left of the plane $x^Tw=\theta$. This is the desired effect. Unfortunately, this minimum is also obtained by a plane that lies entirely on one side of all the points; for instance, $v_{1,i}>0,\ v_{2,i}=0$. We therefore need to modify the mathematical program (3) in some way to ensure that this result is not obtained, and that the resulting plane accomplishes some degree of pattern separation.

This is accomplished by modifying the objective function. Ideally, we would like to have at least one class with a majority of its points on the left side of the plane, and at least one class with a majority on the right. In other words, we would like to have one of the elements of v_1 be large, and one element of v_2 be large. This is equivalent to saying that one element of both v_1 and v_2 should be small. This can be enforced by minimizing the product of the elements of both vectors. These terms prevent the placement of all points on one side of the plane. The new MP is therefore

$$\begin{aligned} \underset{w,\theta}{\text{minimize}} \quad & v_1^Tv_2+\prod_{i=1}^{k}v_{1,i}+\prod_{i=1}^{k}v_{2,i} \\ \text{subject to} \quad & e^T(A_iw-e\theta)_+=v_{1,i}, \quad i=1,\ldots,k \\ & e^T(-A_iw+e\theta)_+=v_{2,i}, \quad i=1,\ldots,k. \end{aligned} \tag{4}$$

The class vectors v_1 and v_2 tend to contain larger values for classes with many points, which leads the mathematical program (4) to favor these large classes when performing the optimization. To overcome this problem, the class vectors are normalized so that each $v_{1,i}+v_{2,i}=1$. The element $v_{1,1}$ is thus interpreted as the percentage of the total distance of points in class 1 from the plane that lie on the left of the plane.

One final change is necessary to make the MP computationally reliable. Since we are performing continuous optimization, the objective function needs to

be differentiable everywhere. This is clearly not the case because of the use of the plus function, which is not differentiable at zero. We therefore substitute the p-function (Chen and Mangasarian 1995), the integral of the sigmoid function, for the plus function:

$$p(x) = x + \frac{1}{\alpha}\log(1 + \epsilon(-\alpha x)), \tag{5}$$

where ϵ is the base of the natural logarithm and α is some predefined constant. This differentiable function approximates the plus function arbitrarily well, with the approximation improving for larger values of α. An α value of 10,000 was used in our experiments. This large value makes the approximation virtually indistinguishable from the plus function. The maximum difference occurs at $x = 0$, with a difference of 0.000069. At $x = \pm 0.001$, the approximation is correct to eight decimal places.

The final version of the OC-SEP MP therefore becomes

$$\begin{aligned}
\underset{w,\theta}{\text{minimize}} \quad & v_1^T v_2 + \prod_{i=1}^{k} v_{1,i} + \prod_{i=1}^{k} v_{2,i} \\
\text{subject to} \quad & p(e^T(A_i w - e\theta)) = u_{1,i}, \quad i = 1, \ldots, k \\
& p(e^T(-A_i w + e\theta)) = u_{2,i}, \quad i = 1, \ldots, k \\
& \frac{u_{1,i}}{(u_{1,i} + u_{2,i})} = v_{1,i}, \quad i = 1, \ldots, k \\
& \frac{u_{2,i}}{(u_{1,i} + u_{2,i})} = v_{2,i}, \quad i = 1, \ldots, k.
\end{aligned} \tag{6}$$

The MP (6) is solved using the sequential quadratic programming method (Biggs 1975) as implemented in the optimization toolbox of the Matlab software package (Coleman et al. 1999).

4. Computational Results

4.1. Classification Accuracy

The OC-SEP method was tested by estimating its predictive accuracy on a collection of data sets from the UCI Machine Learning Repository (Blake and Merz 1998). These results were compared to estimates obtained using C4.5, OC1, and MSM-T. The chosen data sets contain various numbers of numerical attributes, and contain from two to ten classes. The characteristics of the different data sets are shown in Table 2.

All performance estimates were obtained using ten-fold cross-validation (Stone 1974). In this procedure, the data set is partitioned randomly into ten subsets, each maintaining the approximate class distribution of the original set. A classifier is built using nine of the subsets, and tested on the tenth. This is repeated ten times, using each of the subsets in turn as the test set. The average accuracy of these tests is an unbiased estimate of out-of-sample generalization performance. We performed five cross-validation tests for each data set and each classifier. The resulting averages are compared using a difference-of-means test for statistical significance. Note that MSM-T is a two-class decision-tree system, so it was tested only on the two-class problems. Tests were performed using the default tree-pruning settings of each algorithm, and with no pruning. Both MSM-T and OC-SEP use the C4.5 pruning algorithm.

Table 2 Characteristics of Data Sets Used in Computational Comparisons

Data Set	Examples	Features	Classes
Pima Indians Diabetes	768	8	2
Wisconsin Breast Cancer (WBC)	699	9	2
Wisconsin Diagnostic Breast Cancer (WDBC)	569	30	2
Wisconsin Prognostic Breast Cancer (WPBC)	188	32	2
Ionosphere	351	34	2
Sonar	208	60	2
Iris	150	4	3
New thyroid	215	5	3
Wine	178	13	3
Glass	214	9	6
Image segmentation	2,310	19	7
Vowel	900	10	10

Table 3 shows the accuracy of each method on the 12 data sets using the default tree-pruning settings for each method. Entries are mean results of five ten-fold cross-validation runs plus or minus one standard deviation. Superscript 1 indicates performance statistically significantly worse than OC-SEP; superscript 2 indicates significantly better. The best performance on each data set is in boldface type. OC-SEP showed the best performance on six of the 12 data sets, and the worst performance on only one. In head-to-head

Table 3 Accuracy of Decision-Tree Methods with Default Pruning

Data set	OC-SEP	C4.5	OC1	MSM-T
Pima	70.05 ± 1.70	71.48 ± 1.43	$\mathbf{72.14} \pm 0.91^2$	69.13 ± 0.85
WBC	$\mathbf{96.40 \pm 0.08}$	95.04 ± 0.32^1	95.56 ± 0.48^1	95.20 ± 0.52^1
WDBC	$\mathbf{96.10 \pm 0.40}$	95.66 ± 0.63	95.61 ± 0.54	94.55 ± 0.28^1
WPBC	68.62 ± 2.08	68.94 ± 3.37	$\mathbf{75.32 \pm 2.63}^2$	65.53 ± 4.52
Ionosphere	85.93 ± 1.15	$\mathbf{90.38 \pm 0.84}^2$	87.63 ± 1.88	84.62 ± 2.37
Sonar	$\mathbf{76.92 \pm 1.23}$	67.82 ± 3.02^1	68.27 ± 3.74^1	74.81 ± 1.88
Iris	95.60 ± 0.76	94.92 ± 1.01	$\mathbf{96.53 \pm 0.73}$	
New thyroid	$\mathbf{99.16 \pm 0.21}$	93.18 ± 1.13^1	91.56 ± 2.87^1	
Wine	$\mathbf{94.66 \pm 1.48}$	91.76 ± 3.13	90.34 ± 1.88^1	
Glass	60.28 ± 2.66	$\mathbf{69.52 \pm 2.06}^2$	67.57 ± 3.48^2	
Segmentation	$\mathbf{95.05 \pm 0.31}$	88.30 ± 0.98^1	84.57 ± 1.80^1	
Vowel	79.69 ± 0.83	78.88 ± 1.04	$\mathbf{79.72 \pm 1.11}$	

Notes. Superscript 1 indicates performance statistically significantly worse than OC-SEP; superscript 2 indicates significantly better. The best performance on each data set is in boldface type.

Table 4 Accuracy of Decision-Tree Methods with No Pruning

Data set	OC-SEP	C4.5	OC1	MSM-T
Pima	**70.28 ± 1.68**	69.82 ± 0.90	68.70 ± 1.29	68.15 ± 0.56[1]
WBC	**95.37 ± 0.73**	94.22 ± 0.69[1]	94.08 ± 0.31[1]	93.88 ± 0.67[1]
WDBC	**95.47 ± 0.50**	94.62 ± 0.54[1]	94.94 ± 0.70	95.15 ± 0.77
WPBC	**68.83 ± 3.62**	67.70 ± 2.90	67.66 ± 1.89	65.02 ± 4.55
Ionosphere	86.26 ± 0.54	**91.60 ± 2.51**[2]	88.72 ± 1.64[2]	84.61 ± 1.47[1]
Sonar	**77.83 ± 1.83**	68.46 ± 2.19[1]	71.73 ± 3.81[1]	72.50 ± 2.32[1]
Iris	**95.47 ± 1.59**	95.16 ± 0.31	94.93 ± 1.01	
New thyroid	**99.26 ± 0.25**	93.00 ± 0.73[1]	95.35 ± 0.80[1]	
Wine	**94.77 ± 0.48**	92.76 ± 1.07[1]	92.25 ± 1.34[1]	
Glass	60.28 ± 1.72	68.08 ± 1.36[2]	**68.32 ± 1.38**[2]	
Segmentation	**95.10 ± 0.34**	88.02 ± 0.71[1]	85.43 ± 1.09[1]	
Vowel	79.13 ± 0.66	79.84 ± 0.89	**83.47 ± 0.70**[2]	

Notes. Superscript 1 indicates performance statistically significantly worse than OC-SEP; superscript 2 indicates significantly better. The best performance on each data set is in boldface type.

comparisons based only on those results with a significant difference, OC-SEP outperformed C4.5 4-2, OC1 5-3, and MSM-T 2-0. The OC-SEP results on the two-class problems were consistently better than those of MSM-T, supporting the expected superiority of the orthogonality-based objective. Comparisons with OC1 and C4.5 were mixed on the two-class problems but show an advantage for OC-SEP on the multicategory problems.

Table 4 shows the experimental results with pruning turned off, that is, the leaves of all trees contained only one class. Here, OC-SEP performed best on nine of the 12 problems. Looking at head-to-head comparisons, OC-SEP outperformed the three other algorithms: C4.5 6-2, OC1 5-3, and MSM-T 4-0. Surprisingly, pruning appears to offer little advantage for the MP-based algorithms, indicating that the optimization method itself provides some degree of overfitting avoidance.

In summary, OC-SEP appears to offer an advantage over C4.5 based on oblique separating planes and the optimization procedure, and over MSM-T based on the orthogonality-based objective. Further, the optimization method and objective appear to give OC-SEP a more qualified advantage over OC1.

4.2. Computation Time

Table 5 shows the average run-time in seconds for the various algorithms for the ten folds of a single cross-validation run. Tests were run on a 240 MHz Linux workstation.

As expected, the Matlab implementation of OC-SEP was significantly slower than the C implementations of the other multicategory algorithms, by about one (two) order(s) of magnitude compared to OC1 (C4.5). To get some feel for how much of this difference was due to the solver, we compared two different versions of MSM-T: one implemented in C using the MINOS LP solver (MSM-T C), and one in Matlab (MSM-T M).

Table 5 Comparative Run Times (in Seconds) of Decision-Tree Algorithms

Data set	OC-SEP	C4.5	OC1	MSM-T C	MSM-T M
Pima	814	44	5.9	6.1	5,119
WBC	126	29	1.7	1.0	3,493
WDBC	456	45	3.4	1.5	7,341
WPBC	2,331	17	1.8	2.2	1,379
Ionosphere	924	38	3.1	2.6	1,393
Sonar	1,149	13	3.0	1.9	1,447
Iris	13	2.0	0.1		
New thyroid	3.0	3.0	0.2		
Wine	22	4.0	0.3		
Glass	455	10	1.0		
Segmentation	2,832	12	0.9		
Vowel	1,318	138	46.7		

The difference between these two was consistently around two orders of magnitude, and in fact, the MSM-T runs took longer than OC-SEP in five of the six cases. We conclude that the constrained nonlinear optimization search method employed by OC-SEP makes it inherently more computationally demanding than decision-tree builders based on, say, greedy search or linear programming. Still, while the current implementation of OC-SEP is not feasible for large-scale problems, there is reason to believe that a more efficient version using compiled code and a better solver might be.

We also examined the trends in run time against various characteristics of the problem: size (number of features × number of examples), number of classes, and predictive accuracy. The Matlab-based programs showed a strong positive correlation between run time and problem size, while the others did not, again indicating that the speed is largely dependent on the solver. OC-SEP showed the least dependence of the three multicategory algorithms on the number of classes. Finally, all algorithms except for MSM-T M were slightly faster on easier problems.

5. Conclusions and Future Work

This paper presented OC-SEP, a new method for the construction of oblique decision trees based on a mathematical-programming formulation. The method combines the advantages of several well-known decision-tree-induction programs and achieves generally superior classification accuracy, particularly on multicategory problems.

Further refinement of the OC-SEP method will improve its utility as a general tool for classification. Certainly the objective function is not as immediately intuitive as the simpler formulations based on misclassification distance or entropy. We are experimenting with ways to simplify the objective without sacrificing the generalization accuracy. Moreover, the optimization procedure is relatively slow when

compared to the other methods. Future work will focus on the search for specialized optimization methods that take advantage of the particular objective structure, making the induction procedure feasible for larger data sets. The method might also benefit from some heuristic technique for determining a starting point for the optimization problem. Further analysis is also needed to specify the types of problems on which OC-SEP performs particularly well (such as the New thyroid problem) or particularly poorly (such as Glass). Finally, both the generalization accuracy and the interpretability of the learned models could be enhanced by reducing the number of predictive features used at each decision node. We will incorporate a feature-selection term into the objective function to achieve this improvement.

Acknowledgments

This work was funded in part by NSF Grant IIS-99-96044. The author wishes to thank Mark Hanson, Danni Jiao, and Zhang Yi for their assistance with the computational experiments.

References

Bennett, K. P. 1992. Decision tree construction via linear programming. M. Evans, ed. *Proc. 4th Midwest Artificial Intelligence and Cognitive Sci. Soc. Conf.*, Midwest Artificial Intelligence and Cognitive Science Society, Utica, IL, 97–101.

Bennett, K. P., E. J. Bredensteiner. 1997. A parametric optimization method for machine learning. *INFORMS J. Comput.* **9** 311–318.

Bennett, K. P., E. J. Bredensteiner. 2000. Geometry in learning. C. Gorini, ed. *Geometry at Work*, Mathematical Association of America, Washington, DC, 132–145.

Bennett, K. P., O. L. Mangasarian. 1992. Robust linear programming discrimination of two linearly inseparable sets. *Optim. Methods Software* **1** 23–34.

Biggs, M. C. 1975. Constrained minimization using recursive quadratic programming. L. C. W. Dixon, G. P. Szergo, eds. *Toward Global Optimization*. North-Holland, Amsterdam, The Netherlands, 341–349.

Blake, C. L., C. J. Merz. 1998. UCI repository of machine learning databases. Department of Information and Computer Sciences. University of California, Irvine, CA. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Bradley, P. S., U. M. Fayyad, O. L. Mangasarian. 1999. Mathematical programming for data mining: Formulations and challenges. *INFORMS J. Comput.* **11** 217–238.

Bradley, P. S., O. L. Mangasarian, W. N. Street. 1998. Feature selection via mathematical programming. *INFORMS J. Comput.* **10** 209–217.

Breiman, L., J. Friedman, R. Olshen, C. Stone. 1984. *Classification and Regression Trees*. Wadsworth, Inc., Pacific Grove, CA.

Brodley, C. E. 1995. Automatic selection of split criterion during tree growing based on node location. A. Prieditis, S. Russell, eds. *Proc. 12th Internat. Conf. Machine Learning*, Morgan Kaufmann, San Francisco, CA, 73–80.

Chen, C., O. L. Mangasarian. 1995. Smoothing methods for convex inequalities and linear complementarity problems. *Math. Programming* **71** 51–69.

Chen, C., O. L. Mangasarian. 1996. Hybrid misclassification minimization. *Adv. Comput. Math.* **5** 127–136.

Coleman, T., M. A. Branch, A. Grace. 1999. *Optimization Toolbox for Use with MATLAB*. The MathWorks, Inc., Natick, MA.

Fayyad, U. M. 1991. On the induction of decision trees for multiple concept learning. Ph.D. thesis, Electrical Engineering and Computer Science Department, The University of Michigan, Ann Arbor, MI.

Fayyad, U. M., K. Irani. 1992. The attribute selection problem in decision tree generation. *Proc. 11th National Conf. Artificial Intelligence*, MIT Press, San Jose, CA, 322–327.

Fayyad, U. M., G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, eds. 1996. *Adv. Knowledge Discovery Data Mining*. AAAI Press/The MIT Press, Cambridge, MA.

Glover, F. 1990. Improved linear programming models for discriminant analysis. *Decision Sci.* **21** 771–785.

Hertz, J., A. Krogh, R. G. Palmer. 1991. *Introduction to the Theory of Neural Computation*. The Advanced Book Program, Addison-Wesley, Redwood City, CA.

Mangasarian, O. L. 1965. Linear and nonlinear separation of patterns by linear programming. *Oper. Res.* **13** 444–452.

Mangasarian, O. L. 1968. Multi-surface method of pattern separation. *IEEE Trans. Inform. Theory* **IT-14** 801–807.

Mangasarian, O. L. 1993. Mathematical programming in neural networks. *ORSA J. Comput.* **5** 349–360.

Mangasarian, O. L. 1994. Misclassification minimization. *J. Global Optim.* **5** 309–323.

Mooney, R., J. Shavlik, G. Towell, A. Gove. 1989. An experimental comparison of symbolic and connectionist learning algorithms. *Proc. 11th Internat. Joint Conf. Artificial Intelligence*, Detroit, MI, 775–780.

Murthy, S. K., S. Kasif, S. Salzberg. 1994. A system for induction of oblique decision trees. *J. Artificial Intelligence Res.* **2** 1–33.

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

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

Stone, M. 1974. Cross-validatory choice and assessment of statistical predictions. *J. Roy. Statist. Soc. Ser. B* **36** 111–147.

Vapnik, V. N. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.