In the *Fifth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL, January 1995. Proceedings were printed only for conference attendees. A slightly shortened version appears in D. Fisher and H. Lenz (Eds.), *Learning From Data: Artificial Intelligence and Statistics V*, Springer-Verlag, New York, 1996.

Robust Linear Discriminant Trees

George H. John
Computer Science Department
Stanford University
Stanford, CA 94305
gjohn@cs.Stanford.EDU

Abstract

We present a new method for the induction of classification trees with linear discriminants as the partitioning function at each internal node. This paper presents two main contributions: first, a novel objective function called *soft entropy* which is used to identify optimal coefficients for the linear discriminants, and second, a novel method for removing outliers called *iterative re-filtering* which boosts performance on many datasets. These two ideas are presented in the context of a single learning algorithm called DT-SEPIR.

1 Introduction

Recursive partitioning classifiers, or decision trees, are an important nonparametric function representation in Statistics and Machine Learning (Friedman 1977, Breiman, Friedman, Olshen & Stone 1984, Quinlan 1986, Quinlan 1993). Their wide and successful use in fielded applications and their simple intuitive appeal make decision tree learning algorithms an important area of study. In this paper we address both of the main issues in decision tree induction: construction and pruning.

We follow the traditional recursive partitioning approach to tree building, but rather than partitioning the data on a single axis we instead employ a linear discriminant at each node to recursively split the data (Henrichon, Jr. & Fu 1969, Friedman 1977, Breiman et al. 1984). The issue of how to find a good discriminant naturally arises. We discuss problems in previous approaches and propose a new splitting criterion, *soft entropy*. The DT-SE algorithm uses recursive partitioning on this criterion to build a pure tree, one that correctly classifies each instance in the training set.

A problem with working on decision tree splitting criteria is that nobody ever actually uses a decision tree without first pruning it in an attempt to avoid overfitting and address the bias-variance tradeoff. Thus, results on the superiority of one splitting criterion over another should be in the context of the use of some regularization algorithm in order to mirror the results we expect to observe in practice.

To address the overfitting problem, we developed the DT-SEP algorithm which prunes the pure tree using a method similar to a stopping rule. Pruning involves removing an entire subtree and replacing it with a leaf. Implicit in this operation is the assumption that the input patterns which caused the subtree to be built are locally either unimportant or harmful. We suggest that these patterns might be *globally* unimportant or harmful (they might have caused bad splits higher in the tree) and thus the induction algorithm should remove such patterns from the training set and build a new tree. This step of *filtering* the pruned instances out of the training sample and rebuilding yields the DT-SEPIR (Iterative Re-filtering) algorithm. Though common in regression in the guise of *robust* (Hubel 1977) or *resistant* (Hastie & Tibshirani 1990, Chapter 9) fitting, in the context of classification this appears to be novel.

In Section 2 we formalize the problem of finding a splitting function for an arbitrary splitting criterion as a function optimization problem. Section 3 discusses problems with splitting criteria used in previous work in decision trees, and presents the soft entropy criterion. Section 4 presents the pruning algorithm and discusses the iterative re-filtering method in more detail. Experimental results comparing DT with related algorithms are presented and discussed in Section 5. Section 6 discusses related work, and Section 7 gives directions for future work. Finally, Section 8 presents our conclusions.

2 Finding Splitting Functions

In this section we frame the problem of finding the best splitting function at each node in the tree as a function optimization problem. To define this correctly we require a bit of notation summarized in Table 1.

Table 1: Notation used in the rest of the paper.

Symbol	Meaning
$\vec{x} \in \mathcal{X}$	An instance. A vector of attribute values.
$y \in \mathcal{Y}$	The class or output value of an instance.
f	A target function mapping instances to output values. $f : \mathcal{X} \mapsto \mathcal{Y}$.
T	A training set. $T \subseteq \{(\vec{x}, y) \mid \vec{x} \in \mathcal{X}, y = f(\vec{x}) + \epsilon\}$ for some *noise* term ϵ.
z	A training (labeled) instance. $z = (\vec{x}, y)$. $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$.
S	A weighted set of training instances. For each $z \in S$, $w_S(z) \in (0, 1]$ where w_S is the weighting function of S. $w_S(z) = 0$ for all $z \notin S$. Unweighted sets are a special case where $w_S(z) \in \{0, 1\}$.
$\theta \in \Theta$	A vector of parameters for a splitting function. In the case of linear splitting functions, θ is the vector of coefficients.
g_θ	A *soft* binary splitting function, mapping instances into $[0, 1]$. $g_\theta : \mathcal{X} \mapsto [0, 1]$. Given a weighted set S, define S/g_θ to be the tuple of weighted sets (S_0, S_1) gotten by applying g_θ to each $z_x \in S$ such that $w_{S_0}(z) = g_\theta(z_x) w_S(z)$ and $w_{S_0}(z) + w_{S_1}(z) = w_S(z)$. *Hard* splitting functions have the range $\{0, 1\}$.
I	A splitting criterion (impurity function) over binary splits, mapping two weighted sets to the real numbers $\mathbb{R}$.

We view the problem of choosing a splitting function at a node of a decision tree as a function optimization problem. Given a weighted set S of training instances we desire to split, and some fixed parametric form of the splitting function g_θ, our goal is to find θ^* satisfying:

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} \; I(S/g_\theta) \quad . \tag{1}$$

We could as well search over several different forms of g, so that the minimization is over both g and θ. We have investigated this but do not pursue it further in this paper.

In defining a problem as function optimization there are three important decisions: the domain in which the optimization should be performed, the objective function, and the function optimizer used. The domain, Θ, is defined by whatever functional form we set for g—all parameters remaining to be set constitute Θ. The optimization algorithm could be any method for unconstrained optimization, including methods using first derivative information if $\partial I/\partial\theta$ can be derived.

Fortunately, by the chain rule $\partial I/\partial\theta$ can be separated into two terms: $\partial I/\partial g_\theta \; \partial g_\theta/\partial\theta$, so that we need only derive the derivatives for the splitting criterion with respect to the outputs of the splitting function (whatever it may be) and the derivative of the output of the splitting function with respect to its parameters θ (regardless of the impurity function being used).

The essential ingredient is the objective function: the splitting criterion I which we wish to minimize. All of the splitting criteria we are aware of—twoing (Breiman et al. 1984), entropy, Gini, delta (Morgan & Messenger 1973), gain-ratio (Quinlan 1993), C-sep (Fayyad & Irani 1992)—are defined on sets, but have straightforward extensions to weighted sets. The criteria are all just functions of the following counts: the total number of instances in S, the total number in S_0 and S_1, and the number of instances of each class in S_0 and S_1. For weighted sets, we define cardinality as $|S| = \sum_{z \in S} w_S(z)$. Simply by replacing sets with weighted sets and the standard cardinality measure by the new cardinality measure, we can define *soft* versions of all of our favorite splitting criteria.

As an example, consider the softened linear discriminant splitting function

$$g_\theta(\vec{x}) = 1/(1 + \exp(-\theta^T \vec{x})) \quad . \tag{2}$$

Note that the range of g is $(0, 1)$, thus when using g_θ to partition a set S, all $z \in S$ are partially assigned to both S_0 and S_1.

The function optimization algorithm repeatedly evaluates $I(S/g_\theta)$ to get the value of the current θ, makes adjustments to θ (perhaps based on $\partial I/\partial\theta$) and evaluates the new θ. Eventually the optimization algorithm stops, outputting $\widehat{\theta}$, its estimate of θ^* from Equation 1.

3 Building Trees with Soft Entropy

Although several approaches to building linear discriminant trees have used a soft splitting criterion (mean-squared error in logistic regression), and many other approaches have used a hard criterion based on entropy or Gini, we are aware of no approach combining the desirable properties of both. Below we discuss why we prefer a criterion based on entropy and why we prefer a soft criterion. We then give the details of the tree construction algorithm DT-SE.

3.1 Why *Soft* Entropy?

The left graph in Figure 1 shows several candidate hard splits of a small dataset. All of the splits shown have equivalent counts (number of instances of each class in each subset) and hence they are evaluated equally by a hard splitting criterion. Because they use sets instead of weighted sets, hard criteria possess limited granularity and are unable to distinguish between splits that are quite different.

Figure 1: Left: all four splits have equivalent entropy (and Gini, delta, *etc.*). Right: the split found by minimizing soft entropy.

The unique best soft split is shown in Figure 1 on the right. Since the soft split assigns instances partially to both the left and right subset (in the case of Equation 2, based on the distance between the instance and the splitting hyperplane), all hyperplanes are assigned different values by the splitting criterion. Additionally, since soft criteria are differentiable they are amenable to the use of a wider variety of function optimizers.

3.2 Why Soft *Entropy*?

Given that softness is a desirable property, much work has used the soft linear discriminant of Equation 2 combined with an error criterion such as sum-squared error, common in logistic regression and neural networks. However, Breiman et al. (1984) argue against error as a splitting criterion, and our example in Figure 2 demonstrates a shortcoming of error. Figure 2 shows two linear discriminant functions each optimizing some splitting criterion. (In this case they happened to be soft linear discriminants, but the same phenomenon occurs with hard splits.) The entropy-minimizing discriminant does an excellent job of partitioning the data, while the error-minimizing discriminant fails to partition the data at all.

3.3 Finding the Best Discriminant

To build trees using the soft entropy criterion, we must combine several ingredients. Perhaps most important is the agenda for constructing the tree itself. We will use the standard recursive partitioning approach, at each node finding the splitting function that minimizes some impurity measure, splitting the data with this splitting function, and then recursively building trees from the resulting subsets.

The splitting criterion we wish to minimize is still $I(S/g_\theta)$, where I is now specified to be the entropy measure. (Note that I is a function on a pair of sets while entropy is a function of a single set—as in Breiman et al. (1984) we take the weighted average of entropy on the two sets.) The splitting function g is defined in Equation 2. In this case, calculating $I(S/g_\theta)$ and $\partial I/\partial\theta$ is straightforward, so we used the simplest unconstrained function optimization method using first derivative information: steepest descent.

Figure 2: Left: The soft linear discriminant minimizing mean squared error fails to partition the data. Right: The soft linear discriminant minimizing soft entropy.

The function optimization technique used in DT-SE is a modified version of steepest-descent. Steepest descent optimizers begin with some θ_0, then repeatedly move some distance in the direction of the gradient of the objective function at θ_0. Typically the distance is set by the user as a parameter or it is found by an expensive line minimization. In DT-SE we have employed a few tricks to get reasonable speed while retaining the basic steepest descent algorithm. (We do not claim to be at the cutting edge of function optimization here, but the algorithm is included for replicability.) The update equation for going from θ_i, the parameters currently under consideration, to θ_{i+1}, the next parameters to be considered, is:

$$\theta_{i+1} = \theta_i - 0.1 \frac{\frac{\partial I}{\partial \theta}|_{\theta_i}}{||\frac{\partial I}{\partial \theta}|_{\theta_i}||^{0.8}} + 0.8(\theta_i - \theta_{i-1}) \qquad (3)$$

The algorithm stops when the current $I(S/g_\theta)$ drops below 0.00002, when $i > 1000$, or when 20 steps fail to produce a change in I of at least 0.00005. The whole process repeats once, starting from a different θ_0, in an attempt to avoid the perils of getting trapped in a local minimum.

It is important to note that once the best θ is found, we use a hard split to partition the data into a left and right subset, which are themselves recursively split. All of the work on soft splitting functions was done just to discover a good hard split. In the case of the soft linear discriminant, it is replaced by its hard cousin:

$$g_\theta(\vec{x}) = 1 \text{ if } \theta^T \vec{x} \geq 0, 0 \text{ otherwise } . \qquad (4)$$

4 Pruning a Tree

While studying the trees induced by the algorithm, we found that most trees contained several leaves matching on the order of 50 to 100 instances from the training set, and a few leaves matching only a few instances. We conjectured that these isolated instances constituted noise in the training data, and that we might improve performance by pruning such instances.

Our regularization algorithm contains a pruning step and a retraining step. The pruning step replaces a decision node with a leaf whenever the number of positive or negative instances matching that node drops below k. (In our experiments, k was arbitrarily set to 5.) This is applied to all nodes in the tree in a top down fashion.

In what appears to be a novel approach, the regularization algorithm then *retrains* on the reduced training set, building a completely new tree. The reduced training set is defined to the original training set minus the instances which the pruned tree now classifies incorrectly (the "confusing" instances). While retraining may seem odd, it is in fact just an extension of the assumptions underlying pruning. By pruning the tree we essentially assume that these confusing instances are locally not useful. Retraining merely takes this assumption a step further by completely removing these instances from the training set. The regularization algorithm continues pruning and retraining until no further pruning can be done.

The hypothesis behind iterative re-filtering is thus that data which is *locally* un-informative is also *globally* un-informative. In statistics, estimators resistant to the effect of outliers, *robust* estimators, have been studied in some depth. Points with high *leverage*, points with disproportionately high effect on the fitted model, are often removed from training data in order to better fit the remaining points. In the context of decision trees, we may identify a set of points with high leverage by examining the difference in number of nodes between a tree built with and without the set of points. This difference is estimated by starting with a pure tree built using all the points and then pruning. The training instances that it now classifies incorrectly are the high leverage points that were removed. However, the removal of the points by pruning was only approximate—the obvious step is to remove the points from the training set and retrain.

Another way of looking at iterative re-filtering is as a pattern selection method. The selection of patterns for training is an ubiquitous problem in pattern recognition. Many methods are *passive*, accepting data from a training sample in some random order or all at once as a set. Other methods (Aha 1991) actively accept or reject training patterns from a temporal sequence presented to them. The difference between our method and theirs is somewhat akin to the difference between forward and backward feature subset selection (John, Kohavi & Pfleger 1994) or forward and backward (construction/pruning) search methods over neural net or decision tree architectures. In general, unless time is a very limited resource, the best results are achieved by starting "big" and then shrinking (Breiman et al. 1984). This should apply to pattern selection as well, and thus we suspect backward pattern selection will give greater performance than forward selection.

5 Experiments with DT-SEPIR

We ran cross-validation experiments comparing DT-SE, DT-SEP and DT-SEPIR with CART and a CART-like algorithm, OC1. Experimental methodology is first discussed, then results are presented and analyzed.

5.1 Methodology

Five datasets were gathered from the UCI (Murphy & Aha 1994) and Statlog (Michie, Spiegelhalter & Taylor 1994) collections. In the Vote1 dataset, the task is to predict the political party of congress members given their votes on key issues. The most predictive attribute has been removed. In the Australian database, the task is to decide whether or not to give someone a credit card. The Breast-Wisc is the Wisconsin breast cancer database. The task is to classify a lump as benign or malignant. The task in the Diabetes dataset is to classify a Pima Indian female as having diabetes or not. In the Heart database, the algorithms must predict whether or not a patient has heart disease given various information.

All datasets were processed (either by the author, or in the case of the Statlog datasets, by others) so that all unordered categorical attributes were encoded using 0-1 indicator variables and all ordered categorical attributes were encoded as integers. All missing values were either removed or replaced with their mean or mode.

Three different algorithms were used: the DT-SE algorithms discussed in this paper, CART (Breiman et al. 1984), and OC1 (Murthy, Kasif & Salzberg 1994, Murthy, Salzberg & Kasif 1993). CART was set to use linear splits whenever the number of instances at the node was greater than ten times the number of attributes. (When using linear splits, CART seems to be quite numerically fragile. This high setting was required to get it to run on most of the datasets.) CART used ten-fold cross-validation and the 1-SE rule to select the correct pruned tree. Default values were used for all other parameters. OC1 is similar to CART, also learning trees with linear splits. It uses a modified version of CART's optimization algorithm which seems to do a better job of avoiding local minima. It does not offer the option of using cross-validation to select a pruning parameter; rather, it randomly picks 10% of the training set to use as a pruning set. Default values were used

Table 2: Test set accuracy and tree size results from 10-fold cross-validation.

		Vote1	Australian	Breast-Wisc	Diabetes	Heart
DT-SEPIR	Acc	89.0±4.0	85.0±4.6	95.1±2.2	73.7±5.8	77.8±10.5
	Size	8.8±3.7	8.0±3.0	6.2±2.3	16.6±5.1	7.4±2.3
DT-SEP	Acc	86.9±5.6	81.8±3.6	95.3±2.2	71.9±6.7	75.6±10.8
	Size	11.8±2.3	35.4±6.3	8.2±1.0	58.2±6.7	13.2±2.0
DT-SE	Acc	85.5±4.8	80.2±4.3	94.3±1.9	68.9±6.7	74.5±11.0
	Size	20.4±3.3	85.8±11.0	19.8±4.3	127.6±10.1	25.0±2.7
OC1-pruned	Acc	89.9±4.2	85.1±2.9	95.6±2.7	74.3±3.5	75.5±6.8
	Size	5.8±6.0	10.2±9.6	3.6±1.0	13.6±12.1	9.6±10.2
OC1	Acc	87.1±3.5	81.2±4.6	94.9±1.6	68.4±4.3	72.6±7.5
	Size	36.2±5.3	83.6±11.0	35.2±5.4	149.8±7.7	42.2±6.3
CART	Acc		82.0±2.9	96.2±3.1	74.6±4.5	81.5±8.9
	Size		3.0±0.0	3.8±1.8	3.6±9.7	3.4±1.3

for all OC1 parameters.

Ten-fold cross-validation was used to estimate the accuracy of each algorithm on each dataset. (This is a separate step from CART's own internal cross-validation.) In 10-fold CV, the dataset is divided into 10 blocks of roughly equal size, then the learning algorithm is repeatedly trained on nine out of the ten blocks and tested on the block not seen during training. From the ten accuracy results we report the mean and variance.

5.2 Results and Discussion

Table 2 gives 10-fold cross-validation results for the DT-SE algorithms, the OC1 algorithm, and CART. Each box contains 2 measurements: the top is the accuracy, and the bottom is the tree size for the ten runs.

No results are reported for CART on the Vote1 domain because CART failed with a numerical error on each of the ten folds. On the diabetes and heart domains, all ten runs completed successfully; on the rest, only a few runs completed without errors.

We applied the paired t-test to every meaningful set of results we could think of. Comparing the tree sizes of OC1-unpruned and DT-SE, DT-SE produces smaller trees on all domains except Australian with over 99.5% confidence. Within the DT-SE group of algorithms, DT-SEP is superior to DT-SE with nearly 97.5% confidence on the vote, breast, and diabetes datasets, and with 90% on the Australian dataset. DT-SEPIR is superior to both DT-SEP and DT-SE with 95% confidence on vote1 and Australian. DT-SEPIR is superior to DT-SE with 90% confidence on breast and diabetes. Comparing CART, OC1, and DT-SEPIR we found that none of the differences in accuracy between these algorithms is significant at the 90% level, except that CART performed better than OC1 on the Heart data at the 95% confidence level. Comparing the accuracy of OC1-unpruned and DT-SE, OC1-unpruned is better than DT-SE at the 90% confidence level on vote1.

Though the accuracy results are mixed, the sizes of the unpruned trees does support the idea that soft entropy is a good splitting criterion. (Presumably a characteristic of good splitting criteria is that they result in small trees.) Regarding iterative re-filtering, it is promising that the DT-SEPIR results were competitive with OC1-pruned and CART, both of which employ what might be thought of as more sophisticated methods. Iterative re-filtering, when combined with these better pruning methods, might yield an even better boost in performance.

6 Related Work

John (1994) presents preliminary experiments with DT-SEPIR, comparing against other neural network and decision tree algorithms. Friedman (1977) gives an excellent discussion of practically every issue related to the induction of decision trees from data, including linear discriminant splits using Fisher's linear discriminant (Duda & Hart 1973). The earliest attempt at building linear discriminant trees seems to be Henrichon, Jr. & Fu (1969), who use the maximum eigenvector of the covariance matrix to determine the split. Lin & Fu (1983) use k-means clustering to find the splitting function. Bennett & Mangasarian (1992) use a parallel pair of hyperplanes as a splitting function and use linear programming to minimize their splitting criterion, which was the distance between the pair. Sankar & Mammone (1991) present "Neural Tree Networks," which seem to be simply soft linear discriminant trees using mean squared error as the criterion. Loh & Vanichsetakul (1988) present linear discriminant trees where the discriminant hyperplane may be solved for exactly by making normality and homoscedasticity assumptions about the data. Qing-Yun & Fu (1983) present a sophisticated method for not only finding linear splits at each node (based on regression) but also selecting the best subset of variables to use in the split. Brodley & Utgoff (n.d.) present a similar algorithm, also based on linear regression with subset selection.

Although this work was developed independently, Brent (1991) discusses essentially the same technique for softening a related splitting criterion. However, Brent's focus was not on finding a better splitting criterion for multivariate decision trees; rather, he used this impurity measure to train neural networks one node at a time. Sethi (1990) gives the original description of the mapping between neural nets and decision trees, and proposes converting the tree (built with univariate splits) to neural net form for further refinement. Along the same lines, Jordan & Jacobs (1993) present a method that allows simultaneous optimization of all coefficients in the tree by defining the model likelihood and using the EM algorithm (Dempster, Laird & Rubin 1977) for optimization. Soft splits are used not only for tree construction but also during classification. However, their method requires the user to specify the tree structure a priori. Sahami (1993) investigates various optimization algorithms for finding an error-minimizing discriminant at each node in a decision tree.

Bichsel & Seitz (1989) gives an algorithm for training a tree-structured net very similar to the approach by Heath, Kasif & Salzberg (1993), using simulated annealing to search the space of hyperplanes at each node. The procedure for adding nodes is somewhat similar in spirit to that used in TDIDT methods, but their actual construction algorithm assumes the tree will be very small. A related approach in the neural net community was by Koutsougeras & Papachristou (1988). They discuss the benefits of learning entropy-minimizing hyperplanes in a tree-structured neural net, but in the actual algorithm they instead implement Fisher's linear discriminant function.

Regarding robust methods and outlier rejection, Hubel (1977) states "I am inclined to ... prefer technical expertise to any 'statistical' criterion for straight outlier rejection." We are guilty of this sin in our work, but Guyon, Boser & Vapnik (1993) have proposed an interesting method for making use of human expertise in outlier removal to "clean" a dataset.

7 Future Work

A known problem with decision trees is that as one goes deeper in the tree, less data is available to each node, due to the recursive hard-partitioning. Since DT-SE already deals with partial assignments of instances to the left and right subsets, it would be a simple matter to use the soft partitions in the recursive step of the tree-building algorithm and during classification as in Jordan & Jacobs (1993).

As mentioned in the introduction, the procedure for finding a good splitting function is just one part of a whole decision tree learning algorithm. The most important issue with respect to generalization ability is regularization (pruning). The simple pruning strategy described should be extended to use cross-validation to pick the threshold, rather than arbitrarily setting it at five. As Brodley & Utgoff (n.d.) point out, pruning in multivariate trees can be carried out at a finer granularity than removing entire nodes: subset selection should be investigated as an alternate means of regularization. Penalty functions such as AIC, BIC, *etc.*, should also be investigated.

Finally, another benefit of this approach is that it can be easily generalized to learn splitting functions other than linear splits. Neither the function optimizer nor the objective function depend on the actual splitting function used, so this is a completely independent module. We have investigated the use of more complex splitting functions (neural networks with few hidden units) with mixed results but plan to investigate this further.

8 Conclusion

We have provided a general framework to address the problem of finding a splitting function in recursive-partitioning classification tree algorithms. Within this framework we have identified a splitting criterion, soft entropy, which appears to combine the good properties of its hard counterpart entropy as well as soft criteria such as mean squared error in the logistic regression setting. Experimental results indicate that soft entropy is a better objective function than its hard counterpart. A rudimentary pruning algorithm was employed to address the overfitting problem, and a novel data filtering method was used. Accuracy results for the resulting DT-SEPIR algorithm as compared with CART and OC1 are inconclusive. More experiments are needed to isolate the effects of the splitting criterion and pruning methods.

9 Acknowledgments

This work was supported by a National Science Foundation Graduate Research Fellowship and NSF Grant IRI-9116399. We would like to thank Pat Langley, Wray Buntine, Jerome Friedman, Scott Benson, Ron Kohavi and Nils Nilsson for comments on the ideas presented here. Thanks to Richard Olshen and Charles Stone for providing the CART software.

References

Aha, D. W. (1991), "Instance-based learning algorithms", *Machine Learning* **6**(1), 37–66.

Bennett, K. P. & Mangasarian, O. L. (1992), Neural network training via linear programming, *in* P. M. Pardalos, ed., "Advances in Optimization and Parallel Computing", North Holland, Amsterdam, pp. 56–67.

Bichsel, M. & Seitz, P. (1989), "Minimum class entropy: A maximum information approach to layered networks", *Neural Networks* **2**, 133–141.

Breiman, L., Friedman, J., Olshen, R. & Stone, C. (1984), *Classification and Regression Trees*, Chapman & Hall, New York.

Brent, R. P. (1991), "Fast training algorithms for neural networks", *IEEE Transactions on Neural Networks* **2**(3), 346–354.

Brodley, C. E. & Utgoff, P. E. (n.d.), "Multivariate decision trees", *Machine Learning*. Forthcoming.

Dempster, A. P., Laird, N. M. & Rubin, D. B. (1977), "Maximum likelihood from incomplete data via the EM algorithm", *Journal of the Royal Statistical Society B* **39**, 1–38.

Duda, R. & Hart, P. (1973), *Pattern Classification and Scene Analysis*, Wiley.

Fayyad, U. M. & Irani, K. B. (1992), The attribute selection problem in decision tree generation, *in* "AAAI-92: Proceedings of the Tenth National Conference on Artificial Intelligence", American Association for Artificial Intelligence, AAAI Press/MIT Press, pp. 104–110.

Friedman, J. H. (1977), "A recursive partitioning decision rule for nonparametric classification", *IEEE Transactions on Computers* pp. 404–408.

Guyon, I., Boser, B. & Vapnik, V. (1993), Automatic capacity tuning of very large VC-dimension classifiers, *in* S. J. Hanson, J. Cowan & C. L. Giles, eds, "Advances in Neural Information Processing Systems", Vol. 5, Morgan Kaufmann, pp. 147–154.

Hastie, T. J. & Tibshirani, R. J. (1990), *Generalized Additive Models*, Chapman and Hall.

Heath, D., Kasif, S. & Salzberg, S. (1993), Induction of oblique decision trees, *in* R. Bajcsy, ed., "Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence", Morgan Kaufmann.

Henrichon, Jr., E. G. & Fu, K.-S. (1969), "A nonparametric partitioning procedure for pattern classification", *IEEE Transactions on Computers* **C-18**(7), 614–624.

Hubel, P. J. (1977), *Robust Statistical Procedures*, Society for Industrial and Applied Mathematics, Pittsburgh, PA.

John, G. H. (1994), Finding multivariate splits in decision trees using function optimization, *in* "AAAI-94: Proceedings of the Twelfth National Conference on Artificial Intelligence", American Association for Artificial Intelligence, AAAI Press/MIT Press, p. 1463. Abstract.

John, G., Kohavi, R. & Pfleger, K. (1994), Irrelevant features and the subset selection problem, *in* H. Hirsh & W. Cohen, eds, "Machine Learning: Proceedings of the Eleventh International Conference", Morgan Kaufmann, pp. 121–129. Available by anonymous ftp in `starry.Stanford.EDU:pub/gjohn/papers/ml94.ps`.

Jordan, M. I. & Jacobs, R. A. (1993), Supervised learning and divide-and-conquer: A statistical approach, *in* P. Utgoff, ed., "Proceedings of the Tenth International Conference on Machine Learning", Morgan Kaufmann.

Koutsougeras, C. & Papachristou, C. A. (1988), Training of a neural network for pattern classification based on an entropy measure, *in* "IEEE International Conference on Neural Networks", IEEE Press, pp. 247–254.

Lin, Y. K. & Fu, K. S. (1983), "Automatic classification of cervical cells using a binary tree classifier", *Pattern Recognition* **16**(1), 69–80.

Loh, W.-Y. & Vanichsetakul, N. (1988), "Tree-structured classification via generalized discriminant analysis", *Journal of the American Statistical Association* **83**(403), 715–725.

Michie, D., Spiegelhalter, D. J. & Taylor, C. C. (1994), *Machine Learning, Neural and Statistical Classification*, Prentice Hall.

Morgan, J. N. & Messenger, R. C. (1973), *THAID: a sequential analysis program for the analysis of nominal scale dependent variables*, University of Michigan.

Murphy, P. M. & Aha, D. W. (1994), "UCI repository of machine learning databases", Available by anonymous ftp to `ics.uci.edu` in the `pub/machine-learning-databases` directory.

Murthy, S. K., Salzberg, S. & Kasif, S. (1993), "OC1", Available by anonymous ftp in `blaze.cs.jhu.edu:pub/oc1`.

Murthy, S., Kasif, S. & Salzberg, S. (1994), "A system for induction of oblique decision trees", *Journal of Artificial Intelligence Research* **2**, 1–32.

Qing-Yun, S. & Fu, K. S. (1983), "A method for the design of binary-tree classifiers", *Pattern Recognition* **16**(6), 593–603.

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.

Sahami, M. (1993), Learning non-linearly separable boolean functions with linear threshold unit trees and madaline-style networks, *in* "AAAI-93: Proceedings of the Eleventh National Conference on Artificial Intelligence", American Association for Artificial Intelligence, AAAI Press/MIT Press, pp. 335–341.

Sankar, A. & Mammone, R. J. (1991), Optimal pruning of neural tree networks for improved generalization, *in* "IJCNN-91-SEATTLE: International Joint Conference on Neural Networks", IEEE Press, Seattle, WA, pp. II: 219–224.

Sethi, I. K. (1990), "Entropy nets: from decision trees to neural networks", *Proceedings of the IEEE* **78**(10), 1605–1613.