Machine Learning, 40, 203–229 (2000)

A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-three Old and New Classification Algorithms

TJEN-SIEN LIM
WEI-YIN LOH * loh@stat.wisc.edu
Department of Statistics, University of Wisconsin, Madison, WI 53706

YU-SHAN SHIH ** yshih@math.ccu.edu.tw
Department of Mathematics, National Chung Cheng University, Chiayi 621, Taiwan, R.O.C.

Received June 19, 1997 ; Revised May 26, 1999

Editor: William W. Cohen

Abstract. Twenty-two decision tree, nine statistical, and two neural network algorithms are compared on thirty-two datasets in terms of classification accuracy, training time, and (in the case of trees) number of leaves. Classification accuracy is measured by mean error rate and mean rank of error rate. Both criteria place a statistical, spline-based, algorithm called POLYCLASS at the top, although it is not statistically significantly different from twenty other algorithms. Another statistical algorithm, logistic regression, is second with respect to the two accuracy criteria. The most accurate decision tree algorithm is QUEST with linear splits, which ranks fourth and fifth, respectively. Although spline-based statistical algorithms tend to have good accuracy, they also require relatively long training times. POLYCLASS, for example, is third last in terms of median training time. It often requires hours of training compared to seconds for other algorithms. The QUEST and logistic regression algorithms are substantially faster. Among decision tree algorithms with univariate splits, C4.5, IND-CART, and QUEST have the best combinations of error rate and speed. But C4.5 tends to produce trees with twice as many leaves as those from IND-CART and QUEST.

Keywords: Classification tree, decision tree, neural net, statistical classifier.

1. Introduction

There is much current research in the machine learning and statistics communities on algorithms for decision tree classifiers. Often the emphasis is on the accuracy of the algorithms. One study, called the STATLOG Project [34], compares the accuracy of several decision tree algorithms against some non-decision tree algorithms on a large number of datasets. Other studies that are smaller in scale include Brodley and Utgoff (1992), Brown, Corruble and Pittard (1993), Curram and Mingers (1994), and Shavlik, Mooney and Towell (1991).

* Supported in part by grants from the U. S. Army Research Office and Pfizer, Inc. and a University of Wisconsin Vilas Associateship.
** Supported in part by a grant and fellowship from the Republic of China National Science Council.

Recently, comprehensibility of the tree structures has received some attention. Comprehensibility typically decreases with increase in tree size and complexity. If two trees employ the same kind of tests and have the same prediction accuracy, the one with fewer leaves is usually preferred. Breslow and Aha (1997) survey methods of tree simplification to improve their comprehensibility.

A third criterion that has been largely ignored is the relative training time of the algorithms. The StatLog Project finds that no algorithm is uniformly most accurate over the datasets studied. Instead, many algorithms possess comparable accuracy. For such algorithms, excessive training times may be undesirable [19].

The purpose of our paper is to extend the results of the StatLog Project in the following ways:

1. In addition to classification accuracy and size of trees, we compare the training times of the algorithms. Although training time depends somewhat on implementation, it turns out that there are such large differences in times (seconds versus days) that the differences cannot be attributed to implementation alone.

2. We include some decision tree algorithms that are not included in the StatLog Project, such as S-Plus tree [14], T1 [3, 25], Oc1 [38], Lmdt [9], and Quest [30].

3. We also include several of the newest spline-based statistical algorithms. Their classification accuracy may be used as benchmarks for comparison with other algorithms in the future.

4. We study the effect of adding independent noise attributes on the classification accuracy and (where appropriate) tree size of each algorithm. It turns out that except possibly for three algorithms, all the others adapt to noise quite well.

5. We examine the scalability of some of the more promising algorithms as the sample size is increased.

Our experiment compares twenty-two decision tree algorithms, nine classical and modern statistical algorithms, and two neural network algorithms. Several datasets are taken from the University of California, Irvine, Repository of Machine Learning Databases (UCI) [33]. Fourteen of the datasets are from real-life domains and two are artificially constructed. Five of the datasets were used in the StatLog Project. To increase the number of datasets and to study the effect of noise attributes, we double the number of datasets by adding noise attributes to them. This yields a total of thirty-two datasets.

Section 2 briefly describes the algorithms and Section 3 gives some background to the datasets. Section 4 explains the experimental setup used in this study and Section 5 analyzes the results. The issue of scalability is studied in Section 6 and conclusions and recommendations are given in Section 7.

2. The algorithms

Only a short description of each algorithm is given. Details may be found in the cited references. If an algorithm requires class prior probabilities, they are made proportional to the training sample sizes.

2.1. Trees and rules

CART: We use the version of CART [6] implemented in the `cart` style of the IND package [13] with the Gini index of diversity as the splitting criterion. The trees based on the 0-SE and 1-SE pruning rules are denoted by `IC0` and `IC1` respectively. The software is obtained from the *http* address:

`ic-www.arc.nasa.gov/ic/projects/bayes-group/ind/IND-program.html`.

S-Plus tree: This is a variant of the CART algorithm written in the S language [4]. It is described in Clark and Pregibon (1993). It employs deviance as the splitting criterion. The best tree is chosen by ten-fold cross-validation. Pruning is performed with the `p.tree()` function in the `treefix` library [47] from the STATLIB S Archive at `http://lib.stat.cmu.edu/S/`. The 0-SE and 1-SE trees are denoted by `ST0` and `ST1` respectively.

C4.5: We use Release 8 [41, 42] with the default settings including pruning (`http://www.cse.unsw.edu.au/~quinlan/`). After a tree is constructed, the C4.5 rule induction program is used to produce a set of rules. The trees are denoted by `C4T` and the rules by `C4R`.

FACT: This fast classification tree algorithm is described in Loh and Vanichsetakul (1988). It employs statistical tests to select an attribute for splitting each node and then uses discriminant analysis to find the split point. The size of the tree is determined by a set of stopping rules. The trees based on univariate splits (splits on a single attribute) are denoted by `FTU` and those based on linear combination splits (splits on linear functions of attributes) are denoted by `FTL`. The FORTRAN 77 program is obtained from `http://www.stat.wisc.edu/~loh/`.

QUEST: This new classification tree algorithm is described in Loh and Shih (1997). QUEST can be used with univariate or linear combination splits. A unique feature is that its attribute selection method has negligible bias. If all the attributes are uninformative with respect to the class attribute, then each has approximately the same chance of being selected to split a node. Ten-fold cross-validation is used to prune the trees. The univariate 0-SE and 1-SE trees are denoted by `QU0` and `QU1`, respectively. The corresponding trees with linear combination splits are denoted by `QL0` and `QL1`, respectively. The results in this paper are based on version 1.7.10 of the program. The software is obtained from `http://www.stat.wisc.edu/~loh/quest.html`.

IND: This is due to Buntine (1992). We use version 2.1 with the default settings. IND comes with several standard predefined styles. We compare four Bayesian styles in this paper: `bayes`, `bayes opt`, `mml`, and `mml opt` (denoted by `IB`, `IBO`, `IM`, and `IMO`, respectively). The `opt` methods extend the non-`opt` methods by growing several different trees and storing them in a compact graph structure. Although more time and memory intensive, the `opt` styles can increase classification accuracy.

OC1: This algorithm is described in Murthy, Kasif and Salzberg (1994). We use version 3 (`http://www.cs.jhu.edu/~salzberg/announce-oc1.html`) and compare three styles. The first one (denoted by `OCM`) is the default that uses a mixture of univariate and linear combination splits. The second one (option `-a`; denoted by `OCU`) uses only univariate splits. The third one (option `-o`; denoted by `OCL`) uses only linear combination splits. Other options are kept at their default values.

LMDT: The algorithm is described in Brodley and Utgoff (1995). It constructs a decision tree based on multivariate tests that are linear combinations of the attributes. The tree is denoted by `LMT`. We use the default values in the software from `http://yake.ecn.purdue.edu/~brodley/software/lmdt.html`.

CAL5: This is from the Fraunhofer Society, Institute for Information and Data Processing, Germany [36, 37]. We use version 2. CAL5 is designed specifically for numerical-valued attributes. However, it has a procedure to handle categorical attributes so that mixed attributes (numerical and categorical) can be included. In this study we optimize the two parameters which control tree construction. They are the predefined threshold s and significance level α. We randomly split the training set into two parts, stratified by the classes: two-thirds are used to construct the tree and one-third is used as a validation set to choose the optimal parameter configuration. We employ the C-SHELL program that comes with the CAL5 package to choose the best parameters by varying α between 0.10 and 0.90 and s between 0.20 and 0.95 in steps of 0.05. The best combination of values that minimize the error rate on the validation set is chosen. The tree is then constructed on all the records in the training set using the chosen parameter values. It is denoted by `CAL`.

T1: This is a one-level decision tree that classifies examples on the basis of only one split on a single attribute [25]. A split on a categorical attribute with b categories can produce up to $b + 1$ leaves (one leaf being reserved for missing attribute values). On the other hand, a split on a continuous attribute can yield up to $J + 2$ leaves, where J is the number of classes (one leaf is again reserved for missing values). The software is obtained from `http://www.csi.uottawa.ca/~holte/Learning/other-sites.html`.

2.2. Statistical algorithms

LDA: This is linear discriminant analysis, a classical statistical method. It models the instances within each class as normally distributed with a common covariance matrix. This yields linear discriminant functions.

QDA: This is quadratic discriminant analysis. It also models class distributions as normal, but estimates each covariance matrix by the corresponding sample covariance matrix. As a result, the discriminant functions are quadratic. Details on `LDA` and `QDA` can be found in many statistics textbooks, e.g., Johnson and Wichern (1992). We use the SAS PROC DISCRIM [45] implementation of `LDA` and `QDA` with the default settings.

NN: This is the SAS PROC DISCRIM implementation of the nearest neighbor method. The pooled covariance matrix is used to compute Mahalanobis distances.

LOG: This is logistic discriminant analysis. The results are obtained with a polytomous logistic regression (see, e.g., Agresti, 1990) FORTRAN 90 routine written by the first author (`http://www.stat.wisc.edu/~limt/logdiscr/`).

FDA: This is flexible discriminant analysis [23], a generalization of linear discriminant analysis that casts the classification problem as one involving regression. Only the MARS [17] nonparametric regression procedure is studied here. We use the S-PLUS function `fda` from the `mda` library of the STATLIB S Archive. Two models are used: an additive model (`degree=1`, denoted by `FM1`) and a model containing first-order interactions (`degree=2` with `penalty=3`, denoted by `FM2`).

PDA: This is a form of penalized `LDA` [21]. It is designed for situations in which there are many highly correlated attributes. The classification problem is cast into a penalized regression framework via optimal scoring. `PDA` is implemented in S-PLUS using the function `fda` with `method=gen.ridge`.

MDA: This stands for mixture discriminant analysis [22]. It fits Gaussian mixture density functions to each class to produce a classifier. `MDA` is implemented in S-PLUS using the library `mda`.

POL: This is the POLYCLASS algorithm [28]. It fits a polytomous logistic regression model using linear splines and their tensor products. It provides estimates for conditional class probabilities which can then be used to predict class labels. `POL` is implemented in S-PLUS using the function `poly.fit` from the `polyclass` library of the STATLIB S Archive. Model selection is done with ten-fold cross-validation.

2.3. Neural networks

LVQ: We use the learning vector quantization algorithm in the S-PLUS `class` library [47] at the STATLIB S Archive. Details of the algorithm may be found in Kohonen (1995). Ten percent of the training set are used to initialize the algorithm, using the function `lvqinit`. Training is carried out with the optimized learning rate function `olvq1`, a fast and robust `LVQ` algorithm. Additional fine-tuning in learning is performed with the function `lvq1`. The number of iterations is ten times the size of the training set in both `olvq1` and `lvq1`. We use the default values of 0.3 and 0.03 for α, the learning rate parameter, in `olvq1` and `lvq1`, respectively.

RBF: This is the radial basis function network implemented in the SAS `tnn3.sas` macro [44] for feedforward neural networks (`http://www.sas.com`). The network architecture is specified with the `ARCH=RBF` argument. In this study, we construct a network with only one hidden layer. The number of hidden units is chosen to be 20% of the total number of input and output units [2.5% (5 hidden units) only for the `dna` and `dna+` datasets and 10% (5 hidden units) for the `tae` and `tae+` datasets because of memory and storage limitations]. Although the macro can perform model selection to choose the optimal number of hidden units, we did not utilize this capability because it would have taken too long for some of the datasets (see Table 6 below). Therefore the results reported here for this algorithm should be regarded as lower bounds on its performance. The hidden layer is fully connected to the input and output layers but there is no direct connection between the input and output layers. At the output layer, each class is represented by one unit taking the value of 1 for that particular category and 0 otherwise, except for the last one which is the reference category. To avoid local optima, ten preliminary trainings were conducted and the best estimates used for subsequent training. More details on the radial basis function network can be found in Bishop (1995) and Ripley (1996).

3. The datasets

We briefly describe the sixteen datasets used in the study as well as any modifications that are made for our experiment. Fourteen are from real domains while two are artificially created. Fifteen of the datasets are available from UCI.

Wisconsin breast cancer (`bcw`). This is one of the breast cancer databases at UCI, collected at the University of Wisconsin by W. H. Wolberg. The problem is to predict whether a tissue sample taken from a patient's breast is malignant or benign. There are two classes, nine numerical attributes, and 699 observations. Sixteen instances contain a single missing attribute value and are removed from the analysis. Our results are therefore based on 683 records. Error rates are estimated using ten-fold cross-validation. A decision tree analysis of a subset of the data using the FACT algorithm is reported in Wolberg, Tanner, Loh and

Vanichsetakul (1987) and Wolberg, Tanner and Loh (1988, 1989). The dataset has also been analyzed with linear programming methods [32].

Contraceptive method choice (`cmc`). The data are taken from the 1987 National Indonesia Contraceptive Prevalence Survey. The samples are married women who were either not pregnant or did not know if they were pregnant at the time of the interview. The problem is to predict the current contraceptive method choice (no use, long-term methods, or short-term methods) of a woman based on her demographic and socio-economic characteristics [29]. There are three classes, two numerical attributes, seven categorical attributes, and 1473 records. The error rates are estimated using ten-fold cross-validation. The dataset is available from UCI.

StatLog DNA (`dna`). This UCI dataset in molecular biology was used in the StatLog Project. Splice junctions are points in a DNA sequence at which "superfluous" DNA is removed during the process of protein creation in higher organisms. The problem is to recognize, given a sequence of DNA, the boundaries between exons (the parts of the DNA sequence retained after splicing) and introns (the parts of the DNA sequence that are spliced out). There are three classes and sixty categorical attributes each having four categories. The sixty categorical attributes represent a window of sixty nucleotides, each having one of four categories. The middle point in the window is classified as one of exon/intron boundaries, intron/exon boundaries, or neither of these. The 3186 examples in the database were divided randomly into a training set of size 2000 and a test set of size 1186. The error rates are estimated from the test set.

StatLog heart disease (`hea`). This UCI dataset is from the Cleveland Clinic Foundation, courtesy of R. Detrano. The problem concerns the prediction of the presence or absence of heart disease given the results of various medical tests carried out on a patient. There are two classes, seven numerical attributes, six categorical attributes, and 270 records. The StatLog Project employed unequal misclassification costs. We use equal costs here because some algorithms do not allow unequal costs. The error rates are estimated using ten-fold cross-validation.

Boston housing (`bos`). This UCI dataset gives housing values in Boston suburbs [20]. There are three classes, twelve numerical attributes, one binary attribute, and 506 records. Following Loh and Vanichsetakul (1988), the classes are created from the attribute median value of owner-occupied homes as follows: class = 1 if log(median value) ≤ 9.84, class = 2 if $9.84 <$ log(median value) ≤ 10.075, class = 3 otherwise. The error rates are estimated using ten-fold cross-validation.

LED display (`led`). This artificial domain is described in Breiman, Friedman, Olshen and Stone (1984). It contains seven Boolean attributes, representing seven light-emitting diodes, and ten classes, the set of decimal digits. An attribute value is either zero or one, according to whether the corresponding light

is off or on for the digit. Each attribute value has a ten percent probability of having its value inverted. The class attribute is an integer between zero and nine, inclusive. A C program from UCI is used to generate 2000 records for the training set and 4000 records for the test set. The error rates are estimated from the test set.

BUPA liver disorders (`bld`). This UCI dataset was donated by R. S. Forsyth. The problem is to predict whether or not a male patient has a liver disorder based on blood tests and alcohol consumption. There are two classes, six numerical attributes, and 345 records. The error rates are estimated using ten-fold cross-validation.

PIMA Indian diabetes (`pid`). This UCI dataset was contributed by V. Sigillito. The patients in the dataset are females at least twenty-one years old of Pima Indian heritage living near Phoenix, Arizona, USA. The problem is to predict whether a patient would test positive for diabetes given a number of physiological measurements and medical test results. There are two classes, seven numerical attributes, and 532 records. The original dataset consists of 768 records with eight numerical attributes. However, many of the attributes, notably serum insulin, contain zero values which are physically impossible. We remove serum insulin and records that have impossible values in other attributes. The error rates are estimated using ten-fold cross validation.

StatLog satellite image (`sat`). This UCI dataset gives the multi-spectral values of pixels within 3×3 neighborhoods in a satellite image, and the classification associated with the central pixel in each neighborhood. The aim is to predict the classification given the multi-spectral values. There are six classes and thirty-six numerical attributes. The training set consists of 4435 records while the test set consists of 2000 records. The error rates are estimated from the test set.

Image segmentation (`seg`). This UCI dataset was used in the StatLog Project. The samples are from a database of seven outdoor images. The images are hand-segmented to create a classification for every pixel as one of brickface, sky, foliage, cement, window, path, or grass. There are seven classes, nineteen numerical attributes and 2310 records in the dataset. The error rates are estimated using ten-fold cross-validation.

The algorithm `T1` could not handle this dataset without modification, because the program requires a large amount of memory. Therefore for `T1` (but not for the other algorithms) we discretize each attribute except attributes 3, 4, and 5 into one hundred categories.

Attitude towards smoking restrictions (`smo`). This survey dataset [11] is obtained from `http://lib.stat.cmu.edu/datasets/csb/`. The problem is to predict attitude toward restrictions on smoking in the workplace (prohibited, restricted, or unrestricted) based on bylaw-related, smoking-related, and sociodemographic covariates. There are three classes, three numerical attributes,

and five categorical attributes. We divide the original dataset into a training set of size 1855 and a test set of size 1000. The error rates are estimated from the test set.

Thyroid disease (`thy`). This is the UCI `ann-train` dataset contributed by R. Werner. The problem is to determine whether or not a patient is hyperthyroid. There are three classes (normal, hyperfunction, and subnormal functioning), six numerical attributes, and fifteen binary attributes. The training set consists of 3772 records and the test set has 3428 records. The error rates are estimated from the test set.

StatLog vehicle silhouette (`veh`). This UCI dataset originated from the Turing Institute, Glasgow, Scotland. The problem is to classify a given silhouette as one of four types of vehicle, using a set of features extracted from the silhouette. Each vehicle is viewed from many angles. The four model vehicles are double decker bus, Chevrolet van, Saab 9000, and Opel Manta 400. There are four classes, eighteen numerical attributes, and 846 records. The error rates are estimated using ten-fold cross-validation.

Congressional voting records (`vot`). This UCI dataset gives the votes of each member of the U. S. House of Representatives of the 98th Congress on sixteen key issues. The problem is to classify a Congressman as a Democrat or a Republican based on the sixteen votes. There are two classes, sixteen categorical attributes with three categories each ("yea", "nay", or neither), and 435 records. Error rates are estimated by ten-fold cross-validation.

Waveform (`wav`). This is an artificial three-class problem based on three waveforms. Each class consists of a random convex combination of two waveforms sampled at the integers with noise added. A description for generating the data is given in Breiman, Friedman, Olshen and Stone (1984) and a C program is available from UCI. There are twenty-one numerical attributes, and 600 records in the training set. Error rates are estimated from an independent test set of 3000 records.

TA evaluation (`tae`). The data consist of evaluations of teaching performance over three regular semesters and two summer semesters of 151 teaching assistant (TA) assignments at the Statistics Department of the University of Wisconsin–Madison. The scores are grouped into three roughly equal-sized categories ("low", "medium", and "high") to form the class attribute. The predictor attributes are (i) whether or not the TA is a native English speaker (binary), (ii) course instructor (25 categories), (iii) course (26 categories), (iv) summer or regular semester (binary), and (v) class size (numerical). This dataset is first reported in Loh and Shih (1997). It differs from the other datasets in that there are two categorical attributes with large numbers of categories. As a result, decision tree algorithms such as CART that employ exhaustive search usually take much longer to train than other algorithms. (CART has to evaluate $2^{c-1} - 1$ splits for each categorical attribute with c values.) Error rates are estimated using ten-fold cross-validation. The dataset is available from UCI.

A summary of the attribute features of the datasets is given in Table 1.

Table 1. Characteristics of the datasets. The last three columns give the number and type of added noise attributes for each dataset. The number of values taken by the class attribute is denoted by J. The notation "N(0,1)" denotes the standard normal distribution, "UI(m,n)" a uniform distribution over the integers m through n inclusive, and "U(0,1)" a uniform distribution over the unit interval. The abbreviation C(k) stands for UI(1,k).

			No. of original attributes								Noise attributes	
			Num.	Categorical						Tot.	Numerical	Categor.
Set	Size	J		2	3	4	5	25	26			
bcw	683	2	9							9	9 UI(1,10)	
cmc	1473	3	2	3		4				9	6 N(0,1)	
dna	2000	3				60				60		20 C(4)
hea	270	2	7	3	2	1				13	7 N(0,1)	
bos	506	3	12	1						13	12 N(0,1)	
led	2000	10		7						7		17 C(2)
bld	345	2	6							6	9 N(0,1)	
pid	532	2	7							7	8 N(0,1)	
sat	4435	6	36							36	24 UI(20,160)	
seg	2310	7	19							19	9 N(0,1)	
smo	1855	3	3	3	1		1			8	7 N(0,1)	
thy	3772	3	6	15						21	4 U(0,1)	10 C(2)
veh	846	4	18							18	12 N(0,1)	
vot	435	2			16					16		14 C(3)
wav	600	3	21							21	19 N(0,1)	
tae	151	3	1	2				1	1	5	5 N(0,1)	

4. Experimental setup

Some algorithms are not designed for categorical attributes. In these cases, each categorical attribute is converted into a vector of 0-1 attributes. That is, if a categorical attribute X takes k values $\{c_1, c_2, \ldots, c_k\}$, it is replaced by a $(k-1)$-dimensional vector $(d_1, d_2, \ldots, d_{k-1})$ such that $d_i = 1$ if $X = c_i$ and $d_i = 0$ otherwise, for $i = 1, \ldots, k-1$. If $X = c_k$, the vector consists of all zeros. The affected algorithms are all the statistical and neural network algorithms as well as the tree algorithms `FTL`, `OCU`, `OCL`, `OCM`, and `LMT`.

In order to increase the number of datasets and to study the effect of noise attributes on each algorithm, we created sixteen new datasets by adding independent noise attributes. The numbers and types of noise attributes added are given in the right panel of Table 1. The name of each new dataset is the same as the original dataset except for the addition of a '`+`' symbol. For example, the `bcw` dataset with noise added is denoted by `bcw+`.

For each dataset, we use one of two different ways to estimate the error rate of an algorithm. For large datasets (size much larger than 1000 and test set of size at least 1000), we use a test set to estimate the error rate. The classifier is constructed

using the records in the training set and then it is tested on the test set. Twelve of the thirty-two datasets are analyzed this way.

For the remaining twenty datasets, we use the following ten-fold cross-validation procedure to estimate the error rate:

1. The dataset is randomly divided into ten disjoint subsets, with each containing approximately the same number of records. Sampling is stratified by the class labels to ensure that the subset class proportions are roughly the same as those in the whole dataset.

2. For each subset, a classifier is constructed using the records not in it. The classifier is then tested on the withheld subset to obtain a cross-validation estimate of its error rate.

3. The ten cross-validation estimates are averaged to provide an estimate for the classifier constructed from all the data.

Because the algorithms are implemented in different programming languages and some languages are not available on all platforms, three types of UNIX workstations are used in our study. The workstation type and implementation language for each algorithm are given in Table 2. The relative performance of the workstations according to SPEC marks is given in Table 3. The floating point SPEC marks show that a task that takes one second on a DEC 3000 would take about 1.4 and 0.8 seconds on a SPARCstation 5 (SS5) and SPARCstation 20 (SS20), respectively. Therefore, to enable comparisons, all training times are reported here in terms of DEC 3000-equivalent seconds—the training times recorded on a SS5 and a SS20 are divided by 1.4 and 0.8, respectively.

5. Results

We only report the summary results and analysis here. Fuller details, including the error rate and training time of each algorithm on each dataset, may be obtained from `http://id001.wkap.nl/mach/ml-appe.htm`.

5.1. Exploratory analysis of error rates

Before we present a formal statistical analysis of the results, it is helpful to study the summary in Table 4. The mean error rate for each algorithm over the datasets is given in the second row. The minimum and maximum error rates and that of the plurality rule are given for each dataset in the last three columns. Let p denote the smallest observed error rate in each row (i.e., dataset). If an algorithm has an error rate within one standard error of p, we consider it to be close to the best and indicate it by a $\surd$ in the table. The standard error is estimated as follows. If p is from an independent test set, let n denote the size of the test set. Otherwise, if p is a cross-validation estimate, let n denote the size of the training set. The standard error of p is estimated by the formula $\sqrt{p(1-p)/n}$. The algorithm with the largest

Table 2. Hardware and software platform for each algorithm. The workstations are DEC 3000 Alpha Model 300 (DEC), SUN SPARCstation 20 Model 61 (SS20), and SUN SPARCstation 5 (SS5).

Algorithm		Platform	Algorithm		Platform
Tree & Rules			ST1	S-PLUS tree, 1-SE	DEC/S
QU0	QUEST, univar. 0-SE	DEC/F90	LMT	LMDT, linear	DEC/C
QU1	QUEST, univar. 1-SE	DEC/F90	CAL	CAL5	SS5/C++
QL0	QUEST, linear 0-SE	DEC/F90	T1	T1, single split	DEC/C
QL1	QUEST, linear 1-SE	DEC/F90			
FTU	FACT, univariate	DEC/F77	Statistical		
FTL	FACT, linear	DEC/F77	LDA	Linear discriminant anal.	DEC/SAS
C4T	C4.5 trees	DEC/C	QDA	Quadratic discriminant anal.	DEC/SAS
C4R	C4.5 rules	DEC/C	NN	Nearest-neighbor	DEC/SAS
IB	IND bayes style	SS5/C	LOG	Linear logistic regression	DEC/F90
IB0	IND bayes opt style	SS5/C	FM1	FDA, degree 1	SS20/S
IM	IND mml style	SS5/C	FM2	FDA, degree 2	SS20/S
IM0	IND mml opt style	SS5/C	PDA	Penalized LDA	SS20/S
IC0	IND-CART, 0-SE	SS5/C	MDA	Mixture discriminant anal.	SS20/S
IC1	IND-CART, 1-SE	SS5/C	POL	POLYCLASS	SS20/S
OCU	OC1, univariate	SS5/C			
OCL	OC1, linear	SS5/C	Neural Network		
OCM	OC1, mixed	SS5/C	LVQ	Learning vector quantization	SS20/S
ST0	S-PLUS tree, 0-SE	DEC/S	RBF	Radial basis function network	DEC/SAS

Table 3. SPEC benchmark summary

	Workstation	SPECfp92	SPECint92	Source
DEC	DEC 3000 Model 300 (150MHz)	91.5	66.2	SPEC Newsletter Vol. 5, Issue 2, June 1993
SS20	SUN SPARCstation 20 Model 61 (60MHz)	102.8	88.9	SPEC Newsletter Vol. 6, Issue 2, June 1994
SS5	SUN SPARCstation 5 (70MHz)	47.3	57.0	SPEC Newsletter Vol. 6, Issue 2, June 1994

error rate within a row is indicated by an X. The total numbers of $\surd$ and X-marks for each algorithm are given in the third and fourth rows of the table.

The following conclusions may be drawn from the table:

1. Algorithm POL has the lowest mean error rate. An ordering of the other algorithms in terms of mean error rate is given in the upper half of Table 5.

2. The algorithms can also be ranked in terms of total number of $\surd$- and X-marks. By this criterion, the most accurate algorithm is again POL, which has fifteen $\surd$-marks and no X-marks. Eleven algorithms have one or more X-marks. Ranked in increasing order of number of X-marks (in parentheses), they are:

FTL(1), OCM(1), ST1(1), FM2(1), MDA(1), FM1(2),

Table 4. Minimum, maximum, and 'naive' plurality rule error rates for each dataset. A '√'-mark indicates that the algorithm has an error rate within one standard error of the minimum for the dataset. A 'X'-mark indicates that the algorithm has the worst error rate for the dataset. The mean error rate for each algorithm is given in the second row.

	Decision trees and rules																					Statistical algorithms									Nets		Error rates		
	QUO	QU1	QLO	QL1	FTU	FTL	C4T	C4R	IB	IBO	IM	IMO	ICO	IC1	OCU	OCL	OCM	ST0	ST1	LMT	CAL	T1	LDA	QDA	NN	LOG	FM1	FM2	PDA	MDA	POL	LVQ	RBF	Min	Max	Naive
Mean	.221	.226	.208	.211	.238	.234	.220	.220	.229	.219	.220	.219	.215	.227	.227	.260	.230	.232	.233	.220	.270	.354	.208	.301	.281	.204	.242	.217	.213	.207	.195	.269	.257			
#√	8	8	10	9	4	12	7	8	3	8	5	6	4	5	9	4	4	5	5	1	4	4	10	4	4	13	12	12	10	9	15	4	8			
#X	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	3	1	0	1	0	0	11	0	3	4	0	2	1	0	1	0	4	0			
bcw		√	√							√						X										√						√	√	.03	.09	.35
bcw+		√	√																			X				√	√	√		√		√	√	.03	.08	
cmc	√	√																							X						√			.43	.60	.57
cmc+	√	√																							X						√			.43	.58	
dna						√																X					√	√			√			.05	.38	.49
dna+						√																X						√			√			.04	.38	
hea		√	√			√																	√			√			√			X		.14	.34	.44
hea+			√			√																	√						√			X		.15	.31	
bos							√	√			√						√								√		√				√	X	√	.22	.31	.66
bos+							√								√												√	X			√			.23	.42	
led		√				√	√	√		√	√				√	√						X	√	√		√	√	√	√	√	√			.27	.82	.89
led+						√									√							X	√			√	√	√	√	√	√			.27	.81	
bld								√								√	√					X					√	√			√			.28	.43	.42
bld+																						X									√			.29	.44	
pid	√	√	√	√		√		√			√		√		√	X		√			√		√	√		√	√		√	√	√		√	.22	.31	.33
pid+	√	√	√	√		√		√					√			X							√			√	√	√	√	√	√			.22	.32	
sat																						X										√		.10	.40	.77
sat+																												√				X		.12	.41	
seg										√																√		X						.02	.52	.86
seg+										√																		X						.03	.57	
smo	√	√	√	√	√	√	√				√	√		√	√	√	√	√	√		√	√	√	X		√	√	√	√	√	√		√	.30	.45	.31
smo+	√	√	√	√	√	√					√		√	√	√	√	√	√	√	√	√	√	√		X	√		√	√	√	√		√	.31	.45	
thy							√	√	√				√	√	√			√	√					X										.01	.89	.07
thy+							√	√	√		√	√												X										.01	.88	
veh																						X		√										.15	.49	.74
veh+																						X		√										.16	.49	
vot	√	√	√		√					√		√		√	√				√		√							√		X				.04	.06	.39
vot+	√	√	√	√	√	√	√	√		√		√		√	√		X	√	√		√	√	√		√	√	√	√		√		√	√	.04	.07	
wav																						X				√							√	.15	.48	.67
wav+						√																			X	√	√			√		√		.16	.45	
tae						X				√															√									.33	.69	.66
tae+									√	√									X				√		√	√			√	√				.45	.70	

OCL(3), QDA(3), NN(4), LVQ(4), T1(11).

Excluding these, the remaining algorithms rank in order of decreasing number of √-marks (in parentheses) as:

$$\text{POL}(15),\ \text{LOG}(13),\ \text{QLO}(10),\ \text{LDA}(10),\ \text{PDA}(10),\ \text{QL1}(9),\ \text{OCU}(9), \qquad (1)$$

Table 5. Ordering of algorithms by mean error rate and mean rank of error rate

Mean error rate	POL	LOG	MDA	QLO	LDA	QL1	PDA	ICO	FM2	IBO	IMO
	.195	.204	.207	.208	.208	.211	.213	.215	.217	.219	.219
	C4R	LMT	IM	C4T	QUO	QU1	OCU	IC1	IB	OCM	STO
	.220	.220	.220	.220	.221	.226	.227	.227	.229	.230	.232
	ST1	FTL	FTU	FM1	RBF	OCL	LVQ	CAL	NN	QDA	T1
	.233	.234	.238	.242	.257	.260	.269	.270	.281	.301	.354
Mean rank of error rate	POL	LOG	FM1	FM2	QLO	LDA	QUO	C4R	IMO	MDA	PDA
	8.3	12.1	12.2	12.2	12.6	13.7	13.9	14.0	14.0	14.3	14.5
	C4T	IBO	QL1	ICO	IM	FTL	OCU	QU1	IC1	STO	ST1
	14.5	14.7	14.8	14.9	14.9	15.4	16.6	16.7	16.8	17.0	17.6
	LMT	OCM	IB	RBF	FTU	QDA	LVQ	OCL	CAL	NN	T1
	18.5	18.9	19.0	19.1	20.7	22.8	24.0	24.3	25.1	25.5	27.5

QUO(8), QU1(8), C4R(8), IBO(8), RBF(8), C4T(7), IMO(6),
IM(5), IC1(5), STO(5), FTU(4), ICO(4), CAL(4), IB(3), LMT(1).

The top four algorithms in (1) also rank among the top five in the upper half of Table 5.

3. The last three columns of the table show that a few algorithms are sometimes less accurate than the plurality rule. They are NN (at cmc, cmc+, smo+), T1 (bld, bld+), QDA (smo, thy, thy+), FTL (tae), and ST1 (tae+).

4. The easiest datasets to classify are bcw, bcw+, vot, and vot+; the error rates all lie between 0.03 and 0.09.

5. The most difficult to classify are cmc, cmc+, and tae+, with minimum error rates greater than 0.4.

6. Two other difficult datasets are smo and smo+. In the case of smo, only T1 has a (marginally) lower error rate than that of the plurality rule. No algorithm has a lower error rate than the plurality rule for smo+.

7. The datasets with the largest range of error rates are thy and thy+, where the rates range from 0.005 to 0.890. However, the maximum of 0.890 is due to QDA. If QDA is ignored, the maximum error rate drops to 0.096.

8. There are six datasets with only one $\surd$-mark each. They are bld+ (POL), sat (LVQ), sat+ (FM2), seg+ (IBO), veh and veh+ (QDA both times).

9. Overall, the addition of noise attributes does not appear to increase significantly the error rates of the algorithms.

5.2. Statistical significance of error rates

5.2.1. Analysis of variance A statistical procedure called mixed effects analysis of variance can be used to test the simultaneous statistical significance of differences between mean error rates of the algorithms, while controlling for differences between datasets [39]. Although it makes the assumption that the effects of the datasets act like a random sample from a normal distribution, it is quite robust against violation of the assumption. For our data, the procedure gives a significance probability less than 10^{-4}. Hence the null hypothesis that all the algorithms have the same mean error rate is strongly rejected.

Simultaneous confidence intervals for differences between mean error rates can be obtained using the Tukey method [35]. According to this procedure, a difference between the mean error rates of two algorithms is statistically significant at the 10% level if they differ by more than 0.058.

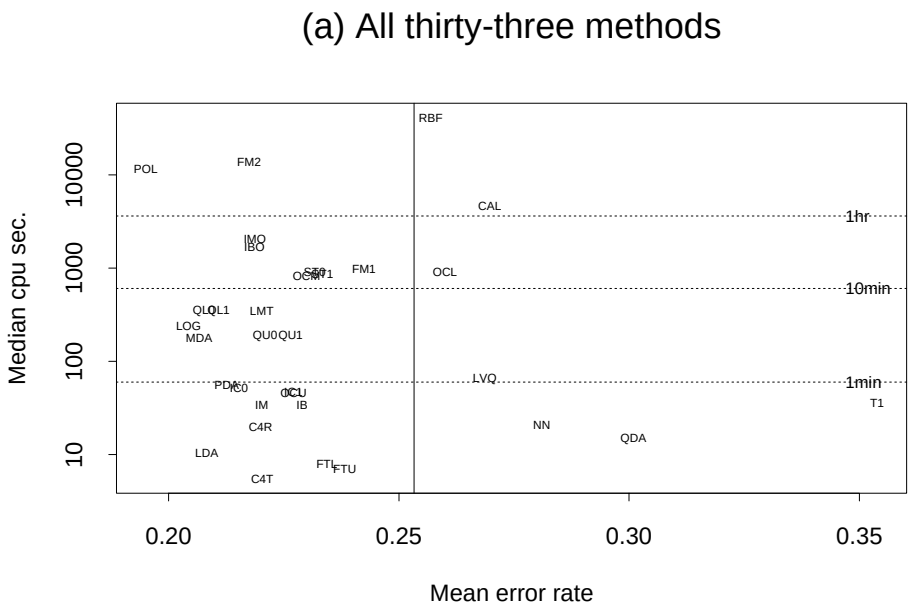
To visualize this result, Figure 1(a) plots the mean error rate of each algorithm versus its median training time in seconds. The solid vertical line in the plot is 0.058 units to the right of the mean error rate for `POL`. Therefore any algorithm lying to the left of the line has a mean error rate that is not statistically significantly different from that of `POL`.

The algorithms are seen to form four clusters with respect to training time. These clusters are roughly delineated by the three horizontal dotted lines which correspond to training times of one minute, ten minutes, and one hour. Figure 1(b) shows a magnified plot of the eighteen algorithms with median training times less than ten minutes and mean error rate not statistically significantly different from `POL`.

5.2.2. Analysis of ranks To avoid the normality assumption, we can instead analyze the ranks of the algorithms within datasets. That is, for each dataset, the algorithm with the lowest error rate is assigned rank one, the second lowest rank two, etc., with average ranks assigned in the case of ties. The lower half of Table 5 gives an ordering of the algorithms in terms of mean rank of error rates. Again `POL` is first and `T1` last. Note, however, that the mean rank of `POL` is 8.3. This shows that it is far from being uniformly most accurate across datasets.

Comparing the two methods of ordering in Table 5, it is seen that `POL`, `LOG`, `QLO`, and `LDA` are the only algorithms with consistently good performance. Three algorithms that perform well by one criterion but not the other are `MDA`, `FM1`, and `FM2`. In the case of `MDA`, its low mean error rate is due to its excellent performance in four datasets (`veh`, `veh+`, `wav`, and `wav+`) where many other algorithms do poorly. These domains concern shape identification and the datasets contain only numerical attributes. `MDA` is generally unspectacular in the rest of the datasets and this is the reason for its tenth place ranking in terms of mean rank.

The situation for `FM1` and `FM2` is quite different. As its low mean rank indicates, `FM1` is usually a good performer. However, it fails miserably in the `seg` and `seg+` datasets, reporting error rates of more than fifty percent when most of the other algorithms have error rates less than ten percent. Thus `FM1` seems to be less robust than the other algorithms. `FM2` also appears to lack robustness, although to a



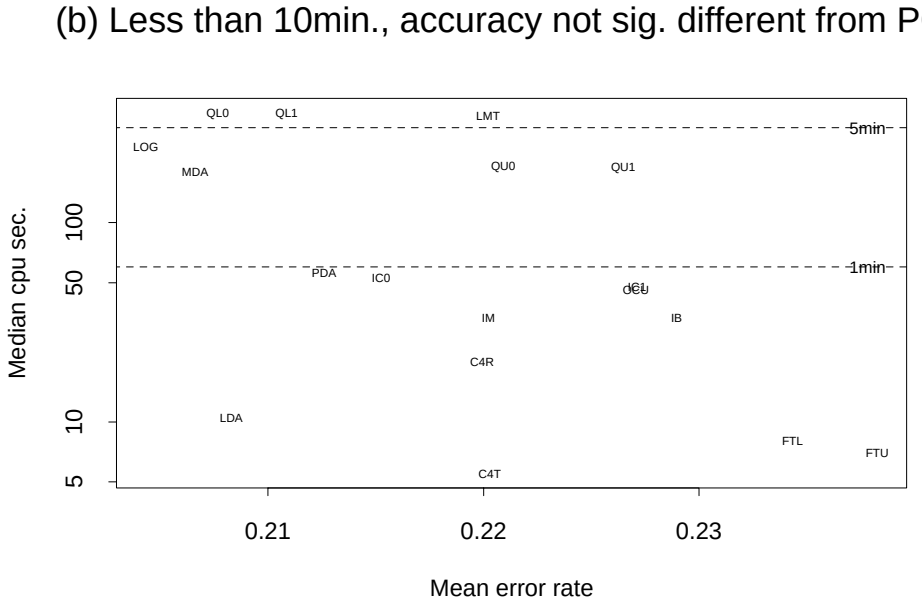


Figure 1. Plots of median training time versus mean error rate. The vertical axis is in log-scale. The solid vertical line in plot (a) divides the algorithms into two groups: the mean error rates of the algorithms in the left group do not differ significantly (at the 10% simultaneous significance level) from that of `POL`, which has the minimum mean error rate. Plot (b) shows the algorithms that are not statistically significantly different from `POL` in terms of mean error rate and that have median training time less than ten minutes.

lesser extent. Its worst performance is in the bos+ dataset, where it has an error rate of forty-two percent, compared to less than thirty-five percent for the other algorithms. The number of X-marks against an algorithm in Table 4 is a good predictor of erratic if not poor performance. MDA, FM1, and FM2 all have at least one X-mark.

The Friedman (1937) test is a standard procedure for testing statistical significance in differences of mean ranks. For our experiment, it gives a significance probability less than 10^{-4}. Therefore the null hypothesis that the algorithms are equally accurate on average is again rejected. Further, a difference in mean ranks greater than 8.7 is statistically significant at the 10% level [24]. Thus POL is not statistically significantly different from the twenty other algorithms that have mean rank less than or equal to 17.0. Figure 2(a) shows a plot of median training time versus mean rank. Those algorithms that lie to the left of the vertical line are not statistically significantly different from POL. A magnified plot of the subset of algorithms that are not significantly different from POL and that have median training time less than ten minutes is given in Figure 2(b).

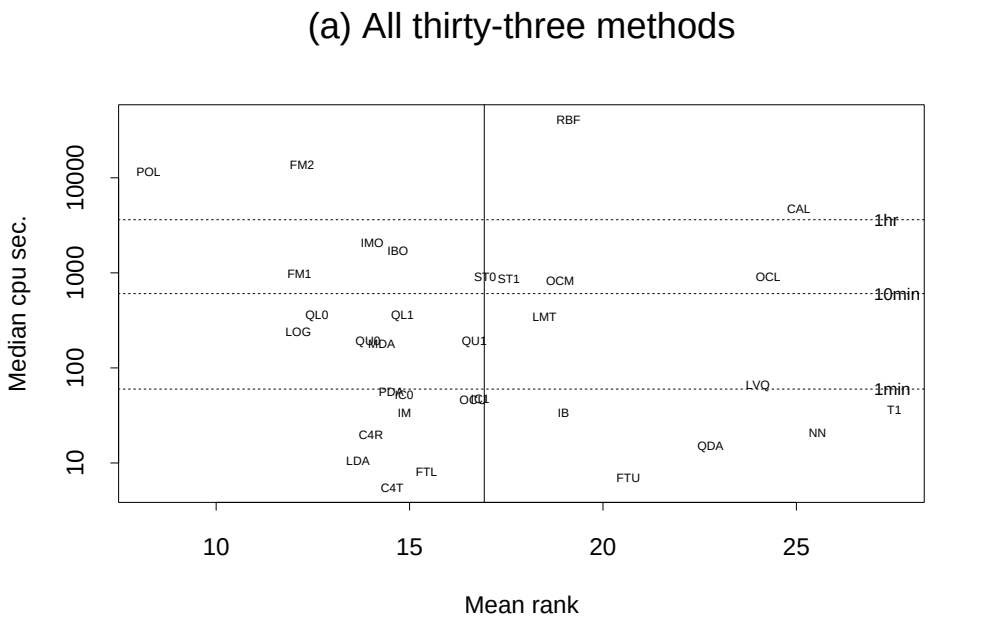
The algorithms that differ statistically significantly from POL in terms of mean error rate form a subset of those that differ from POL in terms of mean ranks. Thus the rank test appears to be more powerful than the analysis of variance test for this experiment. The fifteen algorithms in Figure 2(b) may be recommended for use in applications where good accuracy and short training time are desired.

5.3. Training time

Table 6 gives the median DEC 3000-equivalent training time for each algorithm and the relative training time within datasets. Owing to the large range of training times, only the order relative to the fastest algorithm for each dataset is reported. The fastest algorithm is indicated by a '0'. An algorithm that is between 10^{x-1} to 10^x times as slow is indicated by the value of x. For example, in the case of the dna+ dataset, the fastest algorithms are C4T and T1, each requiring two seconds. The slowest algorithm is FM2, which takes more than three million seconds (almost forty days) and hence is between 10^6 to 10^7 times as slow. The last two columns of the table give the fastest and slowest times for each dataset.

Table 7 gives an ordering of the algorithms from fastest to slowest according to median training time. Overall, the fastest algorithm is C4T, followed closely by FTU, FTL, and LDA. There are two reasons for the superior speed of C4T compared to the other decision tree algorithms. First, it splits each categorical attribute into as many subnodes as the number of categories. Therefore it wastes no time in forming subsets of categories. Second, its pruning method does not require cross-validation, which can increase training time several fold.

The classical statistical algorithms QDA and NN are also quite fast. As expected, decision tree algorithms that employ univariate splits are faster than those that use linear combination splits. The slowest algorithms are POL, FM2, and RBF; two are spline-based and one is a neural network.



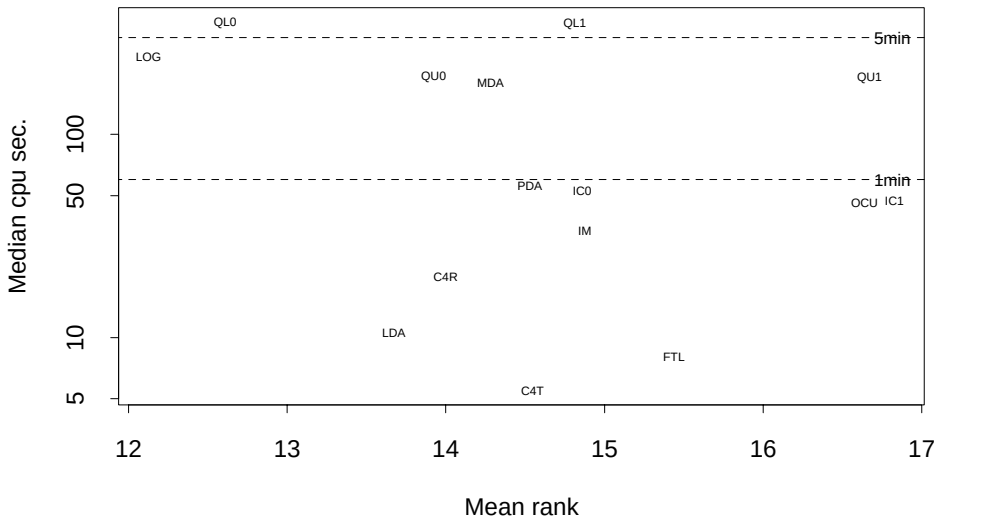


Figure 2. Plots of median training time versus mean rank of error rates. The vertical axis is in log-scale. The solid vertical line in plot (a) divides the algorithms into two groups: the mean ranks of the algorithms in the left group do not differ significantly (at the 10% simultaneous significance level) from that of `POL`. Plot (b) shows the algorithms that are not statistically significantly different from `POL` in terms of mean rank and that have median training time less than ten minutes.

Table 6. DEC 3000-equivalent training times and relative times of the algorithms. The second and third rows give the median training time and rank for each algorithm. An entry of 'x' in the each of the subsequent rows indicates that an algorithm is between 10^{x-1}–10^x times slower than the fastest algorithm for the dataset. The fastest algorithm is denoted by an entry of '0'. The minimum and maximum training times are given in the last two columns. 's', 'm', 'h', 'd' denote seconds, minutes, hours, and days, respectively.

	Decision trees and rules																					Statistical algorithms									Nets		CPU time	
	QUO	QU1	QLO	QL1	FTU	FTL	C4T	C4R	IB	IBO	IM	IMO	ICO	IC1	OCU	OCL	OCM	STO	ST1	LMT	CAL	T1	LDA	QDA	NN	LOG	FM1	FM2	PDA	MDA	POL	LVQ	RBF	Min	Max
Time	3.2m	3.2m	5.9m	5.9m	7s	8s	5s	20s	34s	27.5m	34s	33.9m	52s	47s	46s	14.9m	13.7m	15.1m	14.4m	5.7m	1.3h	36s	10s	15s	20s	4m	15.6m	3.8h	56s	3m	3.2h	1.1m	11.3h	5s	11.3h
Rank	18	17	22	21	2	3	1	6	8	28	9	29	13	12	11	25	23	26	24	20	30	10	4	5	7	19	27	32	14	16	31	15	33		
bcw+	2	2	2	2	1	1	0	1	1	3	1	3	2	2	1	3	3	3	3	2	3	1	1	1	1	2	3	3	2	2	4	2	4	4s	2.7h
cmc+	2	2	3	3	0	1	1	2	1	3	1	3	2	2	2	3	3	3	3	2	4	2	1	1	1	2	3	4	2	2	4	2	4	12s	23.9h
dna+	3	3	3	3	1	2	0	2	1	2	1	2	1	1	3	3	3	3	3	4	5	0	2	2	3	3	5	7	3	3	5	4	6	2s	39.6d
hea+	2	2	2	2	1	1	0	1	1	4	1	3	2	2	1	2	2	3	3	2	4	1	1	1	1	2	3	3	1	2	3	1	4	4s	3.3h
bos+	2	2	2	2	0	0	1	1	1	3	1	3	2	2	1	3	3	3	3	2	4	2	1	1	1	2	3	2	1	2	4	1	4	9s	5.5h
led+	3	3	4	4	1	1	1	2	1	3	1	3	2	1	2	3	3	3	3	3	4	0	1	2	2	3	3	4	2	3	4	2	5	1s	12.4h
bld+	2	2	2	2	1	0	1	1	1	4	1	4	2	2	1	3	3	3	3	2	3	1	1	1	1	1	2	3	1	2	3	1	3	5s	1.5h
pid+	2	2	2	2	0	0	1	1	1	3	1	3	2	2	1	3	3	3	3	2	3	1	1	1	1	1	2	3	1	2	4	1	3	7s	2.5h
sat+	3	3	3	3	0	1	1	2	2	3	2	3	2	2	2	4	3	3	3	3	4	2	1	1	2	3	4	5	2	2	4	3	5	8s	6.1d
seg+	2	2	2	2	1	1	1	1	2	2	2	2	2	2	1	3	3	2	2	2	4	2	0	1	1	3	3	4	2	2	4	2	5	28s	6.3d
smo+	3	3	3	3	0	0	1	2	2	3	2	3	2	2	2	3	3	3	3	3	4	2	1	1	1	2	3	4	2	2	5	2	4	1s	3.8h
thy+	2	2	3	3	1	1	0	1	1	2	1	2	1	1	2	3	2	3	3	3	3	2	1	1	2	2	3	4	2	2	5	3	5	3s	16.1h
veh+	2	2	2	2	1	1	1	1	2	3	2	3	2	2	1	3	3	3	3	2	4	3	0	1	1	2	4	4	1	2	4	2	4	14s	1.2d
vot+	2	2	2	2	1	1	0	1	2	4	2	3	2	2	2	3	3	3	3	3	4	0	1	1	2	3	4	5	2	3	4	2	5	2s	2.1d
wav+	2	2	2	2	1	1	1	1	1	3	1	3	2	2	1	2	2	2	2	2	3	2	0	1	1	2	3	4	1	2	4	1	4	4s	4.3h
tae+	2	2	2	2	0	0	1	1	1	2	1	3	1	1	1	1	2	4	4	2	3	1	1	1	1	2	3	4	1	2	3	1	4	6s	10.2h

Table 7. Ordering of algorithms by median training time

C4T	FTU	FTL	LDA	QDA	C4R	NN	IB	IM	T1	OCU
5s	7s	8s	10s	15s	20s	20s	34s	34s	36s	46s
IC1	ICO	PDA	LVQ	MDA	QU1	QUO	LOG	LMT	QL1	QLO
47s	52s	56s	1.1m	3m	3.2m	3.2m	4m	5.7m	5.9m	5.9m
OCM	ST1	OCL	STO	FM1	IBO	IMO	CAL	POL	FM2	RBF
13.7m	14.4m	14.9m	15.1m	15.6m	27.5m	33.9m	1.3h	3.2h	3.8h	11.3h

Although ICO, IC1, STO and ST1 all claim to implement the CART algorithm, the IND versions are faster than the S-PLUS versions. One reason is that ICO and IC1 are written in C whereas STO and ST1 are written in the S language. Another reason is that the IND versions use heuristics (Buntine, personal communication) instead of greedy search when the number of categories in a categorical attribute is large. This is most apparent in the tae+ dataset where there are categorical attributes

with up to twenty-six categories. In this case IC0 and IC1 take around forty seconds versus two and a half hours for ST0 and ST1. The results in Table 4 indicate that IND's classification accuracy is not adversely affected by such heuristics; see Aronis and Provost (1997) for another possible heuristic.

Since T1 is a one-level tree, it may appear surprising that it is not faster than algorithms such as C4T that produce multi-level trees. The reason is that T1 splits each continuous attribute into $J + 1$ intervals, where J is the number of classes. On the other hand, C4T always splits a continuous attribute into two intervals only. Therefore when $J > 2$, T1 has to spend a lot more time to search for the intervals.

5.4. Size of trees

Table 8 gives the number of leaves for each tree algorithm and dataset before noise attributes are added. In the case that an error rate is obtained by ten-fold cross-validation, the entry is the mean number of leaves over the ten cross-validation trees. Table 9 shows how much the number of leaves changes after addition of noise attributes. The mean and median of the number of leaves for each classifier are given in the last columns of the two tables. IBO and IMO clearly yield the largest trees by far. Apart from T1, which is necessarily short by design, the algorithm with the shortest trees on average is QL1, followed closely by FTL and OCL. A ranking of the algorithms with univariate splits (in increasing median number of leaves) is: T1, IC1, ST1, QU1, FTU, IC0, ST0, OCU, QU0, and C4T. Algorithm C4T tends to produce trees with many more leaves than the other algorithms. One reason may be due to under-pruning (although its error rates are quite good). Another is that, unlike the binary-tree algorithms, C4T splits each categorical attribute into as many nodes as the number of categories.

Addition of noise attributes typically decreases the size of the trees, except for C4T and CAL which tend to grow larger trees, and IMO which seems to fluctuate rather wildly. These results complement those of Oates and Jensen (1997) who looked at the effect of sample size on the number of leaves of decision tree algorithms and found a significant relationship between tree size and training sample size for C4T. They observed that tree algorithms which employ cost-complexity pruning are better able to control tree growth.

6. Scalability of algorithms

Although differences in mean error rates between POL and many other algorithms are not statistically significant, it is clear that if error rate is the sole criterion, POL would be the method of choice. Unfortunately, POL is one of the most compute-intensive algorithms. To see how training times increase with sample size, a small scalability study was carried out with the algorithms QU0, QL0, FTL, C4T, C4R, IC0, LDA, LOG, FM1, and POL.

Training times are measured for these algorithms for training sets of size $N = 1000, 2000, \ldots, 8000$. Four datasets are used to generate the samples—sat, smo+, tae+, and a new, very large UCI dataset called adult which has two classes and six

Table 8. Number of leaves, including means and medians, of decision tree algorithms. The numbers for `bcw`, `cmc`, `hea`, `bos`, `bld`, `pid`, `seg`, `veh`, `vot`, and `tae` are means from ten-fold cross-validation experiments.

	Dataset																	
Alg.	bcw	cmc	dna	hea	bos	led	bld	pid	sat	seg	smo	thy	veh	vot	wav	tae	Mean	Med.
QU0	9	11	13	14	24	31	29	3	140	66	1	20	68	3	34	21	30	21
QU1	7	10	13	6	7	24	10	2	112	44	1	18	31	2	16	15	20	12
QL0	3	24	7	2	6	31	6	5	30	39	1	13	16	2	5	10	13	7
QL1	2	11	5	2	3	15	4	2	11	21	1	6	8	2	5	6	7	5
FTU	5	23	12	6	14	21	5	6	102	53	1	19	38	2	26	9	21	13
FTL	3	3	5	3	4	12	2	3	49	18	1	13	22	2	3	1	9	3
C4T	11	143	97	23	36	29	26	18	216	42	1	12	65	10	54	79	54	33
IB	31	894	325	48	79	103	73	81	348	58	817	17	123	49	66	262	211	80
IB0	27829	26153	7513	35238	26278	415	26862	23889	3174	13759	9884	129	9683	46695	24295	5548	17959	18824
IM	29	965	316	48	79	102	73	79	342	80	824	15	123	48	66	392	224	80
IM0	23939	12325	3582	34839	25500	207	31247	30375	1381	2964	4849	290	1390	19838	18641	11653	13939	11989
IC0	13	12	13	7	18	22	12	9	63	69	6	11	71	5	14	43	24	13
IC1	7	8	12	5	6	21	7	5	29	28	1	6	27	2	9	14	12	8
OCU	19	48	15	6	9	30	4	7	70	39	10	6	51	2	15	39	23	15
OCL	4	12	10	3	10	14	5	8	22	16	4	11	14	2	4	2	9	9
OCM	5	8	16	3	7	14	5	5	33	26	2	5	41	2	15	20	13	8
ST0	7	18	13	9	13	22	16	9	33	16	1	6	30	3	26	13	15	13
ST1	5	8	12	3	8	19	7	4	19	15	1	6	24	2	17	7	10	8
LMT	3	46	3	3	11	38	12	11	20	6	61	6	13	2	4	5	15	9
CAL	7	15	37	7	25	40	22	11	178	72	34	16	79	2	62	34	40	30
T1	3	5	5	4	5	3	3	3	8	9	5	5	6	4	5	27	6	5
Mean	2473	1940	573	3347	2483	58	2782	2597	304	831	786	30	568	3175	2066	867		
Med.	7	15	13	6	11	24	10	7	63	39	2	12	38	2	16	20		

Table 9. Differences in the number of leaves of decision tree algorithms for datasets with and without noise. A negative difference means that the tree with noise is shorter than the one without noise.

	Dataset																
Alg.	bcw	cmc	dna	hea	bos	led	bld	pid	sat	seg	smo	thy	veh	vot	wav	tae	Mean	Med.
QU0	-1	3	0	3	-14	38	-17	-1	-5	-10	0	0	-14	0	-19	-8	-3	-1
QU1	-1	0	-1	2	-2	-8	-5	0	-43	-8	0	0	-1	0	-6	-9	-5	-1
QL0	-1	-4	-2	0	-2	1	0	-2	-12	-24	0	-7	-3	0	0	-6	-4	-2
QL1	0	-4	0	0	0	0	-2	0	1	-9	0	0	-2	0	-2	-3	-1	0
FTU	0	3	0	1	1	-1	1	0	-1	0	0	0	1	0	-1	1	0	0
FTL	0	0	1	0	-1	5	0	-1	0	0	0	-1	0	0	0	0	0	0
C4T	0	116	0	1	1	91	7	13	8	0	222	0	-1	1	-9	30	30	1
IB	-5	-399	-3	-9	-15	583	-17	-19	-37	-2	-343	0	-19	-12	-6	-56	-22	-14
IBO	-7113	-11878	-1304	-2369	-3897	13441	-685	3147	2603	405	-166	-18	-5845	-14858	-21228	6782	-2686	-995
IM	-5	-422	0	-10	-15	589	-19	-19	-30	-25	-390	0	-20	-12	-6	-111	-31	-17
IMO	-8181	3806	-598	-15113	4600	4921	-8466	-1126	4726	4234	1001	357	1261	-17810	-8795	10399	-1549	679
IC0	-1	-1	0	1	-10	31	-7	0	-7	-7	-5	0	-26	1	-6	-19	-4	-3
IC1	-2	0	0	0	-2	7	-3	-1	-3	1	0	0	-8	0	-2	-9	-1	-1
OCU	-2	-15	0	0	2	3	1	12	-5	-1	-8	0	-6	0	1	-24	-3	0
OCL	-1	2	-7	-1	-6	2	0	0	-4	-3	-2	-6	-3	0	8	0	-1	-1
OCM	-1	0	-2	1	4	0	1	-1	17	0	0	-1	-16	0	5	-9	0	0
ST0	2	-9	0	-3	-3	4	-6	2	-6	0	0	0	-2	0	8	-3	-1	0
ST1	0	-1	0	1	-3	-1	-4	0	4	0	0	0	-1	0	-1	-1	0	0
LMT	-1	-20	0	-1	-7	-28	-7	-5	-2	-3	-51	-1	-6	0	-1	-3	-9	-3
CAL	4	5	-1	0	1	-27	-6	2	250	12	-26	7	3	0	-47	-10	10	1
T1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Mean	-729	-420	-91	-833	30	936	-440	95	355	217	11	16	-224	-1557	-1434	807		
Med.	-1	0	0	0	-2	3	-4	0	-2	0	0	0	-3	0	-2	-3		

continuous and seven categorical attributes. Since the first three datasets are not large enough for the experiment, bootstrap re-sampling is employed to generate the training sets. That is, N samples are randomly drawn with replacement from each dataset. To avoid getting many replicate records, the value of the class attribute for each sampled case is randomly changed to another value with probability 0.1. (The new value is selected from the pool of alternatives with equal probability.) Bootstrap sampling is not carried out for the `adult` dataset because it has more than 32,000 records. Instead, the nested training sets are obtained by random sampling without replacement.

The times required to train the algorithms are plotted (in log-log scale) in Figure 3. With the exception of `POL`, `FM1` and `LOG`, the logarithms of the training times seem to increase linearly with $\log(N)$. The non-monotonic behavior of `POL` and `FM1` is puzzling and might be due to randomness in their use of cross-validation for model selection. The erratic behavior of `LOG` in the `adult` dataset is caused by convergence problems during model fitting.

Many of the lines in Figure 3 are roughly parallel. This suggests that the relative computational speed of the algorithms is fairly constant over the range of sample sizes considered. `QLO` and `C4R` are two exceptions. Cohen (1995) had observed that `C4R` does not scale well.

7. Conclusions

Our results show that the mean error rates of many algorithms are sufficiently similar that their differences are statistically insignificant. The differences are also probably insignificant in practical terms. For example, the mean error rates of the top ranked algorithms `POL`, `LOG`, and `QLO` differ by less than 0.013. If such a small difference is not important in real applications, the user may wish to select an algorithm based on other criteria such as training time or interpretability of the classifier.

Unlike error rates, there are huge differences between the training times of the algorithms. `POL`, the algorithm with the lowest mean error rate, takes about fifty times as long to train as the next most accurate algorithm. The ratio of times is roughly equivalent to hours versus minutes, and Figure 3 shows that it is maintained over a wide range of sample sizes. For large applications where time is a factor, it may be advantageous to use one of the quicker algorithms.

It is interesting that the old statistical algorithm `LDA` has a mean error rate close to the best. This is surprising because (i) it is not designed for binary-valued attributes (all categorical attributes are transformed to 0-1 vectors prior to application of `LDA`), and (ii) it is not expected to be effective when class densities are multi-modal. Because it is fast, easy to implement, and readily available in statistical packages, it provides a convenient benchmark for comparison against future algorithms.

The low error rates of `LOG` and `LDA` probably account for much of the performance of the better algorithms. For example, `POL` is basically a modern version of `LOG`. It enhances the flexibility of `LOG` by employing spline-based functions and automatic

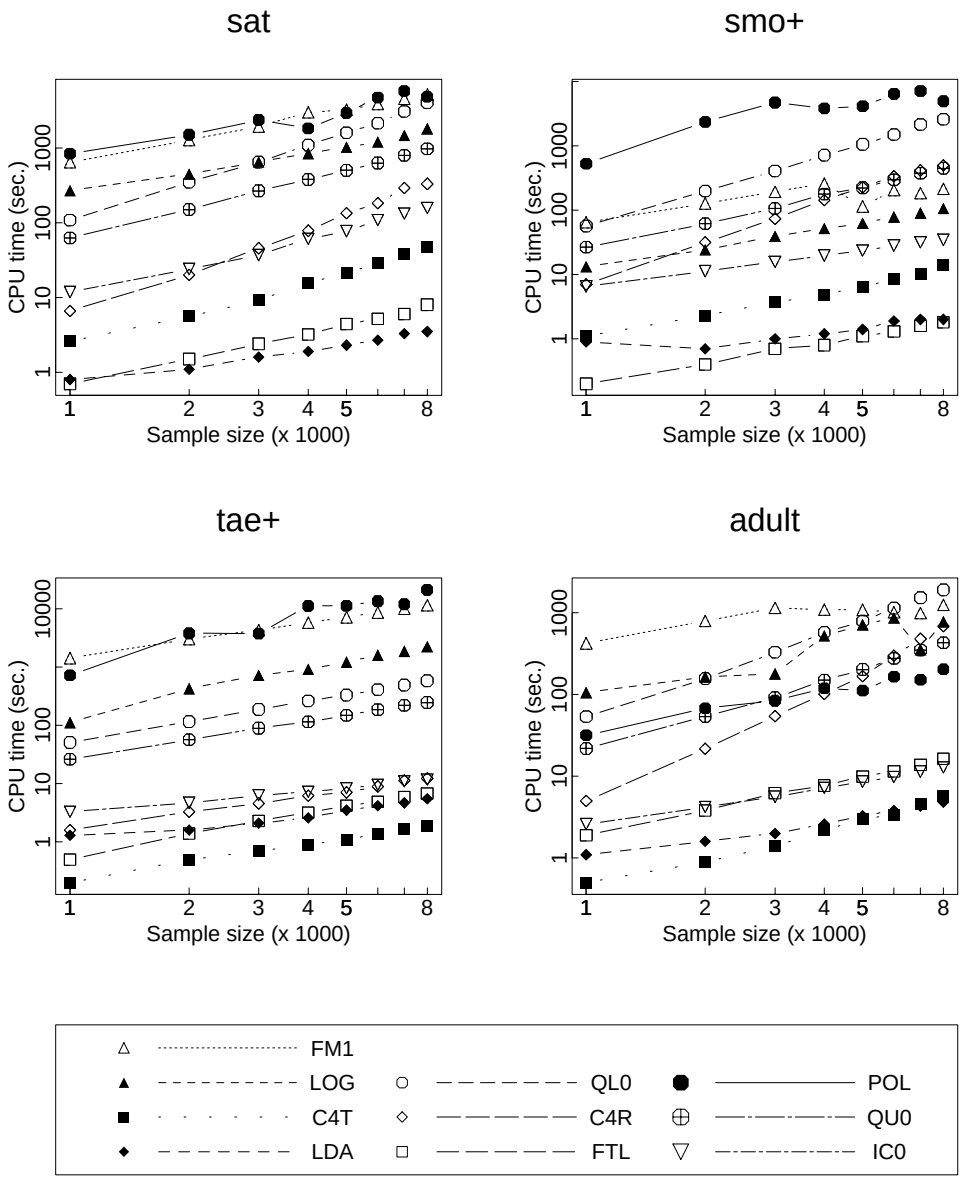


Figure 3. Plots of training time versus sample size in log-log scale for selected algorithms.

model selection. Although this strategy is computationally costly, it does produce a slight reduction in the mean error rate—enough to bring it to the top of the pack.

The good performance of QL0 may be similarly attributable to LDA. The QUEST linear-split algorithm is designed to overcome the difficulties encountered by LDA in multi-modal situations. It does this by applying a modified form of LDA to partitions of the data, where each partition is represented by a leaf of the decision tree. This strategy alone, however, is not enough, as the higher mean error rate of FTL shows. The latter is based on the FACT algorithm which is a precursor to QUEST. One major difference between the QUEST and FACT algorithms is that the former employs the cost-complexity pruning method of CART whereas the latter does not. Our results suggest that some form of bottom-up pruning may be essential for low error rates.

If the purpose of constructing an algorithm is for data interpretation, then perhaps only decision rules or trees with univariate splits will suffice. With the exception of CAL and T1, the differences in mean error rates of the decision rule and tree algorithms are not statistically significant from that of POL. IC0 has the lowest mean error rate and QU0 is best in terms of mean ranks. C4R and C4T are not far behind. Any of these four algorithms should provide good classification accuracy. C4T is the fastest by far, although it tends to yield trees with twice as many leaves as IC0 and QU0. C4R is the next fastest, but Figure 3 shows that it does not scale well. IC0 is slightly faster and its trees have slightly fewer leaves than QU0. Loh and Shih (1997) show, however, that CART-based algorithms such as IC0 are liable to produce spurious splits in certain situations.

Acknowledgments

We are indebted to P. Auer, C. E. Brodley, W. Buntine, T. Hastie, R. C. Holte, C. Kooperberg, S. K. Murthy, J. R. Quinlan, W. Sarle, B. Schulmeister, and W. Taylor for help and advice on the installation of the computer programs. We are also grateful to J. W. Molyneaux for providing the 1987 National Indonesia Contraceptive Prevalence Survey data. Finally, we thank W. Cohen, F. Provost, and the reviewers for many helpful comments and suggestions.

References

1. A. Agresti. *Categorical Data Analysis*. John Wiley & Sons, New York, NY, 1990.
2. J. M. Aronis and F. J. Provost. Increasing the efficiency of data mining algorithms with breadth-first marker propagation. In D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 119–122, Menlo Park, CA, 1997. AAAI Press.
3. P. Auer, R. C. Holte, and W. Maass. Theory and applications of agnostic PAC-learning with small decision trees. In A. Prieditis and S. Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 21–29, San Francisco, CA, 1995. Morgan Kaufmann.
4. R. A. Becker, J. M. Chambers, and A. R. Wilks. *The New S Language*. Wadsworth, 1988.
5. C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, NY, 1995.

6. L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Chapman and Hall, New York, NY, 1984.
7. L. A. Breslow and D. W. Aha. Simplifying decision trees: A survey. *Knowledge Engineering Review*, 12:1–40, 1997.
8. C. E. Brodley and P. E. Utgoff. Multivariate versus univariate decision trees. Technical Report 92-8, Department of Computer Science, University of Massachusetts, Amherst, MA, 1992.
9. C. E. Brodley and P. E. Utgoff. Multivariate decision trees. *Machine Learning*, 19:45–77, 1995.
10. D. E. Brown, V. Corruble, and C. L. Pittard. A comparison of decision tree classifiers with backpropagation neural networks for multimodal classification problems. *Pattern Recognition*, 26:953–961, 1993.
11. S. Bull. Analysis of attitudes toward workplace smoking restrictions. In N. Lange, L. Ryan, L. Billard, D. Brillinger, L. Conquest, and J. Greenhouse, editors, *Case Studies in Biometry*, pages 249–271. John Wiley & Sons, New York, NY, 1994.
12. W. Buntine. Learning classification trees. *Statistics and Computing*, 2:63–73, 1992.
13. W. Buntine and R. Caruana. *Introduction to IND Version 2.1 and Recursive Partitioning*. NASA Ames Research Center, Moffet Field, CA, 1992.
14. L. A. Clark and D. Pregibon. Tree-based models. In J. M. Chambers and T. J. Hastie, editors, *Statistical Models in S*, pages 377–419. Chapman & Hall, New York, NY, 1993.
15. W. W. Cohen. Fast effective rule induction. In A. Prieditis and S. Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123, San Francisco, CA, 1995. Morgan Kaufmann.
16. S. P. Curram and J. Mingers. Neural networks, decision tree induction and discriminant analysis: an empirical comparison. *Journal of the Operational Research Society*, 45:440–450, 1994.
17. J. Friedman. Multivariate adaptive regression splines (with discussion). *Annals of Statistics*, 19:1–141, 1991.
18. M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32:675–701, 1937.
19. D. J. Hand. *Construction and Assessment of Classification Rules*. John Wiley & Sons, Chichester, England, 1997.
20. D. Harrison and D. L. Rubinfeld. Hedonic prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5:81–102, 1978.
21. T. Hastie, A. Buja, and R. Tibshirani. Penalized discriminant analysis. *Annals of Statistics*, 23:73–102, 1995.
22. T. Hastie and R. Tibshirani. Discriminant analysis by Gaussian mixtures. *Journal of the Royal Statistical Society, Series B*, 58:155–176, 1996.
23. T. Hastie, R. Tibshirani, and A. Buja. Flexible discriminant analysis by optimal scoring. *Journal of the American Statistical Association*, 89:1255–1270, 1994.
24. M. Hollander and D. A. Wolfe. *Nonparametric Statistical Methods*. John Wiley & Sons, New York, NY, 2nd edition, 1999.
25. R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–90, 1993.
26. R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, Englewood Cliffs, NJ, 3rd edition, 1992.
27. T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Heidelberg, 1995.
28. C. Kooperberg, S. Bose, and C. J. Stone. Polychotomous regression. *Journal of the American Statistical Association*, 92:117–127, 1997.
29. C. Lerman, J. W. Molyneaux, S. Pangemanan, and Iswarati. The determinants of contraceptive method and service point choice. In *Secondary Analysis of the 1987 National Indonesia Contraceptive Prevalence Survey*, volume 1: Fertility and Family Planning, Honolulu, HI, 1991. East-West Population Institute.
30. W.-Y. Loh and Y.-S. Shih. Split selection methods for classification trees. *Statistica Sinica*, 7:815–840, 1997.

31. W.-Y. Loh and N. Vanichsetakul. Tree-structured classification via generalized discriminant analysis (with discussion). *Journal of the American Statistical Association*, 83:715–728, 1988.
32. O. L. Mangasarian and W. H. Wolberg. Cancer diagnosis via linear programming. *Siam News*, 23:1–18, 1990.
33. C. J. Merz and P. M. Murphy. *UCI Repository of Machine Learning Databases*. Department of Information and Computer Science, University of California, Irvine, CA, 1996. (`http://www.ics.uci.edu/~mlearn/MLRepository.html`).
34. D. Michie, D. J. Spiegelhalter, and C. C. Taylor, editors. *Machine Learning, Neural and Statistical Classification*, London, 1994. Ellis Horwood.
35. Ruppert G. Miller, Jr. *Simultaneous Statistical Inference*. Springer-Verlag, New York, 2nd edition, 1981.
36. W. Müller and F. Wysotzki. Automatic construction of decision trees for classification. *Annals of Operations Research*, 52:231–247, 1994.
37. W. Müller and F. Wysotzki. The decision-tree algorithm CAL5 based on a statistical approach to its splitting algorithm. In G. Nakhaeizadeh and C. C. Taylor, editors, *Machine Learning and Statistics: The Interface*, pages 45–65. John Wiley & Sons, New York, NY, 1997.
38. S. K. Murthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–33, 1994.
39. J. Neter, W. Wasserman, and M. H. Kutner. *Applied Linear Statistical Models*. Irwin, Boston, MA, 3rd edition, 1990.
40. T. Oates and D. Jensen. The effects of training set size on decision tree complexity. In D. H. Fisher, Jr., editor, *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 254–262, San Francisco, CA, 1997. Morgan Kaufmann.
41. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.
42. J. R. Quinlan. Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.
43. B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, 1996.
44. W. S. Sarle. Neural networks and statistical models. In *Proceedings of the Nineteenth Annual SAS Users Groups International Conference*, pages 1538–1550, Cary, NC, 1994. SAS Institute, Inc. (`ftp://ftp.sas.com/pub/neural/neural1.ps`).
45. SAS Institute, Inc. *SAS/STAT User's Guide, Version 6*, volume 1 & 2. SAS Institute, Inc., Cary, NC, 1990.
46. J. W. Shavlik, R. J. Mooney, and G. G. Towell. Symbolic and neural learning algorithms: an empirical comparison. *Machine Learning*, 6:111–144, 1991.
47. W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S-Plus*. Springer, New York, NY, 2nd edition, 1997.
48. W. H. Wolberg, M. A. Tanner, and W.-Y. Loh. Diagnostic schemes for fine needle aspirates of breast masses. *Analytical and Quantitative Cytology and Histology*, 10:225–228, 1988.
49. W. H. Wolberg, M. A. Tanner, and W.-Y. Loh. Fine needle aspiration for breast mass diagnosis. *Archives of Surgery*, 124:814–818, 1989.
50. W. H. Wolberg, M. A. Tanner, W.-Y. Loh, and N. Vanichsetakul. Statistical approach to fine needle aspiration diagnosis of breast masses. *Acta Cytologica*, 31:737–741, 1987.