StatLog: Comparison of Classification Algorithms on Large Real-World Problems

R.D. King[*], C. Feng[†] and A. Sutherland[‡]

Abstract

This paper describes work in the StatLog project comparing classification algorithms on large real-world problems. The algorithms compared were from: symbolic learning (CART, C4.5, NewID, AC^2, ITrule, Cal5, CN2), statistics (Naive Bayes, k-nearest neighbor, kernel density, linear discriminant, quadratic discriminant, logistic regression, projection pursuit, Bayesian networks), and neural networks (back-propagation, radial basis functions). Twelve datasets were used: five from image analysis, three from medicine, and two each from engineering and finance. We found that which algorithm performed best depended critically on the dataset investigated. We therefore developed a set of dataset descriptors to help decide which algorithms are suited to particular datasets. For example, datasets with extreme distributions ($skew > 1$ and $kurtosis > 7$) and with many binary/categorical attributes ($> 38\%$) tend to favor symbolic learning algorithms. We suggest how classification algorithms can be extended.

[*]Department of Statistics, Strathclyde University, Glasgow G1, UK. Present address: Biomolecular Modelling Laboratory, PO.Box 123, Lincoln's Inn Fields, London WC2A 3PX (email: rd_king@icrf.ac.uk). Author whom correspondence should be sent to.

[†]The Turing Institute Ltd., Glasgow, UK. Present address: Computer Science Dept, 150 Louis Pasteur/Priv, Ottawa University, Ottawa, Ont., Canada K1N 6N5 (email: cfeng@csi.uottawa.ca).

[‡]Department of Statistics, Strathclyde University, Glasgow G1, UK. Present address: Hitachi Dublin Laboratory, Dublin 2, Ireland

[§]Running title; Comparison of Classification Algorithms

1 Introduction

Classification algorithms have been developed in symbolic learning, statistics, and neural networks, usually using different data modelling and representation techniques. Classification is here defined to be the problem of correctly predicting the probability that an example has a predefined class from a set of attributes describing the example. There exist many different algorithms, but their relative merits and practical usefulness are unclear. Thus, the need arises to evaluate their relative performances, in particular on large-scale industrial problems. For this purpose, the European Commission initiated the StatLog project. The project's full title is *The Comparative Testing of Statistical and Logical Learning Algorithms on Large-Scale Applications to Classification, Prediction and Control*. Its main focus was on classification algorithms, and this paper will examine only them.

To ensure objectivity and completeness, StatLog was a collaboration, or perhaps a *competition*, between researchers, using established state-of-the-art algorithms and industrial data. StatLog had ten academic and industrial partners; each of them has interest to see that their algorithm performs to its maximum potential. (Contact information is in Appendix C.)

1.1 StatLog structure

For the StatLog project a representative collection of algorithms were compared. Six algorithms were collected from symbolic learning, eight from statistics, and two from neural networks. (No Genetic or Inductive Logic Programmming (ILP) algorithms were included, because initial trials showed that these methods were at too early a stage in their research to make them easily applicable to the datasets used in StatLog.)

We tested the algorithms on twelve large real-world, and therefore possibly noisy, classification problems: five from image analysis (satellite image interpretation, two for character recognition, vehicle recognition, and image recognition), three from medicine (heart disease, head injury, and diabetes), and two each from engineering (space shuttle, belgian power) and finance (credit assignment in the U.K. and Germany).

The basic methodology can be viewed as a table: with the algorithms along one axis and the datasets along the other, and performance measures at each position in the matrix. The objective measures of performance we use are time (for the training and test data), and accuracy (or cost if there is a cost function/matrix available). Subjective measures are much more difficult to use and we have relied on the response from the users, they include: understandability of the decision rule, ease of use of the algorithm (particularly as perceived by the "naive user"), and robustness to required parameter input.

As an aid in our interpretation of results we developed a set of dataset descriptors so that we can analyze the results to discern what features of datasets suit particular algorithms. Finally, we developed a methodology for testing the algorithms that is designed to be as consistent as possible.

Earlier results have been presented elsewhere [Sutherland *et al.*, 1992, Feng *et al.*, 1993, King *et al.*, 1992], and the complete StatLog project (including non classification studies) will be described in a forthcoming book [Michie *et al.*, 1994].

2 Previous Comparison Studies

A number of criticisms can be leveled at previous comparative studies; most concern algorithms and datasets.

Many comparisons have been intra-subject comparisons, e.g., within symbolic learning [Clark and Boswell, 1991, Sammut, 1988, Quinlan *et al.*, 1986, Aha, 1992], within statistics [Cherkaoui and Cleroux, 1991, Remme *et al.*, 1980], and within neural networks [Huang *et al.*, 1991, Fahlman, 1991, Xu *et al.*, 1991, Ersoy and Hong, 1991]. There have been fewer, but still many, inter-subject studies involving algorithms from two or more of these fields, e.g., [Weiss and Kapouleas, 1989, Weiss and Kulikowski, 1991], [Ripley, 1992], [Mooney *et al.*, 1989, Shavlik *et al.*, 1991], [Fisher and McKusick, 1989], [Thrun *et al.*, 1991], [Kirkwood *et al.*, 1989], [Tsaptsinos *et al.*, 1990], [Spikovska and Reid, 1990], [Atlas *et al.*, 1991], and [Gorman and Sejnowski, 1988].

The comparisons by [Weiss and Kapouleas, 1989, Weiss and Kulikowski, 1991] found that symbolic algorithms (PVM [Weiss *et al.*, 1990] and CART [Breiman *et al.*, 1984]) were more accurate (on most datasets) than back-propagation or a number of statistical algorithms (e.g., linear and logistic discriminants). In similar trials involving ID3 [Quinlan, 1986], a perceptron, and back-propagation [Mooney *et al.*, 1989, Shavlik *et al.*, 1991] and [Fisher and McKusick, 1989] reported that back-propagation was at least as accurate as ID3 and the perceptron algorithm. Ripley [1992] compared a diverse set of algorithms on two related problems of Tetse fly distribution, and found that the nearest-neighbor and conjugate gradient (neural net) algorithms had the highest accuracy.

The comparisons by [Kirkwood *et al.*, 1989] found that ID3 performed better than discriminant analysis for classifying the gait cycle of artificial limps. Tsaptsinos *et al.* [1990] reported that ID3 was preferable on an engineering control problem to two neural network algorithms. Spikovska & Reid [1990] showed that a higher order neural network (HONN) performed better than ID3. Atlas *et al.* [1991] reported back-propagation did better than CART. Gorman & Sejnowski [1988] showed that back-propagation outperformed nearest neighbor for classifying sonar targets. Many comparisons reported various neural networks performed similarly to, or slightly better than, symbolic and statistical algorithms [Huang and Lippmann, 1987, Bonelli and Parodi, 1991, Sethi and Otten, 1990].

A cynic would say: the only *general* conclusion that can be made from these studies, is that the algorithms in which the authors have most interest tend to do best.

It is gradually becoming recognized [Buntine, 1989] that there need be analysis to show what features of an algorithm make it successful on particular datasets, or what features of a dataset favor a particular algorithm. This requires a more complete and standardized study.

2.1 Criticisms on algorithm use

Studies rarely consider more than a handful of different algorithms, and these are often restricted to one or two classes of algorithms. Ripley [1992] examined ten varied algorithms and their variations (linear discrimination, generalized discrimination, projection pursuit, k-nearest-neighbor, S-tree, back-propagation, quickprop, conjugate gradients, cascade correlation, and learning vector quantization), but included only one symbolic learning algorithm. Weiss & Kulikowski [1991] examined nine different algorithms (linear discrimination, quadratic discrimination, nearest neighbor, naive Bayes, Bayes 2nd order, back-propagation, PVM, Assistant, and CART), but did not include any of the newer statistical methods. Fisher and McKusick [1989] examined only ID3 and back-propagation. Mooney *et al.* [1989] and Shavlik *et al.* [1991] examined only a perceptron, ID3, and back-propagation. Thrun *et al.* [1991] studied more than ten algorithms, but none of them are from statistics. A full comparative trial should include standard and up-to-date algorithms from symbolic learning, statistics, and neural networks.

Sometimes *outdated* algorithms are used. Three comparison studies were presented at the 1989 Int. Joint Conf. on AI [Mooney *et al.*, 1989, Fisher and McKusick, 1989, Weiss and Kapouleas, 1989]. Despite care taken by the authors there was still concern that certain algorithms were not up-to-date, e.g., an old version of ID3 was used.

Most comparisons have overlooked the problem that some learning methods are *not complete* and require parameters to be tuned for a particular problem. Many algorithms need an experienced "expert" to use them properly, i.e., to set the parameters correctly.

Ideally all algorithms would be automatic, the user would just feed in the data and run the algorithm. However, at present this is not the case. For example, [Sammut, 1988] reported that this "tweaking" was considerably important to achieve reasonable performance level with some algorithms. Ripley [1992] described a "degree of frustration" in getting some algorithms to produce the eventual results (especially neural networks). The problem of parameter tuning is probably most acute for neural network algorithms and least so for symbolic ones. (Genetic algorithms were found to need so much work for tuning to particular problems that they were excluded from the main body of StatLog work.) When comparing learning methods, the studies should be carried out by *experts* in the different methods. This is important because novices are less experienced in techniques required for careful parameter adjustment.

One further problem in drawing conclusions from these studies is that there often exist various *versions*, or re-implementations, of algorithms, and it is not always made explicit what version and what implementation is used. The parameter settings are also not commonly given. This makes repeating the work problematic.

2.2 Criticisms on datasets and methods

Most comparative studies have considered *too few* datasets to draw very general conclusions. Weiss & Kulikowski [1991] examined four dataset, three medical and Fisher's iris data. Ripley [1992] examined only two variations of the same problem. Mooney *et al.* [1989] and Shavlik *et al.* [1991] looked at five datasets. Fisher *et al.* [1989] looked at four datasets (including two artificial datasets). Thrun *et al.* [1991] used only one with three variations of an artificial dataset. The datasets used are also often small (e.g., Fisher's iris data is a small and popular dataset), or the datasets are in some sense "toy" problems.

Just as there are complications caused by different variations of algorithms, there are complications caused by variations of datasets. For example, there are several "thyroid" datasets. At the 1989 Int. Workshop on Machine Learning [Segre(ed.), 1989], there were several papers giving the accuracy for ID3 on the thyroid data: each was different. The authors explained this by variations in datasets and algorithms. It would be helpful if a fuller study could establish some common standards in datasets.

Datasets are also commonly *artificially created*: [Cherkaoui and Cleroux, 1991] investigated the performance of six statistical discriminants applied to data with a mixture of binary, nominal, ordinal, and continuous attributes. [Thrun *et al.*, 1991] studied the MONK's problem involving simulated robots, classified into two classes using six attributes. Using artificial data has the advantage that conditions can be altered at will, but the disadvantage that in defining the class and type of noise one almost defines the best algorithm for finding the class. For example, if the class is defined by some logical rule, then some symbolic algorithm would be expected to be successful, but linear discrimination may have problems. To judge on the empirical applicability of algorithms on *large real-world* problems, it is best to use *large real-world* data.

There has also been various different measures of success and testing methodologies used. The most common measure of success has been prediction *accuracy* or error rate. Statistician often use *costs* of miss-classification, but this has been largely ignored by symbolic learning and neural net researchers. Prediction accuracy is normally measured on a separate test set. Traditionally this is 30% of the total data — although there appears to be no strong reason for using this split. Cross-validation should be used, or if the dataset is small leave-one-out should be used. If the dataset is very small the bootstrap method can be used. [Weiss and Kulikowski, 1991] describes these various techniques. Workers within symbolic learning have been particularly fond of the the resampling technique of randomly splitting a dataset into a training and test set n times; they would probably be better *advised* to use cross-validation.

Learning speed is also studied. Here the literature is clearer, [Mooney *et al.*, 1989, Shavlik *et al.*, 1991], [1], [Weiss and Kulikowski, 1991], and [Ripley, 1992] all found that back-propagation was slower than decision tree algorithms (this can also be expected from theory). If the speed of classification is important, a nearest-neighbor algorithm would be slow. One other important measure of success of an algorithm is the *understandability* of the classification function/rule. It is an often touted advantage of symbolic learning that it produces human understandable rules. The trouble in testing this is that understandability is very subjective. There is however a lot of anecdotal evidence that the results of symbolic learning are more human friendly than the black box results of statistical and neural network algorithms, e.g., [Ripley, 1992] (a statistician) states the tree results for his Tetse fly data, "nearest neighbor and LVQ ... provide no explanation", "Tree-based methods rapidly provided a very interpretable and rather good fit".

3 Algorithms and Datasets

The algorithms that we have chosen are representative of the *established* state-of-the-art in the respective fields. They can be divided into four categories:

- Symbolic classifiers, including C4.5 [Quinlan, 1987a], NewID, AC^2, CN2 [Clark and Boswell, 1991], ITrule [Goodman and Smyth, 1989], CART [Breiman *et al.*, 1984], and INDCART;
- Statistical classifiers, including k-nearest-neighbor, "naive" Bayes, linear discriminant (Discrim), quadratic discriminant (Quadra), and logistic regression (LogReg) [Hand, 1981];
- Modern statistics, including projection pursuit (SMART) [Friedman and Stuetzle, 1981], kernel density estimate (Alloc80) [Hermans *et al.*, 1974], and a polytree learner (CASTLE) [Acid *et al.*, 1991];

- Neural networks, including back-propagation (Backprop) [Rumelhart *et al.*, 1986] and radial basis functions (RBF) [Poggio and Girosi, 1990].

A brief description of these algorithms is presented in Appendix A.

The datasets were selected by the criteria of being large (example/class ratio) and of real-world interest. No other criteria were used, and the twelve datasets used were the first twelve we could find. The datasets are described briefly in Appendix B, and are: satellite image interpretation (Satellite), character recognition (Digits, KL), vehicle recognition (Vehicle), image recognition (Segment), heart disease (Heart), head injury (Head), diabetes (Diabetes), space shuttle (Shuttle), belgian power (Belgian), and U.K. credit assignment (Credit) and German credit assignment (German). Most datsets can be obtained from the University of Strathclyde (some are not public); they have also been placed in the U.C.Irvine ML database.

3.1 Statistical characteristics of datasets

An important objective in StatLog is to investigate why certain algorithms do better on particular datasets. We have used several statistical measures on the datasets that will help to explain our findings. The statistical figures are in Table 1.

1) Homogeneity of covariances. This is the geometric mean ratio of standard deviations of the populations of individual classes to the standard deviations of the sample, and is tabulated as SD_ratio. The SD_ratio is strictly greater than unity if the covariances differ, and is equal to unity if and only if all individual covariances are equal to the covariances of the whole sample. SD_ratio is calculate by: $\gamma \sum_{i=1}^{k} (n_i - 1) \log |S_{ui}^{-1} S_u|$, where for p attributes, k classes, n_i examples of the ith class,

$$\gamma = 1 - \frac{2p^2 + 3p - 1}{6(p+1)(k-1)} \Big\{ \sum \frac{1}{n_i - 1} - \frac{1}{n-k} \Big\},$$

and S_{ui} is the variance of the ith class and S_u is that of the whole dataset. S_{ui} and S_u are the unbiased estimators $S_u = (n/(n-k))S$, and $S_{ui} = (n_i/(n_i - 1))S_i$. This statistic has an asymptotic $\chi^2_{p(p+1)(k-1)/2}$ distribution. The approximation is good if each n_i exceeds 20, and if k and p are both much smaller than every n_i.

2) Mean absolute correlation coefficient. The correlations ρ_{ij} between all pairs of attributes indicate the dependence between the attributes. They are calculated for each class separately. The absolute values of these correlations are averaged over all pairs of attributes and over all populations. The mean measure ρ gives a measure of interdependence between attributes. If the correlation is near unity, there is much redundant information in the attributes; some procedures such as logistic discriminants may have technical problems associated with this. Also, CASTLE may be misled substantially by fitting relationships to the attributes instead of concentrating on relationship between the classes and the attributes.

3) Canonical discriminant correlations. Canonical discriminants systematically project the n-attribute space to n-1, maximizing the ratio of between-mean distances to within-cluster (population centers of examples) distances; successive discriminants are orthogonal to earlier discriminants. As a result, the first canonical discriminant gives the best single linear combination of attributes that discriminates between the examples. The second canonical discriminant is the best single linear combination orthogonal to the first, and so on. The success of these discriminants can be measured by the canonical correlations ($cancor_{1...4}$). For example, if the first canonical correlation is close to unity, the k means lie nearly along a straight line; if the q+1th canonical correlation is near zero, the means lie in q-dimensional space.

4) Variation explained by first four canonical discriminants. The sum of the first q eigenvalues of the canonical discriminant matrix divided by the sum of all the eigenvalues represents the "proportion of total variation" explained by the first q canonical discriminants. We tabulate, as $fract_q$, the values of $(\lambda_1+...+\lambda_q)/(\lambda_1+\lambda_2+\ldots+\lambda_p)$ for $q = 1, ..., 4$.

This measures the collinearity of the class means. When the classes form an ordered sequence, e.g., soil types might be ordered by wetness (e.g., satellite image), the class means typically lie along a curve in low dimensional space. The λ's are the squares of the canonical correlations. The significance of the λ's can be judged from the χ^2 statistics produced by the MANOVA method.

5) Univariate skew and kurtosis. Skew measures the non-normality of the attributes when considered separately. Each $\beta_1(i, j)$ can usually be calculated for the ith attribute in the jth class. As a single measure of skew for the whole dataset, we quote the mean square value of $\beta_1(i, j)$, averaged over all attributes and over all classes. This gives the measure *skew*. For a normal population *skew* should be zero: for uniform and exponential variables, the theoretical values of *skew* are zero and 16 respectively. Similarly, we find the average of the univariate kurtosis $\beta_2(i, j)$, averaged over all attributes and populations. This gives the measure *kurtosis*. For a normal population, *kurtosis* $= 3$ exactly, and the corresponding figures for a uniform and an exponential are 1.8 and 9 respectively.

3.2 Pre-processing

Algorithms differ in their ability to cope with different types of data. Some algorithms cannot process missing values whereas others can. Some cannot process real-valued, or categorical or binary attributes. For the purpose of testing the algorithms we standardized all the datasets; the methods we used are mostly the simplest available.

Missing values. They were replaced by the mode in the case of categorical attributes, or the median in the case of continuous variables for that attribute. In the training set, the mode or median was calculated over the training set and in the test set it was calculated over the whole population. This made comparisons between different algorithms easier since any differences in results would be due to the algorithm and not to the method of substituting the missing values. (It is important to compare on data with exactly the *same* characteristics.)

Restricted attribute values. Many algorithms cannot process certain types of data. For example, linear and quadratic discriminants cannot process categorical values. To overcome this problem categorical values are replaced with n new binary attributes, where n is the number of categories. The ith binary attribute is set equal to one to represent the ith category, the others being set equal to zero. CASTLE cannot process real attributes so continuous values must be discretized before it can produce a polytree for classification.

4 Evaluation Methodology

To ensure the fairness of comparison, a number of measures were taken to reduce conscious or unconscious bias towards one particular algorithm or another. Care was taken before the datasets were sent, 1) to eliminate possible pre-knowledge the users of the datasets, and 2) to avoid pre-processing the data such that it was biased to particular algorithms. To address the problem of expert vs. naive user, we devised a two stage schedule for selected trials. The first stage involved an expert user testing the algorithm on the dataset sent to him, and the second stage a novice user validating that result. The complete procedure is:

- All the datasets were collected at one center for pre-processing. Each dataset was sent to all testing sites with the same pre-processing done, so they have the same data characteristics.
- When possible, part of the data was kept back from the testing sites in case the results were disputed.
- Default parameter settings of algorithms were used where available.

- All algorithms were tested by experienced "experts". For each trial they filled in a report about the method of application of the algorithm and the "tuned" system parameters.
- The sample trials were validated by a third party relatively new to the algorithms. The results were accepted when they were close, otherwise another evaluation had to be attempted.
- Finally, the results were transmitted to a center where comparison and analysis were carried out. The individual testing sites were not aware of the results of other sites.

We hope that these measures are reasonably adequate to address the points raised in Section 2. (When there is unusual pre-processing by the testing site, we explicitly state what had been done along with the results.)

4.1 Evaluation criteria

Algorithms are evaluated using a number of criteria. Three are objectively measurable criteria. They are accuracy, misclassification cost, and the time taken to produce results. There are also two subjective criteria: the comprehensibility of the results and the ease-of-use of the algorithm to relatively naive users.

4.1.1 Accuracy and confusion matrices

Accuracy is the percentage of examples that the algorithm classifies correctly. We did not study the *learning curve* of algorithms. Rather than a single measure of accuracy, in our tests each algorithm produced a *confusion matrix*. A confusion matrix has n dimensions, where n is the number of classes. The rows of a confusion matrix represent the *true* classifications of the test examples and the columns represent the classifications predicted by the algorithm. For example, one element c_{ij} of the confusion matrix is the number of the members of class i that were assigned to class j by the algorithm. The confusion matrices were used in calculating costs.

4.1.2 Cost and cost matrix

In many practical situations there are different costs associated with different classification errors (this is different from the cost associated with testing attributes). A confusion matrix distinguishes different types of errors. Thus, we may give different costs (penalties) to different errors, e.g., it is more *costly* to misclassify a case of moderate injury as deadly than to misclassify a deadly injury as moderate; a larger penalty would be given to the error of misclassifying moderate to deadly.

From the penalty difference, a *cost matrix* can be constructed. An element p_{ij} represents the penalty for misclassifying an example in class i to j. Jointly with the confusion matrix, $p_{ij}c_{ij}$ represents the total given to one type of error; the larger the penalty the more numerous and undesirable the errors are. The total cost of the algorithm is judged by the sum: $\sum c_{ij} \times p_{ij}$. The (average) cost is the total cost divided by the total number of examples in the confusion matrix; this quantity should be minimized.

4.1.3 Run-time speed

Training and testing time were calculated. Training time is the time taken to learn plus the time to classify the training data: test time is the time to classify the test data. All partners used SUN SPARCstations and times were corrected to the standard benchmark machine (SPARCstation IPC). Note that if we ignore the effect of how algorithms are encoded, the run-time (training) speed should be proportional to the product of total example presentations and the processing time per presentation. This is to a certain degree consistent with [Fisher and McKusick, 1989].

4.1.4 Comprehensibility and ease-of-use

Some algorithms can produce more comprehensible results, e.g., rules are often considered more understandable than a trained neural network. To be less subjective, we arrived at the conclusions on comprehensibility through consulting each partner. We also asked the independent partners to provide an estimate of the ease-of-use of each algorithm. This is to be based on their experience with using the algorithm, the amount of tuning for the algorithm, and the length of time that was taken to get the algorithm to work correctly.

4.2 Testing strategies

Depending on the size of the datasets, different training strategies were adopted (Table 1).

In *Train/Test*, the data is randomly split into training and test sets. An estimate of the accuracy is made by applying the algorithm to the test data. The number of examples in the test set determines the reliability of accuracy estimates. When the test set is large enough, we can normally have very accurate estimates. Roughly, for typical accuracy rates ($\geq 80\%$) 1000 examples for testing is extremely accurate compared to the true accuracy [Weiss and Kulikowski, 1991]. At 5000 examples, the test sample estimate is virtually identical to the true value. The number of classes is also important, the more classes there are the more test examples are needed to get accurate estimates. We chose a single train/test split for large datasets.

For smaller datasets we used *cross-validation* [Stone, 1974, Weiss and Kulikowski, 1991]. It is more computationally expensive than a single train/test split but more accurate and less biased estimates of the accuracy. In cross-validation the data is divided into n sub-samples, each sub-sample is classified via the classification rule constructed from the remaining (n-1) sub-samples, and the estimated accuracy is the average of the n sub-samples.

5 Classification Results on Datasets

Tables are used to show the various measures of the algorithms on datasets, including accuracy, cost and time. (We also have the reports of the original tests and parameter settings). The leading performance indicator is the number, either accuracy or cost (with its standard deviation), for the test data. These results are frequently related to the data characteristics in Section 3.1. The standard deviation for accuracy is estimated by: $\sqrt{A(1-A)/N}$, where A is the measured accuracy and N is the number of test examples. The standard deviation for cost is estimated by: $\sqrt{abs(\sum_{i,j} c_{ij}p_{ij}^2 - (\sum_{i,j} c_{ij}p_{ij})^2)/N}$, where c is the confusion matrix produced by the algorithm and p is the cost matrix of the data.

The default accuracy is the accuracy of predicting all examples to be in the most common class. The default cost is the minimum cost of predicting all examples to be in one class.

Items marked by "*" indicates that the entries for time include both training and testing times. (INDCART is marked by "?" because it does not correctly implement the cost processing mechanism of CART.)

5.1 Satellite image data

The k-N-N algorithm does very well on this data (in terms of accuracy) (Table 2). The related algorithms: radial basis functions and Alloc80 also do well. The success of these algorithms may be because the attributes are roughly equally scaled and equally relevant. There appears to be little to choose between any of the other algorithm, except that Naive Bayes does badly. This dataset has the highest correlation between attributes ($\rho = 0.5977$), which partly explains the failure of Naive Bayes. Note that only three linear discriminants are sufficient to separate all six class means ($fract_3 = 0.9691$). This points to the seriation of the classes "grey soil", "damp grey soil", and "very damp grey soil". So it is hard to improve accuracy further. Equally, this result can be interpreted as indicating that the original four attributes may be successfully reduced to three with no (or little) loss of information.

5.2 Hand written digits data

The results for the digits dataset (Table 3), and the KL digits dataset (Table 4) are very similar so are treated together. Most algorithms perform a few percent better on the KL digits data. The KL digits dataset is very close to normal ($skew = 0.1802, kurtosis = 2.9200$, which did not effect k-N-N) as against the exact normal values of $skew = 0$ and $kurtosis = 3$. In both digits datasets, k-N-N, and the related algorithms radial basis functions and Alloc80, do fairly well. As in the satellite dataset, the attributes are relatively equally scaled and equally important. The KL version appears to be well suited to Quadra: there is a substantial difference in variances ($SD\_ratio = 1.9657$), while at the same time the distribution is not much from normality with skew nearly 0 and kurtosis nearly 3. The performance of back-propagation confirms the results in the literature [McClelland *et al.*, 1986].

5.3 Vehicle silhouettes data

The attributes for the vehicle silhouettes dataset are derived from the low level bit information, and are not equally scaled. This may have caused the lower performances of k-N-N and radial basis functions (Table 5). However, Alloc80 appears to be more robust than the other two algorithms. (The original [Siebert, 1987] paper showed that an ID3-based algorithm did better than k-N-N.) Quadratic discriminant do well (moderate skew, 0.8282, kurtosis, 5.1800, and higher SD_ratio, 1.5392). The high value of $fract_2 = 0.8189$ indicates that linear discrimination could be based on just two discriminants. This relates to the fact that the two cars are not easily distinguishable, so can be treated as one (reducing dimensionality of the mean vectors to 3D). However, although the fraction of discriminating power for the third discriminant is low $(1 - 0.8189)$, it is still statistically significant, so cannot be discarded without a small loss of discrimination. Back-propagation also does well on this dataset.

5.4 Segment data

The attributes for the segment dataset, like the Vehicle dataset, are processed and so are less equally scaled (Table 6). Judging by skew (2.9580) and kurtosis (24.4813), the attributes are much further from normal than the Vehicle dataset. This dataset has the highest SD_ratio (4.0014). This may explain the robustness of Alloc80 compared to k-N-N. The high skew and kurtosis appears to suit symbolic learning well, with six of the top seven being such algorithms. In contrast, it severely disrupts the performance of the the three discriminants. The segment dataset has some attributes that are linear combinations ($\rho = 0.1425$) of others, which decreases the performance the discriminants, Naive Bayes, and CASTLE.

5.5 Credit risk

Eight attributes of this datasets are categorical (Table 7); many are also irrelevant: it is possible to get 87% accuracy using just one attribute (many algorithms do worse than this). Most symbolic algorithms agree that this attribute produces the best rule (NewID, AC^2, CART, and CN2), and this is also the conclusion of CASTLE. Due credit should be given to these procedures for finding a rule that is not only simple but efficient. On the other hand, SMART and Backprop achieve about the same accuracy but the simple structure of the problem is completely masked. The three discriminants also fail to find the simple rule. k-N-N is disrupted by the presence of symbolic attributes (which are difficult to scale) and the irrelevant attributes.

5.6 Belgian data

On this dataset SMART and logistic regression produced high accuracy (Table 8). Symbolic algorithms are also quite successful; this confirms the work of [Cutsem *et al.*, 1991]. It is interesting to note that seven of the original attributes are linear functions of the others. These are considered important in the work of [Cutsem *et al.*, 1991], but they were removed in the application of logistic regression (marked by "#"). The algorithm still produced a low error rate. The presence of attributes that are linear functions ($\rho = 0.3503$) of the others may be responsible for the poor performance of k-nearest neighbor, CASTLE, and Naive Bayes.

5.7 Shuttle control data

This dataset (Table 9) is the largest (58000 examples) and is far from normal ($skew = 4.4371$, $kurtosis = 160.3108$). There appears to be very little "noise": it seems possible to get arbitrarily close to 100% accuracy. The attributes are numerical and exhibit multimodality (analyzed using S-plus, but we do not have a good statistical test to measure this). All the algorithms achieved very high accuracy. Symbolic algorithms generally produce accuracy close to 100% on the test set. SMART, k-N-N, and Alloc80 do also very well.

We analyzed the decision trees produced for the Shuttle data. We found that only a few attributes are needed to classify the data. For example, a tree with five leaves using two attributes would perfectly classify two classes "High" and "Rad_flow". This seems to indicate that attributes selected in this data are very well suited to the symbolic algorithms.

The data seem to consist of isolated islands or clusters of points, each of which is pure (belongs to only one class), with one class comprising several such islands. However, neighboring islands may be very close and yet come from different classes. Their boundaries seem to be parallel with the coordinate axes. This suggest that it may be possible to classify the combined dataset with 100% accuracy using a decision tree. It is then of interest to ask which of our algorithms are *guaranteed* that given an arbitrarily large learning dataset. We believe that k-N-N , Bayes, CASTLE, Backprop, and Alloc80 should produce 100% accuracy. (Our implementation of radial basis functions had a singularity problem, but we think it should also.) Decision trees should also find the perfect rule with some tuning of the pruning parameter, but may not do so under all circumstances as it is occasionally necessary to override the splitting criterion [Gordon and Olshen, 1978]. The three discriminants would not improve much.

5.8 Diabetes data

This dataset has no categorical attributes (Table 10). The three discriminants perform well, and so does SMART. (Note only the first discriminant function is meaningful, $cancor_1 = 0.5507$, so it is hard to separate the reamining 1-0.5507.) k-N-N and Alloc80 do very badly, close to default accuracy.

5.9 Heart disease and head injury

Both Head (Table 12) and Heart (Table 11) have cost matrices associated with them. In the results for both datasets the top 10 algorithms (algorithms with the lowest costs) are all capable of utilizing costs in the testing phase. (Note that SMART is the only algorithm that as standard can utilize costs directly in the training phase, though an experimental back-propagation algorithm exists, which could utilize costs.) Many algorithms have a higher cost than the default cost, this is because they are trying to maximise accuracy, not minimise cost. In the Head data, the discriminants do best. LogReg do better than Discrim and Quadra; this can be explained by the presence of binary/categorical attributes. The same ordering is repeated in the Heart data. Naive Bayes performed best on the Heart dataset, CASTLE also does well. Both have moderate skew (1.0071, 0.9560 respectively) and kurtosis (5.0408, 3.6494); this may reflect the careful selection of attributes by doctors. Symbolic algorithms do badly on both datasets (both have cost matrices). CART is the only symbolic algorithm that incorporates costs (at the classification stage), and it performs best among the symbolic algorithms. In Head, only the first discriminant ($fract_1$ is near unity) contributes to discriminate between the classes. Therefore, this dataset is very close to linearity. In turn, this suggests that the class values reflect some underlying continuum of severity (Tests using MANOVA shows that class 2 is slightly nearer to class 3). Because of the low SD_ratio (1.1231), Quadra is no better than Discrim. In Heart, $cancor_1$ (0.7384) is not very high, so the discriminants are only moderate (outperformed by naive Bayes).

5.10 German credit data

This data (Table 13) has similar characteristics to those of the other credit dataset, but differs in having a cost matrix (so lower cost is desired), and higher skew (1.6986) and kurtosis (7.7943). The statistical algorithms do well. CART is the best symbolic algorithm. The high skew and kurtosis ought to favor symbolic algorithms, but this effect is being masked by the presence of a cost matrix.

5.11 Subjective evaluation

The algorithms were also evaluated by the users who validated the algorithms and who were relatively new to the algorithms. The results were marked according to five grades: from the lowest $G1$ to the highest $G5$. Because of their subjectivity, these marks should be only taken as general guidelines.

Ease-of-use assessment

The ease-of-use of an algorithm is graded by: 1) how easily it can be set up and be run, and 2) how difficultly it is to tune to process different datasets.

C4.5, NewID, CN2, CART, INDCART, Cal5, k-N-N, and naive Bayes are graded in $G5$, meaning that they are most easy to use (e.g., statisticians had no problems with symbolic algorithms).

Discrim, Quadra, LogReg, and ITrule are graded next with a mark of $G4$. They were also relatively easy to use, but required more familiarity. For example, symbolic learning experts found that manually specifying the attributes and class combinations was necessary when running linear discriminant.

AC^2 is given $G3$, because its user interface hindered efficient input and output of data (it is designed for a different purpose from that in StatLog). For example, AC^2 took a typical time of about one hour to input data alone. CASTLE is also in $G3$, because of the number of parameters that needed to be adjusted.

Radial basis functions, SMART and Alloc80 are categorized in grade $G2$ because they normally needed several trials to decide various parameters such as the number of hidden nodes. For example, SMART has two user controlled parameters; it is not obvious what values should be chosen for them from the result of one trial. But they are easier than the other neural network algorithm Backprop.

In the case of Backprop, which is in $G1$, the users not only need to determine the number of hidden nodes but also to decide the time to terminate training. Typically, multiple runs are necessary to determine good parameter settings.

Comprehensibility of results

On the comprehensibility of algorithm's results, CN2 and ITrule were given $G5$, because their results are simple and are very straightforward to interpret.

Cal5, CART, C4.5, and INDCART are in grade $G4$ because they have sophisticated and automatic pruning mechanism and generally produced smaller trees. Cal5, and CART, produced the smallest trees.

Grade $G3$ includes NewID, AC^2, and CASTLE. The results of AC^2 are less comprehensible, because it often produces extra large trees. CASTLE produced directed networks of probabilistic weights and generally the weights can provide insight to the relationships between attributes and classes.

Naive Bayes and the three discriminants are given grade $G2$, because their results are the easiest to explain in statistical terms. We also have the algorithms Alloc80 and k-N-N in this grade. Both are similar and simple in principle. Their results only involve an implicit evaluation of the similarity between new examples and stored examples. Normally k-N-N is easier to interpret than Alloc80 because it has a more straightforward evaluation mechanism.

In the last grade $G1$, we include SMART and the neural networks. SMART is in fact a three layered statistical algorithm; it produces a level of intermediate values from the input attributes. The neural network algorithms are all multi-layered and make use of a hidden layer. It is usually much more difficult to make sense of the values in the hidden layer.

6 Discussion and Analysis

6.1 Symbolic learning

Symbolic learning methods performed best (in terms of accuracy) on the Shuttle, Segment, and Credit datasets. Examining Table 1, we note that Shuttle and Segment have the highest values for kurtosis (> 7) and skew (> 1); indicating that they are the furthest from that of (multivariate) normal. Symbolic methods are generally non-parametric, i.e., they do not make any assumptions about the underlying distributions. Theoretically this should make them robust to distributions with large kurtosis and skew. The Shuttle dataset is also multimodal (Section 5.7), so it may be easier for symbolic algorithms to partition its example space. Many other algorithms in StatLog are nonparametric (e.g., Alloc80, SMART, and Backprop), but symbolic algorithms are shown here, in some sense, to be more robust to extreme distributions than these other nonparametric methods.

The characteristic that stands out about the Credit dataset is that it has a large number (50%) of binary/categorical attributes, eight out of sixteen, which suits symbolic algorithms. The other datasets that have high number of relevant categorical attributes are Heart ($>$ 38%) and German (65%), both of which have cost matrices. The presence of cost matrices degrade the performance of symbolic algorithms compared to statistical ones.

One point to ponder is that all of the datasets on which symbolic learning performed well came from workers within the symbolic learning community. (This also seems to be true in statistics and neural networks.)

6.1.1 Intra-symbolic learning comparisons

Examining the symbolic algorithms, they all performed very similarly. There is no obvious best algorithm. If the problem involves a cost matrix, then CART should be used – as it is the only algorithm designed to deal with such problems (INDCART has a new version that fully implement the cost processing capability of CART). This conclusion is based on theory, but it is also supported by the empirical evidence. In the problems with cost matrices that CART has been applied (Head, Heart, and German), CART was best among the symbolic algorithms.

Comparing the three algorithms based on ID3 (C4.5, NewID, and AC^2), C4.5 and NewID produce very similar results. AC^2 is slightly worse in terms of accuracy; it is certainly much slower (being written in LISP with a large interfact overhead). It also can produce over large trees. Comparing CART and INDCART, there was little difference in accuracy, CART was significantly faster.

Cal5 differs significantly from the other decision tree algorithms in its tree growing method. This tends to produce much smaller trees but lower accuracy, the exceptions being the Belgian and Diabetes data. The results on the Segment and Diabetes datasets are based on a modified version of Cal5 (version 5.2) that is much closer in form to the other decision tree algorithms.

CN2 and ITrule produce rules and not trees, but this does not seem to affect accuracy significantly. There is a strong similarity between CN2 and other tree algorithms. This may be explained by the fact that CN2 has two kinds of rules: ordered and unordered rules. (The case of ITrule is not entirely clear.) Ordered rules were used on: Satellite, KL Digits, Digits, Head, Vehicle, Credit, Shuttle, Belgian, Segment, Diabetes, and German. Ordered rules are also called *decision lists* [Rivest, 1987], which is closely related to (binary) decision trees. One point of interest to many algorithm creators may be the fact that ordered rules almost always did better on these numeric datasets than unordered rules. (However, the reason for this effect is unclear.)

6.1.2 Decision tree sizes

By far the most numerous symbolic algorithms are tree-based. There is often a very large difference in size between the trees produced by the different algorithms – commonly an order of magnitude. However, this difference in size may not be accompanied by much difference in accuracy. In such a case, a simpler tree may be preferred. Based on limited data we found that CART and Cal5 tend to produce the smallest trees, NewID produces the next largest trees whereas AC^2 normally produce the largest trees. INDCART produced trees larger than CART but smaller than NewID and AC^2. We believe that CAL5 produces trees similar to INDCART, larger than CART but smaller than NewID.

For equal amounts of data, smaller trees produce more reliable results. *Reliability* is inversely related to the difference between the training and testing accuracies. This is because as you move down a tree the number of examples on which to base the splits becomes smaller and statistically less reliable. Examining the sizes of trees produced by the different algorithms confirms this. In the Digits data for the 4×4 set: CART produces a tree with 164 nodes and has a training accuracy of 84% and a test accuracy of 82% (a difference of 2%); C4.5 has 704 nodes and has a training accuracy of 96.0% and a test accuracy of 85.1% (a difference of over 10%).

We examined the case of CART and its variant INDCART more closely. In both algorithms, cross-validation is used to estimate a parameter that defines the most successful level of pruning in the training data. INDCART uses *0-SE* default rule in pruning whereas CART uses the *1-SE* rule. The *0-SE* rule applies the estimated parameter directly to decide on the size of the final tree (the maximum likelihood estimate). In contrast, the *1-SE* rule estimates the standard error in the parameter and then uses the original parameter minus one standard error to grow the tree; this has the effect of biasing the trees towards simplicity. The empirical data shows that the trees produced by 0-SE are between two and ten times the size of those produced by 1-SE. The question arises, which is better?

The empirical evidence is unclear: with 0-SE doing better on Digits and the Credit datasets, and 1-SE doing better on the Head, Heart, Belgian, Segment, Diabetes, and German (Head, Heart, and German datasets have cost matrices). It seems that there is no single best rule, instead it depends on how much *noise* there is in the data (this would be consistent with the design of CART [Breiman *et al.*, 1984]). If there is little noise in the data, then the estimate of the pruning parameter is likely to be good, and the 0-SE rule should be used. If there is a lot of noise, the estimate of the parameter is likely to be bad; the 1-SE rule should be used.

6.2 Statistical analysis

6.2.1 Discriminant and regression algorithms

The three discriminant and regression algorithms are generally quite successful methods, in terms of accuracy and cost. They are also simpler than the other statistical and neural network algorithms. They performed very well on the three datasets with associated cost matrices: on the Head dataset (top three), Heart (three out of top five), German (Linear and Logistic top, Quadratic sixth). Logistic regression and linear discriminant appear to do better than quadratic discriminant. This may be explained by the low covariance ($SD\_ration < 1.2$) and moderate (or near moderate) skew (< 1.7) and kurtosis (< 8) of the three datasets.

Logistic regression and linear discriminant do very well on the Belgian (second and third) and Diabetes datasets (first and second), and quadratic discriminant does very well on the Digits (second), KL (third), and Vehicle (first) datasets. All three fail badly on the Satellite, Shuttle, and Segment datasets and their performance on the Credit dataset is also poor. It is intersting to note that these are precisely the datasets on which symbolic algorithms perform well. This is consistent with what is known about the properties of the discriminant algorithms that the properties we hypothesize favor symbolic algorithms: higher skew (> 1) or kurtosis (> 7), along with the presence of binary/categorical variables should disrupt these algorithms. The algorithms also fail on data with high correlation between the attributes ($\rho > 0.59$, e.g., Satellite).

In theory, logistic regression should be the most robust to binary/categorical variables. However, the data shows that the simple model of linear discriminant does just as well. The performance of linear discriminant and logistic regression seems more closely correlated than that with quadratic discriminant, they also appear more robust. Quadra performs best out of the three when SD_ratio is high (> 1.2), and skew and kurtosis are moderate ($skew < 1$, $0.27 < kurtosis < 5.5$ respectively), e.g., Satellite, Digits, KL, and Vehicle. However, it is more sensitive to extreme skew (> 1) and/or kurtosis (< 0.27 or > 5.5), e.g., Shuttle and German.

In terms of speed linear discriminant is one of the fastest algorithms in StatLog. Quadratic discriminant is slower (about the speed of a symbolic algorithm). Logistic regression is one of the slowest algorithms.

6.2.2 Nearest neighbor algorithms

The two nearest neighbor methods in StatLog are: k-nearest neighbor and Alloc80. k-N-N performs best, in terms of accuracy, on the image analysis datasets: Satellite, Digits, and KL; and rather poorly on all the others. In these three image analysis datasets, the attributes are taken more or less directly from brightness levels. This appears to favor k-N-N as there are no binary/categorical attributes (with no measure between attributes) and the attributes are relatively equally scaled (100%). It might have been expected that the correlation ($\rho = 0.5977$) between the attributes in the Satellite dataset might have disrupted k-N-N. However, this does not seem to be the case. The value of k that was found to be best was 1 for all datasets except Satellite (4), Digits (3), and Shuttle (3).

We have not experimented with more sophisticated distance functions such as those employed in instance-based learning [Aha *et al.*, 1991]. However, the literature suggests that this would perhaps improve results on some datasets, but might also decrease results on other datasets, compared to k-N-N [Aha, 1992].

Alloc80, like k-nearest neighbor, performs very well on the image analysis datasets: Satellite (third), Digits (third), and KL (second). It also performs well on the two other image analysis datasets with transformed attributes: Vehicle (second), and Segment (third). In addition Alloc80 does well on the German dataset. It is clear that Alloc80 is much more robust to binary/categorical attributes and scaling than is k-nearest neighbor. However, it is not clear what features are most suitable for Alloc80. It may have something to do with the size of the clustering of examples in attribute space, with small clusters favoring Alloc80. However we have no statistical measure of this.

Alloc80 is slower than k-nearest neighbor, both are very slow in testing.

6.2.3 Bayesian classification algorithms

The two directly "Bayesian" classifiers are Naive Bayes and CASTLE. The performance of Naive Bayes is poor, appearing near the bottom of the performance table for almost all the datasets. This might be expected as the assumptions on which it is based are very "naive". The only bright spot for Naive Bayes are the three medical datasets, especially Heart where it is the best performing algorithm. Theory indicates that it ought to perform well when the attributes are conditionally independent of the class. This is true for these two medical datasets. From Table 1 the correlation between attributes in Head is ($\rho = 0.1217$), Heart ($\rho = 0.1236$), and Diabetes ($\rho = 0.1439$). All are less than 0.15. Doctors may be expected to chose attributes that are conditionally independent. However, the two financial datasets also have low correlation between attributes, and Naive Bayes does not perform especially well on them (although better than the normal performance for Naive Bayes). These datasets have more binary/categorical attributes.

CASTLE had a slightly better record than Naive Bayes, but still performed quite poorly. The datasets on which it performed best are similar to Naive Bayes: Head, Heart, Diabetes, and Credit. The poor performance of CASTLE can be explained by that it is not primarily designed for classification. It is designed to give an easily comprehensible "picture" of the structure of the data, by indicating which attributes influence one another most strongly and can identify which subset of attributes are most strongly connected to the decision class. However, it ignores weak connections, some of which may have an influence on the final decision class.

Naive Bayes is the fastest algorithm in StatLog. CASTLE is also reasonably fast.

6.3 Multi-layered classifiers

The three multi-layered classifiers are SMART, Backprop, and radial basis functions. As expected, they perform similarly, especially between SMART and Backprop.

SMART is a very reliable algorithm, it does not perform very badly (in terms of accuracy and cost) on any of the StatLog datasets. Yet it performs well on the Belgian (first), Credit (second), Diabetes (third), and German (fourth) datasets. German has high skew (> 1) and kurtosis (> 7). SMART is non-parametric, and would be expected to perform well under such circumstances. These are similar conditions to where symbolic algorithms are expected to do well. SMART is a serious competitor to symbolic learning. An interesting point may be that these results cannot be improved by their successive discriminants (Table 1), but the reason for this is unclear.

SMART appears to be able to deal well with binary/categorical attributes and cost matrices. SMART does not however do so well on the Shuttle dataset, this may be due to the extreme multimodality of this dataset.

Back-propagation does reasonably well (in terms of accuracy) on the four image analysis datasets: Satellite, Digits, KL, and Vehicle, as well as on Head. It performs rather badly on the Heart (with cost matrix) and Credit datasets. The good results on the image datasets, especially the digits data (Digits and KL) where it is fourth, are consistent with the good results obtained by [McClelland *et al.*, 1986] for this problem. The poor performance on the Heart (and Credit) dataset may be caused by the presence of binary/categorical attribute (Head has only one). We have not experimented with variations of back-propagation (each of which may improve some aspect of the original algorithm), so the effect of these improvements is unclear on these datasets.

Radial basis functions performed well on the satellite dataset (second in terms of accuracy) and reasonably on Digits and KL Digits, but rather badly on Vehicle, Head, Heart and Credit. This is a similar pattern of results to that of k-N-N, the algorithms share a similarity of methodology.

SMART is very slow and is difficult to run because of its many parameters. Back-propagation is very slow, probably the slowest of all the StatLog algorithms. Radial basis functions runs about an order of magnitude faster. Improvement in the speed of neural network algorithms is a large research area, e.g., Quickprop [Fahlman, 1989]. Again, they have not been included in these comparisons.

A major drawback of neural networks is that they are very difficult to set up to produce good results. For example, to run back-propagation properly several parameters needed to be adjusted to the task: *temperature*, *stopping criterion*, *hidden nodes*, *the rate of updating weights*, *momentum coefficient*, *learning rate*, and *normalization method*. Any small change in these could decrease the performance substantially; much of the work involves a trial-and-error process. Given the mathematical arguments and results reviewed in [Ripley, 1992], despite the effort to use expert users, we are still not certain that our neural networks results are the best that they could have achieved — one more change of variables may have produced better results. Automatic methods of parameter selection for back-propagation is an important current research topic.

6.4 Misclassifcation cost versus accuracy

Some of the StatLog algorithms can use cost matrices in the training and/or testing phases. When cost must be minimized (as opposed to accuracy maximization), these algorithms have an obvious advantage. The two algorithms that can use costs in both the training and testing phases are: SMART and an experimental version of back-propagation (which was implemented by Statistics Department, Strathclyde University).

Most algorithms from the statistical community can process costs in the testing phase. This includes the symbolic algorithm CART, which originated from statistics. Such algorithms produce rules that maximize the accuracy on the training set. However, when the dataset is tested they minimize the cost of misclassification by multiplying the estimated probability of an example being misclassified in each class and chosing the lowest. For example, for a decision tree the probability of a predicted class is multiplied by the corresponding entry in the cost matrix.

Symbolic learning algorithms, for historical reasons, do not incorporate costs. Neither do most neural network algorithms. So they normally classify examples with the assumption that each misclassification costs the same, which departs from reality and differs sharply from the practice in statistics.

6.5 Human understanding critical problems

In many problems human understanding is likely to be crucial if the learned rules are to be useful, e.g., in legal decision making, medical and engineering diagnosis, knowledge discovery in chemistry and biology, pharmaceutical and engineering design and etc. StatLog researchers (both naive and expert users) confirm that symbolic algorithms have a definite advantage in this respect. For example, to construct an autopilot it is important to understand the reasons for each action of control. Neural networks and statistical algorithms can produce very little insight in these problems.

Researchers in statistics have recently been working on methods that can improve human comprehensibility. The Bayes networks methods seem to be the most promising. We already included a limited Bayes network algorithm, i.e., CASTLE, which constructs a polytree. [Spiegelhalter, 1990, Lauritzen and Spiegelhalter, 1988] has improved the understandability of Bayes networks by adding a graphical interpreter for causal reasoning. However, until now building such networks from data has proved to be difficult. CASTLE overcomes this difficulty by restricting the learned network to be polytrees.

6.6 Recommendation for Future Research

We believe that symbolic algorithms could be improved if a more sophisticated example space partitioning method is used. One alternative that is possible is the use of convex hulls or simplified forms of them.

Both symbolic and neural network algorithms need to be extended to incorporate (misclassification) costs. It seems that there is no great technical difficulty in doing this, but it has not previously been considered important.

Some datasets have seriation of, or closely related, classes, e.g., Satellite and Vehicle, as measured by $fract_{1 \ldots 4}$ ($fract_1$, $fract_2$, or $fract_3$ is greater than 90%). In such situations, it may be better to merge the series of classes to develop a set of classification rules (with a merged class), and then develop a regression equation for the classes that have been merged. However, no algorithm in StatLog can do this.

Our analysis has made a start at recognising features that favor particular algorithms. But much more work is required. The aim being to develop mechanisms that can automatically analyze data and then select the best algorithm.

7 Conclusion

This paper presents results from the StatLog project on classification algorithms. This project involves a comprehensive comparison between seventeen established, state-of-the-art algorithms from symbolic learning, statistics, and neural networks on twelve large real-world classification tasks. Although any comparative study is by its nature limited, a number of tentative conclusions can be drawn from the StatLog results.

- The main conclusion is that: there is no single *best* algorithm, and it is a case of "horses for courses". The best algorithm for a particular dataset depends crucially on features of that dataset. We have made a start at investigating what these features are.

- Symbolic algorithms are a good choice for maximizing accuracy, if the data has extreme distribution, i.e., it is far from normality as measured by its skew (> 1) and kurtosis (> 7), and/or if there is a high percentage of binary/categorical attributes ($> 38\%$). If ease-of-use and human understandability is of a high priority symbolic algorithms should also be chosen. Symbolic algorithms are poor choices if: misclassification cost should be minimized (perhaps CART is an exception), or the dataset has equally scaled and equally important numerical attributes. The performance of symbolic algorithms are by and large similar; CART should be chosen if costs are involved.

- The three discriminants perform well: if cost should be minimized, and if the data is close to (multivariate) normal (measured by skew < 1 and kurtosis < 7). When the covariance between attributes is higher ($SD\_ratio > 1.2$), the quadratic version can be expected to do better but it is more sensitive to extreme skew (> 1) or kurtosis (< 0.27 or > 5.5). Because of their parametric nature, the performance of these three algorithms are bound by the canonical discriminant correlations of the data ($cancor_{1...4}$). All three are bad choices, if the data is far from normal, e.g., high skew ($>$ 1) and high kurtosis (> 7), if the data has high percent of relevant binary/categorical attributes ($> 38\%$), and if the correlation ($\rho > 0.59$) is high. Logistic regression would be too demanding on machine resources when the dataset is large.

- Nearest neighbor algorithms are a good choice for accuracy and cost, if the attributes are all equally scaled and equally important. Alloc80 is the most robust of the nearest neighbor algorithms.
- Bayes classifiers should not normally be considered for classification unless the dataset has very low correlation (ρ near zero) and few other complications (e.g., low covariance SD_ratio, or < 1.1).
- SMART may be chosen on datasets between those favoured by symbolic algorithms and discriminants. SMART is a robust algorithm and rarely produces low accuracies. SMART also deals very well with datasets with costs.
- Back-propagation may be chosen is situations similar to SMART. Both Backprop and SMART require large machine resources overhead.

Acknowledgements

This work has been supported by the European Commission (ESPRIT 5170). C. Feng thanks Ottawa Machine Learning Group and R.D. King thanks the Imperial Cancer Research Fund for providing facility to complete this paper. We would like to thank the help of Dr. R. Henery of Strathclyde University and other Partners of StatLog for their help at various stages of the work in this paper (contact information in Appendix C). Thanks are due to A. Srinivasan and J. Catlett for contributing their databases. We thank D. Aha for comments on a draft of this paper, and the reviewers for many valuable suggestions to improve it.

A Description for Algorithms

A.1 Symbolic algorithms

C4.5, NewID and AC^2

C4.5 [Quinlan, 1987a, Quinlan, 1987b], NewID, AC^2 are descendant of ID3 [Quinlan, 1986]. They select an attribute for splitting a dataset into subsets according to the result of a test conducted on the attribute. This set is used to form the root of the tree and the chosen attribute divides it into subsets. Each subset is used to construct a subtree recursively. For C4.5, NewID and AC^2, to choose an attribute they examine the entropy of potential splits resulting from attributes. The *entropy* of a node with dataset s is $I(s) = -\sum_i P_i \log P_i$,, where P_i is the probability of the ith class in that set. The *entropy gain* is the difference of entropy between the current node and the nodes that are introduced by a split:

$$Gain(N) = I(N) - \sum_k I(N_k)$$

where $I(N_k)$ is the entropy of the kth child depending on the probabilities P_{kj} (for the jth class). The entropy *gain ratio* takes both the entropy gain and the information on attributes into consideration: $Ratio(s) = Gain(s)/I(a)$. The entropy of an attribute a is $I(a) = -\sum_j P_j \log P_j$, where P_j is the probability of the jth value of the attribute a. The attribute with the highest gain (or ratio) is selected.

The entropy criterion is a local measure so it does not guarantee to yield the smallest tree. Noise can also increase tree size and therefore decreases performance. So many algorithms apply *pre-pruning* and/or *post-pruning*. Pre-pruning stops testing when the number and/or mixture of examples belonging to different classes falls below a prespecified level. In post-pruning, a complete tree is built that classifies all the training data correctly. Then successive nodes are removed, which changes the ratio of examples classified into different classes at the leaves, until the tree is optimal according to a certain criterion.

C4.5 uses the entropy gain ratio instead of ID3's gain. C4.5 incorporates pre-pruning and post-pruning. NewID contains post-pruning, and produces probabilities of each class. All can also handle attributes that are continuous numeric values. NewID is from the Turing Institute Ltd., UK, and AC^2 is from ISoft, France.

CART and INDCART

The acronym CART comes from *Classification and Regression Tree* [Breiman *et al.*, 1984]. CART is not really a single algorithm, but a collection of tree algorithms and analysis methods for classification and regression (or discrimination and forecasting). The algorithm described in this section is the most commonly used version.

CART is a binary tree algorithm, i.e., each node has only two branches. It has a different measure for attributes, i.e., the *Gini* function $1 - \sum_j P_j^2$, which measures the "impurity" of a node. Its most crucial difference from the other decision tree algorithms is the post-pruning process known as the "minimal cost-complexity pruning" [Breiman *et al.*, 1984] to determine the *tradeoff* between getting the right size of the tree and getting the accurate estimates of the true probabilities of misclassification. This method uses a separate *pruning* dataset.

The cost complexity for each tree is the summation of misclassifications on the pruning dataset and a proportion of the number of terminal nodes. The proportion is defined by a weight α, which measures the cost per leaf. In the first stage, for each α, CART finds a subtree that minimizes the cost complexity. Because of the finiteness of the subtrees, there is a sequence of *pruned* trees that have decreasing number of nodes corresponding to increasing α. In the second stage, CART selects the tree in the sequence that has the least expected classification error. Cross-validation is used to obtain a more honest estimate of the error. CART uses the *1-SE* rule after cross-validation, meaning that the error of the selected tree is within one standard error of the minimal error.

INDCART is a variant of CART and can be obtained from NASA Ames Center (we used version 1.3). It differs from CART [Breiman *et al.*, 1984] by using a different method for processing missing values (not used in the study), in a different default setting for pruning (i.e., the no error *0-SE* rule after cross-validation), and in not implementing the regression part of CART.

Cal5

Cal5 is based on statistical methods and is designed for continuous variables. It is from the Fraunhofer Institute, Berlin [Meyer-Broetz and Schuermann, 1970a, Unger and Wysotzki, 1981], and was developed independently of the work on decision trees in the West. In Cal5, trees are constructed sequentially starting with one attribute and branching with other attributes recursively, if no sufficient discrimination of classes can be achieved. That is, if at a node no decision for a class c_i according to the above procedure can be made, a branch formed with a new attribute is appended. If this attribute is continuous, a discretization, i.e., intervals corresponding to qualitative values have to be used.

Let N be a certain non-leaf node in the tree construction process. At first the attribute with the best local discrimination measure at this node has to be determined. For that a method working without any knowledge about the result of the desired discretization is used. For continuous attributes, $quotient(N) = A^2/(A^2 + D^2)$ is a discrimination measure for a single attribute [Meyer-Broetz and Schuermann, 1970a], where A is the standard deviation of examples in N from the centroid of the attribute value and D is the mean value of the square of distances between the classes. This measure has to be computed for each attribute. The attribute with the least value of $quotient(N)$ is chosen as the best one for splitting at this node. Note that at each current node N all available attributes $a_1, a_2, \ldots, a_n$ will be considered again. If a_i is selected and occurs already in the path to N, than the discretization procedure leads to a refinement of an already existing interval.

CN2 and ITrule

CN2 [Clark and Niblett, 1988] and ITrule [Goodman and Smyth, 1989] are rule-based algorithms. They produce a set of "*If* ... *Then* ..." rules. The conditions are constraints on attribute values and the conclusion is a class. They generate decision rules for each class in turn. Given a class, they start with a *universal rule*, which assigns all examples to the current class. Specializations are then repeatedly generated and explored until all rules *consistent* with the data are found. Each rule must correctly classify at least a prespecified percentage of the examples belonging to the current class. As few as possible *negative* examples, i.e., examples in other classes, should be covered. Specializations are obtained by adding a condition to the left-hand side of the rule.

CN2 is an extension of AQ [Michalski, 1983] with several techniques to cope with noise in data. It can also process continuous numeric values in attributes. The main technique for reducing error is to minimize $(k+1)/(k+n+c)$ (Laplacian function), where k is the number of examples classified correctly by a rule, n is the number of examples classified incorrectly, and c is the total number of classes.

ITrule [Goodman and Smyth, 1989] produces rules of the form "*If* ... *Then* ... *with probability* ...". This algorithm contains probabilistic inference through the J-measure, which evaluates its rule candidates. It differs from CN2 since it may drop a condition to generalize a rule. J-measure is a product of prior probabilities for each class and the cross-entropy of class values conditional on the attribute values. ITrule cannot process continuous numeric values.

A.2 Statistical algorithms

Bayes classifiers

Naive Bayes and CASTLE are both Bayesian classification algorithms (They should not be confused with Bayesian statistical methods.) The version of Naive Bayes used in StatLog is part of the IND package with INDCART. It directly applies the Bayes rule: $P(c|e) = P(e|c) \cdot P(c)/P(e)$, where c is the class and e is a given example. Given an example, its most probable class is c_i, if for all j $(i \neq j)$, $P(e|c_i)P(c_i) > P(e|c_j)P(c_j)$. Bayes classification requires complete and accurate probability data. For real problems this is impossible so it is frequently assumed that all attributes are independent conditional on the classes, which entails $P(e|c) = P(a_1|c) \cdot P(a_2|c) \cdot ... \cdot P(a_n|c) for$ the n attributes a_i $(i = 1, ..., n)$. In principle, it is possible to include prior information into the Bayesian analysis, but this is rarely done in practice. Naive Bayes can cope with missing values by simply ignoring them when estimating the joint probability.

CASTLE (CAusal STructures for inductive Learning) [Acid *et al.*, 1991] is an implementation of the polytree algorithm defined by [Pearl, 1988]. A causal network [Pearl, 1988] is a directed acyclic graph (DAG) in which the nodes represent propositions (or variables), the arcs signify the existence of direct causal dependencies between the linked propositions, and the strength of these dependencies is quantified by conditional probabilities. CASTLE is restricted to polytrees in which no more than one path exists between any two nodes of the same network.

The structure of a causal network can be determined in the following way: each variable in the domain is identified with a node in the graph. We then draw arrows to each node X_i from a set of nodes $C(X_i)$ considered as direct causes of X_i. The strengths of these direct influences are quantified by assigning to each variable X_i a matrix $P(X_i|C(X_i))$ of conditional probabilities (Kullback-Leibler information [Read and Cressie, 1988]) of the events

$X_i = x_i$ given any combination of values of the parent set $C(X_i)$. The conjunction of these local probabilities defines a consistent global model, i.e., a joint probability distribution. Once the network is constructed it constitutes an efficient device to perform probabilistic inferences. The problem of building such a network remains. The structure and conditional probabilities necessary for characterizing the network could be provided either externally by experts or from direct empirical observations.

Under the Bayesian approach, the learning task in CASTLE separates into two related subtasks: 1) to identify the topology of the network, and 2) to compute the numerical parameters (conditional probabilities) for a given network topology.

k-Nearest neighbor

This algorithm (k-N-N) normally maintains a set of training examples and then classifies new examples by trying to estimate their various properties in relation to the old examples. It finds a sample of k closest examples that are most similar to a new example in distance (i.e., we used the Euclidian distance function, which defines its behavior) and classifies the new example as the most frequent class among the sample. In training, k-nearest-neighbor chooses a value for k. It evaluates the classification resulting from different values of k, and selects the one that gives the best performance.

It should be noted that instance-based algorithms in symbolic learning [Aha *et al.*, 1991] have extended k-nearest-neighbor in various ways.

Alloc80

Alloc80 is a density estimation algorithm, whose underlying principle for classification is similar to that of k-nearest-neighbor. The algorithm used is from University of Leeds, UK. This version allows for all types of continuous and discrete data.

The program performs multigroup discriminant analysis; within each group the variability is modeled using a non-parametric density estimator, based on kernel functions. Suppose that we have to estimate the p-dimensional density function $f(x)$ of an unknown distribution, information about f is given by n independent observations from this distribution, i.e., $Y_i = (Y_{i1}, \ldots, Y_{im}, \ldots, Y_{ip})$ with $i = 1, 2, \ldots, n$. Let $K^{(p)}(X; Y_i, \lambda)$ be a kernel function centred at Y_i and let λ denote the window width of the kernel, the estimate of $f(x)$ is:

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^{n} K^{(p)}(X; Y_i, \lambda).$$

Examples in the test data are then allocated to classes based on a calculation of the posterior odds by a standard Bayesian calculation. The smoothness of the kernel density estimates, as governed by the window width, is determined by a pseudo maximum likelihood method. The program can handle continuous as well as mixed (discrete) data.

This program is computationally expensive, both in storage and CPU terms. The methods to choose the smoothing parameter automatically are not always successful, and the version used in StatLog is rather unwieldy. It is expected to do better than standard methods where the data are highly non-*normal*. For more bizarre datasets, such as two interlocking spirals, this method performs very well. In general, any situation where the boundaries between classes is not easily modeled by a straight line or quadratic will lend themselves well to this approach. The main difficulty, as with most nonparametric density estimators, is to ensure a good choice of smoothing parameter (λ).

Discriminants

Fisher's linear discriminant (Discrim) is perhaps the single most commonly available algorithm for discrimination. It assumes that a linear combination of the attribute values can separate the classes. Linear discriminant is optimal in accuracy (not necessarily in terms speed or comprehensibility), when the populations of each class are all *multivariate normal* with common covariance matrix (i.e., there is a geometrical center of the examples in the, say, two classes, and the distribution of the examples is ellipsoidal around that center for the given data).

To classify an example, a linear discriminant function is calculated for each class and the one with the largest value is selected. So for the kth class: $f(k) = \alpha_{k1}a_1 + ... + \alpha_{kn}a_n$, where a_i is the ith attribute and α_{ki} $(i = 1, ..., n)$ are constant coefficients estimated from the data. This is equivalent to constructing a separating hyperplane of dimension one less than the number of attributes. For example, to separate data with two attributes a line will suffice. If the examples are evenly distributed, i.e., with same prior probabilities, the coefficients α_{ki} $(k = 1, 2, i = 1, ..., n)$ are computed such that the decision plane is mid-way between the two centers of gravity of the two classes. Linear discriminant algorithm cannot process missing values but it is easy to apply and its results are relatively easy to interpret. Variants of linear discriminant include quadratic discriminant (Quadra). The difference is in the function $f(k)$ – a quadratic function of the attributes. Both Discrim and Quadra are from University of Strathclyde, UK, and both are implemented in S-plus/Fortran.

Logistic regression

Logistic regression is designed to process combinations of continuous, categorical and qualitative attributes [Cox, 1966, Day and Kerridge, 1967]. The probability of an example e given the class of $c_e = k$ $(k = 1, ..., m)$ can be computed as:

$$P(e|c_e = k) = (R(e|c_e = k))/(\sum_{i=1}^{m} R(e|c_e = i)).$$

$R(e|c_e = k)$ is the relative probability to a fixed class (the last one m, say), which is a logistic function of the parametric linear combination of attributes: $R(e|c_e = k) = P(e|c_e = k)/P(e|c_e = m) = exp^{-(\alpha_{k1}a_1^e + ... + \alpha_{kn}a_n^e)}$, where a_i^e $(i = 1, ..., n)$ are the attribute values of the example e and $R(e|c_e = m) = 1$. The coefficients α_{kj} $(k = 1, ..., m, j = 1, ..., n)$ are estimated for each class and it must maximize the total likelihood: $p(\{e|e \text{ is in the sample}\})$. Assume that the sample is randomly chosen so the examples are independent:

$$p(\{e|e \text{ is in the sample}\}) = \prod_{\{e|c_e=1\}} p(e|c_e = 1) \cdot \prod_{\{e|c_e=2\}} p(e|c_e = 2) \cdot \ldots \cdot \prod_{\{e|c_e=m\}} p(e|c_e = m).$$

$p(e|c_e = k)$ has close links with the binomial distribution, thus its arguments can be categorical.

The logistic regression (LogReg) method produces a linear separation of classes. It can start from the coefficients estimated by Discrim. Its main difference from Discrim is in the way that the coefficients $\alpha_{kj}(k = 1, ..., m, j = 1, ..., n)$ (the parameters for the separation hyperplanes) are estimated. Discrim optimizes a quadratic error function whereas logistic regression optimizes on the total likelihood.

LogReg is also implemented in S-plus/Fortran and is from University of Strathclyde, Glasgow, UK.

Projection pursuit

SMART (Smooth Multiple Additive Regression Technique) is implemented in FORTRAN by Friedman [Friedman and Stuetzle, 1981]. It is from University of Leeds, UK. It is a generalization of projection pursuit regression. It is a nonparametric and iterative process. It starts with estimates of "ridge" functions – a function of one linear combination of the attributes – chosen to best model the relations between the classes and attribute values among training examples. Each subsequent iteration incorporates a new ridge function to improve the model. The classes are calculated from this intermediate level of values.

The regression models take the form:

$$E[Y_i|x_1, x_2, \ldots, x_p] = \overline{Y}_i + \sum_{m=1}^{M} \beta_{im} f_m(\sum_{j=1}^{p} \alpha_{jm} x_j), \tag{1}$$

with $\overline{Y}_i = EY_i$, $Ef_m = 0$, $Ef_m^2 = 1$ and $\sum_{j=1}^{p} \alpha_{jm}^2 = 1$. The coefficients β_{im}, α_{jm} and the ridge functions f_m are parameters of the model and are estimated by least squares. The criterion:

$$L_2 = \sum_{i=1}^{q} E\big[Y_i - \overline{Y}_i - \sum_{m=1}^{M} \beta_{im} f_m(\alpha_m^T x)\big]^2,$$

is minimized with respect to the parameters β_{im}, $\alpha_m^T = (\alpha_{1m}, \ldots, \alpha_{pm})$ and the functions f_m.

Classification is closely related. The objective here is to minimize the misclassification risk:

$$R = E\big[\min_{1\leq j\leq q} \sum_{i=1}^{q} l_{ij} p(i|x_1, x_2, \ldots, x_p)\big],$$

where l_{ij} is the specified cost for predicting $Y = c_j$ when its true value is c_i ($l_{ii} = 0$). The conditional probability is reformulated using a conditional expectation that is then modeled by (1).

This algorithm can be expected to perform very well whenever a cost matrix is applicable, because it (unusually) uses the cost matrix in the *training* phase as well as in the classification stage. Although the training time is not competitive, the algorithm does generally produce good misclassification rates.

A.3 Neural networks

Back-propagation

Backprop was implemented by R. Rohwer while at the Center for Speech Technology Research, Edinburgh, UK. It is a neural network algorithm, which consists of a network of "neurons" arranged in a number of layers, where each neuron is connected to every neuron in the adjacent layers. Each neuron sends a signal along the connections to the neurons in the layer above. The signals are multiplied by weights corresponding to each connection.

We used a version with three layers: an input layer, a hidden layer, and an output layer: *strictly layered, 3-layer MLP*, which is the mapping:

$$y_i^{(H)} = f^{(H)}\left(\sum_j w_{ij}^{(HI)} y_j^{(I)}\right), \quad \text{and} \quad y_i^{(T)} = f^{(T)}\left(\sum_j w_{ij}^{(TH)} y_j^{(H)}\right), \tag{2}$$

from the inputs $y^{(I)}$ to the targets $y^{(T)}$, via the *hidden nodes* $y^{(H)}$. The parameters are the *weights* $w^{(HI)}$ and $w^{(TH)}$. The univariate functions $f^{(\cdot)}$ are usually each set to be logistic, which varies smoothly from zero at $-\infty$ to one at ∞.

It can be shown that, given enough hidden nodes, this can approximate any mapping to an arbitrary accuracy. This mapping has a (controversial) biological interpretation: Node (model neuron) i produces a strong output if its *activation potential* $\sum_j w_{ij} y_j$ is positive, and a weak output if it is negative. The activation is increased if node i is connected to an active node j via an *excitatory* synapse ($w_{ij} > 0$), and is decreased by active nodes connected via *inhibitory* synapses ($w_{ij} < 0$).

One input to each layer, say node 0, is traditionally assigned the constant value 1 so that w_{i0} provides a constant offset or *bias* for the activation of i. This device allows the *threshold* activation (due to the remaining nodes), at which $y_i = 0.5$, to be adjusted away from 0.

Radial basis functions

Radial basis functions methods are closely related to the kernel estimator methods discussed under nonparametric discriminant analysis [Poggio and Girosi, 1990]. This version of radial basis functions is implemented by R. Rohwer [Renals and Rohwer, 1989]. The radial basis function network mapping is given by:

$$y_i^{(H)} = f^{(H)}\left(\frac{\sqrt{\sum_j (y_j^{(I)} - c_{ij})^2}}{r_i}\right), \quad \text{and} \quad y_i^{(T)} = \sum_j w_{ij}^{(TH)} y_j^{(H)}. \tag{3}$$

It has a linear output layer like an MLP with such an option, but the hidden layer is different. Hidden node i computes a function of the Euclidian distance of an input from its *center* $c_{i\cdot}$, on a scale determined by its *radius* r_i. Usually the chosen function is the Gaussian $f^{(H)}(x) = e^{-x^2}$. As in the MLP, a bias node is introduced into the linear layer by setting, say, $Y_0^{(H)} = 1$.

Sometimes non-Euclidian distance measures are used. The loosely-defined region for which a radial basis functions has a significant output is its *receptive field*. The functions computed at the hidden nodes are the "radial basis functions" *per se*. They are "radial" in that their receptive fields are spherically symmetric; there is an r_i for each center i but not an r_{ij} for each center and input coordinate. Such a generalization is often made, however, and if it is not, then it is desirable to prescale the input data to give it equal variance in each dimension.

B Dataset Description

1) Satellite image. This sample database was generated by taking a small section (82 rows and 100 columns) from the original Landsat Multi-Spectral Scanner image data from a part of Western Australia. This data is originally purchased from NASA by the Australian Center for Remote Sensing. There are six classes (red soil, cotton crop, grey soil, damp grey soil, very damp grey soil and soil with vegetation stubble) of different soil conditions to be classified according to the four spectral bands of the 256 grey-level image pixels and the spectrals of 3×3 neighboring pixels.

2) Hand written digits. This data is used for the recognition of the digits 0-9 from 4×4 pixellated images of 2000 individual's handwritten postcodes on letters passing through the German Federal Post [Kressel, 1991]. Each example of 18000 handwritten digits was originally digitized onto a 16×16 pixel array with grey levels 0(white)-255(black). The pixel values were then averaged over 4×4 neighborhoods to produce the 4×4 images. A very small number of mistaken images were allowed to appear. The data is supplied by Daimler-Benz, Forschung, Ulm, Germany.

3) Karhunen-Loeve digits (KL). This dataset is produced by a linear transformation of the original 16×16 images (Karhunen-Loeve transformation [Kendall *et al.*, 1983]). The eigenvectors of the covariance matrix of the original 16×16 images were computed. The scalar products of the top 40 eigenvectors with the original images were calculated. It was found that the original images could be reconstructed from these 40 eigenvectors with few loss of information. These 40 scalar products are the attributes of the KL dataset.

4) Vehicle silhouettes recognition. This dataset is concerned with the recognition of double deck bus, Chevrolet van, Opel Manta 400, and Saab 9000 models from features extracted from their silhouette images taken from different angles [Siebert, 1987]. It is from The Turing Institute Ltd., Glasgow, UK. The images were acquired by a camera looking downwards at the model vehicles from a fixed angle of elevation. The vehicles were also rotated and their angle of orientation was measured with "head on", "rear" and two side views respectively.

5) Segment. This dataset came from a database of outdoor images, the problem being to classify a 3×3 pixel region. The data is from the databases at the University of California, Irvine. It was donated originally by C. Brodley with permission of the Vision Group, University of Massachusettes.

6) Shuttle control. In this dataset the classes are the appropriate actions under given conditions of the space shuttle in flight. This data is believed to concern the position of radiators on the space shuttle. The data was taken from an actual space shuttle flight and came from the NASA Space Center in Huston. It was acquired from Sydney University by Strathclyde University. In the data approximately 80% (default accuracy) of the data belongs to class 1. There appears to be little (if any) noise in the data.

7) Credit risk. The data classifies a customer of the credit industry as a good or bad credit risk given his financial history. Sixteen of the 39 attributes in the original data were retained (with eight numerical and eight binary/categorical attributes). The ratio of good to bad customers in the dataset is almost 1:1. This is not representative of the population as a whole, where the ratio is more like 10:1. The data is from Attar Software Ltd, UK.

8) Belgian. This dataset is concerned with the detection of emergency voltage conditions. The data comes from a simulated model of a power system. The dataset was provided by L. Wehenkel of the Institut Montefiore, University of Liege, B-4000, Liege, Belgium [Cutsem *et al.*, 1991].

9) Diabetes. This dataset is concerned with the prediction of the onset of diabetes mellitus. All the examples describe female patients at least 21 years old of Pima Indian heritage [Smith *et al.*, 1991]. It is also from the ML database at U.C.Irvine.

10) Head injury. The dataset is a series of patients with severe head injury collected prospectively by neurosurgeons between 1968 and 1976 [Titterington *et al.*, 1981]. Outcome was categorized according to the five categories of the Glasgow Outcome Scale, but they were reduced to three for the purpose of prediction. The attributes (five numerical attributes and one binary/categorical) are age and various indicators of the brain damage, as reflected in brain dysfunction. The attributes are either binary or ordered. The predicted classes are dead, severe and moderate. So the cost of misclassifying a "moderate" injury as a "dead" one is very high. The cost matrix is: [0,10,75;10,0,90;750,100,0].

11) Heart disease. This data is for the diagnosis of heart disease from thirteen different symptoms. This is the Cleveland data and is from the ML database of U.C.Irvine. Six of the original 303 examples were discarded because they contain missing values. Then, 27 of these were left out as a validation set, leaving a final total of 270. Two classes are the presence and absence of heart disease (reduced from the original four). The attributes are a mixture of binary, ordered categorical, and real values. It is assumed that the cost of misclassifying a unhealthy patient is five times that of doing the opposite misclassification. (This data has been studied in the literature before, but without taking a cost matrix into account.) The cost matrix is: [0,1;5,0].

12) German. This dataset is for the classification of credit risk in Germany. Its cost matrix is in the right of the above table. Thirteen attributes are binary/categorical and seven are numerical. This data is from Daimler-Benz, Forschung Ulm, Germany. The cost matrix is: [0,1;5,0].

C StatLog Contacts

R. Henery (statistics and technical information), Department of Statistics and Modeling Science, Livingstone Tower, University of Strathclyde, Glasgow, UK.

r.j.henery@vaxa.strath.ac.uk

G. Nakhaeizadeh (symbolic algs.), Daimler Benz AG., Forschung, Ulm, Germany

reza

C. Taylor (statistics), Department of Statistics, University of Leeds, Leeds, UK

charles@amsta.leeds.ac.uk

H. Perdrix (AC^2), ISoft SA., Paris, France

hp@is21.isoft.fr

J. Stender (neural nets, Genetic Algorithms), Brainware GmbH., Berlin, Germany

gary@uucp.brainwr

R. Molina (CASTLE), Department of Computer Science, University of Granada, Granada, Spain

rms@es.ugr

F. Wysotzki (Cal5), Fraunhofer Institute, Berlin, Germany

References

[Acid *et al.*, 1991] S. Acid, S. L. de Campos, A. González, R. Molina, and Pérez de la Blanca. 1991. CASTLE : A Tool for Bayesian Learning. *Esprit Conference 1991.*

[Aha *et al.*, 1991] D. W. Aha, D. Kibler, and M. K. Albert. 1991. Instance-based learning algorithms. *Machine learning*, 6.

[Aha, 1992] D. Aha. 1992. Generlising case studies: a case study. *9th Int. Conf. on Machine Learning*, pp 1–10, San Mateo, Cal., Morgan Kaufmann.

[Atlas *et al.*, 1991] L. Atlas, J. Connor, D. Park, M. El-Sharkawi, R. Marks, A. Lippman, R. Cole, and Y. Muthasamy. 1991. A performance comparison of trained multi-layer perceptrons and trained classification trees. *Systems, man, and cybernetics: proceedings of the 1989 IEEE international conference*, pp 915 – 920, Cambridge, Ma. Hyatt Regency.

[Bonelli and Parodi, 1991] P. Bonelli and A. Parodi. 1991. An efficient classifier system and its experimental comparisons with two representative learning methods on three medical domains. *ICGA-91: genetic algorithms: proceedings of the fourth international conference*, pp 288 – 295, San Mateo, CA. Morgan Kaufmann.

[Breiman *et al.*, 1984] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. 1984. *Classification and Regression Trees.* Wadsworth, Belmont.

[Buntine, 1989] W. Buntine. 1989 Learning classification rules using bayes. *Proceedings of the Sixth International Machine Learning Workshop*, pp 94 – 98, Cornell, New York. Morgan Kaufman, San Mateo, CA.

[Cherkaoui and Cleroux, 1991] O. Cherkaoui and R. Cleroux. 1991. Comparative study of six discriminant analysis procedures for mixtures of variables. *Proceddings of Interface Conference 1991.* Morgan Kaufmann.

[Clark and Boswell, 1991] P. Clark and R. Boswell. 1991 Rule induction with cn2: some recent improvements. *EWSL '91*, pp 151–163, Porto, Portugal. Berlin: Springer-Verlag.

[Clark and Niblett, 1988] P. Clark and T. Niblett. 1988. The CN2 induction algorithm. *Machine Learning*, 3(4) 261–283.

[Cox, 1966] D. R. Cox. 1966. Some procedures associated with the logistic qualitative response curve. In *Research Papers in Statistics: Frestschrift for J. Neyman*, volume 45. ed. F. N. David. Wiley, New York, 1966.

[Cutsem *et al.*, 1991] T. Van Cutsem, L. Wehenkel, M. Pavella, B. Heilbronn, and M. Goubin. 1991. Decision trees for detecting emergency voltage conditions. *2nd Int. Workshop on Bulk Power System Voltage Phenomena – Voltage Stability and Security*, U.S.A. McHenry.

[Day and Kerridge, 1967] N. Day and D. Kerridge. 1967. A general maximum likelihood discriminant. *Biometrics*, 23 313–323.

[Ersoy and Hong, 1991] O. K. Ersoy and D. Hong. 1991. Parallel, self-organizing, hierarchical neural networks for vision and systems control. In *Intelligent motion control: proceedings of the IEEE international workshop*, ed. O. Kaynak. New York. IEEE.

[Fahlman, 1989] S. Fahlman. 1990. Distributed connectionist systems for ai: prospects and problems. *Concepts and characteristics of knowledge-based systems: selected and reviewed papers from the IFIP TC 10/WG 10.1 workshop. Mount Fuji, Japan. 9-12 November, 1987.*, pp 23 – 35, Amsterdam. North-Holland.

[Fahlman, 1991] S. Fahlman. 1991. The cascade-correlation learning algorithm on the monk's problems. In *The MONK's problems - a performance comparison of different learning algorithms*, ed. S. Thrun, J. Bala, E. Bloedorn, and I. Bratko pp 107 – 112. Carnegie Mellon University,Computer Science Department.

[Feng *et al.*, 1993] C. Feng, A. Sutherland, R. King, and B. Henery. Comparing machine learning classifiers to statistics and neural networks. 1993. *Proceedings of International Conference on Artificial Intelligence and Statistics.*, Florida. Society of AI and Statistics and IASC.

[1] D. H. Fisher and K. B. McKusick. 1989. An empirical comparison of id3 and back-propagation (vol 1). *IJCAI 89*, pp 788 – 793, San Mateo, CA. Morgan Kaufmann.

[Fisher and McKusick, 1989] D. Fisher, K. McKusick, R. Mooney, J .W. Shavlik, and G. Towell. 1989. Processing issues in comparisons of symbolic and connectionist learning systems. *Proceedings* *of the sixth international workshop on machine learning. Cornell University, Ithaca, New York. June 26-27, 1989.*, pp 169 – 173, San Mateo. Morgan Kaufmann.

[Friedman and Stuetzle, 1981] J. H. Friedman and W. Stuetzle. 1981. Projection pursuit regression. *Journal of American Statistics Association*, 76 817–823.

[Goodman and Smyth, 1989] R. M. Goodman and P. Smyth. 1989. The induction of probabilistic rule sets - the itrule algorithm. *Proceedings of the sixth international workshop on machine learning. Cornell University, Ithaca, New York.*, pp 129 – 132, San Mateo, CA. Morgan Kaufmann.

[Gordon and Olshen, 1978] L. Gordon and R. A. Olshen. 1978. Asymptotically efficient solutions to the classification problem. *Ann, Statist.,*, 6 515–533.

[Gorman and Sejnowski, 1988] R. P. Gorman and T. J. Sejnowski. 1988. Analysis of hidden units in a layered network trained to classify sonar targets. In *Neural networks*, 1 (Part 1):75 – 89.

[Hand, 1981] D. J. Hand. 1981. *Discrimination and Classification.* Chichester, UK, Wiley.

[Hermans *et al.*, 1974] J. Hermans, J. D. F. Habbema, and A. T. Van der Burght. 1974. Cases of doubt in allocation problems, k populations. *Bulletin of International Statisics Institute*, 45 523–529.

[Huang and Lippmann, 1987] W. Y. Huang and R. P. Lippmann. 1987. Comparisons between neural net and conventional classifiers. *Proceedings of the IEEE first international conference on neural networks.*, pp 485 – 494, Piscataway, NJ. IEEE.

[Huang *et al.*, 1991] H. H. Huang, C. Zhang, S. Lee, and H. P. Wang. 1991. Implementation and comparison of neural network learning paradigms: back propagation, simulated annealing and tabu search. *Intelligent Engineering Systems Through Artificial Neural Networks: Proceedings of the Artificial Neural Networks in Engineering Conference.*, New York. American Society of Mechanical Engineers.

[Karalic, 1992] A. Karalic. 1992. Employing linear regression in regression tree leaves. *Proceedings of the tenth European conference on artificial intelligence*, pp 440 – 441, Vienna, Austria. England: John Wiley and Sons.

[Kendall *et al.*, 1983] M. G. Kendall, A. Stuart, and J. K. Ord. 1983. *The advanced Theory of Statistics, Vol III: Design, Analysis and Time Series.* Griffin, London, UK.

[King *et al.*, 1992] R. King, R. Henery, C. Feng, and A. Sutherland. 1994. A comparative study of classification algorithms: Statistical, machine learning, and neural network. In D. Michie, editor, *Machine Intelligence 13.* ed. D. Michie, S. Muggleton. K. Furakawa. Oxford University Press, (in press).

[Kirkwood *et al.*, 1989] C. Kirkwood, B. Andrews, and P. Mowforth. 1989. Automatic detection of gait events: a case study using inductive learning techniques. *Journal of biomedical engineering*, 11(23) 511 – 516.

[Kressel, 1991] U. Kressel. 1991. The impact of the learning set size in handwritten digit recognition. *Proc. of the ICANN.* Helsinki.

[Lauritzen and Spiegelhalter, 1988] S. L. Lauritzen and D. J. Spiegelhalter. 1988. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, 50 157–224.

[McClelland *et al.*, 1986] J. L. McClelland, D. E. Rumelhart, and G.E. Hinton. 1986. *Parallel distributed processing: explorations in the microstructure of cognition. Volume I, II and III.* MIT Press, Cambridge, MA.

[Meyer-Broetz and Schuermann, 1970a] G. Meyer-Broetz and J. Schuermann. 1970. *Methoden der automatischen Zeichenerkennung.* Akademie Verlag, Berlin.

[Michalski, 1983] R.S. Michalski. 1983. A theory and methodology of inductive learning. *Artificial Intelligence*, 20(2) 111–161.

[Michie *et al.*, 1994] D. Michie, D.J. Spiegelhalter, and C.C. Taylor. 1994. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, Hemel Hempstead, UK.

[Mooney *et al.*, 1989] R. Mooney, J. Shavlik, G. Towell, and A. Gove. An experimental comparison of symbolic and connectionist learning algorithms (vol 1). *IJCAI 89: proceedings of the eleventh international joint conference on artificial intelligence. 20-25 August 1989, Detroit, MI.*, pp 775 – 780, San Mateo, CA. Morgan Kaufmann.

[Pearl, 1988] J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems.* Morgan and Kauffman.

[Poggio and Girosi, 1990] T. Poggio and F. Girosi. 1991. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9, September):1481 – 1497.

[Preparata and Shamos, 1985] F. P. Preparata and M. I. Shamos. 1985. *Computational Geometry: an introduction.* Springer-Verlag, New York.

[Quinlan *et al.*, 1986] J. R. Quinlan, P. J. Compton, K. A. Horn, and L. Lazarus. 1986. Inductive knowledge acquisition: a case study. *Proceedings of the Second Australian Conference on applications of expert systems, Sydney, 14-16 May 1986*, pp 183 – 204, Sydney. New South Wales Institute of Technology.

[Quinlan, 1986] J. Quinlan. 1986. Induction of decision trees. *Machine Learning*, 1(1) 81 – 106.

[Quinlan, 1987a] J. R. Quinlan. 1987. Simplifying decision trees. *International journal of man-machine studies*, 27(3) 221 – 234.

[Quinlan, 1987b] J.R. Quinlan. 1987. Generating production rules from decision trees. *International Joint Conference on Artificial Intelligence*, pp 304–307, Milan.

[Quinlan, 1992] J. R. Quinlan. 1992. Learning with continuous classes. *Proc. of the 1992 Australian Joint Conference on AI*, Singapore. World Scientific.

[Read and Cressie, 1988] T. R. C. Read and N. Cressie. 1988. *Goodness of Fit Statistics for Discrete Multivariate Data.* Springer Verlag, New York.

[Remme *et al.*, 1980] J. Remme, J.D.F. Habbema, and J. Hermans. 1980. A simulative comparison of linear, quadratic and kernel discrimination. *J. Statist. Comput. Simul.*, 11 87–106.

[Renals and Rohwer, 1989] S. Renals and R. Rohwer. 1989. Phoneme classification experiments using radial basis functions. *IJCNN-89: proceedings of the international joint conference on neural networks. Washington, D.C., USA. 1989. [Volume 1]*, pp 461 – 467. IEEE.

[Ripley, 1992] B. D. Ripley. 1992. Statistical aspects of neural networks. In *SemSat, Sandbjerg, Denmark, April.* Chapman and Hall.

[Rivest, 1987] R.L. Rivest. 1987. Learning decision lists. *Machine Learning*, 2(3) 229–246.

[Rumelhart *et al.*, 1986] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. 1986. Learning representations by back-propagating errors. In *Neurocomputing: foundations of research*, pp 696 – 699. MIT Press, Cambridge, MA.

[Sammut, 1988] C. Sammut. 1988. Experimental results from an evaluation of algorithms that learn to control dynamic systems. In *Proceedings of the fifth international conference on machine learning. Ann Arbor, Michigan. June 12-14, 1988*, pp 437 – 443, San Mateo, CA, 1988. Morgan Kaufmann.

[Segre(ed.), 1989] A.M. Segre(ed.). 1989. *Proceedings of the Sixth International Machine Learning Workshop.* Morgan Kaufman, San Mateo, CA, Cornell, New York.

[Sethi and Otten, 1990] I. K. Sethi and M. Otten. 1990. Comparison between entropy net and decision tree classifiers. In *IJCNN-90: proceedings of the international joint conference on neural networks.*, pp 63 – 68, Ann Arbor, MI. IEEE Neural Networks Council.

[Shavlik *et al.*, 1991] J. W. Shavlik, R. J. Mooney, and G. G. Towell. 1991. Symbolic and neural learning algorithms: an experimental comparison. *Journal of Machine learning*, 6 111 – 143.

[Siebert, 1987] J. P. Siebert. 1987. Vehicle recognition using rule-based methods. *Tirm-87-018*, The Turing Institute, Glasgow, UK.

[Smith *et al.*, 1991] J.W. Smith, J.E. Everhart, W.C. Dickson, W.C. Knowler, and R.S. Johannes. 1991. Using the adap learning algorithm to forecast the onset of diabetes mellitus. *Proceedings of the Symposium on Computer Applications and Medical Care*, pp 261 –265, U.S.A. IEEE Computer Society Press.

[Spiegelhalter, 1990] D. J. Spiegelhalter. 1990. Probabilistic methods and the interface with statistics. In *Colloquium on reasoning and uncertainty organised by professional group C4 (Artificial Intelligence).* IEE Computing and Control Division.

[Spikovska and Reid, 1990] L. Spikovska and M. B. Reid. 1990. An empirical comparison of id3 and honns for distortion invariant object recognition. *TAI-90: tools for artificial intelligence: proceedings of the 2nd international IEEE conference*, Los Alamitos, CA. IEEE Computer Society Press.

[Stone, 1974] M. Stone. 1974. Cross-validatory choice and assessment of statistical predictions. *Journal of Royal Statistics Society*, B 36 111 –133.

[Sutherland *et al.*, 1992] A. Sutherland, B. Henery, and C. Feng. 1992. Statlog - an esprit project for the comparison of statistical and logical learning algorithms. *Conference on New Techniques and Technology in Statistics*, Germany. Bonn.

[Thrun *et al.*, 1991] S. B. Thrun, T. Mitchell, and J. Cheng. 1991. The monk's comparison of learning algorithms - introduction and survey. *The MONK's problems - a performance comparison of different learning algorithms*, eds. S. Thrun, J. Bala, E. Bloedorn, and I. Bratko pp 1–6. Carnegie Mellon University,Computer Science Department, 1991.

[Titterington *et al.*, 1981] D.M. Titterington, G.D. Murray, L.S. Murray, D.J.Spiegelhalter, A.M. Skene, J.D.F. Habbema, and G.J.Gelpke. 1981. Comparison of discrimination techniques applied to a complex data set of head injured patuents. *J. Royal Statist. Soc.*, 144 145–175, 1981.

[Tsaptsinos *et al.*, 1990] D. Tsaptsinos, A. Mirzai, and B. Jervis. Comparison of machine learning paradigms in a classification task. *Applications of artificial intelligence in engineering V: proceedings of the fifth international conference*, Berlin, 1990. Springer-Verlag.

[Unger and Wysotzki, 1981] S. Unger and F. Wysotzki. 1981. *Methoden der automatischen Zeichenerkennung.* Akademie Verlag, Berlin.

[Weiss and Kapouleas, 1989] S. M. Weiss and I. Kapouleas. 1989. An empirical comparison of pattern recognition, neural nets and machine learning classification methods (vol 1). *IJCAI 89. 20-25 August 1989, Detroit, MI.*, pp 781 – 787, San Mateo, CA. Morgan Kaufmann.

[Weiss and Kulikowski, 1991] S. M. Weiss and C. A. Kulikowski. 1991. *Computer systems that learn: classification and prediction methods from statistics, neural networks, machine learning, and expert systems.* Morgan Kaufmann, San Mateo, CA.

[Weiss *et al.*, 1990] S. M. Weiss, R. S. Galen, and P. V. Tadepalli. 1990. Maximizing the predictive value of production rules. *Artificial intelligence*, 45 47 – 71.

[Xu *et al.*, 1991] L. Xu, A. Krzyzak, and E. Oja. 1991. Neural nets for dual subspace pattern recognition method. *International Journal of Neural Systems*, 2 169 – 184.

	Satellite	Digits	KL	Vehicle	Head	Heart	Credit	Shuttle	Belgian	Segment	Diabetes	German
Ex. No.	6435	18000	18000	846	900	270	8900	58000	2500	2310	768	1000
Strategy	tt	tt	tt	9x	9x	9x	tt	tt	tt	10x	12x	10x
Train No.	4435	9000	9000	752	800	240	6230	43500	1250	2079	704	900
Test No.	2000	9000	9000	94	100	30	2670	14500	1250	231	64	100
Att. No.	36	16	40	18	6	13	16	9	21	11	8	20
Cat.Att.	0	0	0	0	1	5	8	0	0	0	0	13
Classes	6	10	10	4	3	2	2	7	2	7	2	2
SD_ratio	1.2970	1.5673	1.9657	1.5392	1.1231	1.0612	1.0273	1.6067	1.5124	4.0014	1.0377	1.0369
ρ	0.5977	0.2119	0.1093	0.4828	0.1217	0.1236	0.0825	0.3558	0.3503	0.1425	0.1439	0.0848
$cancor_1$	0.9366	0.8929	0.9207	0.8420	0.7176	0.7384	0.7618	0.9668	0.8869	0.9760	0.5507	0.5044
$cancor_2$	0.9332	0.8902	0.9056	0.8189	0.1057	0.0000	0.0000	0.6968	0.0001	0.9623	0.0000	0.0000
$cancor_3$	0.7890	0.7855	0.8440	0.3605	0.0000	NA	NA	0.2172	NA	0.8345	NA	NA
$cancor_4$	0.2385	0.6982	0.7761	0.0000	NA	NA	NA	0.1458	NA	0.5878	NA	NA
$fract_1$	0.3586	0.2031	0.1720	0.4696	0.9787	1.0000	1.0000	0.6252	1.0000	0.3098	1.0000	1.0000
$fract_2$	0.7146	0.4049	0.3385	0.9139	1.0000	1.0000	1.0000	0.9499	1.0000	0.6110	1.0000	1.0000
$fract_3$	0.9691	0.5621	0.4830	1.0000	1.0000	NA	NA	0.9814	NA	0.8375	NA	NA
$fract_4$	0.9923	0.6862	0.6053	1.0000	NA	NA	NA	0.9957	NA	0.9499	NA	NA
skew	0.7316	0.8562	0.1802	0.8282	1.0071	0.9560	1.2082	4.4371	0.4334	2.9580	1.0586	1.6986
kurtosis	4.1737	5.1256	2.9200	5.1800	5.0408	3.6494	4.4046	160.3108	2.6581	24.4813	5.8270	7.7943

Table 1: Statistical characteristics of datasets: the number of examples (Ex. No.), the resampling strategy (Strategy) (tt = train and test, 9x = 9 fold cross-validation), the number of attributes (Att. No.), the number of binary/categorical attributes (Cat. Att.), the number of classes (Classes), the homogeneity of covarience (SD_ratio), ρ mean absolute correlation coefficient (ρ), cannonical discriminant correlation (cancor), variation explained by first four canonical discriminants (fract), average skew (skew), and average kurtosis (kurtosis).

Algorithm	*Accuracy(%)*		*Time(sec.)*	
	Train	Test	Train	Test
k-N-N	91.1	90.6±0.65	2105	944
RBF	88.9	87.9±0.73	723	74
Alloc80	96.4	86.8±0.76	63840	28757
INDCART	98.9	86.3±0.77	2109	9
CART	NA	86.2±0.77	348	14
Backprop@	88.8	86.1±0.77	54371	39
NewID	93.3	85.0±0.80	296	53
C4.5	95.7	84.9±0.80	449	11
CN2	98.6	84.8±0.80	1718	16
Quadra	89.4	84.7±0.80	276	93
Cal5!	87.8	84.6±0.81	1345	13
AC^2	NA	84.3±0.81	8244	17403
SMART	87.7	84.1±0.82	83068	20
LogReg	88.1	83.1±0.84	4414	41
Discrim	85.1	82.9±0.84	68	12
ITrule	NA	81.2±0.87	253*	NA
CASTLE	81.4	80.6±0.88	75	80
Bayes	71.3	69.3±1.00	56	12
Default	24.2	23.0	–	–

Table 2: Satellite image result. Backprop (marked by "@") was terminated after 20 hours of CPU time (ten hidden nodes were used). Cal5 (marked by "!") used a newer version (Cal5.2).

Algorithm	*Accuracy(%)*		*Time(sec.)*	
	Train	Test	Train	Test
k-N-N	98.4	95.3±0.22	2231	2039
Quadra	94.8	94.6±0.24	194	152
Alloc80	93.4	93.2±0.27	3250	134370
Backprop	92.8	92.0±0.29	28910	110
RBF	92.0	91.7±0.29	1150	250
LogReg	92.1	91.4±0.30	5110	138
SMART	90.4	89.6±0.32	51435	33
Discrim	89.0	88.6±0.34	65	30
CN2	99.9	86.6±0.36	2229	78
NewID	91.9	85.5±0.37	516	80
C4.5	95.9	85.1±0.38	778	60
INDCART	98.9	84.6±0.38	3615	51
AC^2	NA	84.5±0.38	32965	22384
CASTLE	82.5	82.1±0.40	4341	4090
ITrule	NA	77.8±0.44	8283*	NA
Bayes	78.0	76.7±0.45	104	62
Cal5	78.5	71.5±0.48	570	55
Default	10.0	10.0	–	–

Table 3: Handwritten digits results

Algorithm	*Accuracy(%)*		*Time(sec.)*	
	Train	Test	Train	Test
k-N-N	100.0	98.0±0.15	0	13881
Alloc80	100.0	97.6±0.16	48106	48188
Quadra	98.4	97.5±0.16	1990	1648
Backprop	95.9	95.1±0.23	129840	240
LogReg	96.8	94.9±0.23	3538	1713
RBF	95.2	94.5±0.24	2280	580
SMART	95.7	94.3±0.24	389448	58
Discrim	93.0	92.5±0.28	141	54
CASTLE	87.4	86.5±0.36	49162	45403
NewID	100.0	83.8±0.39	785	109
AC^2	100.0	83.2±0.39	27382	24791
INDCART	99.7	83.2±0.39	3550	53
CN2	96.4	82.0±0.40	9183	103
C4.5	95.0	82.0±0.40	1437	35
ITrule	NA	78.4±0.43	8175*	NA
Bayes	79.5	77.7±0.44	141	76
Cal5	75.2	66.9±0.50	3049	64
Default	10.0	10.0	–	–

Table 4: Karhunen-Loeve digits results

Algorithm	*Accuracy(%)*		*Time(sec.)*	
	Train	Test	Train	Test
Quadra	91.5	85.0±3.68	251	29
Alloc80	100.0	82.7±3.90	30	10
LogReg	83.3	80.9±4.05	758	8
Backprop	83.2	79.3±4.18	14411	4
Discrim	79.8	78.4±4.24	16	3
SMART	93.8	78.3±4.25	3017	1
C4.5	93.5	73.4±4.56	153	1
k-N-N	100.0	72.5±4.61	164	23
CART	NA	71.6±4.65	25	1
AC^2	NA	70.3±4.71	595	23
NewID	97.0	70.2±4.72	18	1
INDCART	95.3	70.2±4.72	85	1
RBF	90.28	69.3±4.76	1736	12
CN2	98.2	68.6±4.79	100	1
ITrule	NA	67.6±4.83	985*	NA
Cal5	70.3	64.9±4.92	41	1
CASTLE	49.5	45.5±5.14	23	3
Bayes	48.1	44.2±5.12	4	1
Default	23.2	23.2	–	–

Table 5: Vehicle silhouettes results

Algorithm	*Accuracy(%)*		*Time(sec.)*	
	Train	Test	Train	Test
Alloc80	96.8	97.0±1.12	1248	279
AC^2	100.0	96.9±1.14	3198	84
NewID	100.0	96.6±1.19	64	69
CART	99.5	96.0±1.29	139	4
C4.5	98.7	96.0±1.29	142	93
CN2	99.7	95.8±1.32	114	3
INDCART	98.8	95.5±1.36	248	234
SMART	96.1	94.8±1.46	16362	1
Cal5#	95.8	93.8±1.59	259*	NA
k-N-N	100.0	92.3±1.75	5	28
LogReg	90.3	89.1±2.05	302	8
CASTLE	98.9	88.8±2.07	377	31
Discrim	88.8	88.4±2.11	74	7
Quadra	84.5	84.3±2.39	50	16
Bayes	74.0	73.5±2.90	92	3
Default	14.3	14.3	–	–

Table 6: Segment data results (Cal5, marked by "#", uses the newer Cal5.2.)

Algorithm	*Accuracy(%)*		*Time(sec.)*	
	Train	Test	Train	Test
INDCART	92.0	92.0±0.53	206	193
C4.5	94.3	91.2±0.53	1086	11
SMART	89.5	89.1±0.60	2151	5
CASTLE	88.3	88.1±0.63	81	33
Cal5	89.5	87.4±0.64	76	12
CART	87.2	87.0±0.65	19	19
NewID	100	87.0±0.65	380	4
Discrim	87.2	87.0±0.65	71	16
AC^2	100	87.0±0.65	7970	410
RBF	87.5	87.0±0.65	837	54
LogReg	87.3	86.9±0.65	251	30
Backprop	88.2	86.9±0.65	28819	19
CN2	100	86.7±0.66	2309	13
Quadra	86.3	86.0±0.67	78	20
Bayes	85.0	84.4±0.70	44	8
ITrule	83.9	83.3±0.72	773*	NA
Alloc80	97.7	83.0±0.72	24	738
k-N-N	100.0	80.6±0.77	0	1851
Default	50.1	50.0	–	–

Table 7: Credit risk (UK) results

Algorithm	*Accuracy(%)*		*Time(sec.)*	
	Train	Test	Train	Test
SMART	99.7	99.4±0.22	5853	12
LogReg#	99.8	99.3±0.24	103	27
Discrim	97.8	97.5±0.44	46	28
Cal5	97.5	97.1±0.47	24	13
CN2	100.0	96.8±0.50	272	17
NewID	100.0	96.7±0.51	32	19
CART	99.1	96.6±0.51	280	18
INDCART	99.3	96.6±0.51	151	150
AC^2	100.0	96.6±0.51	2712	70
Alloc80	97.4	95.6±0.58	3676*	0
C4.5	98.8	95.5±0.59	66	12
CASTLE	97.1	95.3±0.60	477	199
Quadra	96.4	94.8±0.63	85	41
k-N-N	100.0	94.1±0.67	0	137
Bayes	95.4	93.8±0.68	30	14
Default	63.7	63.8	–	–

Table 8: Belgian data results

Algorithm	*Accuracy(%)*		*Time(sec.)*	
	Train	Test	Train	Test
NewID	100	99.99±0.008	6180*	NA
CN2	100.0	99.97±0.014	11160*	NA
C4.5@	99.90	99.96±0.017	11131	11
INDCART@	99.96	99.92±0.023	1152	16
AC^2@	100.0	99.68±0.047	4493	3397
Cal5	NA	99.60±0.052	552	18
k-N-N	99.61	99.56±0.055	65270	21698
SMART	99.39	99.41±0.064	110010	93
Alloc80	99.05	99.17±0.075	55215	18333
CASTLE	96.34	96.23±0.158	819	263
LogReg@	96.06	96.17±0.159	6946	106
Bayes@	95.42	95.45±0.173	1030	22
Discrim@	95.02	95.17±0.178	508	102
Backprop	95.10	95.10±0.179	28800	75
Quadra@	93.65	93.28±0.208	709	177
Default	78.41	79.16	–	–

Table 9: Shuttle control data results. The algorithms marked by "@" indicates that only a sample of the training data was used (because some algorithms could not use the full set with given machine resources): 32760 for C4.5 (43500 for the full training data); 4351 for AC^2; 32625 for Bayes and INDCART; and 20000 for Discrim, Quadra, and LogReg.

Algorithm	*Accuracy(%)*		*Time(sec.)*	
	Train	Test	Train	Test
LogReg	78.1	77.73±5.20	31	7
Discrim	78.0	77.47±5.22	27	6
SMART	82.3	76.82±5.28	314*	NA
Cal5#	76.8	75.00±5.41	40	1
CART	77.3	74.48±5.45	61	2
CASTLE	74.0	74.22±5.47	29	4
Bayes	76.1	73.83±5.49	2	1
Quadra	76.3	73.83±5.49	24	6
C4.5	86.9	73.05±5.55	12	1
INDCART	92.1	72.92±5.55	18	17
AC^2	100.0	72.40±5.59	648	29
CN2	99.0	71.09±5.67	38	3
NewID	100.0	71.09±5.67	10	10
Alloc80@	71.2	69.92±5.73	115*	NA
k-N-N	100.0	67.58±5.85	1	2
Default	65.10	65.10	–	–

Table 10: Diabetes data results (Alloc80, marked by "@", achieved its best results using only attributes 2, 8, 6, and 7. Cal5, marked by "#", used the newer version of Cal5.2)

Algorithm	*Av. Cost*		*Time(sec.)*	
	Train	Test	Train	Test
Bayes	0.351	0.374±1.50	6	3
Discrim	0.315	0.393±1.06	14	3
LogReg	0.271	0.396±1.00	128	7
Alloc80	0.394	0.407±1.03	31	5
Quadra	0.274	0.422±1.10	60	16
CASTLE	0.374	0.441±1.07	16	3
CART	0.463	0.452±0.69	7	1
k-N-N	0	0.478±0.50	0	1
SMART	0.264	0.478±1.03	725	1
ITrule	NA	0.515±1.37	5*	NA
Default	0.556	0.556	–	–
Cal5	0.517	0.559±1.40	8*	NA
Backprop	0.381	0.574±1.50	128	13
INDCART	0.261	0.630±1.47	8	1
AC^2	0	0.744±1.65	250*	NA
CN2	0.206	0.767±1.73	25	5
C4.5	0.439	0.781±1.71	34	1
RBF	0.303	0.781±1.73	26	4
NewID	0	0.844±1.78	12*	NA

Table 11: Heart disease results

Algorithm	*Av. Cost*		*Time(sec.)*	
	Train	Test	Train	Test
LogReg	16.6	18.0±40.69	736	7
Discrim	19.8	19.9±40.90	28	3
Quadra	17.8	20.1±48.38	253	32
CART	19.8	20.4±33.72	20	1
CASTLE	18.9	20.9±53.71	30	3
Backprop	18.2	21.5±48.30	656	32
SMART	13.6	21.8±58.90	420	4
Bayes	23.6	25.0±55.14	2	1
INDCART	21.9	29.3±72.88	56	1
k-N-N	9.2	35.3±94.56	9	11
ITrule	NA	37.6±137.5	7*	NA
Cal5	32.3	38.4±107.9	5	1
Alloc80	45.3	46.1±158.2	322	276
Default	47.3	47.3	–	–
NewID	18.9	53.6±166.5	16	2
RBF	53.4	63.1±191.8	17	5
C4.5	59.8	82.0±213.5	49	1

Table 12: Head injury results

Algorithm	*Av. Cost*		*Time(sec.)*	
	Train	Test	Train	Test
Discrim	0.509	0.535±1.03	50	7
LogReg	0.499	0.538±1.02	56	7
Alloc80@	0.597	0.584±1.16	912*	NA
SMART	0.389	0.601±1.13	489*	NA
CART	0.581	0.613±1.17	114#	1#
Quadra	0.431	0.619±1.23	53	8
k-N-N	0.0	0.694±1.28	2	9
Default	0.70	0.70	–	–
Bayes	0.600	0.703±1.55	5	1
INDCART	0.069	0.761±1.61	59	44
CN2	0.0	0.856±1.72	117	3
AC^2	0.0	0.878±1.74	1701	41
NewID	0.0	0.925±1.77	13	15
C4.5	0.640	0.985±1.89	14	1
CASTLE	1.276	1.340±2.16	182	12

Table 13: German credit data results (Alloc80, marked by "@", achieved its best results using attributes 1, 3, and 13. CART's times, marked by "#", was quoted for a Solbourne SPARC clone)