Characterizing the Applicability of Classification Algorithms Using Meta-Level Learning

Pavel Brazdil [1]
João Gama [1]
Bob Henery [2]

[1] LIACC, University of Porto, Rua Campo Alegre 823, 4100 Porto, Portugal.
email: pbrazdil, jgama @ncc.up.pt.

[2] Dept. of Statistics, University of Strathclyde, Livingston Tower, 26 Richmond Street, Glasgow, UK. email: r.j.henery @strathclyde.ac.uk.

Abstract. This paper is concerned with a comparative study of different machine learning, statistical and neural algorithms and an automatic analysis of test results. It is shown that machine learning methods themselves can be used in organizing this knowledge. Various datasets can be characterized using different statistical and information theoretic measures. These together with the test results can be used by a ML system to generate a set of rules which could also be altered or edited by the user. The system can be applied to a new dataset to provide the user with a set of recommendations concerning the suitability of different algorithms and these are graded by an appropriate information score. The experiments with the implemented system indicate that the method is viable and useful.

1 Introduction

Project StatLog carried out perhaps the most comprehensive comparative study of different machine learning, neural and statistical classification (D.Michie, et al., 1994). About 22 different algorithms were evaluated on more than 20 different datasets of industrial interest. It is interesting that no particular algorithm could be considered 'best' when considering the error rates. With some datasets one particular algorithms could be better than other algorithms, but with other datasets this could easily be the other way round. As there was no algorithm which could be considered best overall, a question arises whether one could adopt a method that would identify the promising algorithm(s) on the basis of the existing test results. The purpose of this paper is to show how we can do that.

The aim of our work is to obtain a set of rules characterizing the applicability of different algorithms. It appears that datasets can be characterized using certain features such as number of attributes, proportion of binary, categorical or numeric attributes, unknown values etc. In addition, we can use other more complex statistical or information theoretic measures. It is reasonable to try to match the features of datasets with our knowledge concerning the performance of algorithms. If we select the algorithm that most closely matches the features of the dataset on which the algorithm performed well, then we increase the chances of obtaining useful results. The advantage is that not all algorithms need to be

tried out. Those algorithms that do not match the data can be excluded, and so, a great deal of computing effort effort can be saved.

In order to achieve this aim, we need to determine which dataset features are relevant. After that, various instances of learning tasks can be examined with the aim of formulating a 'theory' concerning the applicability of different machine learning and statistical algorithms. The process of constructing such a theory can be considered as a kind of *meta-level learning*. The knowledge concerning as to which algorithm is applicable can be summarized in the form of rules of the form: If the given dataset has characteristics C1..Cn, then classification algorithm A is (may be) applicable.

Each rule can in addition be qualified using certain statistical measures, such as estimates of correctness. For reasons explained later we prefer to use estimates of how informative each rule is. Rules concerning applicability of algorithms can be constructed manually, or with the help of some (semi-)automatic method. In this paper we are concerned with the application of machine learning methods to this problem.

Previous work on comparative studies has usually considered only a few algorithms (e.g. neural networks and decision trees) and these were done usually on few datasets only (a comprehensive review of previous work appears in D.Michie, et al. (1994)). As the number of tests was generally limited, few people have attempted to automate the formulation of a theory concerning the applicability of different algorithms. One exception was the work of Aha (1992) who represented this knowledge using rule schemata. One example of such a rule schema is:

```
If  (# training instances < 737 )     AND
    (# prototypes per class > 5.5 )  AND
    (# relevant attributes > 8.5 )
Then IB1 >> C4
```

where IB1 >> C4 means that algorithm IB1 is predicted to have significantly higher accuracies than algorithm C4. Our approach differs from Aha's in that we are not concerned with just a comparison between two algorithms, but rather a group of algorithms. This task is more challenging than a set of pairwise comparisons. Also, out rules are suitably quantified and so our recommendations are more refined.

There is, however, an alternative solution to the problem of how we can go about selecting a suitable classification algorithm. Instead of attempting to *predict* which classification algorithm would perform well, we can try out all different algorithms and examine the outcome. This approach has been adopted by Shaffer (1993) who compared three different classification algorithms - C4.5, C4.5rules and back propagation. The algorithms were evaluated on five different datasets using cross-validation (CV). The objective of the method is to consider each dataset in turn, and in each case select that algorithm that appears to give the highest success rate. In this context CV could be considered as another (more complex) classification algorithm. The experimental results have shown that CV is superior to any of the individual algorithms. This method has, of course, the disadvantage that it requires a lot of processing time before the final answer can be given. For some classification algorithms the learning times may be of the order of hours, particularly if we use large industrial datasets (such as the ones used in StatLog). Besides, we have to have all classification algorithms available and the expertise to use them. Our approach avoids these difficulties.

Organization of this Paper

This paper is organized as follows. Sections 2 describes a set of basic dataset measures used to characterize datasets. Section 3 describes the preprocessing of test results, and in particular, the criteria used to determine whether a particular algorithm can be considered applicable or non-applicable. This section describes also how a particular machine learning algorithm (C4.5) can be used to generate a set of rules on the basis of the available data (dataset measures and classified test results). Section 4 is concerned with the issue of assessing the informativity of the individual rules generated. Section 5 gives an overview of the results and analysis of some of the rules generated. Section 6 shows how the system can provide an advice concerning which algorithm is most suitable for a given dataset. In this section we discuss also the correlation between the advice and the actual performance.

2 Characteristics of Datasets

Our overall aim is to be able to match the features of datasets with our past knowledge concerning the algorithms. In order to achieve this, we need to determine which dataset features are relevant. In this section we will briefly describe some of the features used in this study. The features can be divided into simple measures, statistical measures and information based measures. The simple measures include:

N	Number of examples in the dataset.
p	Number of attributes.
k	Number of classes.
Bin_att	Proportion of binary attributes.
Cat_att	Proportion of categorical attributes.
Un_attrib	Proportion of unknown attributes.
Cost	Cost matrix indicator (it is equal to 1 if costs are used and 0 otherwise)

As for the statistical measures, we have adopted those suggested by R.Henery and C.Taylor (Michie D.et al, 1994). These measures include:

SD_ratio	Standard deviation ratio. Geometric mean ratio of standard deviations of the individual populations to the pooled standard deviation.
Corr.abs	Mean value of correlations between attributes (mean over all pairs of attributes and all populations).
Cancor1	Canonical correlation for the best single linear combination of attributes that discriminates between populations.
Fract1	The first eigenvalue of canonical discriminant matrix divided by the sum of all eigenvalues.
Skewness	Mean of $\mid E(X-\mu)^3 \mid / \sigma^3$.
Kurtosis	Mean of $\mid E(X-\mu)^4 \mid / \sigma^4$.

where $E(X-\mu)$ represents a moment, μ mean and σ standard deviation. The information based measures include:

H(A)	Mean entropy (complexity) of attributes.
H(C)	Entropy (complexity) of class.
M(C,A)	Mean mutual information of class and attributes.
EN.attr	Equivalent number of attributes H(C) / M(C,A)

Each dataset is then simply described using a vector of numeric values.

3 Using Test Results as Data in Meta-level Learning

The test results obtained during the StatLog project were used as raw data which was preprocessed in a particular way. The results for each dataset were analyzed with the objective of determining which algorithms achieved *low error rates* (or *costs*). All algorithms with low error rates were considered *applicable* to this dataset. The other algorithms were considered *inapplicable*.

This categorization of the test results can be seen as preparatory step for the metalevel learning task. Of course, the categorization will permit us also to make predictions regarding which algorithms are applicable on a new dataset.

Of course, the question whether the error rate is *high* or *low* is rather relative. The error rate of 15% may be excellent in some domains, while 5% may be bad in others. This problem is resolved using a method similar to *subset selection* in statistics. First, the best algorithm is identified according to the error rates. Then an acceptable margin of tolerance is calculated. All algorithms whose error rates fall within this margin are considered *applicable*, while the others are labeled as *inapplicable*.

The level of tolerance can reasonably be defined in terms of the standard deviation of the error rate, but since each algorithm achieves a different error rate, the appropriate standard deviation will vary across algorithms. The standard deviation can be estimated by calculating the error margin

$$EM = \mathrm{sqrt}(ER * (1 - ER) / NT),$$

where *ER* represents the error rate of the 'best' algorithm, *NT* the number of examples in the test set. Then all algorithms whose error rates fall within the interval <ER, ER + k*EM> are considered *applicable*. Of course we still need to choose a value for k which determines the size of the interval. This affects the confidence that the truly best algorithm appears in the group considered. The larger the value of k, the higher the confidence that the best algorithm will be in this interval. As a rough guide, a value of k=3 corresponds to a 95% confidence level.

For example, let us consider the tests on the Segment dataset consisting of 2310 examples. The best algorithm appears to be ALLOC80 with the error rate of 3% (ER=0.03). Then

$$EM = \mathrm{sqrt}(0.03 * (1 - 0.03) / 2310) = 0.0035$$

which is 0.35%. In this example, we can say with high confidence that the best algorithms is in the group with error rates between 3% and k * 0.35%. If k=1, the interval is relatively small - <3%, 3.35%> and includes only two other algorithms (AC2, BayesTree) apart from ALLOC80. All the algorithms that lie in this interval can be considered *applicable* to this dataset, and the others inapplicable. If we enlarge the margin, by considering larger values

of k (e.g. 2, 4, 8 or 16), we get a more relaxed notion of applicability. If k=16, for example, the relatively broad band will include most of the "average" algorithms (see Fig.1).

Algorithm	Error rate	Class (k=16)	Margin	Note
ALLOC80	.030	Appl	0.030	Margin for k=0
AC2	.031	Appl		
BayesTree	.033	Appl		
			0.0335	Margin for k=1
NewID	.034	Appl		
			0.037	Margin for k=2
C4.5	.040	Appl		
CART	.040	Appl		
DIPOL92	.040	Appl		
CN2	.043	Appl		
			0.044	Margin for k=4
IndCART	.045	Appl		
LVQ	.046	Appl		
SMART	.052	Appl		
Backprop	.054	Appl		
			0.058	Margin for k=8
Cal5	.062	Appl		
Kohonen	.067	Appl		
RBF	.069	Appl		
kNN	.077	Appl		
			0.086	Margin for k=16
Logdisc	.109	Non-Appl		
CASTLE	.112	Non-Appl		
Discrim	.116	Non-Appl		
Quadisc	.157	Non-Appl		
Bayes	.265	Non-Appl		
ITrule	.455	Non-Appl		
Default	.900	Non-Appl		

Fig. 1 Test Results on *Segment* dataset and Error Margins

The decision as to where to draw the line (by choosing a value for k) is, of course, rather subjective. In this work we had to consider an additional constraint related to the purpose we had in mind. As our objective is to generate rules concerning applicability of algorithms we have opted for the more relaxed scheme of applicability (k=8 or 16), so as to have enough examples in each class (Appl, Non-Appl). This point will be discussed in more detail later.

Some of the tests results analyzed are not characterized using error rates, but rather *costs*. Consequently the notion of error margin discussed earlier has to be adapted to costs. The *standard error of the mean cost* can be calculated from the confusion matrices (obtained by testing), and the cost matrix[1].

[1] The values obtained on the basis of some tests were:

Dataset	Algorithm	Mean cost	Standard error (of mean cost)
German credit	Discrim	0.525	0.0327

We notice that the categorization of error rates (into Appl, Non-Appl) seems to be a bit unnatural, but it is necessary, given our aims. We require it simply for the next step, that involves meta-level learning. As we use a classification algorithm we need to introduce such categorization. Had we chosen a regression algorithm, this would not be necessary.

Preprocessing of Test Results

The problem of learning was divided into several phases. In each phase all the test results relative to just one particular algorithm (e.g. CART) were joined, while all the other results (relative to other algorithms) were temporarily ignored. The purpose of this strategy was to simplify the class structure. For each algorithm we would have just two classes (Appl and Non-Appl). This strategy worked better than the obvious solution that included all available data for training. For example, when considering CART algorithm and margin of k =16 we get:

CART-Appl,	Satim	CART-Non-Appl,	KL
CART-Appl,	Vehic	CART-Non-Appl,	Dig44
CART-Appl,	Head	CART-Non-Appl,	Chrom
CART-Appl,	Heart	CART-Non-Appl,	Shut
CART-Appl,	Belg	CART-Non-Appl,	Tech
CART-Appl,	Segm	CART-Non-Appl,	CUT
CART-Appl,	Diab	CART-Non-Appl,	Cr.Man
CART-Appl,	Cr.Ger	CART-Non-Appl,	Letter
CART-Appl,	Cr.Aust	CART-Non-Appl,	Simdat
CART-Appl,	DNA		
CART-Appl,	New-Bel		
CART-Appl,	Faults		
CART-Appl,	Tset		

Fig. 2 Classified test results relative to one particular algorithm (CART)

The classified test results are then modified as follows. The dataset name is simply substituted by a vector containing the corresponding dataset characteristics (measures). Values which are not available or missing are simply represented by "?". This extended dataset is then used in the meta-level learning.

Prior Ordering of Algorithms

The classified test results are interesting per se. They enable us to make quick comparisons among different classification algorithms. Obviously, the more often a particular algorithm is classified as *applicable*, the better.

If we consider the classified test results of CART for k=16 shown earlier, we see that this algorithm can be considered applicable in 13 cases and inapplicable in 9. These numbers

---

Heart disease	Discrim	0.415	0.0688
Head injury	Logdiscr	18.644	1.3523

In the experiments reported later the error margin was simply set to the values 0.0327, 0.0688 and 1.3523 respectively, irrespective of the algorithm used.

can be used to estimate the probability P(Ai-Appl) that the algorithm Ai is applicable. Here we prefer to calculate the information associated with this probability, suitably normalized. (The method takes into account the overall probability of algorithms falling into the class "applicable". Full description appears in Section 4).

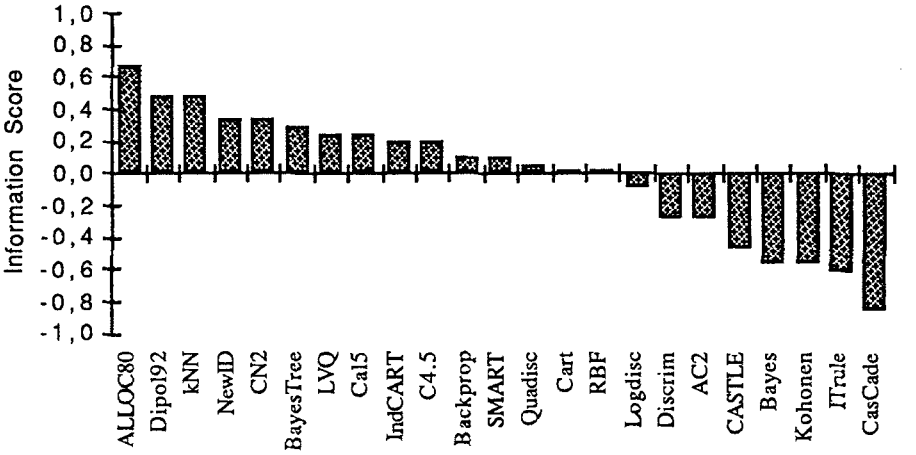

Fig. 3 Information associated with the prior probability of applicability of different classification algorithms tested in Statlog

Fig. 3 shows the information values for all algorithms tested in StatLog. We have decided to show here the mean of two values, one relative to the error margin k=8 and the other relative to k=16. All values are arranged in a descending order. ALLOC80, for example, is the first place in the ordering. The associated information value is around .7 bits. That is, the information that ALLOC80 is applicable is worth about .7 bits. Similarly the information that Dipol92 or kNN is applicable is worth about .5 bits and so on.

If the information is negative, this can be interpreted as an argument that the algorithm is likely to be non-applicable. This does not mean, of course, that we cannot find situations in which the algorithm could be useful. If we do not have any other information, finding such situations may simply be somewhat less likely.

The information values provide us with useful default decisions which need not be followed. Indeed, the purpose of meta-level learning can be viewed as the process of finding useful rules which enable us to make better decisions than those incorporated in these defaults.

Choice of Algorithm for Meta-Level Learning (C4.5)

A question arises as to which algorithm we should use in the process of meta-level learning. We have decided to use C4.5 (Quinlan, 1992) for the following reasons. First, as our results have demonstrated, this algorithm achieves quite good results overall. Secondly,

the decision tree generated by C4.5 can be inspected and analyzed. This is not the case with some statistical and neural learning algorithms.

So, for example, when C4.5 has been supplied with the partial test results relative to CART algorithm, it generated the following decision tree:

```
N > 6435 : Non-Appl (8.0)
N <= 6435:
|  Skew <= 0.57 : Appl (2.0)
|  Skew > 0.57 : Non_Appl (12.0)
```

It has been argued that rules are more legible than trees. The decision tree shown earlier can be transformed into a rule form using a very simple process, where each branch of a tree is simply transcribed as a rule. The applicability of CART can thus be characterized using the following set of rules:

```
CART-Appl      <-- N <= 6435, Skew > 0.57
CART-Non-Appl  <-- N > 6435
CART-Non-Appl  <-- N <= 6435, Skew <= 0.57
```

Quinlan (1992) has argued that rules obtained from decision trees can be improved upon in various ways. For example, it is possible to eliminate conditions that are irrelevant, or even drop entire rules that are irrelevant or incorrect. In addition it is possible to reorder the rules according to certain criteria, introduce a default rule to cover the cases that have not been covered. System C4.5 provides a command that permits to transform a decision tree into a such a rule set. The rules produced by the system are characterized using (pessimistic) error rate estimates.

As is shown in the next section, error rate (or its estimate) is not ideal measure, however. This is particularly evident when dealing with continuous classes. This problem has motivated us to undertake a separate evaluation of all candidate rules and characterize them using a new measure. The aim is to identify those rules that appear to be most informative. The details are given in the following section.

4 Characterizing the Rules Generated

We notice that the amount of data used in generating the rules concerning applicability of one particular algorithm is rather modest. As we used only the StatLog test results, we had only about 22 examples at our disposal (each case represents the results of particular test on a particular dataset). Of these, only a part represented 'positive examples', corresponding to the datasets on which the particular algorithm performed well. Also, the set of dataset descriptors used may not carry too much information. We could thus expect that the rules generated capture a mixture of relevant and fortuitous regularities.

In order to strengthen our confidence in the result the rules generated are evaluated. Our aim is to determine whether the rules could actually be used to make useful predictions concerning its applicability. A rule is considered useful if it appears to make better predictions than the appropriate default rule. The default rule simply states that the algorithm is applicable (with no conditions attached), if the algorithm performs well on relatively many datasets. If the algorithm performs poorly, the default is that the algorithm is non-applicable.

The predictive power of the rules should thus be compared to the default. If a rule appears to be less informative than the default, the default rule should be used instead. All rules (including defaults) are quantified using a measure of how predictive they are. The method of how this is done is given below.

The evaluation of how successful the rules are in predicting the applicability (or non-applicability) of a given algorithm is done using a leave-one-out procedure. Following this procedure *all but one* items is used in training, while the remaining item is used for testing. Of course, the set of rules generated in each pass could be slightly different, but the form of the rules was not our primary interest here.

Let us consider, for example, the problem of predicting applicability of CART. This can be characterized using a confusion matrix, such as the ones shown below, showing results relative to the error margin k=16.

	Appl	Non-Appl
Appl	11	2
Non-Appl	1	8

The rows represent a true class. The columns refer to the predictions made using the rules generated.

The confusion matrix shows that the rules generated were capable of correctly predicting the applicability of CART on a unseen dataset in 11 cases. Incorrect prediction was made only in 1 case. Similarly, if we consider non-applicability, we see that the correct prediction is made in 8 cases, and incorrect one in 2. This gives a rather good overall success rate of 86 %.

Had we decided to use a default rule stating that CART is simply applicable, the number of correct predictions would increase to 13 (11+2), and the number of incorrect ones to 9 (1+8). This gives a success rate of about 59 %. The default stating that CART is inapplicable gives correct predictions in 9 cases, and incorrect one in 13. The corresponding success rate is about 41%.

We notice that success rate is not the ideal measure, however. As the margin of applicability is extended (by making k larger), more cases will get classified as applicable and consequently, the estimates concerning applicability will be easier to make. If we consider an extreme case, when the margin covers all algorithms, we will get an apparent success rate of 100 % (equal to the default). This apparent paradox can be resolved by adopting the measure called *information score* described by Kononenko and Bratko (1991). This measure takes into account prior probabilities.

The information score associated with a definite positive classification[2] is defined as -log P(C), where P(C) represents the prior probability of class C. The information scores can be used to weigh all classifier answers. In our case we have two classes *Appl* and *Non-Appl*.

---

[2] The term 'definite positive classification' is used here to indicate that the classifier affirms that the case belongs to class C with a probability of 1.

The weights can be represented conveniently in the form of a information score matrix as follows:

	Appl	Non-Appl
Appl	- log P(Appl)	- log (1 - P(Appl))
Non-Appl	- log (1 - P(Non-Appl))	- log P(Non-Appl)

The information scores can be used to calculate the total information provided by a rule on the given dataset. This can be done simply by multiplying each element of the confusion matrix by the corresponding element of the information score matrix.

The quantities P(Appl) and P(Non-Appl) can be estimated from the appropriate frequencies. If we consider the frequency of Appl and Non-Appl for *all* algorithms (irrespective of the algorithm in question), we get a kind of absolute reference point. This enables us to make comparisons right across different algorithms.

For example, if we consider tests of 23 algorithms on 22 datasets, we get a dataset consisting of 506 cases (23 x 22). As it happens 307 cases fall into the class *Appl*. The information associated with -logP(Appl) is -log (307 / 506) = 0.721. Similarly, the value of -log P(Non-Appl) is -log (199 /506) = 1.346.

We notice that due to the distribution of data (given by the relatively large margin of applicability of k=16), the examples of applicable cases are relatively common. Consequently, the information concerning applicability has a somewhat smaller weight (.721) than the information concerning non-applicability (1.346).

If we multiply the elements of the confusion matrix for CART by the corresponding elements of the information score matrix we get the following matrix:

	Appl	Non-Appl
Appl	7.93	2.69
Non-Appl	.72	10.77

This matrix is in a way similar to the confusion matrix shown earlier with the exception that the error counts have been weighted by the appropriate information scores. To obtain an estimate of the *average* information relative to *one* case, we need to divide all elements by the number of cases considered (i.e. 22). This way we get:

	Appl	Non-Appl
Appl	.360	.122
Non-Appl	.033	.489

The information provided by classification of *Appl* is .360 - .033 = .327 bits. The information provided by classification of *Non-Appl* is similarly .489 - .122 = .367 bits.

This information obtained in the manner described can be compared to the information provided by the appropriate default rule. If we assume that CART is applicable by default, the corresponding rule will give a correct answer in 13 cases and incorrect one in 9 (we note that as the number of correct answers exceeds the number of errors, this is actually the right default to consider). The number of correct answers and errors can be expressed using a confusion matrix. Then we can calculate the overall information score in a similar way as

before. Each answer needs to be weighed using the appropriate information score. In our case we get (13* -logP(Appl) - 9* -log(1-P(Non-Appl))) / 22 = 0.131 bits.

This information score of the default rule can be compared to the information score provided by the rules. A rule can be considered *useful* if it provides us with more information than the default. If we come back to our example, we see that the classification for Appl provide us with .327 bits, while the default classification provides only .131 bits. This indicates that the rules are more informative than the default for CART. In consequence, the actual rule should be kept and the default rule discarded.

5 Rules Generated in Meta-level Learning

This section contains some rules generated using the method described. As we have not used a uniform notion of applicability throughout, each rule is qualified by additional information. The symbol Appl $\downarrow_k$ represents the concept of applicability derived on the basis of the best error rate. In case of Appl Appl $\downarrow_{16}$ the interval of applicability is <Best error rate, Best error rate + 16 STD's> and the interval of non-applicability is <Best error rate + 16 STD's, 1>.

The set of rules generated includes a number of 'default rules' which can be easily recognized (they do not have any conditions on the right hand side of "←"). Each rule included shows also the normalized information score. This parameter gives an estimate of the usefulness of each rule. Only those rules that could be considered minimally useful (with information score > .200) have been included here. In the implemented system we use a few more rules which are a bit less informative (with inf. scores down to .100).

Decision Tree and Rule Base Algorithms:		Inf. score:
C4.5-Appl $\downarrow_{16}$	←	.477
C4.5-Non-Appl $\downarrow_8$	← k > 2	.226
NewId-Appl $\downarrow_{16}$	←	.609
AC2-Non-Appl $\downarrow_8$	←	.447
CART-Appl $\downarrow_8$	← N <= 4999, Kurtosis > 2.92	.186
CART-Appl $\downarrow_{16}$	← N <= 6435, Skew > 0.57	.328
CART-Non-Appl $\downarrow_8$	← N > 4999	.226
CART-Non-Appl $\downarrow_{16}$	← N > 6435	.367
IndCART-Appl $\downarrow_8$	← Cancor1 <= 0.78	.274
IndCART-Appl $\downarrow_{16}$	←	.384
Cal5-Appl $\downarrow_{16}$	← k <= 7	.524
Cal5-Non-Appl $\downarrow_{16}$	← k > 7	.244
CN2-Appl $\downarrow_{16}$	←	.702
ITRule-Non-Appl $\downarrow_8$	← N > 768	.549
ITRule-Non-Appl $\downarrow_{16}$	← N > 1000	.918
Statistical Algorithms:		Inf. score:
Discrim-Appl $\downarrow_8$	← N <= 1000	.247

Discrim-Non-Appl $\downarrow_8$	← N > 1000	.453
Discrim-Non-Appl $\downarrow_{16}$	← k > 4	.367
QuaDisc-Appl $\downarrow_8$	← N <= 1000	.309
QuaDisc-Appl $\downarrow_{16}$	← Skew <= 2.56, EnAttr > 2.98	.229
QuaDisc-Non-Appl $\downarrow_8$	← N > 1000, Hx <= 5.58	.226
LogDisc-Appl $\downarrow_8$	← N <= 3186	.495
LogDisc-Non-Appl $\downarrow_8$	← N > 3186	.323
LogDisc-Non-Appl $\downarrow_{16}$	← k > 4	.367
Alloc80-Appl $\downarrow_8$	←	.406
Alloc80-Appl $\downarrow_{16}$	←	.797
kNN-Appl $\downarrow_{16}$	←	.766
Smart-Appl $\downarrow_{16}$	← Fract1 > 0.63	.262
Bayes-Non-Appl $\downarrow_8$	←	.418
Bayes-Non-Appl $\downarrow_{16}$	←	.705
BayTree-Appl $\downarrow_{16}$	← k <= 7	.557
BayTree-Non-Appl $\downarrow_{16}$	← k > 7	.305
Castle-Non-Appl $\downarrow_8$	← Cost <= 0, N > 768	.420
Castle-Non-Appl $\downarrow_{16}$	← Bin_att <= 0	.734
Neural Network Algorithms:		Inf. score:
Dipol92-Appl $\downarrow_8$	←	.341
Dipol92-Appl $\downarrow_{16}$	←	.544
Radial-Non-Appl $\downarrow_8$	←	.401
LVQ-Appl $\downarrow_{16}$	←	.498
BackProp-Appl $\downarrow_8$	← N <= 3000	.495
BackProp-Appl $\downarrow_{16}$	← k <= 7, EnAttr > 2.88	.229
BackProp-Non-Appl $\downarrow_8$	← N > 3000, Cancor1 > 0.61	.259
Kohonen-Non-Appl $\downarrow_8$	←	.641
Cascade-Non-Appl $\downarrow_8$	←	.706
Cascade-Non-Appl $\downarrow_{16}$	←	.866

Fig. 4 Some Rules Generated During Meta-Level Learning

The rules presented could be supplemented by another set generated on the basis of the worst error rate (i.e. the error rate associated with the choice of most common class or worse). In case of Appl $\uparrow_{16}$ the interval of applicability is <0, Default error rate - 16 STD's> and the interval of non-applicability is <Default error rate - 16 STD's, 1>.

Discussion

The problem of learning rules for all algorithms simultaneously is formidable. We want to obtain a substantial number rules to qualify each algorithm. In addition, there are perhaps far too many measures given the available data. To limit the complexity of problem we

have considered one algorithm at a time. This facilitated the construction of rules. Considering that the problem is difficult, what confidence can we have that the rules generated are minimally sensible?

One possibility is to try to evaluate the rules, by checking whether they are capable of giving useful predictions. This is what we have done and the method was described earlier. Note that measuring simply the success rate has the disadvantage that it does not distinguish between predictions that are easy to make, and those that are more difficult. This is why we have evaluated the rules by examining how informative they are in general. Our analysis showed that the rules generated can indeed provide us with a useful information. For example, the following rule

Discrim-Appl $\downarrow_8$ $\leftarrow$ N <= 1000

provides us with .247 bits of information, if invoked, which is reasonable. On quick glance the condition "N <= 1000" is a bit puzzling, however. Why should Discrim perform simply well, if the number of examples is less than this number?

One possible answer to this question is that the condition shown is simply fortuitous. The rules could contain some fortuitous conditions, given that they were generated on the basis of relatively few data. Unless we have more data, it is difficult to determine which conditions are really relevant. However, it is necessary to note that each condition, such as ""N <= 1000", should be interpreted contextually. The condition cannot be simply interpreted as "Discrim performs will perform well if such and such condition is satisfied". The correct interpretation is something like - "Discrim is likely to compete well under the conditions stated, provided no other more informative rule applies".

However, the condition seems to make sense in the light of the following additional evidence. Some algorithms have a faster learning rate than others. These algorithms compete well with others provided the number of examples is small. The fast learning algorithms may however be overtaken by others later. The experiments with learning curves on the Satellite Image dataset show that Discriminant algorithm is among the first six algorithms in terms of error rate as long as the number of examples is relatively small (100, 200 etc.). This algorithm seems to quickly pick up what is relevant and so we could say, it competes well under these conditions. When the number of examples is larger, however, Discriminant is overtaken by other algorithms. For example, with 3200 examples Discriminant is in the 15th place in the ranking. This supports the view that the system has 'discovered' a new piece of experimental knowledge from the regularities in the data.

There is of course a well recognized problem that should be tackled. Many conditions contain numeric tests which are either true or false. It does not make sense to consider Discriminant algorithm applicable if the number of examples is less than 1000, and inapplicable, if this number is just a bit higher. Obviously a more flexible approach is needed (e.g. using flexible matching). We note, however, that this problem is somewhat attenuated by the fact *different* error margins are used in the process of rule generation. The rules for CART generated by the system were:

CART-Non-Appl $\downarrow_8$	$\leftarrow$ N > 4999	.226
CART-Non-Appl $\downarrow_{16}$	$\leftarrow$ N > 6435	.367

These rules suggest that there may be a functional dependence between non-applicability and N. It is conceivable that if we used more error margins (e.g. k=4, 8, 12, 16 etc.), we

could get a more precise model of this dependence. So we can get some benefits of flexible matching without further work.

6 Giving Advice Concerning Application

Rules generated in the way described permit us to give recommendations as to which classification algorithm could be used with a given dataset. This is done with the help of a kind of expert system called an *Application Assistant (AplAs)*. This system contains a *knowledge base* containing all the rules shown earlier (the actual rule set includes a few extra rules with lower information scores). The interpreter is quite standard, but uses a particular method for resolution of conflicts.

We notice that the knowledge base may contain potentially conflicting rules. In general several rules may apply, some of which may recommend the use of a particular algorithm while others may be against it. Some people believe that knowledge bases should always be cleaned up so that such situations would not arise. This would amount to obliterating certain potentially useful information and so we prefer to deal with the problem in a different way.

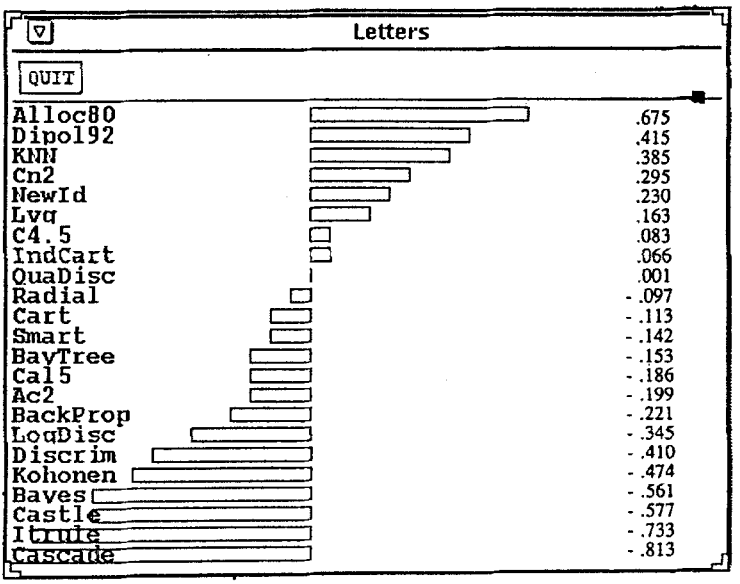

Fig. 5 Recommendations concerning applicability of algorithm (for Letters dataset)

For every algorithm we consider all the rules satisfying the conditions, sum all the information scores and then normalize them. The information scores associated with the recommendation to apply an algorithm are taken with a positive sign, the others with a negative one. The sum of information scores is then normalized. In our case, as we use two margins (k=8 and k=16), the mean is divided by 2. The output of this phase is a list of algorithms ordered by their information scores. A positive score can be interpreted as an

argument to apply the algorithm. A negative score can be interpreted as an argument against the application of the algorithm. Moreover, the higher the score, the more informative is the recommendation in general. The information score can be then considered as a strength of the recommendation. Figure 5 shows the recommendations obtained for the Letters dataset.

The recommendations given are of course not perfect. They do not guarantee that the first algorithm in the recommendation ordering will have the best performance in reality. However, our results demonstrate that the algorithms accompanied by a strong recommendation do perform quite well in general. The opposite is also true. The algorithms that have not been recommended have a poorer performance in general. In other words, we observe that there is a reasonable degree of correlation between the prediction and the actual test results. This is illustrated in Fig. 6 which shows the correlation between the information score and the success rate for one particular dataset (Letters).

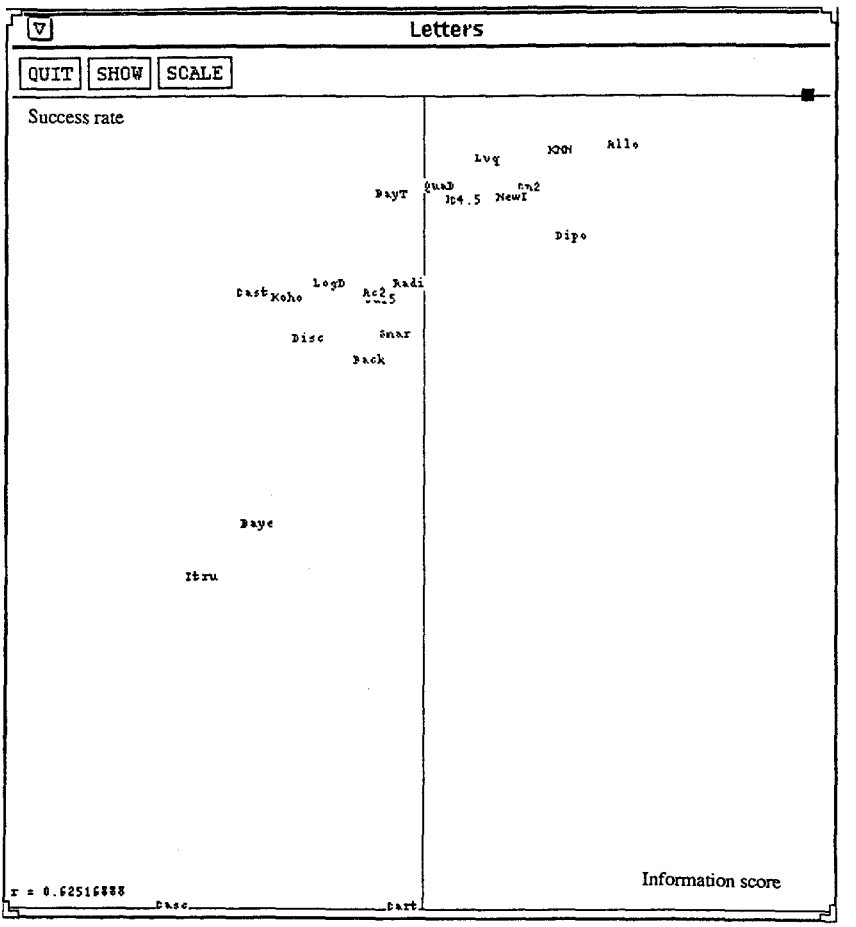

Fig. 6 Correlation between information score and success rate (for Letters dataset)

The top part shows the algorithms with high success rates. The right part shows the algorithms accompanied by a strong recommendation concerning applicability. We notice that several algorithms with high success rates appear there. The algorithm that is most strongly recommended for this dataset is Alloc80 (Inf. score = .601 bits). This algorithm has also the highest success rate of 93.6 %. The algorithms kNN and Dipol92 share the second place in the recommendation ordering. We note that kNN is a very good choice, while Dipol92 is not too bad either.

The correlation between the information score and success rate could, of course, be better. For example, Dipol92 is a bit overvalued, Castle undervalued etc. The correlation could be improved, in the first place, by obtaining more test results. This would give us the opportunity to possibly refine the rule set. It would be beneficial to consider also other potentially useful sets of rules, including the ones generated on the basis of other error margins.

Some Problems and Future Work

Some of the measures used in estimating the performance of algorithms are not too simple to calculate. For example, the programming effort in calculating SD_ratio is greater than that in running the linear discriminant on the available data. Indeed to find SD_ratio requires virtually all the quantities needed in finding the quadratic discriminant. This poses the question: if it is easier to run, say linear discriminants and NewID, why not run them and use the performance of these procedures as yardsticks by which to judge the performance of other algorithms? The similarities evident in the empirical results strongly suggest that the best predictor for logistic regression is linear discriminants (with logistic regression doing that little better on average), and AC2 is very similar to NewID (if there is no hierarchy), and so on. This idea has been followed up and described in (D.Michie, et al., 1994).

There is scope for further work. As almost certainly there are insufficient data to construct reliable rules, it is worth considering an interactive method, capable of incorporating prior expert's knowledge. As one simple example, if it is known that an algorithm can handle cost matrices, this could simply be provided to the system. As another example, the knowledge that the behaviour of NewID and AC2 is likely to be similar could also be useful to the system. The rules for AC2 could be then be constructed from the rules for NewID, by adding suitable conditions concerning e.g. the hierarchical structure of the attributes. Also, some algorithms have in-built checks on applicability, such as linear or quadratic discriminants, and these should be incorporated into the rules constructed by the system.

Despite the fact that there is space for possible improvements, the method is sound in principle and seems to produce very promising results. The user can get a recommendation as to which algorithm could be used with a new dataset. Although the recommendation is not guaranteed to give the best possible advice, it narrows down the user's choice.

Acknowledgements

This work was supported by Esprit II Project StatLog (No.5170). The authors wish to thank Commission of European Communities for this support. Also, we wish to thank the following partners for providing the individual test results:

• Dept. of Statistics, Univ. of Strathclyde, Glasgow, UK; • Dept. of Statistics, Univ. of Leeds, UK; • Aston University, Birmingham, UK; • Forschungszentrum Ulm, Daimler-Benz AG, Germany; • Brainware GmbH, Berlin, Germany; • Fraunhofer Gesellschaft IITB-EPO, Berlin, Germany; • Institut fuer Kybernetik, Bochum, Germany; • ISoft, Gif sur Yvette, France; • Dept. of CS and AI, University of Granada, Spain.

The authors wish to thank Luís Torgo for some corrections, and also, to anonymous referees for their comments.

References

Aha D. (1882): Generalizing from Case Studies: A Case Study, in *Proc. of the Ninth International Workshop on Machine Learning (ML92)*, ed. D.Sleeman and P.Edwards, Morgan Kaufmann.

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

Kononenko I. and Bratko I. (1991): "Information-Based Evaluation Criterion for Classifier's Performance", in *Machine Learning, Vol.6, No. 1*, Kluwer Academic Publishers.

Quinlan R. (1992): *C4.5: Programs for Machine Learning*, Morgan Kaufmann.

Shaffer C. (1993): Selecting a Classification Method by Cross-Validation, in *Machine Learning, Vol.13, No. 1*, Kluwer Academic Publishers.

Appendix

1. Classification Algorithms Used in StatLog

Decision Tree Classifiers:

C4.5	- Inductive Decision Tree
NewID	- New Inductive Decision Tree
AC2	- Decision Trees with Knowledge Acquisition
CART	- Classification and Regression Tree
IndCART	- Classification and Regression Tree
Cal5	- Numeric Decision Tree Classifier

Rule Classifiers:

CN2	- Decision Rule Classifier
ITrule	- Probabilistic Decision Rule Classifier

Classical Statistical Algorithms:

Discrim	- Fisher's Linear Discriminants
Quadisc	- Quadratic Discriminats
Logdisc	- Logistic Discriminants

Non-Parametric Statistical Algorithms:

ALLOC80	- Density Estimation (Kernel Classifier)

kNN	- k-Nearest Neighbour
SMART	- Projection Pursuit (Smooth Additive Regression)
Bayes	- Naive Bayes
BayesTree	- Extension of Naive Bayes
CASTLE	- Causal Networks
DIPOL92	- Discriminate Analysis with Post-Optimisation

Neural Network Classifiers:

RBF	- Radial Basis
LVQ	- Linear Vector Quantizer
Backprop	- Multi Layer Perceptron (Back Propagation)
Kohonen	- Self Organizing Feature Map

Our experiments included one additional algorithm (Cascade) which is officially not included among the algorithms evaluated under StatLog.

2. Datasets Used in StatLog

*	Cr.Aust - Australian credit		Belg - Belgian power
*	Cr.Ger - German credit	*	DNA - DNA sequence
*	Satim - Landsat Satellite image		Tech - Technical
	Dig44 - Handwritten digits (Digits)		Faults - Finance of maintenance
	KL - Karhunen-Loeve Digits		New-Bel - New Belgian power
*	Vehic - Vehicle silhouttes		Tset - Tsetse Fly Distribution
*	Segm - Image Segmentation		CUT - Character Segmentation
	Chrom - Chromosomes		Cr.Man - Credit Management
	Head - Head injury	*	Letter - Letter Recognition
*	Heart - Heart disease		Simdat - Simulated data
*	Shut - Shuttle control		(withdrawn later)
*	Diab - Diabetes of pima-indians		

The dataset marked with "*" were authorized for public distribution and available via ftp (see the next section). The datasets mentioned contain typically several thousands of examples. The largest dataset contains 58000 examples, and the smallest only 270 examples. The datasets are characterized using a varied number of *attributes*. The Australian credit data (Cr.Aust), for example, is characterized using 14 attributes. The number of attributes can be much larger, however. The DNA, Technical and New-Belgian power datasets are characterized by more than 50 different attributes. The number of classes also varies, and is between 2 and 26 for the datasets shown.

3. Support for Further Comparative Testing

LIACC can offer various datasets used in the comparative testing within StatLog as well as some software that has been written during the StatLog project. In particular, LIACC can provide the source code of Evaluation Assistant which can help users to carry out further comparative testing.

General information about this can be obtained from LIACC, University of Porto, from ftp.ncc.up.pt (192.26.239.52), directory pub/statlog, file README. Alternatively, interested parties can contact P.Brazdil or J.Gama, at LIACC, University of Porto, Rua Campo Alegre 823, 4100 Porto, Portugal, Tel.: +351 600 1672, Fax.: +351 600 3654, or by email statlog-adm@ncc.up.pt.

3.1 Datasets

All public domain datasets used in StatLog can be obtained from LIACC, University of Porto, from ftp.ncc.up.pt, directory pub/statlog/datasets. This directory contains several subdirectories, one for each dataset. Each subdirectory contains an associated .doc file with a brief description of the dataset and previous test results on this dataset. Some larger datasets have been split into train and test set (as used in the StatLog project).

The main source of datasets is the UCI Repository of Machine Learning Databases and Domain Theories which is managed by D.W.Aha. Some datasets were processed and the repository mentioned contains both the unprocessed and processed versions. The datasets available from LIACC, contain only the processed datasets. These datasets can also be obtained from University of Strathclyde, via ftp.strath.ac.uk (130.159.248.24), directory Stams/statlog.

3.2 Conducting New Tests with Evaluation Assistant

New tests can be carried out by interested parties with the help of Evaluation Assistant, which is a software tool developed within Project StatLog. Its aim is to facilitate testing of statistical, machine learning and neural algorithms on given datasets and provide standardized performance measures. The Evaluation Assistant is oriented towards classification tasks. Two versions of Evaluation Assistant exist: a command version, and an interactive one.

The command version of Evaluation Assistant consists of a set of basic commands that enable the user to test learning algorithms. This version is implemented as a set of Cshell scripts and C programs. The interactive version of Evaluation Assistant provides an interactive interface that enables the user to set up the basic parameters for testing. The interactive interface is implemented in C and exploits X windows. This version generates a customized version of some scripts which can be examined and modified before execution.

The source code of the Evaluation Assistant is available from LIACC via ftp.ncc.up.pt. The command version is stored in the directory pub/statlog/eac. The source code of the interactive version is stored in the directory pub/statlog/eai. Both versions run on SUN SPARCstation IPC and other compatible workstations.

3.3 Application Assistant

This software prototype analyses previous test results and generates rules concerning applicability of different machine learning, statistical and neural network algorithms. The rules can be used to provide the user with a recommendation concerning which classification method is appropriate for a given dataset.

The rules referred to earlier are generated on the basis of previous test results and dataset characteristics. The semi-automatic analysis of previous test results is done with the help of one particular ML algorithm (C4.5). The result is transcribed in the form of rules which can be altered or edited by the user. The rules constitute, in effect, a knowledge base of an expert system. The system can be applied to a new dataset to provide the user with a set of recommendations concerning the suitability of different algorithms, graded by a score.

3.4 Future Plans Concerning Support

In future database accesible by ftp from LIACC will be organized in such a way that it is easy to add new datasets, classification algorithms, test methods etc.; as these become publicly available. The database will be maintained and new test results validated whenever this will be feasible. Datasets will only be added to the database if they are of industrial and/or commercial relevance. One of the principal aims of the database will be to give algorithm developers access to the expersise developed earlier. In this way, the developers of new algorithms will be able to compare results with chosen classification procedures that were used in the StatLog project. This will facilitate the evaluation of new procedures, and should extend the range of algorithms available to potential industrial users.