Transactions on Information and Communications Technologies vol 29, © 2003 WIT Press, www.witpress.com, ISSN 1743-3517

Alternative strategies for decision list construction

D. A. Newlands[1] & G. I. Webb[2]
[1]*Deakin University, Victoria 3217, Australia*
[2]*Monash University, Victoria 3800, Australia*

Abstract

This work surveys well-known approaches to building decision lists. Some novel variations to strategies based on default rules for the most common class and insertion of new rules before the default rule are presented. These are expected to offer speed up in the construction of the decision list as well as compression of the length of the list. These strategies and a testing regime have been implemented and some empirical studies done to compare the strategies. Experimental results are presented and interpreted. We show that all strategies deliver decision lists of comparable accuracy. However, two techniques are shown to deliver this accuracy with lists composed of significantly fewer rules than alternative strategies. Of these, one also demonstrates significant computational advantages. The prepending strategy is also demonstrated to produce decision lists which are as much as an order of magnitude shorter than those produced by CN2.

1 Introduction

A decision list [1] is an ordered list of classification rules. This contrasts with unordered rules [2, 3] for which there must be a resolution procedure to select between candidates when two or more rules cover a single case. For a decision list, the first rule which covers the case is applied. As a result, a decision list may be less difficult to understand as the inter-relationships between the rules are represented explicitly and concisely. A widely used approach to constructing decision lists is the covering algorithm [4]

```
set rule list to empty
DO
    find a good rule
    add the rule to the list
    remove cases covered by the rule
UNTIL no new rules found
add default rule to rule list
```

Perhaps the best known example of a decision list algorithm is CN2 [5]. In a rule list constructed in this fashion, the first rule covers the majority of cases. An initial default rule for the most common class might offer an alternative approach to rule construction. This could offer compression of the size of the rule list and a speed-up in the construction process if we can take account of the default rule during learning. Such an approach can be implemented by *prepending* new rules [6] using the algorithm

```
add default rule to list
DO
    find best rule for unclassified and
        misclassified examples
    add it to front of list
UNTIL no new rule can be found
```

Results are reported which offer comparable accuracy to the classical covering algorithm but with shorter rule lists.

Another approach is BBG [7] which adds a default rule to the list initially and then generates subsequent best rules for each class and position. The difference is the testing of all positions for each new rule. The algorithm is reported to outperform C4.5 and C4.5 rules on artificial data sets but to perform less well on a small number of data sets from the UCI Repository [8]. There is no comparison with the standard covering algorithm approach.

This paper proposes a new decision list induction algorithm, *penpend*, that we expect to deliver compact rule lists with minimal computation.

2 Proposed approach to decision list construction

The general approach in this work is to aim for rule list compression by use of a default rule for the most common class and to prepend rules to this list as exceptions to the default class. The main part of the work will investigate generating new rules for every possible class at every possible insertion position to construct reasonably optimal (but not guaranteed to be globally optimal) rule lists. Another approach to be investigated is to find the best rule which might be prepended to the list and to investigate other insertion positions without modifying the rule. The intent being to insert the rule in the best position choosing the deeper of equal best positions. The last approach will be to investigate rules for all classes but only for the insertion position just before the default rule.

The fact that we need different rules at different insertion positions can be justified by noting that the rule training sets depend on the target class and the insertion position because, for any given new rule

- examples of the target class which are not yet correctly classified below the insertion point must be retained
- training examples of the target class which are correctly classified by rules below the insertion point can be excluded as accidentally covering them twice has no effect on accuracy.
- counter-examples, not of the target class, which are misclassified by rules below the insertion point may be excluded, as long as the construction strategy will attempt to cover these by a rule inserted closer to the front of the list. The basic strategy (labelled cn_multi) will include these examples and an adaption (labelled cn_multi/a) will exclude them.
- counter-examples which are correctly classified by rules below the insertion point must be kept so that the new, earlier rule will not accidentally overlap the cover of a deeper rule and possibly misclassify these points.

This work will compare a basic covering strategy as implemented in CN2 with the multiple insertion point strategy, the prepending as deeply as possible strategy and the insertion just before the default rule strategy. The software will be constructed by modifying CN2 in several different ways but always constructing ordered rules. CN2 stops when all the examples are covered and there is no default rule. CN2 supports partial matching of cases with missing values and this would complicate the conceptualisation and implementation of our various strategies and, to simplify the treatment of missing values, examples will be counted as covered by any test on a missing attribute. This can be expected to place the various new implementations at a disadvantage to the native CN2 so a second version, called CNx which will not handle partial matching, will be implemented to give a better comparison with the other algorithms. This approach, of prepending rules, will magnify the problem of using small disjuncts since these are placed before the rules with best support. To minimise this problem, a stopping criterion will be introduced which requires the list, after the latest rule is inserted, to have significantly better accuracy ($p <= 0.05$, z test) than the prior list. The versions will be compared by running them on matched data sets and comparing their predictive accuracies and the lengths of the decision lists induced.

The MULTI variant (cn_multi) will examine all potential new rules for all positions, preceding the default rule, in a decision list which is initialised with a default rule for the majority class in the training set. The metric for measuring the quality of each insertion position will be the classification accuracy of the whole list. The stopping condition will be when no rule can be found which improves the predictive accuracy of the whole list. A potential rule which, if inserted into the list, may overlap a previously constructed rule later in the list and misclassify examples which were previously correctly classified, will be rejected during rule induction because it reduces the accuracy of the list. The prepending variant (cn_pre) will also construct ordered rules but the new rules will be constructed for only for the first position in the list although, once constructed, it will be tested in every possi-

ble insertion position and eventually inserted as deeply as possible, consistent with best performance, in the list. The penpending variant (cn_pen) only constructs the rules for the penultimate position, in front of the default rule and only tests the rules in this single position.

3 Experimental method

Twenty eight well known data sets from the UCI repository [8] will be used for the experimental comparison of the performance of

- CN2: CN2 in its native format producing ordered rules, the output is labelled as CN2.
- CNx: CN2 with a modification which disables partial covering, the output labelled as CNx.
- cn_multi: the output of the multi variant, with output labelled as MULTI.
- cn_multi/a: the output of cn_multi but with the modification, noted in section 2, to training set construction and the output labelled as MULTI/a.
- cn_pre: the prepending variant output is labelled as PRE.
- cn_pen: the penpending variant output is labelled as PEN

For each data set, the examples are shuffled and partitioned into training (80%) and test (20%) sets. All six decision list construction algorithms are run on these two ordered sets of items and the classification accuracy and list length on the test set is noted. This procedure is repeated 20 times for each data set.

4 Results

4.1 Accuracy

The average accuracy of each algorithm on each of the data sets is shown in Table 1 below.

In Table 2 the win:draw:loss (respectively and where win is for the left name) ratios for the pairs of algorithms are shown as well as the significance using a sign test.

CN2 is seen to be significantly (at $p <= 0.05$) better than CNx and MULTI/a and close to significantly better than PEN. While this is not ideal, it needs to be recalled that the ability to handle partial covering has been removed from CN2 when we modified the algorithm for these experiments. The difference between CN2 and CNx gives some measure of how much damage was done to the CN2 algorithm by this change and it is clearly substantial. However, the main purpose was to compare the other algorithms with CNx which is a version of CN2 which is denatured to the same extent in all the various implementations.

Comparing CNx with the other algorithms, there is no significant difference between it and MULTI, PRE and PEN but the superiority to MULTI/a is significant. Comparing MULTI with MULTI/a, PRE and PEN, there is no significant difference between them. Similarly, there is no significant difference between

Table 1: Average accuracy of algorithms.

Domain	CN2	CNx	MULTI	MULTI/a	PRE	PEN
allbp :	96.4	96.3	96.0	95.9	95.9	95.9
anneal :	93.0	77.2	97.4	98.2	98.6	98.1
audio :	69.1	69.0	76.1	76.9	78.2	74.6
balance :	81.3	81.3	67.9	64.3	67.9	64.3
bf :	91.1	91.1	88.7	82.8	85.8	78.9
big-pole-a :	84.6	84.6	56.1	56.1	56.1	56.1
echocardio :	67.8	63.4	67.9	67.9	67.9	67.9
ecoli :	79.4	79.4	80.8	80.9	81.0	81.3
glass :	62.3	62.3	69.8	65.7	68.3	66.5
heart-clev :	74.9	74.9	72.5	72.5	72.5	72.5
heart-hung :	75.4	74.9	76.6	76.6	76.6	76.6
hepatitis :	79.9	81.3	78.9	78.9	78.9	78.9
hypo :	99.1	98.9	99.0	98.6	98.9	97.9
iris :	94.1	94.1	93.7	93.5	94.0	93.9
oring_e :	72.8	72.8	68.3	75.8	68.3	75.8
oring_b :	64.3	64.3	54.8	58.2	57.1	58.2
page-block :	87.4	87.4	90.0	86.7	89.4	87.4
satimage :	90.9	90.9	89.8	91.1	88.2	91.3
segment :	94.8	94.8	92.7	94.7	88.8	94.3
shuttle :	100.0	100.0	99.9	99.9	99.8	99.9
sonar :	75.0	75.0	65.4	65.4	65.4	65.4
soya :	85.5	84.7	84.2	82.8	83.6	84.4
thyroid :	96.3	96.2	96.1	96.1	96.1	96.1
vehicle :	80.6	80.6	79.0	77.6	74.0	76.7
votes :	93.5	93.5	91.4	91.4	91.4	91.4
waveform :	70.9	70.9	68.0	67.5	67.2	67.3
wine :	87.4	87.4	88.2	84.8	90.1	85.2
yeast :	52.3	52.3	56.2	49.7	55.3	49.8

MULTI/a and PRE or PEN and no significant difference between PRE and PEN.

From these results it would seem that MULTI/a offers the worst performance in terms of accuracy. The performance of PRE and PEN seem indistinguishable

Table 2: Comparison of accuracy of algorithms.

versus	CNx	MULTI	MULTI/a	PRE	PEN
CN2	8:19:1	19:0:9	20:0:8	19:0:9	19:1:8
	p=0.039	p=0.087	p=0.036	p=0.087	p=0.052
CNx		18:0:10	20:0:8	18:1:9	19:1:8
		p=0.185	p=0.036	p=0.122	p=0.052
MULTI			12:9:7	12:10:6	11:9:8
			p=0.359	p=0.230	p=0.648
MULTI/a				7:9:12	7:13:8
				p=0.359	p=1.0
PRE					10:9:9
					p=1.0

and while MULTI has some perceptible advantage, it is not significantly superior to either. The denatured version of CN2 cannot offer clearly better performance than MULTI, PRE or PEN. Thus, apart from rejecting MULTI/a, there is little to choose, on the basis of accuracy, between CNx, MULTI, PRE and PEN.

4.2 List compression

Table 3 shows the average length of the decision lists constructed by the various algorithms.

In Table 4 the win:draw:loss (respectively and where win is for the left name) ratios for the pairs of algorithms are shown as well as the significance using a sign test.

It is easily seen that MULTI, MULTI/a, PRE and PEN all yield list lengths which are obviously and statistically significantly less than those of CN2 and CNx. MULTI is significantly better than MULTI/a and PEN but not significantly different from PRE. MULTI/a, PRE and PEN are not significantly different from each other. It clearly matches expectation that MULTI should be better than PRE or PEN because of the more thorough nature of its search for the best next rule. It also is expected that MULTI/a should produce longer lists since, by excluding some examples from the training set, it is committing to later insertion of extra rules to correct any errors resulting from such exclusions.

5 Conclusions

This work set out to investigate approaches to building decision lists which involved using a default rule for the majority class and various strategies for con-

Table 3: Average lengths of lists.

Domain	CN2	CNx	MULTI	MULTI/a	PRE	PEN
allbp	26.6	21.5	6.2	6.3	6.3	6.3
anneal	16.9	8.2	10.9	11.4	11.6	11.8
audio	21.2	21.4	14.7	22.8	12.9	19.5
balance	56.4	56.4	6.3	5.4	6.3	5.4
bf	94.8	94.8	19.2	21.6	16.2	19.5
big-pole-a	184.7	184.7	5.1	5.1	5.1	5.1
echocardio	14.6	11.8	3.5	3.5	3.5	3.5
ecoli	18.9	18.9	11.8	14.2	13.2	13.7
glass	15.7	15.7	9.9	11.1	10.2	11.1
heart-clev	17.3	17.3	4.0	4.0	4.0	4.0
heart-hung	19.4	18.0	3.5	3.5	3.5	3.5
hepatitis	11.6	8.0	3.7	3.7	3.7	3.7
hypo	17.0	13.2	10.2	8.0	8.4	7.9
iris	5.9	5.9	3.7	3.8	3.6	3.8
oring_e	3.5	3.5	3.2	3.2	3.4	3.2
oring_b	4.0	4.0	4.1	3.9	4.2	3.9
page-block	9.9	9.9	6.7	7.8	6.8	7.3
satimage	7.6	7.6	6.9	7.2	7.1	7.3
segment	22.9	22.9	14.2	19.5	15.8	19.8
shuttle	9.8	9.8	10.6	10.4	8.3	11.3
sonar	9.7	9.7	3.5	3.5	3.5	3.5
soya	24.2	23.4	23.1	22.5	24.0	21.1
thyroid	21.9	17.4	5.0	5.1	5.1	5.1
vehicle	6.7	6.7	6.1	6.2	6.7	6.2
votes	15.4	15.4	4.3	4.3	4.3	4.3
waveform	6.1	6.1	4.9	5.4	5.4	5.4
wine	4.8	4.8	3.8	4.0	4.0	4.0
yeast	84.7	84.7	17.1	18.5	16.5	17.9
average	26.9	25.8	8.1	8.8	8.0	8.5

structing the remainder of the rule list. The strategies compared were CN2 appending to the list, CNx appending to the list but having the CN2 partial matching facility removed, MULTI which searches through the best rule for every class in every position, MULTI/a which incorporates an adaptation to the construction of training sets, PRE which constructs a prependable rule but inserts it as deeply into the list as possible without losing accuracy and PEN which constructs only

Table 4: Comparison of length of lists.

versus	CNx	MULTI	MULTI/a	PRE	PEN
CN2	1:19:8	2:0:26	2:0:26	1:1:26	1:0:27
	$p<0.05$	$p<0.01$	$p<0.01$	$p<0.01$	$p<0.01$
CNx		3:0:25	3:0:25	3:1:24	2:0:26
		$p<0.01$	$p<0.01$	$p<0.01$	$p<0.01$
MULTI			15:8:5	14:8:6	16:8:4
			$p<0.05$	$p>0.10$	$p<0.05$
MULTI/a				7:11:10	4:17:7
				$p>0.25$	$p>0.25$
PRE					11:11:6
					$p>0.25$

rules which can be inserted just before the default rule. Only the MULTI and PRE strategies could achieve accuracies which were not significantly worse than that of CN2. However, both MULTI and PRE provided significantly shorter decision lists than CN2. Although no direct measure was taken of the speed, it is clear that PRE which only constructs rules for a single position before trying various insertion positions is substantially quicker than MULTI which constructs all possible rules in all possible positions. Since this was a research implementation, no attempt was made to make the algorithm efficient so actual compute times have little relevance.

Within the constraints of the design of these experiments, the adaption to the construction of training sets (cn_multi/a) has proven not to be valuable. However, the notion of using an initial default rule and prepending other rules has proven to result in decision lists which are significantly shorter than CN2 while offering comparable classification accuracy. The MULTI strategy should offer reasonably optimal results because of its nature and is comparable to native CN2 despite having an inbuilt disadvantage in not handling partial covering.

It is desirable that a future study investigate how often the PRE strategy places a rule in a deeper position to understand whether the extra complexity in rule construction, relative to a pure prepending strategy, is worthwhile.

References

[1] Rivest, R., Learning decision lists. *Machine Learning*, **2**, pp. 229–246, 1987.

[2] Cohen, W., Fast effective rule induction. *Machine Learning: Proceedings of 12th International Conference*, Morgan Kaufmann, pp. 115–123, 1995.

[3] Quinlan, J., *C4.5 programs for Machine Learning*. Morgan Kaufmann: San Mateo, California, 1995.

[4] Michalski R. S., On the quasi-minimal solution of the general covering problem. *Proceedings of the Fifth International Symposium on Information Processing (FCIP 69)*, volume A3, pp. 125–128, 1969.
[5] Clark, P. and Niblett, T., The cn2 induction algorithm. *Machine learning*, **3**, pp. 261–284, 1989.
[6] Webb, G., Recent progress in learning decision lists by prepending inferred rules. *Second Singapore International Conference on Intelligent Systems*, pp. B280–B285, 1994.
[7] Van Horn, K. and Martinez, T., The bbg rule induction algorithm. *Proc. 6th Australian Joint Conf. on AI*, pp. 348–355, 1993.
[8] Murphy, P. and Aha, D., The uci repository of machine learning databases, http://www.ics.uci.edu/ mlearn/mlrepository.html.