Rule Induction with CN2: Some Recent Improvements

Peter Clark Robin Boswell
The Turing Institute, 36 N.Hanover St., Glasgow
email: {pete,robin}@turing.ac.uk

Abstract

The CN2 algorithm induces an ordered list of classification rules from examples using entropy as its search heuristic. In this short paper, we describe two improvements to this algorithm. Firstly, we present the use of the Laplacian error estimate as an alternative evaluation function and secondly, we show how unordered as well as ordered rules can be generated. We experimentally demonstrate significantly improved performances resulting from these changes, thus enhancing the usefulness of CN2 as an inductive tool. Comparisons with Quinlan's C4.5 are also made.

Keywords: learning, rule induction, CN2, Laplace, noise

1 Introduction

Rule induction from examples has established itself as a basic component of many machine learning systems, and has been the first ML technology to deliver commercially successful applications (eg. the systems GASOIL [Slocombe et al., 1986], BMT [Hayes-Michie, 1990], and in process control [Leech, 1986]). The continuing development of inductive techniques is thus valuable to pursue.

CN2 is an algorithm designed to induce 'if...then...' rules in domains where there might be noise. The algorithm is described in [Clark and Niblett, 1989] and [Clark and Niblett, 1987], and is summarised in this paper. The original algorithm used entropy as its search heuristic, and was only able to generate an ordered list of rules. In this paper, we demonstrate how using the Laplacian error estimate as a heuristic significantly improves the algorithm's performance, and describe how the algorithm can also be used to generate unordered rules. These improvements are important as they enhance the accuracy and scope of applicability of the algorithm.

2 An Improved Evaluation Function

2.1 The Original Entropy Function

The CN2 algorithm consists of two main procedures: a search algorithm performing a beam search for a good rule (shown in Appendix 2) and a control algorithm for repeatedly executing the search (shown later in Figure 1).

During the search procedure, CN2 must evaluate the rules it finds to decide which is best. One possible metric of rule quality is its accuracy on training data (eg. an option for AQ15 [Michalski et al., 1986]). An alternative is entropy, used by ID3 and the original CN2, which behaves very similarly to apparent accuracy. Entropy also prefers rules which cover examples of only one class.

The problem with these metrics is that they tend to select very specific rules covering only a few examples, as the likelihood of finding rules with high accuracy on the training data increases as the rules become more specific. In the extreme case, a maximally specific rule will just cover one example and hence have an unbeatable score using the metrics of accuracy (scores 100% accuracy) or entropy (scores 0.00, a perfect score). This is undesirable as rules covering few examples are unreliable, especially with noise in the domain. Their accuracy on the training data does not adequately reflect their true predictive accuracy (ie. accuracy on new test data) which may appear.

2.2 Significance Testing: A Partial Solution

To avoid selecting highly specific rules, CN2 uses a significance test (see [Clark and Niblett, 1989]) which ensures that the distribution of examples among classes covered by the rule is *significantly different* from that which would occur by chance. In this way, many rules covering only a few examples are eliminated, as the significance test deems their apparent high accuracy likely to be simply due to chance.

However, while a significance test eliminates rules which are below a certain threshold of significance, there is still the problem that rules which just pass the significance test will tend to be preferred over more general and reliable but less apparently accurate rules. Consider a domain with two equally likely classes C1 and C2, and consider three rules R1, R2 and R3, where:

R1 covers 1000 examples of class C1 and 1 of C2 (we denote this by [1000, 1])
R2 covers 5 examples of C1 and 0 of C2 (ie. [5, 0])
R3 covers [1, 0])

Here, the algorithm should ideally prefer R1 as its accuracy on new test data is likely to be the best – rules R2 and R3 only cover a few examples and their apparent accuracies of 100% are not fully reflective of performance on new test data. However, although a 99% significance test eliminates R3, R2 will just pass and be selected in preference to R1. Raising the significance level further does not solve the problem as a rule R1.5 (say) may exist which again just passes the raised significance threshold.

We can describe the metrics of apparent accuracy/entropy as having an undesirable 'downward bias', ie. preference for rules low down in the general (top) to specific (bottom) search space. Raising the significance threshold causes the level of specificity at which the search terminates to raise, but does not eliminate the downward bias itself.

2.3 The Use of Laplace

In fact, as reported by other authors (eg. [Niblett, 1987]) an approximate measure does exist to measure the expected accuracy directly, namely 1 - the Laplace expected error estimate. This expected accuracy is given by the formula:

$$\texttt{LaplaceAccuracy} = (n_c + 1)/(n_{tot} + k) \tag{1}$$

where

k is the number of classes in the domain
n_c is the number of examples in the predicted class c covered by the rule
n_{tot} is the total number of examples covered by rule

When generating a rule list, the predicted class c for a rule is simply the class with the most covered examples in it.

This formula is a special case of the m-probability-estimate developed by Cestnik [Cestnik, 1990]:

$$\texttt{mPAccuracy} = (n_c + p_o(c)\ m)/(n_{tot} + m)$$

where uniform prior probabilities p_o for classes are assumed (ie. $p_o(c) = 1/k$) and the tunable parameter m is set to k. The m-probability-estimate is analysed further in [Cestnik and Bratko, 1991].

For our example above the Laplace accuracy estimates for predicting the class with the most covered examples in are 99.8% for R1, 85.7% for R2 and 66.6% for R3. Thus Laplace avoids the undesirable 'downward bias' of entropy, and which significance testing only partly overcame.

A final check must be included to ensure the expected accuracy is at least better than that of a default rule predicting the class for all examples.

2.4 The New Role of Significance Testing

Significance testing can still be included to prune out the most specialised (and hence less frequently applicable) rules in the rule list. This reduces the complexity of the rule list, but at a slight cost in predictive accuracy. Interestingly, the behaviour of significance testing with Laplace is qualitatively different to that with entropy. With entropy, raising the significance threshold causes CN2 to select slightly more general rules during induction. With Laplace, general rules tend to be favoured anyway, and significance testing instead alters the point at which CN2 stops searching for further rules. In other words, with entropy the test affects which rules are chosen as 'best', but with Laplace acts solely as a termination criterion for the algorithm.

Table 1: Details of Experimental Domains

Domain†	Description	Number of		
		Exs	Atts	Classes
lymphography	disease diagnosis	148	18	4
pole-and-cart	predict human balancing action from exs	1044	4	2
soybean	disease diagnosis	307	35	19
heart-diseaseC	disease diagnosis (data from Cleveland)	303	13	2
heart-diseaseH	disease diagnosis (data from Hungary)	294	13	2
glass	predict glass type from chem. content	194	7	9
primary-tumour	predict tumour type	330	17	15
voting-records	predict democrat/republican from votes	435	16	2
thyroid	disease diagnosis	1960	29	3
breast-cancer	predict if recurrence is likely	286	9	2
hepatitis	predict if survival likely	157	19	2
echocardio	predict if survival from heart problem likely	131	7	2

† (Sources: Lymph, prim-tumour, breast-cancer from Ljubljana, 1985. Pole-&-cart from Turing Inst., 1990. Remainder from UCI, 1989. See end of paper for details of any data conversions made.)

2.5 Experimental Comparison

2.5.1 Experimental Method

Experiments were performed to measure the improvement in predictive accuracy using the Laplace heuristic. As demonstrated by previous authors (eg. [Buntine and Niblett, 1990]), tests on a single domain are not sufficient to draw reliable conclusions about the relative performance of algorithms. Thus experiments on twelve domains shown in Table 1 were conducted.

CN2 using entropy and Laplace were compared. Also, comparisons with Quinlan's C4.5 [Quinlan et al., 1987, Quinlan, 1987] were performed. Data was split into 67% for training and 33% for testing, and the results averaged over 20 runs. For CN2, a star size of 20 was used and significance testing was switched off. (The effect of significance testing is examined later). For C4.5 a single, pruned tree was generated for each run.

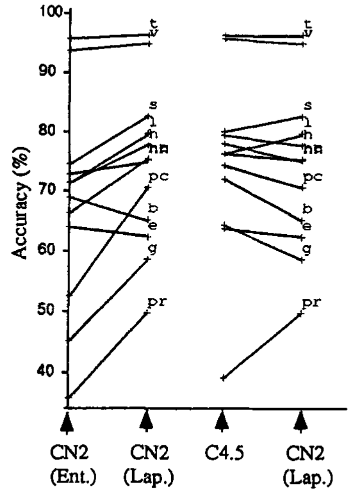
2.5.2 Results: Comparative Accuracies

Table 2 shows the average accuracies obtained over the above domains. To make an overall comparison between the algorithms, a paired, two-tailed t-test was used, whose results are also shown in this table. From this t-test, it can be seen that using the Laplacian heuristic significantly (>99% significant, from the 2-tail prob.) improves CN2's accuracy, with an average improvement of 6.4%. The comparison between CN2 (Laplace) and C4.5 did not reveal any significant difference in accuracy. Additionally, the average size of the rule lists induced by CN2 (Laplace) was smaller than for CN2 (Entropy). The sizes are tabulated in Appendix 1.

Table 2: Percentage Accuracies of Algorithms

The table shows percentage accuracies (σ denotes their standard deviations). The graph schematically re-presents the data as follows: Each line corresponds to a different domain, and connects the observed accuracy using one algorithm with another. Thus an upward slope reflects an improvement in accuracy, and a downward slope a worsening. The average improvement of CN2 (Laplace), and the significance of this improvement, is summarised in the second table.

Domain	Algorithm			
	CN2		C4.5	Default
	(Entropy)	(Laplace)		
lymphography	71.5 σ6.3	79.6 σ5.7	76.4 σ6.2	54.2 σ6.7
pole-and-cart	52.5 σ1.9	70.6 σ3.1	74.3 σ2.0	48.8 σ1.0
soybean	74.7 σ6.7	82.7 σ3.9	80.0 σ3.6	10.3 σ1.5
heart-diseaseC	66.3 σ8.5	75.4 σ3.6	76.4 σ4.5	53.1 σ3.8
heart-diseaseH	73.0 σ4.6	75.0 σ3.8	78.0 σ5.5	64.9 σ3.5
glass	45.2 σ8.1	58.5 σ5.0	64.2 σ5.1	34.0 σ4.4
primary-tumour	35.6 σ5.2	49.7 σ9.8	39.0 σ4.0	24.5 σ2.8
voting-records	93.6 σ1.8	94.8 σ1.7	95.6 σ1.1	61.6 σ2.9
thyroid	95.6 σ0.7	96.3 σ0.7	96.4 σ0.9	95.4 σ0.8
breast-cancer	69.0 σ3.6	65.1 σ5.3	72.1 σ3.7	71.3 σ2.3
hepatitis	71.3 σ5.2	77.6 σ5.9	79.3 σ5.8	78.0 σ4.6
echocardio	63.9 σ5.4	62.3 σ5.1	63.6 σ5.3	64.4 σ4.9



Comparison of mean accuracies using paired, two-tailed t-test on the above data:

Algorithms Compared:	Mean Improvement (Mean X - Y)	Significance of improvement
CN2 (Laplace) - CN2 (Entropy)	6.4%	99.3%
CN2 (Laplace) - C4.5	-0.5%	30.0%

2.5.3 Results: Effect of Pruning

In the original CN2 (ie. using entropy), using a significance test caused the algorithm to select a smaller number of more general rules (possibly with counter-examples against them) in preference to a large number of highly specific rules. The Laplace heuristic, however, is sufficient on its own to bias the search towards those general rules with higher predictive accuracy, tending to find rules of highest predictive accuracy (and thus also high significance) first. It would thus be expected that removing less significant rules using a significance test would have a different effect, namely that CN2 would still select the same rules early on during the search but would terminate earlier. This was indeed observed (see Appendix 1) with the same early rules tending to appear in the rule list but with the number of rules decreasing and the overall accuracy also slightly decreasing.

3 Generating Unordered Rules

3.1 The Disadvantage of Ordered Rules

The original CN2 algorithm generates rules assembled in a particular order, described as a rule list by Rivest [Rivest, 1987]. During classification of a new example, each rule is tried in order until one fires. The algorithm then exits, assigning the class which that rule predicted to the example.

Rule lists have the nice property of being 'logical', in the sense that clashes between rules cannot occur as only one rule can ever fire. Thus there is no need to include probabilistic machinery for resolving clashes between rules.

However, there is also a corresponding problem in understanding the rules, in that the meaning of any single rule is dependent on all the other rules which precede it in the rule list. Consider, for example, a rule list:

```
       If    feathers = yes     then   class = bird
else   if    legs = two         then   class = human
else    ...
```

The rule "`if legs=two then class=human`", when considered alone, is not correct as birds also have two legs. Thus to understand the rule, all the previous rules in the list must also be taken into consideration. This problem becomes acute with a large number of rules, making it difficult for an expert to understand the true meaning of a rule far down in the list. As induced rules must generally be validated by experts before their use in applications, this is a significant disadvantage.

3.2 Generating Unordered Rules Using CN2

3.2.1 The CN2 (unordered) Algorithm

CN2 consists of a search procedure and a control procedure. Fortunately CN2 can be easily modified to generate an *unordered* rule set by changing only the control procedure, leaving the beam search procedure unchanged (apart from the evaluation function, described below). The original control procedure for ordered rules is shown in Figure 1, and the control procedure for unordered rules is shown in Figure 2. (The search procedure is shown in Appendix 1).

The main modification to the algorithm is to iterate the search for each class in turn, removing only covered examples *of that class* when a rule has been found. Unlike for ordered rules, the negative examples remain because now each rule must independently stand against all negatives. The covered positives must be removed to stop CN2 repeatedly finding the same rule.

To effect this rule search for each class in turn, the Laplace heuristic (Equation 1) must be applied differently: with ordered rules the predicted class c is taken simply as the one with the most covered examples in it, but with unordered rules the predicted class is fixed to be the `class` selected by the revised control procedure.

Figure 1: The CN2 Ordered Rules Algorithm

```
procedure CN2ordered(examples, classes):
let rulelist = []
repeat
   call FindBestCondition(examples) to find bestcond
   if   bestcond is not null
   then let class be the most common class of exs. covered by bestcond
      & add rule 'if bestcond then predict class' to end of rulelist
      & remove from examples all examples covered by bestcond
until bestcond is null
return rulelist
```

Figure 2: The CN2 Unordered Rules Algorithm

```
procedure CN2unordered(allexamples, classes):
let ruleset = {}
for each class in classes:
   generate rules by CN2ForOneClass(allexamples,class)
   add rules to ruleset
return ruleset.

procedure CN2ForOneClass(examples,class):
let rules = {}
repeat
   call FindBestCondition(examples, class) to find bestcond
   if   bestcond is not null
   then add the rule 'if bestcond then predict class' to rules
      & remove from examples all exs in class covered by bestcond
until bestcond is null
return rules
```

3.3 Applying Unordered Rules

With an unordered rule list, all rules are tried and those which fired collected. If a clash occurs (ie. more than one class predicted), some probabilistic method is needed to resolve clashes. The method used here is to tag each rule with the distribution of covered examples among classes, and then to sum these distributions to find the most probable class should a clash occur. For example, consider the three rules:

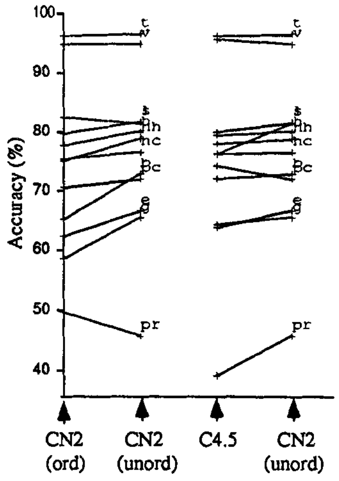
```
if  legs=two and feathers=yes  then class=bird,     covers [13,0].
if  size=large and flies=no    then class=elephant, covers [2,10].
if  beak=yes                   then class=bird,     covers [20,0].
```

Here the two classes are [bird, elephant], [13,0] denoting that the rule covers 13 (training) examples of bird and 0 of elephant. Given a new example of a large, beaked, two-legged,

Table 3: Percentage Accuracies of Algorithms

(See Table 2 for explanation of graph and tables)

Domain	Algorithm			
	CN2 (Laplace)		C4.5	Default
	unordered	ordered		
lymphography	81.7 σ4.3	79.6 σ5.7	76.4 σ6.2	54.2 σ6.7
pole-and-cart	72.0 σ2.9	70.6 σ3.1	74.3 σ2.0	48.8 σ1.0
soybean	81.6 σ3.8	82.7 σ3.9	80.0 σ3.6	10.3 σ1.5
heart-diseaseC	76.7 σ3.9	75.4 σ3.6	76.4 σ4.5	53.1 σ3.8
heart-diseaseH	78.8 σ4.1	75.0 σ3.8	78.0 σ5.5	64.9 σ3.5
glass	65.5 σ5.6	58.5 σ5.0	64.2 σ5.1	34.0 σ4.4
primary-tumour	45.8 σ3.6	49.7 σ9.8	39.0 σ4.0	24.5 σ2.8
voting-records	94.8 σ1.8	94.8 σ1.7	95.6 σ1.1	61.6 σ2.9
thyroid	96.6 σ0.9	96.3 σ0.7	96.4 σ0.9	95.4 σ0.8
breast-cancer	73.0 σ4.5	65.1 σ5.3	72.1 σ3.7	71.3 σ2.3
hepatitis	80.1 σ5.7	77.6 σ5.9	79.3 σ5.8	78.0 σ4.6
echocardio	66.6 σ7.3	62.3 σ5.1	63.6 σ5.3	64.4 σ4.9



Comparison of mean accuracies using paired, two-tailed t-test on the above data:

Algorithms Compared:	Mean Improvement (Mean X - Y)	Significance of improvement
CN2 (unordered) - CN2 (ordered)	2.0%	95.0%
CN2 (unordered) - C4.5	1.5%	94.0%

feathered, non-flying thing, all three rules fire. The clash is resolved by summing the covered examples (sum is [35, 10]) and then predicting the most common class in the sum (bird).

3.4 Comparative Performance

3.4.1 Experimental Method

The same experimental method as performed for the earlier experiments on ordered rules (Section 2.5.1) was followed in order to compare the performances of ordered and unordered rule sets. Additionally, a comparison with C4.5 was again made.

3.4.2 Results: Comparative Accuracies

The results are shown in Table 3. Surprisingly, the CN2 (unordered) algorithm had an even higher accuracy than that of CN2 (ordered), with a small (2%) but significant (at the 95% level) higher average accuracy. The comparison also showed a slight (1.5%) but again significant (at the 94% level) improvement over C4.5.

One possible explanation for this high performance is that, with unordered rules, several rules may contribute to the classification of one example thus reducing effects of noise and an occasional poorly performing rule. Spreading of the classification decision over several rules has been termed using 'multiple knowledge' [Gams et al., 1991] and algorithms specifically designed to generate 'extra' rules have been designed elsewhere (eg. [Gams, 1989, Cestnik and Bratko, 1988]). Cestnik and Bratko report this technique resulted in significantly improved accuracies, and it seems likely a similar phenomenon is occurring here. The possible presence of extra classificational information in the unordered rules, compared with the ordered rules and C4.5's trees, is supported by examination of the rule set sizes. Unordered rule sets were about twice the size of ordered rule lists, and about four times the size of C4.5 trees, as tabulated in Appendix 1. Pruning the unordered rule sets by significance testing using a significance threshold of 99.5% reduced them to a size similar to C4.5's trees, but also slightly reduced the accuracy to one no longer significantly different from that of C4.5.

3.4.3 Effect of Pruning

As for ordered rules, applying a significance test reduced the number of rules found by the algorithm while also slightly reducing the predictive accuracy (see Appendix 1).

3.4.4 Worse-than-Default Domains

An interesting finding, worthy of brief comment, was that CN2 (ordered), in the breast-cancer and echocardio domains, induced rules performing significantly (ie. outside the bounds of one standard error) worse than the default rule (confirmed by repeating the experiments over 250 runs). The simple explanation for this is that, in these cases, CN2 was still slightly overfitting the rules to the data. To understand how induced rules can actually do worse than the default rule, consider the worst case of overfitting where a ruleset/decision tree is grown so every rule/leaf covers only one training example. Given 70% of examples are in class c1 and 30% in class c2, and the classes are completely independent of the attributes (ie. 100% noise), the overfitted rules/tree will be correct with probability 0.7 for rules predicting c1 and 0.3 for c2. With 70% of the examples in c1 and 30% c2, the overall probability correct will thus be $0.7 \times 0.7 + 0.3 \times 0.3 = 0.58$, worse than the default accuracy of 0.7. The overfitting observed in our experiments reflects behaviour between these two extremes, and suggests the pruning of ordered rules could still be slightly improved.

4 Conclusion

In this paper we have described two important extensions to the CN2 algorithm. Most importantly, we has shown how the algorithm can be extended to generate unordered as well as ordered rules, thus contributing to the comprehensibility of the induced rule set. Secondly, we have described a different evaluation function for CN2, and experimentally demonstrated a significantly improved performance resulting from this change. These two extensions thus contribute to CN2's utility as a tool for inductively building knowledge-based systems.

Acknowledgements

This work was funded under the Esprit MLT project, Number 2154. Thanks to Ivan Bratko, Tim Niblett, Matjaž Gams and Donald Michie for their valuable comments on the paper, and to Tim Niblett for his assistance in analysing the experimental data. We are grateful to G. Klanjšček, M. Soklič and M. Zwitter of the University Medical Center, Ljubljana for the use of the lymphography, breast-cancer and primary-tumour data sets and to I. Kononenko for their conversion to a form suitable for the induction algorithms. We are also grateful to David Aha (UCI) for the compilation and use of the UCI Repository of Machine Learning Databases.

Availability of CN2

As part of Esprit MLT project 2154, CN2 has been implemented in C and runs on Sun workstations. To request copies of the algorithm, please contact Robin Boswell at robin@turing.ac.uk.

References

Buntine, W. and Niblett, T. A further comparison of splitting rules for decision-tree induction. (Submitted to the Machine Learning Journal), 1990.

Cestnik, B. Estimating probabilities: A crucial task in machine learning. In *ECAI-90*, 1990.

Cestnik, B. and Bratko, I. Learning redundant rules in noisy domains. In Kodratoff, Y., editor, *ECAI-88*, pages 348–350, London, Pitman, 1988.

Cestnik, B. and Bratko, I. On estimating probabilities in tree pruning. In Kodratoff, Y., editor, *Proc. EWSL-91*, 1991.

Clark, P. and Niblett, T. Induction in noisy domains. In Bratko, I. and Lavrač, N., editors, *Progress in Machine Learning (proceedings of the 2nd European Working Session on Learning)*, Sigma, Wilmslow, UK, 1987.

Clark, P. and Niblett, T. The CN2 induction algorithm. *Machine Learning Journal*, 3(4):261–283, 1989.

Gams, M. New measurements highlight the importance of redundant knowledge. In Morik, K., editor, *EWSL-89*, pages 71–79, London, Pitman, 1989.

Gams, M., Bohanec, M., and Cestnik, B. A schema for using multiple knowledge. (submitted to IJCAI-91), 1991.

Hayes-Michie, J. E., editor *Pragmatica: Bulletin of the Inductive Programming Special Interest Group*, volume 1. Turing Institute Press, Glasgow, UK, 1990.

Leech, W. J. A rule-based process control method with feedback. *Advances in Instrumentation*, 41:169–175, 1986.

Michalski, R., Mozetic, I., Hong, J., and Lavrac, N. The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In *AAAI-86*, volume 2, pages 1041–1045, Ca. Kaufmann, 1986.

Niblett, T. Constructing decision trees in noisy domains. In Bratko, I. and Lavrač, N., editors, *Progress in Machine Learning (proceedings of the 2nd European Working Session on Learning)*, pages 67–78. Sigma, Wilmslow, UK, 1987.

Quinlan, J. R. Simplifying decision trees. *Int. Journal of Man-Machine Studies*, 27(3):221–234, 1987.

Quinlan, J. R., Compton, P. J., Horn, K. A., and Lazarus, L. Inductive knowledge acquisition: a case study. In *Applications of Expert Systems*, pages 157–173, Addison-Wesley, Wokingham, UK, 1987.

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

Slocombe, S., Moore, K., and Zelouf, M. Engineering expert system applications. In *Annual Conference of the BCS Specialist Group on Expert Systems*, 1986.

Data Conversion Notes

Key: cl=classes, ex=examples, att=attributes. lymph: orig data 9cl/150ex reduced to 4cl/148ex by removing 3cl (populations 1,1,0) & merging 2 × 2cl (forms 'X' and 'maybe X'). p-tumour: orig data 22cl/339ex reduced to 15cl/330ex by removing 6cl (popns. 0,1,2,1,2,2,1). b-cancer: 8ex replaced illegal att val with 'unknown'. soybean: UCI file soybean-large.data. h-disease{C,H}: orig 5cl reduced to 2cl (0=absence, 1-4=presence). glass: remove att1 (ex no.). thyroid: 1960ex randomly drawn from UCI file allbp.data. echocardio: Predict for att 2 ('alive'/'dead'), delete atts 1 & 13 (alternative class vals), 10-12 (meaningless), delete 1ex with unknown cl val. Others: conversion straightforward.

Appendix 1: Effect of Pruning on CN2

(See Table 2 for explanation of tables)

Accuracy:

	CN2 (Entropy)		CN2 (Laplace)				C4.5
Domain ↓	(Ordered rules)		Ordered Rules		Unordered Rules		
Sig. Threshold →	0%	99.5%	0%	99.5%	0%	99.5%	
lymphography	71.5 σ6.3	68.4 σ8.6	79.6 σ5.7	74.4 σ5.9	81.7 σ4.3	76.5 σ5.3	76.4 σ6.2
pole-and-cart	52.5 σ1.9	52.2 σ1.7	70.6 σ3.1	67.9 σ3.3	72.0 σ2.9	63.0 σ3.2	74.3 σ2.0
soybean	74.7 σ6.7	54.2 σ6.5	82.7 σ3.9	57.5 σ4.6	81.6 σ3.8	76.1 σ4.4	80.0 σ3.6
heart-diseaseC	66.3 σ8.5	67.1 σ9.2	75.4 σ3.6	76.1 σ4.4	76.7 σ3.9	76.6 σ3.7	76.4 σ4.5
heart-diseaseH	73.0 σ4.6	81.6 σ3.4	75.0 σ3.8	74.9 σ4.7	78.8 σ4.1	77.8 σ3.9	78.0 σ5.5
glass	45.2 σ8.1	44.4 σ7.7	58.5 σ5.0	56.9 σ7.7	65.5 σ5.6	61.6 σ8.3	64.2 σ5.1
primary-tumour	35.6 σ5.2	33.0 σ3.5	49.7 σ9.8	38.7 σ5.3	45.8 σ3.6	41.4 σ5.8	39.0 σ4.0
voting-records	93.6 σ1.8	94.0 σ1.8	94.8 σ1.7	92.8 σ1.8	94.8 σ1.8	93.3 σ2.1	95.6 σ1.1
thyroid	95.6 σ0.7	95.6 σ0.9	96.3 σ0.7	96.3 σ0.5	96.6 σ0.9	96.1 σ1.2	96.4 σ0.9
breast-cancer	69.0 σ3.6	68.7 σ4.3	65.1 σ5.3	64.2 σ7.6	73.0 σ4.5	70.8 σ3.5	72.1 σ3.7
hepatitis	71.3 σ5.2	77.5 σ5.6	77.6 σ5.9	78.1 σ5.9	80.1 σ5.7	80.8 σ4.5	79.3 σ5.8
echocardio	63.9 σ5.4	67.5 σ5.6	62.3 σ5.1	63.2 σ7.7	66.6 σ7.3	69.4 σ6.8	63.6 σ5.3
Average	67.7	67.0	74.0	70.1	76.1	73.6	74.6

Rule list/rule set/decision tree size:
(Number of nodes inc. leaves in tree, or total number of att. tests in rule list/set)

	CN2 (Entropy)		CN2 (Laplace)				C4.5
Domain ↓	(Ordered rules)		Ordered Rules		Unordered Rules		
Sig. Thr. →	0%	99.5%	0%	99.5%	0%	99.5%	
lymph	24.6 σ4.4	5.1 σ1.1	21.1 σ3.8	8.2 σ2.4	40.4 σ4.6	13.5 σ2.3	16.4 σ6.3
pole-&-cart	16.8 σ6.8	3.5 σ2.8	133.6 σ6.3	80.3 σ15.3	255.8 σ8.3	46.5 σ8.2	90.2 σ10.2
soybean	213.2 σ38.8	21.6 σ1.7	55.8 σ7.4	31.3 σ2.7	113.9 σ9.7	83.5 σ6.3	65.9 σ8.4
heart-disC	60.0 σ10.3	9.1 σ2.7	35.1 σ2.5	28.4 σ3.2	68.6 σ5.4	22.8 σ4.1	22.7 σ4.6
heart-disH	37.0 σ7.7	6.1 σ2.6	40.9 σ4.0	26.1 σ5.4	83.4 σ7.5	20.7 σ4.5	7.2 σ3.7
glass	79.0 σ9.4	4.7 σ2.3	32.8 σ3.0	17.2 σ3.0	49.8 σ3.6	30.8 σ3.5	30.9 σ5.8
p-tumour	313.9 σ24.7	5.4 σ2.1	85.2 σ9.6	23.0 σ5.2	351.0 σ23.4	131.4 σ9.3	55.9 σ13.1
voting	11.8 σ3.4	8.1 σ2.0	41.6 σ8.2	15.8 σ5.2	64.8 σ12.1	19.9 σ3.1	7.7 σ3.4
thyroid	1.3 σ0.8	1.1 σ0.5	48.4 σ5.8	37.2 σ7.1	95.6 σ9.9	30.6 σ4.5	15.5 σ7.4
b-cancer	27.9 σ6.0	3.8 σ1.5	53.7 σ5.4	25.8 σ7.4	100.5 σ6.7	18.0 σ5.6	13.0 σ7.0
hepatitis	18.2 σ4.9	2.2 σ1.2	24.0 σ5.5	12.6 σ3.0	43.4 σ6.7	12.6 σ2.3	6.4 σ2.6
echocardio	16.5 σ5.0	1.9 σ0.9	26.4 σ4.0	13.3 σ4.4	48.6 σ3.6	13.1 σ2.1	9.2 σ4.7
Average	68.4	6.1	49.9	26.6	109.7	37.0	28.4

Appendix 2: The CN2 Rule Search Algorithm

```
procedure FindBestCondition(examples[,class][a]):
let mgc = the most general condition ('true')
let star initially contain only the mgc (ie. = { mgc })
let newstar = {}
let bestcond = null
while star is not empty
   for each condition cond in star:
      for each possible attribute test not already tested on in cond
         let cond' = a specialisation of cond, formed by adding test
            as an extra conjunct to cond (ie. cond' = 'cond & test')
         if    cond' is better than bestcond
            &  cond' is statistically significant
         then let bestcond = cond'.
         add cond' to newstar
         if    size of newstar > maxstar (a user-defined constant)
         then remove the worst condition in newstar.
   let star = newstar
return bestcond
```

[a] class is only required for generating unordered rules