Design and Evaluation of the RISE 1.0 Learning System

Pedro Domingos
pedrod@ics.uci.edu

Technical Report 94-34

August 30, 1994

Design and Evaluation of the RISE 1.0 Learning System

Pedro Domingos
Department of Information and Computer Science
University of California, Irvine
Irvine, California 92717, U.S.A.
pedrod@ics.uci.edu

Abstract

Current rule induction systems (e.g. CN2) typically rely on a "separate and conquer" strategy: they induce one rule at a time, removing the newly covered examples from the training set after each step. This results in a dwindling number of examples being available for learning successive rules, which in turn causes several problems that adversely affect the accuracy of the resulting rules. The research reported here investigates the alternative: learning all rules simultaneously using the entire training set for each. A viable approach using this strategy is proposed and implemented in the RISE 1 system. Empirical comparison of the new system with CN2 suggests that "conquering without separating" performs similarly to its counterpart in simple domains, but achieves increasingly substantial gains in accuracy as the domain difficulty grows, without sacrificing speed.

1 Introduction and motivation

Current machine learning approaches to the induction of concept definitions from examples fall mainly into two categories: "divide and conquer" and "separate and conquer." "Divide and conquer" methods [11, 14] recursively partition the instance space until regions of roughly constant class membership are obtained. This approach has often worked well in practice, but is plagued by the splintering of the sample that it causes, resulting in decisions being made with less and less statistical support as induction progresses. "Separate and conquer" methods [7, 4, 10] alleviate this problem somewhat by inducing instead a set of rules that covers all or most positive examples, while covering no or few negative examples. This way, as long as the tests/conditions used are multiway (i.e. not just binary), the induction of each rule starts with a larger fraction of the training set than the corresponding decision tree branch. Rules, however, are induced sequentially, and when learning a new rule only the positive examples not covered by previous rules are used. The availability of fewer examples to learn from means that statistical anomalies are harder to weed out and sensitivity to noise is greater. As a result, it may not be possible to reliably induce some rules to their full length, causing either overly general or incorrect rules to be produced, negatively affecting accuracy.

A related problem, first identified by Holte and coworkers [6, 13], is that of small disjuncts. While covering relatively few examples, small disjuncts tend to be responsible for a disproportionate share of the classification errors committed. Many of these disjuncts undoubtedly represent actual small disjuncts in the domain. The small size of others, however, may be an artifact of the "separate and conquer" strategy: since each such disjunct only attempts to cover a subset of the examples comprising the corresponding target disjunct (the rest having been removed because they were also covered by previously induced disjuncts), it may come out smaller than it should actually be.

All of these problems are alleviated if each rule is learned taking into consideration the entire training set, i.e. if the induction algorithm "conquers without separating." Other problems arise in this case, however. One is that the obvious advantage of "separate and conquer" is forgone: each disjunct now has a confusing influence on the induction of all others, since the positive examples covered only by it will appear as "noise" (false positives) when trying to induce them. Whether this noxious effect prevails over the benefits accrued by conquering without separating is a matter for investigation; conceivably one or the other case may occur, depending on the characteristics of the domain. The aim of the research described in this report is to begin elucidating this question.

Another problem is that a typical rule induction system (e.g. CN2 [4], or FOIL [12] in relational domains) would keep producing the same rule if each cycle started with the whole training set. One way to avoid this is to invert the direction of search: instead of general-to-specific search, use specific-to-general, starting one search from

each example. This is the solution adopted in the algorithm proposed here, RISE (Rule Induction from a Set of Examples). Another question that arises is whether the new procedure, using more searches and more examples in each search, is still acceptably efficient; analysis and experimentation show that it is.

The remainder of this report is organized as follows. The next section describes the representation, control structure, heuristics and classification procedure used by RISE, and its extension to numeric and missing attributes. The time complexity of the algorithm is then analyzed. RISE is then tested in several natural and artificial domains, and its performance compared to CN2's. Finally the results obtained are discussed and some directions for future work are proposed.

2 The RISE algorithm

2.1 Representation

The RISE 1 system induces a set of rules (the hypothesis) from a set of examples (the training set). Rules and examples have a similar representation. Each rule or example has a left-hand side and a right-hand side. The left-hand side is a conjunction of attribute values, the antecedents. Attributes are identified implicitly by the order in which they appear. Each attribute may be nominal or numeric. Nominal attributes take on values which are symbols from some alphabet. Numeric antecedents are ranges represented by their limits; in the case of an example the upper and lower limits coincide. Antecedents of both types can take on the special value $*$, which stands for "any," i.e. for a disjunction of all possible values in the case of a nominal attribute, or the range $[-\infty, \infty]$ in the case of a numeric one. Setting an attribute's value to $*$ is equivalent to dropping it. The right-hand side (or consequent) of each rule or example is a single attribute value, the class, which must be nominal, and cannot be $*$.

2.2 Control structure

RISE conducts a batch, hill-climbing search through the space of rule sets, i.e. (1) it uses all the training examples at once, and (2) it maintains at each step only the best rule set found so far. The initial rule set is simply the training set. A heuristic value is associated with each possible rule set (the computation of this value is the subject of the next subsection). At each step RISE attempts to improve the heuristic value of the current rule set by tentatively generalizing each rule, i.e. it performs specific-to-general search. A rule is generalized by generalizing one of its antecedents. A nominal antecedent is generalized by assigning it the value $*$. The generalization of numeric attributes is described in a later section. For each rule, RISE tentatively generalizes each antecedent, and selects the generalization that results in the greatest heuristic value for the rule set. This generalization is then adopted if the

```
Input: ES is the training set.

Procedure RISE (ES)

Let RS be ES.
Compute H(RS).
Repeat
    For each rule in RS,
        Let R be the current version of the rule.
        For each attribute-value pair AV in R's antecedent,
            Try generalizing AV, and
            Compute the resulting H'(RS).
        Select the AV whose generalization
            results in the greatest H'(RS).
        If this H'(RS) >= H(RS),
        Then Generalize AV,
             If R' is identical to another rule in RS,
             Then delete R' from RS,
             Replace H(RS) with the maximum H'(RS),
Until no increase in H is obtained.
Return RS.
```

Table 1: The RISE algorithm.

corresponding heuristic value is greater than or equal to the current one. Generalizing in the case of equal heuristic values corresponds to a preference for shorter rules, i.e. to a direct application of Occam's razor. If the new rule is identical to an existing one, it is deleted. Search terminates when no generalization yields further improvement. This procedure can be summarized in the pseudo-code presented in Table 1, where `H(.)` is the heuristic evaluation function.

2.3 Heuristics

RISE currently considers the heuristic value of a rule set to be the sum of the heuristic values of the rules that comprise it, and the heuristic value of an individual rule to be of the form:

$$h(r) = p(r) - (1 - \eta)\, S\, n(r)$$

where r is the rule, $p(r)$ is the number of examples covered by the rule whose consequents are identical to the rule's ("positive" examples) , $n(r)$ is the number of examples covered by the rule whose consequents are different from the rule's ("negative" ones), S is the sample (training set) size, and $\eta \in [0, 1]$ is a noise tolerance coefficient.

The idea behind η is that in noisier domains rules should be allowed to cover a greater proportion of negative examples; in noiseless domains this proportion should be 0. The ideal "coarseness" of the rules is domain-dependent and there is no universally good level for it; hence the use of a parameter. If $\eta = 0$, the S factor in h ensures that no number of positive examples will make up for covering even a single negative example, and RISE is thus completely noise intolerant, i.e. any generalization that covers one or more negative examples will necessarily be rejected. If $\eta = 1$ RISE is completely noise tolerant, i.e. all rules will be generalized to a null left-hand side and the most frequent class will always be selected by the classification procedure below. In practice, depending on domain characteristics, the useful range for η will be more restricted. A notable point is $\eta = 1 - \frac{1}{S}$; for this value the heuristic becomes simply $h(r) = p(r) - n(r)$.

Note that in two-class (single-concept) domains RISE forms rules for both classes, and so increasing the value of η does not necessarily increase the fraction of the instance space covered by the concept.

2.4 Classification

To classify a test example RISE matches it with every rule in the induced rule set. Since more than one rule may match the example, a policy is needed to resolve conflicts. RISE uses weighted voting among the successful rules. Currently each rule's weight is simply its heuristic value, on the assumption that rules with higher heuristic values will be better classifiers. With the heuristic described above, this results in a preference for rules that (1) cover many examples, i.e. have substantial statistical support, and (2) cover few examples that are not of the predicted class, i.e. are accurate classifiers. The test example is assigned to the most voted class.

When no rule matches, RISE defaults to a best-match policy, instead of the usual policy of selecting the most frequent class. The number of attributes along which each rule matches the example is counted, with $*$-valued attributes always producing a match, and numeric attributes producing a match when the example's value falls in the specified range. If there is a tie for the highest match score the corresponding rules are selected as the voting set, otherwise the single best rule wins. Perfect match is the special case where the highest match score is equal to the number of attributes. This is easily implemented and may improve accuracy, as well as forming a bridge to instance-based (IBL) algorithms [1]: with appropriate match metrics and voting weights, RISE will act like an IBL algorithm when the voting rules are ungeneralized examples.

With the current version of RISE and the domains used so far, however, no-match situations are extremely rare, so this feature has little or no impact on the results presented in this report.

2.5 Handling of numeric and missing values

Most real-world domains have a mixture of nominal and numeric values; extending RISE to handle the latter is therefore necessary to make it useful. The approach taken is the following. Initially, for each numeric attribute, all occurring values are collected and sorted into ascending order. The midpoint of each pair of consecutive values is then computed, and an ordered list of these midpoints stored, as done in C4.5 [14]. At each generalization step, two generalizations of the attribute are considered: replacing its upper limit by the next higher point on the list (or ∞ if there is none), and replacing its lower limit by the next lower point (or $-\infty$) . The results of each are then treated in the same way as symbolic generalizations.

Missing values also occur frequently in real domains. RISE currently handles them simply by taking them to be $*$, i.e. RISE considers missing attributes to be pre-generalized. This means that a rule generalized from an example with missing attributes will match all values of those attributes, and therefore will match each value with the frequency with which it appears in the testing set. Similarly, a test example with a missing value for one attribute is considered to match all rules along that attribute. This is a good "first-order" approximation, but more sophisticated strategies are possible.

3 Time complexity of RISE

In this section, an upper bound for the time complexity of RISE is derived, i.e. RISE is guaranteed to find a local optimum of the evaluation function in this time. This will be done first for the purely nominal version of the algorithm, and then extended to the numeric case. Let E represent the number of examples in the training set, and A the number of attributes used to describe each example. The initialization phase of the algorithm consists simply in copying the examples to the rules, which takes $O(EA)$ time, and computing the initial value of the heuristic, which takes $O(E)$ time because it is known to be 1 for each rule (assuming a consistent dataset).

The heart of the algorithm is computing the heuristic evaluation of a tentatively generalized rule, and this takes $O(EA)$ time because it involves comparing the rule with each example along each attribute (at worst) to determine if it matches. In each `repeat` cycle generalization is attempted for each attribute of each rule, i.e. $O(EA)$ times, and the cycle repeats at worst until all attributes have been removed. Since each rule is generalized independently of the others, i.e. the decision whether to generalize a rule depends only on the examples covered by its pre-generalized and post-generalized versions, each `repeat` cycle is guaranteed to remove one attribute

from each rule, excepting those rules for which generalization has stopped completely because there are no more useful removals to be made. Thus the cycle will be repeated at most $O(A)$ times. Another way to look at this is to consider that all rules are generalized in parallel, with each `repeat` cycle performing $O(E)$ generalizations. Unless this process is stopped earlier for some rules, after $O(A)$ cycles all antecedents of all rules will have been removed.

The time needed to find and delete duplicates of a rule is $O(EA)$, and this is done at worst only $O(E)$ times in each of the $O(A)$ cycles. The total complexity is therefore $O(EA + EA \times EA \times A + EA \times E \times A) = O(E^2A^3)$. This is competitive with e.g. CN2. (Note that the computations in [4] are only for the basic step of the algorithm, which is embedded in loops that may run $O(EA)$ in the worst case.) In particular, RISE is quadratic in the number of examples, meaning it can be applied directly to all but very large databases. Average-case time is also likely to be substantially smaller than the worst case, due to the unlikelihood of some of the assumptions above.

Let V be the average number of observed values per attribute. The introduction of numeric attributes causes the initialization phase to become $O(EA + AV \log V)$ due to scanning the examples, sorting values (assumed $O(V \log V)$ for each attribute) and computing midpoints. The search phase becomes $O(E^2A^3V)$, since each numeric attribute may now take at worst $O(V)$ steps to generalize instead of a single one. The latter term will always dominate, since by definition $V \leq E$, and so worst-case time will simply increase by a factor of V. The practical effect of this will depend on the domain.

4 Empirical evaluation

With the goal of empirically evaluating the usefulness of the "conquering without separating" strategy, RISE was compared with a current "separate-and-conquer" rule induction algorithm on a number of natural and artificial domains. CN2 [4] was chosen for this purpose because it is probably the most extensively evaluated noise-tolerant algorithm of this type. A recent, enhanced version was used [3]. All options were set so as to maximize similarity to RISE: unordered rules, star size 1, and 0 significance threshold. Results in [3] indicate that induction and use of unordered rules tend to significantly increase performance over decision lists. Star size 1 implements a hill-climbing search, losing some power relative to a beam one, but in practice this seldom decreases accuracy significantly; in fact, in some domains the results reported here for CN2 are the best ever. In the new version of CN2 a significance threshold of 0 tends to produce the highest accuracies. Laplacian accuracy was chosen as the heuristic, since it is the one that yields the best results [3]. All options save the star size are the defaults.

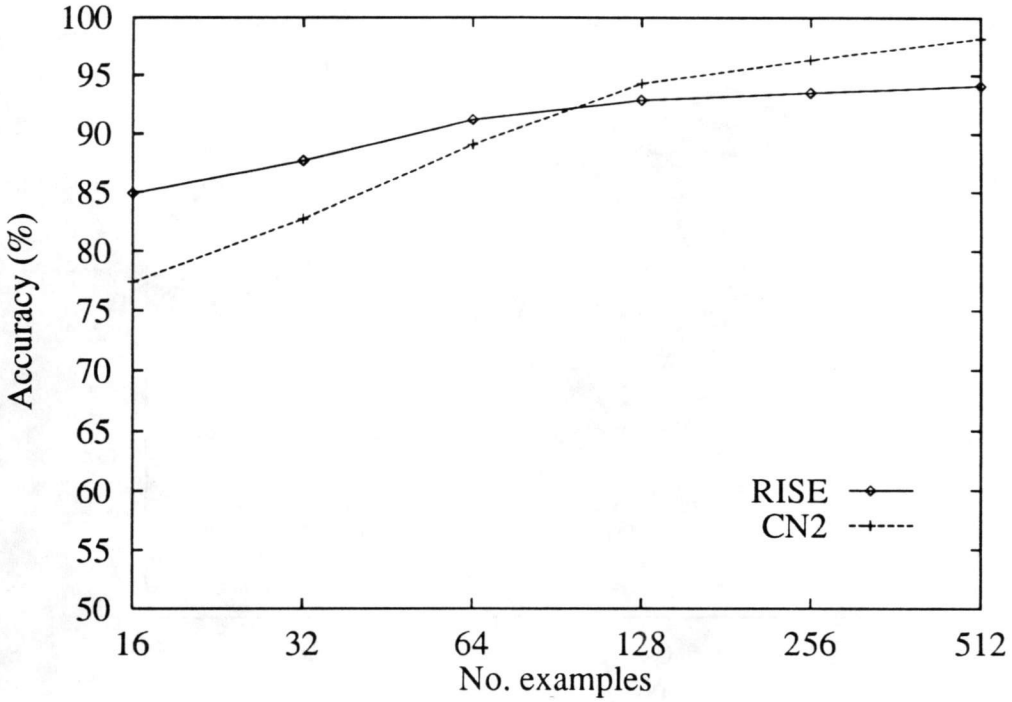

Figure 1: Performance on the task $Concept = A \vee BC \vee DEF \vee GHIJ$.

4.1 Artificial domains

RISE and CN2 were initially tested on two boolean functions of 10 attributes, one thought to be easier for current rule induction algorithms ($Concept = A \vee BC \vee DEF \vee GHIJ$) and one harder ($Concept = ABC \vee BCD \vee CDA \vee DAB$, with 6 irrelevant attributes). Learning was carried out on sets of 16, 32, 64, 128, 256 and 512 examples, randomly chosen with uniform probability from the 1024 possible, and the induced rules were tested on the remaining examples. This procedure was repeated 20 times for each domain. The averaged results are presented graphically in Figs. 1 and 2. All nonzero differences in performance between RISE and CN2 are statistically significant at the 5% level, using a one-tailed paired t test.[1]

Overall RISE and CN2 seem to perform similarly in the simpler case, with RISE faring better when there are fewer examples and conversely when there are many, but RISE achieves consistently higher accuracy in the harder concept, and by a wider margin. In order to verify this an even harder domain was tried, $Concept = ABCD \vee \overline{BCD}E \vee CD\overline{EA} \vee \overline{D}EA\overline{B} \vee \overline{EABC}$ with 5 irrelevant attributes (notice that all attributes have zero information gain). The results, averages of 20 runs as before

[1]Right tail if the difference was positive, left one if negative.

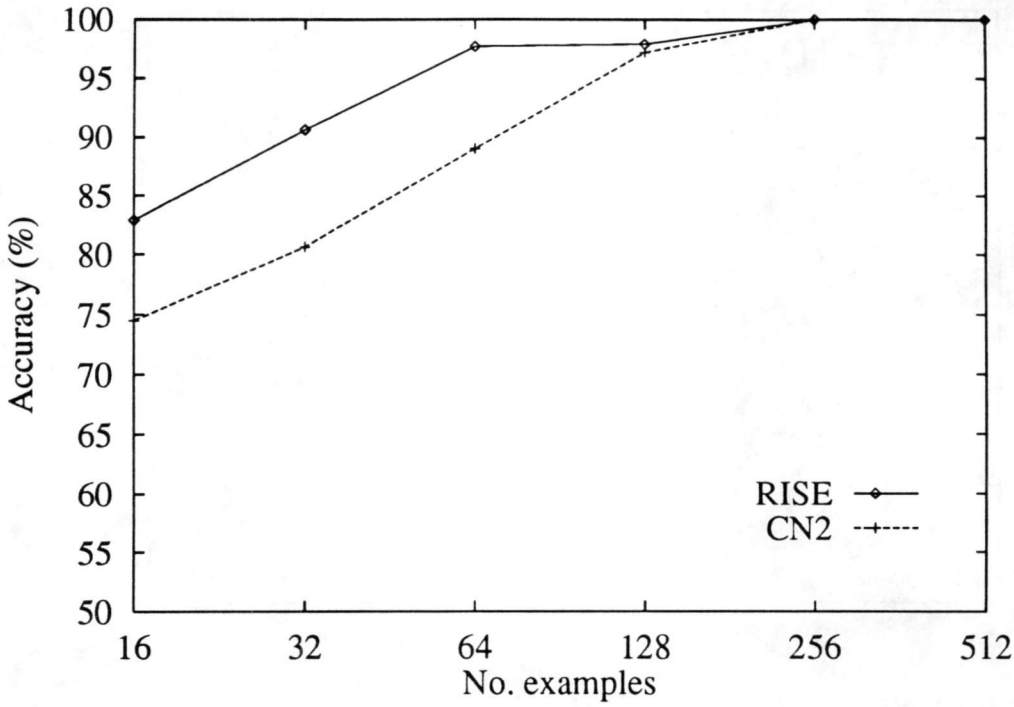

Figure 2: Performance on the task $Concept = ABC \vee BCD \vee CDA \vee DAB$, with 6 irrelevant attributes.

and significant everywhere at the 5% level, are shown in Fig. 3. The difference in performance between RISE and CN2 has again increased substantially, leading to the hypothesis that conquering without separating becomes more advantageous as domain difficulty increases.

4.2 Natural domains

Tests were conducted in 16 domains from the UCI repository [9] to determine if this behavior is observable in practical situations. Selection of domains obeyed the following criteria: prefer widely used domains, provide a selection of symbolic, numeric and mixed domains, and provide a spectrum of difficulty. The domains chosen were: breast cancer recurrence, credit screening, congressional voting records, contact lens fitting, hepatitis, iris flower classification, king-rook-versus-king-pawn chess endgames, labor negotiations (as supplied by Quinlan with C4.5), lung cancer, lymphography, Pima Indians diabetes, primary tumor detection, shuttle landing control, soybean diseases (the small dataset), and wine classification. The voting domain was tried in two forms: with and without the "physician fee freeze" attribute. Remov-

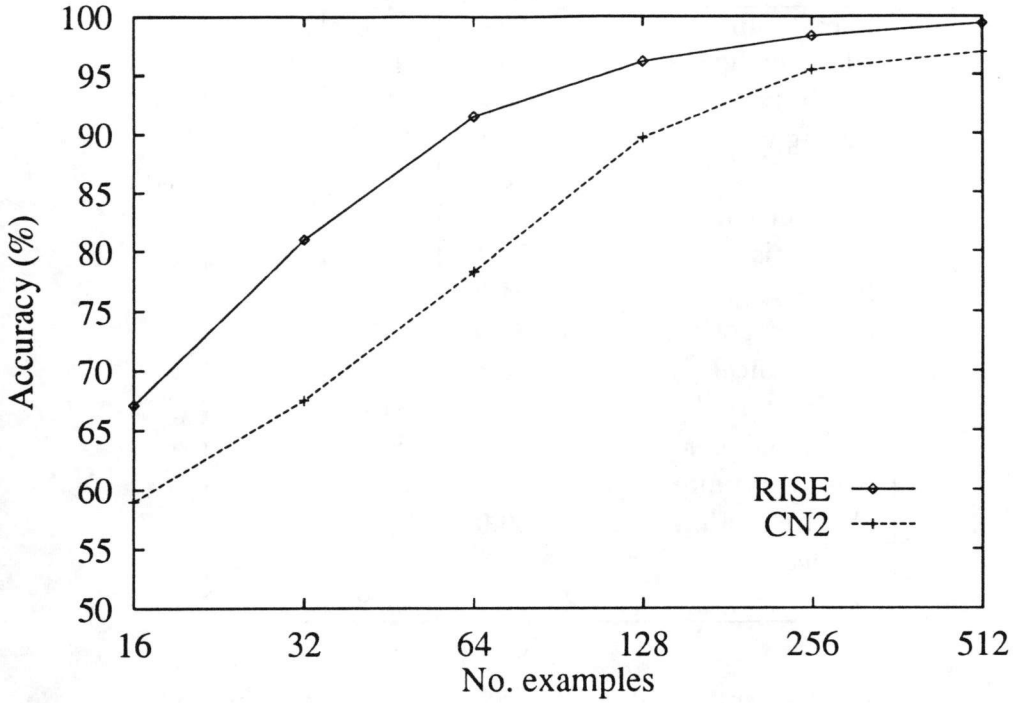

Figure 3: Performance on the task $Concept = ABCD \vee \overline{BCD}E \vee CD\overline{EA} \vee \overline{D}EA\overline{B} \vee \overline{EABC}$, with 5 irrelevant attributes.

ing this attribute causes the difficulty of the domain to increase substantially [2, 5], making possible an evaluation of performance variation with difficulty while holding other factors constant.

Ten-fold cross-validation was performed for RISE and CN2, using the same training and test sets for the two at each step. The following policy was used to set RISE's noise tolerance parameter η: in domains with inconsistent examples, i.e. domains that are guaranteed *a priori* to be noisy, it was set to $1 - \frac{1}{S}$, yielding $h = p - n$ as explained in a previous section; otherwise it was set to 0.[2] The results obtained are summarized in Table 2, where an asterisk indicates that the difference between RISE and CN2 was significant at the 5% level using a one-tailed paired t test.[3]

The results here are not as clear; in most cases, there is no significant difference between RISE and CN2. Two observations, however, support the hypothesis above that RISE improves relative to CN2 as the complexity of the domain increases. One comes from the voting domain, where CN2 performs significantly better in the "easy"

[2]Several other variations of RISE have been tested, often with superior results, but they are not reported here.

[3]See the footnote on the t test in the previous subsection.

Domain	RISE	CN2	Signif.
Breast cancer	69.9	71.6	
Chess endgames	99.1	95.4	*
Credit screening	80.6	80.2	
Voting records	90.8	96.1	*
Voting records modif.	89.2	90.6	
Contact lenses	68.3	65.0	
Hepatitis	79.4	80.7	
Iris	94.0	19.4	*
Labor negotiations	80.0	74.7	
Lung cancer	40.8	39.2	
Lymphography	77.0	81.6	
Pima diabetes	59.0	65.1	
Primary tumor	42.5	37.2	*
Shuttle landing	30.0	40.0	
Soybean	98.0	98.0	
Wine	82.1	39.9	*

Table 2: Results in domains from the UCI repository.

version, but suffers a more pronounced drop when the "hard" one is tried, resulting in no significant difference in this case. The other observation comes from the trio of medical domains (lymphography, breast cancer, and primary tumor) where CN2 was originally tested [4] and that is probably the most widely used in the machine learning literature: there is no significant difference in the "easier" domain (lymphography) or the intermediate one (breast cancer), but RISE is significantly better in the "harder" one (primary tumor). Overall RISE does better than CN2, both on average and in number of significant wins; this remains true even if the iris domain, where CN2's performance was anomalously low (only the default rule was induced), is omitted.

5 Discussion and future work

The preliminary results in artificial domains presented here suggest that "conquering without separating" and "separate-and-conquer" may perform similarly in simple domains, but the former may be substantially better in harder ones. Observations in natural domains do not yet unambiguously confirm this, however. This could be due to several reasons: the domains studied may in fact all be at the easier end of the spectrum, or further effects may be at work in them that are absent from the idealized domains. Also, the type of specific-to-general search used in the current version of RISE may be disadvantageous, because for each single rule the effective sample size

(i.e. the number of examples covered by it and its immediate generalizations) is initially small, and this may cause early bad decisions to lock the system out of better areas of the search space, even if good decisions are made later on.

Performance may be improvable by using more sophisticated heuristics, by generalizing more than one attribute at a time, or by a combination of both. This would overcome the current problem that generalization decisions are "blind" if there are no examples in the immediate vicinity of the rule (i.e. differing from it by at most one attribute). This problem is especially acute at the beginning.

Another direction is to implement and test a general-to-specific version of RISE, while maintaining the "conquering without separating" strategy. With the goal of discriminating between the two strategies, it would also be interesting to minimally modify an existing system like CN2 so as to make it "conquer without separating," and compare the results with the original's.

It will also be interesting to measure the size and accuracy of the individual disjuncts produced, and to relate performance to quantitative measures of concept difficulty, as done in e.g. [15].

The current, far from optimized version of RISE is somewhat slower than the C implementation of CN2, but still quite competitive. After eliminating the more obvious sources of inefficiency in the code, speed may still be improvable without compromising accuracy by the use of sampling techniques, eg. as done by GOLEM [8].

In any case, if accuracy and speed are comparable, then conquering without separating is probably the strategy of choice, since it yields simpler algorithms. It also lends itself better to parallelization, because the search for each rule is independent of the others, and can therefore be carried out on a separate processor, deferring the deletion of duplicates until the end of the search.

Acknowledgments

This work was partly supported by JNICT/Programa Ciência and Fulbright scholarships. The author is grateful to Dennis Kibler and Mike Pazzani for many helpful comments and suggestions, to Peter Clark for supplying CN2, and to M. Zwitter and M. Soklic of the University Medical Centre, Ljubljana, for supplying the lymphography, breast cancer and primary tumor datasets.

References

[1] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] W. Buntine and T. Niblett. A further comparison of splitting rules for decision tree induction. *Machine Learning*, 8:75–86, 1992.

[3] P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. In *Proc. EWSL-91*, pages 151–163, 1991.

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

[5] R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

[6] R. C. Holte, L. E. Acker, and B. W. Porter. Concept learning and the problem of small disjuncts. In *Proc. IJCAI-89*, pages 813–818, 1989.

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

[8] S. Muggleton and C. Feng. Efficient induction of logic programs. In *Proc. 1st Workshop on Algorithmic Learning Theory*, pages 368–381, 1990.

[9] P. M. Murphy and D. W. Aha. UCI repository of machine learning databases. Machine-readable data repository, University of California, Department of Information and Computer Science, Irvine, CA, 1992.

[10] G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 3:71–99, 1990.

[11] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[12] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[13] J. R. Quinlan. Improved estimates for the accuracy of small disjuncts. *Machine Learning*, 6:93–98, 1991.

[14] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[15] H. Ragavan and L. Rendell. Lookahead feature construction for learning hard concepts. In *Proc. 10th Machine Learning Conf.*, pages 252–259, 1993.