Using Partitioning to Speed Up Specific-to-General Rule Induction

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

Abstract

RISE (Domingos 1995; in press) is a rule induction algorithm that proceeds by gradually generalizing rules, starting with one rule per example. This has several advantages compared to the more common strategy of gradually specializing initially null rules, and has been shown to lead to significant accuracy gains over algorithms like C4.5RULES and CN2 in a large number of application domains. However, RISE's running time (like that of other rule induction algorithms) is quadratic in the number of examples, making it unsuitable for processing very large databases. This paper studies the use of partitioning to speed up RISE, and compares it with the well-known method of windowing. The use of partitioning in a specific-to-general induction setting creates synergies that would not be possible with a general-to-specific system. Partitioning often reduces running time and improves accuracy at the same time. In noisy conditions, the performance of windowing deteriorates rapidly, while that of partitioning remains stable.

Introduction

RISE (Domingos 1995; in press) is a rule induction algorithm that searches for rules in a specific-to-general direction, instead of the general-to-specific one used by most rule learners. This has several advantages, among them the ability to detect with confidence a higher level of detail in the databases, and a reduction of sensitivity to the fragmentation (Pagallo & Haussler 1990) and small disjuncts problems (Holte, Acker, & Porter 1989). In a study comparing RISE with several induction algorithms (including C4.5RULES and CN2) on 30 databases from the UCI repository (Murphy & Aha 1995), RISE was found to be more accurate than each of the other algorithms in about two-thirds of the databases, in each case with a confidence of 98% or better according to a Wilcoxon signed-ranks test (DeGroot 1986), and had the highest average accuracy and highest rank.

RISE's running time, like that of previous algorithms, is quadratic in the number of examples, and thus the question arises of whether it is possible to reduce this time to linear without compromising accuracy. This paper proposes, describes and evaluates the application of partitioning to RISE. This raises issues and opportunities that are not present in general-to-specific systems. Partitioning is also compared with windowing, a commonly used speedup method (Catlett 1991; Quinlan 1993).

The structure of the paper is as follows. The next three sections describe pure RISE, RISE with partitioning, and RISE with windowing. This is followed by an empirical study comparing the three, and discussion of the results.

The RISE Algorithm

In RISE, each example is a vector of attribute-value pairs, together with a specification of the class to which it belongs; attributes can be either nominal (symbolic) or numeric. Each rule consists of a conjunction of antecedents and a predicted class. Each antecedent is a condition on a single attribute, and there is at most one antecedent per attribute. Conditions on nominal attributes are equality tests; for numeric attributes they take the form of allowable intervals. An example can be viewed as a maximally specific rule, with conditions on all attributes and degenerate (point) intervals for numeric attributes. A rule is said to *cover* an example if the example satisfies all of the rule's conditions; a rule is said to *win* an example if it is the nearest rule to the example according to the distance metric that will be described below.

The RISE algorithm is summarized in Table 1. RISE searches for "good" rules in a specific-to-general fashion, starting with a rule set that is the training set of examples itself. RISE looks at each rule in turn, finds the nearest example of the same class that it does not already cover (i.e., that is at a distance greater than zero from it), and attempts to minimally generalize the rule to cover it, by dropping conditions (in the case of differing symbolic attributes) and/or expanding intervals (for numeric attributes). This procedure is outlined in Table 2. If the change's effect on global accuracy is positive, it is retained; other-

Table 1: The RISE algorithm.

```
Input: $ES$ is the training set.

Procedure RISE ($ES$)

Let $RS$ be $ES$.
Compute $Acc(RS)$.
Repeat
    For each rule $R$ in $RS$,
        Find the nearest example $E$ to $R$ not already
            covered by it (and of the same class).
        Let $R' =$ Most_Specific_Generalization($R$, $E$).
        Let $RS' = RS$ with $R$ replaced by $R'$.
        If $Acc(RS') \geq Acc(RS)$
        Then Replace $RS$ by $RS'$,
                If $R'$ is identical to another rule in $RS$,
                Then delete $R'$ from $RS$.
Until no increase in $Acc(RS)$ is obtained.
Return $RS$.
```

Table 2: Generalization of a rule to cover an example.

```
Inputs: $R = (a_1, a_2, \ldots, a_A, c_R)$ is a rule,
        $E = (e_1, e_2, \ldots, e_A, c_E)$ is an example.
$a_i$ is either True, $x_i = r_i$, or $r_{i,lower} \leq x_i \leq r_{i,upper}$.

Function Most_Specific_Generalization (R, E)

For each attribute $i$,
    If $a_i =$ True then Do nothing.
    Else if $i$ is symbolic and $e_i \neq r_i$ then $a_i =$ True.
    Else if $e_i > r_{i,upper}$ then $r_{i,upper} = e_i$.
    Else if $e_i < r_{i,lower}$ then $r_{i,lower} = e_i$.
```

wise it is discarded. Generalizations are also accepted if they appear to have no effect on accuracy; this reflects a simplicity bias. This procedure is repeated until, for each rule, attempted generalization fails. The accuracy $Acc(RS)$ is measured using an approximate leave-one-out methodology: when attempting to classify an example, the corresponding rule is left out, unless it already covers other examples as well. With careful optimization to avoid redundant computations, RISE's worst-case time complexity has been shown to be quadratic in the number of examples and the number of attributes, which is comparable to that of other commonly-used rule induction algorithms (Domingos in press).

At performance time, and when gauging the effect of a rule change on global accuracy during learning, classification of each test example is performed by finding the nearest rule to it, and assigning the example to the rule's class. Thus RISE's behavior is in many ways similar to that of nearest-neighbor or instance-based algorithms (Aha, Kibler, & Albert 1991). The distance measure used in RISE is a combination of Euclidean distance for numeric attributes, and a simplified version of Stanfill and Waltz's value difference metric for symbolic attributes (Stanfill & Waltz 1986).

When two or more rules are equally close to a test example, the rule that was most accurate on the training set wins. So as to not unduly favor more specific rules, the Laplace-corrected accuracy is used (Niblett 1987):

$$LAcc(R) = \frac{N_{corr}(R) + 1}{N_{won}(R) + C} \tag{1}$$

where R is any rule, C is the number of classes, $N_{won}(R)$ is the total number or examples won by R, $N_{corr}(R)$ is the number of examples among those that R correctly classifies, and C is the number of classes. The effect of the Laplace correction is to make the estimate of a rule's accuracy converge to the "random guess" value of $1/C$ as the number of examples won by the rule decreases. Thus rules with high apparent accuracy are favored only if they also have high statistical support (i.e., if that apparent accuracy is not simply the result of a small sample).

Partitioning

In the partitioning speedup approach (Chan & Stolfo 1995b), the training data is divided into a number of disjoint subsets, and the learning algorithm is applied to each in turn. The results of each run are combined in some fashion, either at learning or at classification time. In RISE, partitioning is applied by predetermining a maximum number of examples e_{max} to which the algorithm can be applied at once (100 by default). When this number is exceeded, the training set is randomly divided into $\lceil e/e_{max} \rceil$ approximately equal-sized partitions, where e is the total number of training examples. RISE is then run on each partition separately, but with an important difference relative to a direct application: the rules grown from the examples in partition p are not evaluated on the examples in that partition (see Table 1 and accompanying discussion), but on the examples in partition $p + 1$ (modulo the number of partitions). This should help combat overfitting, and the resulting improvement in accuracy may partly offset the degradation potentially caused by using smaller training sets. It is not possible in general-to-specific algorithms, where there is no connection between a specific rule and a specific example.

Because the number of partitions grows linearly with the number of training examples, and RISE's quadratic factor is confined to the examples within each partition and thus cannot exceed a given maximum (e.g., 100^2 if $e_{max} = 100$), the algorithm with partitioning is guaranteed a linear worst-case running time. However, depending on e_{max}, the multiplicative constants can become quite large.

Two methods of combining the results of induction on the individual partitions have been implemented and empirically compared. In the first, all the rule sets produced are simply merged into one, which is output by the learning phase. In the second, the rule sets are kept separate until the performance phase, and each partition classifies the test instance independently. A winning class is then assigned to the example by voting among the partitions, with each partition's weight being the Laplace accuracy of the rule that won within it (Equation 1). The second method was found to achieve consistently better results, and was therefore adopted. More sophisticated combination methods based on Bayesian theory are currently being studied, but have so far yielded inferior results. Many other combination schemes are possible (e.g., (Chan & Stolfo 1995b)).

Windowing

Windowing is applied to RISE in a fashion similar to C4.5's (Quinlan 1993), and proceeds as follows. Initially, only $2\sqrt{e}$ examples randomly extracted from the training set are used for learning. This sample is stratified (i.e., it contains approximately equal proportions of all classes); this makes it possible to still learn classes that have few representatives in the original training set. If the remaining training examples are correctly classified by the resulting rule set, this set is output. Otherwise, the misclassified examples are added to the initial example set, and this process repeats until it produces no improvement in accuracy on two successive expansions. This policy of requiring two successive failures to stop has been verified empirically to lead to better results in the case of RISE than the policy followed by C4.5, of stopping as soon as there is no improvement in accuracy. The latter is more prone to premature stopping (i.e., stopping at a local minimum of the accuracy improvement curve).

In the best case, only $O(\sqrt{e})$ examples are used, and the algorithm becomes linear in the training set size. In the worst case, the window grows to include the entire training set (or nearly so), and the process is more costly than learning directly on that set. This is particularly likely in noisy domains, where it has been observed to lead to serious performance degradation in the case of C4.5 (Catlett 1991). To avoid this, the implementation used in RISE also limits the number of times the window is grown to a prespecified maximum (5 by default). This should help prevent the system from attempting to fit the noise in domains where this is a problem, and has been found empirically to sometimes achieve large reductions in running time compared to the unlimited-expansion version, without seriously affecting accuracy.

Empirical Evaluation

The two speedup methods were tested on seven of the UCI repository's largest databases (Murphy & Aha 1995) (in increasing order of size: credit screening (Australian), Pima diabetes, annealing, chess endgames (kr-vs-kp), hypothyroid, splice junctions, and mushroom). Of these, at least one (Pima diabetes) is thought to be quite noisy, and at least two (chess and mushroom) aer known to be almost entirely noise-free. Partitioning was tested with $e_{max} = 100$, 200, and 500. Ten runs were carried out for each database, in each run randomly dividing the data into two-thirds for training and one-third for testing. The averaged results are shown in Tables 3 (running times) and 4 (accuracies).

Partitioning is quite effective in speeding up RISE. Its running time is (as might be expected) sensitive to the choice of e_{max}, but it appears to increase less than linearly with it. Linear growth would be expected, since, if p is the number of partitions and t is the total running time, $t = O(pe_{max}^2)$, and since $p \simeq e/e_{max}$, $t = O(ee_{max})$, i.e., for a given e, $t \propto e_{max}$. Examination of the rules produced shows that RISE tends to stop earlier within each partition when the partitions are larger, presumably because the additional information available warrants the induction of more specific rules, and while in general-to-specific systems this means that the algorithm will take longer to run because more antecedents will be added, in RISE the opposite is the case, since fewer antecedents will be deleted. This will tend to partly offset the increase in running time due to increasing partition size.

The effect of partitioning on accuracy is more variable than that of windowing. In some domains a trade-off between partition size and accuracy is observed; however, only in the chess domain does increasing e_{max} from 200 to 500 substantially increase accuracy. More interestingly, in the credit, diabetes and splice junctions domains the opposite trend is observed (i.e., partitioning increases accuracy, and smaller partitions more so than larger ones); this may be attributed to the reduction in overfitting derived from inducing and testing rules on different partitions, to the increase in accuracy that can result from combining multiple models (Wolpert 1992; Breiman in press), and possibly to other factors. On the splice junctions dataset, the success of applying partitioning to RISE using a simple combination scheme contrasts with the results obtained by Chan and Stolfo for general-to-specific learners (Chan & Stolfo 1995a). In general, the best partition size should be determined by experimentation on the specific database RISE is being applied to, starting with smaller (and therefore faster) values.

To test partitioning on a larger problem, and obtain a clearer view of the growth rate of its running time compared to that of RISE and windowing, experiments were conducted on NASA's space shuttle database. This database contains 43500 training examples from one shuttle flight, and 14500 test examples from a different flight. Each example is described by nine numeric attributes obtained from sensor readings,

Table 3: Experimental results: running times (in minutes and seconds).

Database	RISE	Windowing	Partitioning		
			$e_{max} = 100$	$e_{max} = 200$	$e_{max} = 500$
Credit	4:31	3:21	1:37	1:11	4:38
Pima diabetes	4:15	6:20	1:32	1:13	2:47
Annealing	4:26	2:44	1:43	2:33	2:17
Chess	33:26	10:40	3:10	6:04	12:06
Hypothyroid	105:23	14:46	5:08	10:42	24:06
Splice junctions	110:39	51:28	5:22	12:45	25:48
Mushroom	70:07	10:07	5:55	7:26	14:32

Table 4: Experimental results: accuracies and standard deviations.

Database	RISE	Windowing	Partitioning		
			$e_{max} = 100$	$e_{max} = 200$	$e_{max} = 500$
Credit	82.6±1.5	83.6±1.5	86.4±1.9	86.4±1.5	82.6±1.6
Pima diabetes	71.6±2.5	70.6±2.7	74.4±2.1	73.6±3.3	72.8±2.6
Annealing	97.5±0.9	98.0±1.0	93.6±1.6	96.1±1.6	96.5±1.1
Chess	98.4±0.6	98.4±0.7	94.5±0.5	95.2±0.6	96.6±0.9
Hypothyroid	97.9±0.2	97.5±0.5	97.0±0.3	97.5±0.3	97.9±0.4
Splice junctions	92.5±0.8	92.8±0.7	95.0±0.7	94.6±0.7	94.7±0.6
Mushroom	100.0±0.0	100.0±0.0	98.9±0.1	99.5±0.3	99.8±0.1

and there are seven possible classes, corresponding to states of the shuttle's radiators (Catlett 1991). The goal is to predict these states with very high accuracy (99–99.9%), using rules that can be taught to a human operator.

Figure 1 shows the evolution of running time with the number of examples for RISE, RISE with partitioning (using $e_{max} = 100$), and RISE with windowing, on a log-log scale. Recall that on this type of scale the slope of a straight line corresponds to the exponent of the function being plotted. Canonical functions approximating the asymptotic curves for RISE and RISE with partitioning are also shown.[1] Partitioning's running time grows linearly with the number of examples, as expected, and is quickly dwarfed by that of RISE, which is approximately quadratic. On the full training database, RISE consumes over a week of CPU time, while partitioning takes less than an hour. Partitioning's accuracy (not shown) lags slightly behind RISE's (0.55% on average). Partitioning is also much faster than windowing, whose asymptote is unclear.

The shuttle data is known to be relatively noise-free. To investigate the effect of noise, the three algorithms (pure RISE, partitioning and windowing) were also applied after corrupting the training data with 20% class noise (i.e., each class had a 20% probability of being changed to a random class, including itself). The learning time curves obtained are shown in Figure 2, again on a log-log scale and with approximate asymptotes shown. The time performance of windowing degrades markedly, making it worse than the pure algorithm for all training set sizes greater than 500. In contrast, partitioning remains almost entirely unaffected. Noise reduces the accuracy of pure RISE and windowing by 3 – 8%, with the smaller differences occurring for larger training set sizes. (Recall that noise was added only to the training set.) The accuracy of partitioning is barely affected, making it consistently more accurate than pure RISE at this noise level.

An interesting observation is that noise substantially reduces RISE's running time, even though it remains much larger than that obtained with partitioning. This may be attributed to the difference between specific-to-general and general-to-specific systems already discussed: noise tends to make rule induction algorithms produce more specific rules, which take less time to induce in RISE and more in systems like C4.5RULES (which, in addition, may then prune back those rules, further adding to their running time; in RISE pruning and initial induction are the same operation). This means that RISE may be more appropriate than general-to-specific systems for noisy databases.

A potential disadvantage of partitioning when compared to windowing is that it sometimes (but not always) tends to produce rule sets that are larger overall, even if the individual rule sets learned from each partition are comparatively small. However, from the point of view of human-comprehensible output (often a desirable goal), this is not necessarily a serious problem,

[1]The constants a and b were chosen so as to make the respective curves fit conveniently in the graph.

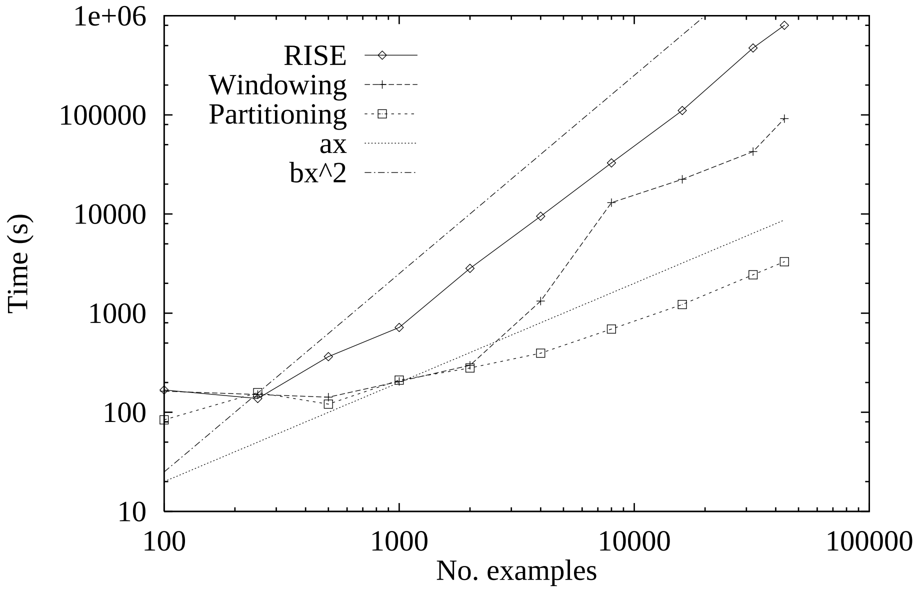

Figure 1: Learning times for the shuttle database.

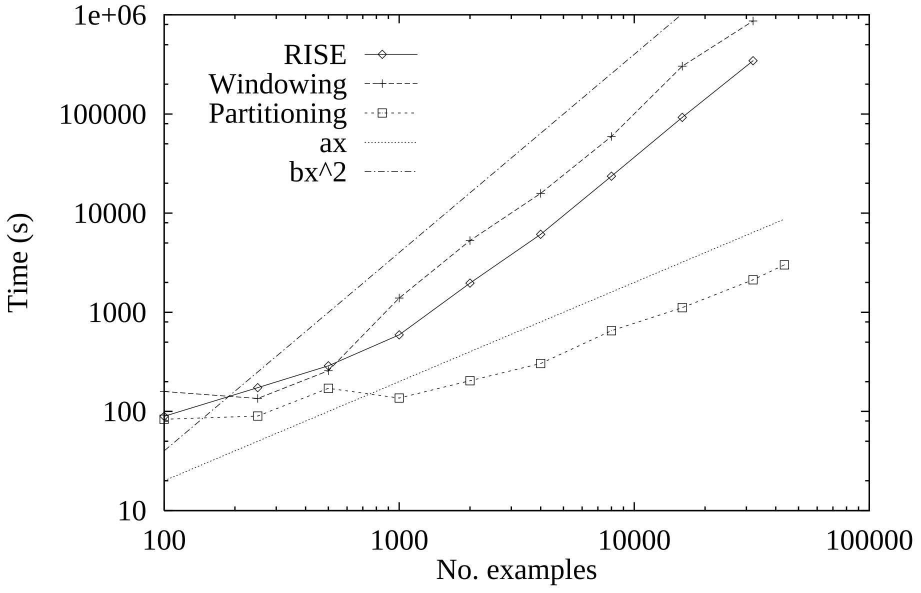

Figure 2: Learning times for the shuttle database with 20% noise.

since understanding can still be gleaned by looking at one of the individual rule sets, or one at a time.

Conclusion

This paper studied the application of partitioning to the RISE rule induction system. Subject to a correct choice of partition size, it was found to effectively reduce the growth of running time with database size, while sometimes improving accuracy. Its superiority over the commonly-used method of windowing is particularly apparent in the case of noisy data.

Directions for future research include testing and developing more sophisticated methods of combining the outputs of the individual partitions (e.g., (Chan & Stolfo 1995b)), automating the selection of partition size, and testing partitioning on a larger variety of larger databases.

Acknowledgments

This work was partly supported by a JNICT/PRAXIS XXI scholarship. The author is grateful to all those who provided the datasets used in the empirical study. Please see the documentation in the UCI Repository for detailed information.

References

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

Breiman, L. Bagging predictors. *Machine Learning.* In press.

Catlett, J. 1991. *Megainduction: Machine Learning on Very Large Databases.* Ph.D. Dissertation, Basser Department of Computer Science, University of Sydney, Sydney, Australia.

Chan, P. K., and Stolfo, S. J. 1995a. A comparative evaluation of voting and meta-learning on partitioned data. In *Proceedings of the Twelfth International Conference on Machine Learning*, 90–98. Tahoe City, CA: AAAI Press.

Chan, P. K., and Stolfo, S. J. 1995b. Learning arbiter and combiner trees from partitioned data for scaling machine learning. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, 39–44. Montréal, Canada: AAAI Press.

DeGroot, M. H. 1986. *Probability and Statistics.* Reading, MA: Addison-Wesley, 2nd edition.

Domingos, P. 1995. Rule induction and instance-based learning: A unified approach. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1226–1232. Montréal, Canada: Morgan Kaufmann.

Domingos, P. Unifying instance-based and rule-based induction. *Machine Learning.* In press.

Holte, R. C.; Acker, L. E.; and Porter, B. W. 1989. Concept learning and the problem of small disjuncts. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 813–818. Detroit, MI: Morgan Kaufmann.

Murphy, P. M., and Aha, D. W. 1995. UCI repository of machine learning databases. Machine-readable data repository, Department of Information and Computer Science, University of California at Irvine, Irvine, CA.

Niblett, T. 1987. Constructing decision trees in noisy domains. In *Proceedings of the Second European Working Session on Learning*, 67–78. Bled, Yugoslavia: Sigma.

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

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

Stanfill, C., and Waltz, D. 1986. Toward memory-based reasoning. *Communications of the ACM* 29:1213–1228.

Wolpert, D. 1992. Stacked generalization. *Neural Networks* 5:241–259.