Efficient 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 introduces a method for reducing RISE's running time based on partitioning the training set, evaluating rules from one partition on examples from another, and combining the final results at classification time. Partitioning guarantees a learning time that is linear in the number of examples, even in the presence of numeric attributes and high noise. Windowing, a well-known speedup method, is also studied as applied to RISE. In low-noise conditions, both methods are successful in reducing running time whilst maintaining accuracy (partitioning sometimes improves it significantly). In noisy conditions, the performance of windowing deteriorates, while that of partitioning remains stable.

Introduction and Previous Work

Rule induction is one of the major technologies underlying data mining. Given a number of classified examples represented by vectors of symbolic and/or numeric attributes, algorithms like C4.5RULES (Quinlan 1993) and CN2 (Clark & Niblett 1989) produce sets of "if ... then ..." rules that allow us to predict the classes of new examples by performing tests on their attributes. However, the running time of these algorithms is typically quadratic or worse in the number of examples, making it difficult to apply them to the very large databases that are now common in many fields. In C4.5RULES, noise can lead to a cubic running time (Cohen 1995). While some faster variants of rule induction have been proposed (Fürnkranz & Widmer 1994; Cohen 1995), none achieve the ideal goal of linear time. An alternative approach, and the one that is followed in this paper, is to employ some form of sampling, like windowing (Catlett 1991; Quinlan 1993), peepholing (Catlett 1991), or partitioning (Chan & Stolfo 1995). While often (though not always) reducing running time, sampling techniques can sometimes substantially reduce accuracy, and there may be a trade-off between the two. A reduced-accuracy rule set is preferable to a more-accurate one that is never reached due to lack of time; ideally, however, the loss in accuracy should be as small as possible, given the available time.

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). RISE also 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 windowing and partitioning to RISE; in both cases, this raises issues and opportunities that are not present in general-to-specific systems.

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

The RISE Algorithm

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, 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). If the change's effect on the rule set's leave-one-out accuracy on the training set is positive or null, it is retained; otherwise it is discarded. This procedure is repeated until, for each rule, attempted generalization fails.

At performance time, classification of each test example is performed by finding the nearest rule to it, and assigning the example to the rule's class. The distance measure used 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).

Windowing

Windowing is applied to RISE in a fashion broadly 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, or a maximum number of expansions is reached (5 by default).

Partitioning

In the partitioning approach (Chan & Stolfo 1995), 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 pre-determining 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 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 (see section on RISE). 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 1995)).

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). 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 1 (running times) and 2 (accuracies).

Both speedup methods are effective in reducing RISE's running time, generally without seriously affecting accuracy (chess and annealing with partitioning, and diabetes with windowing, are the exceptions). Windowing often has practically no effect on accuracy. Thus, overall this method appears to be more useful in RISE than in decision tree induction (Catlett 1991). This may be due to several factors, including RISE's lower sensitivity to the global proportions of different classes, and its higher resistance to the fragmentation problem, which enables it to correctly approximate class frontiers using fewer examples.

The effect of partitioning on accuracy is more vari-

Table 1: 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 2: 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

able 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. In practice, 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 the algorithms on a larger problem, and obtain a clearer view of the growth rate of their running times, 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, 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.

The learning time curves obtained for RISE, RISE with windowing and RISE with partitioning (using $e_{max}=100$) are shown in Figure 1 using a log-log scale. Approximate asymptotes are also shown. Windowing reduces running time, but its growth appears to remain roughly quadratic; partitioning reduces it to linear, as expected. The accuracy curves (not shown) are very similar for all systems, converging rapidly to very high values (99% by e = 1000, etc.), with partitioning lagging slightly behind the other two.

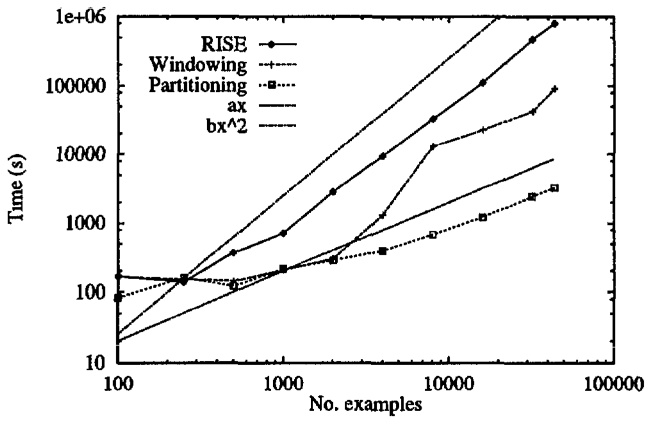

Figure 1: Learning times for the shuttle database.

The shuttle data is known to be relatively noise-free. To investigate the effect of noise, the three algorithms (pure RISE, windowing and partitioning) were also applied after corrupting the training data with

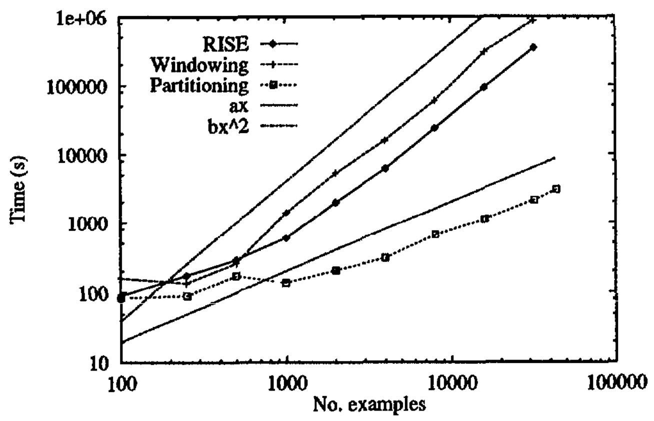

Figure 2: Learning times for the shuttle database with 20% noise.

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. The time performance of windowing degrades markedly, even with the pre-imposed limit on the number of window expansions, becoming a liability for all $e > 500$. 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. Thus partitioning appears to be the speedup method of choice for noisy databases.

Conclusions and Future Work

This paper studied the application to the RISE rule induction system of two speedup methods, windowing and partitioning. With noise-free data, both were found to effectively reduce running time while maintaining accuracy, but only partitioning reduced the growth rate to linear; in noisy conditions, the use of windowing has a negative effect, while partitioning remains unaffected.

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

Acknowledgments

This work was partly supported by a JNICT/PRAXIS XXI scholarship.

References

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. 1995. 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.

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

Cohen, W. W. 1995. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning,* 115–123. Tahoe City, CA: Morgan Kaufmann.

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.

Fürnkranz, J., and Widmer, G. 1994. Incremental reduced error pruning. In *Proceedings of the Eleventh International Conference on Machine Learning,* 70–77. New Brunswick, NJ: Morgan Kaufmann.

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.