Improving the Performance of the RISE Algorithm

Aloísio Carlos de Pina and Gerson Zaverucha

Universidade Federal do Rio de Janeiro, COPPE/PESC, Department of Systems Engineering and Computer Science, C.P. 68511 - CEP. 21945-970, Rio de Janeiro, RJ, Brazil
{long,gerson}@cos.ufrj.br

Abstract. RISE is a well-known multi-strategy learning algorithm that combines rule induction and instance-based learning. It achieves higher accuracy than some state-of-the-art learning algorithms, but for large data sets it has a very high average running time. This work presents the analysis and experimental evaluation of SUNRISE, a new multi-strategy learning algorithm based on RISE, developed to be faster than RISE with similar accuracy.

1 The SUNRISE Algorithm

RISE (Rule Induction from a Set of Exemplars) [2] induces all the rules together. If a generalization of a rule has positive or null effect on the global accuracy, the change is kept. The RISE algorithm is presented in Table 1.

SUNRISE tries to generalize the rules more than once before including them in the rule set and only accepts the changes if the effect on the global accuracy is strictly positive; i.e., a new rule is only added to the rule set if the set achieves a higher accuracy than before its inclusion. The SUNRISE algorithm is presented in Table 2. The fact that SUNRISE does not use Occam's Razor increases the algorithm's speed because it increases the probability of no modification in the rule set after an iteration of the outermost loop Repeat, thus causing the algorithm's stop. Only after k generalizations a rule is evaluated to determine if it must or not belong to the rule set. It makes SUNRISE faster than RISE, since the latter evaluates each generalization made in each rule. The value k is a parameter of the SUNRISE algorithm whose value has to be experimentally determined.

2 Experimental Evaluation

In the experiments, 22 data sets [1] were used to compare the performance of the new algorithm, SUNRISE, to that of the RISE algorithm. The test method used in this research was the paired t test with n-fold cross-validation [5]. To adjust the parameter k, an internal cross-validation was made [5]. The value for k that achieved better performance in most of the data sets was $k \leq 3$. All tests were carried through in a Pentium III 450MHz computer with 64MBytes RAM.

Table 3 presents the running time (training and testing) of each algorithm for each one of the data sets. The two last columns show the results obtained by the SUNRISE

J.-F. Boulicaut et al. (Eds.): PKDD 2004, LNAI 3202, pp. 518–520, 2004.

algorithm when using Occam's Razor, so as to evaluate independently the behavior of the system considering only the effect of the parameter *k*.

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
    covered by it and of the same class of R.
    Let R'=MostSpecificGeneralization(R,E).
    Let RS' = RS with R replaced by R'.
    If Acc(RS') ≥ 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. The SUNRISE algorithm.

```
Input: ES is the training set, k is the SUNRISE
parameter.
Procedure SUNRISE (ES, k)
Let RS be ES.
Compute Acc(RS).
Repeat
  For each rule R in RS,
    Let R' be R.
    Repeat k times
      Find the nearest example E to R' not
      covered by it and of the same class
      of R'.
      R'=MostSpecificGeneralization(R',E).
    Let RS' = RS with R replaced by R'.
    If Acc(RS') > 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 3. Running times (in seconds).

Data Set	RISE	SUNRISE				SUNRISE-OR	
		$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 2$	$k = 3$
Annealing	48.53	3.48	11.83	17.60	19.85	42.82	46.28
Chess endgames	2107.71	261.17	1171.89	1428.81	1744.14	1787.16	1519.58
Credit screening	174.30	5.68	26.94	34.19	42.43	138.04	137.82
DNA promoters	0.69	0.14	0.36	0.36	0.36	0.36	0.36
Echocardiogram	0.72	0.06	0.28	0.28	0.33	0.55	0.55
Glass	2.23	0.31	0.86	0.97	1.24	1.74	1.68
Heart disease	18.97	1.17	4.96	7.00	9.47	18.59	15.40
Hepatitis	5.75	0.20	0.80	0.97	1.24	5.25	3.39
Horse colic	51.34	1.39	6.17	8.26	10.35	37.60	32.82
Iris	0.91	0.09	0.25	0.25	0.31	0.69	0.58
LED	2.05	0.51	1.83	1.83	2.05	2.16	2.32
Liver disease	19.19	1.12	5.57	6.94	9.69	15.46	13.48
Mushroom	4930.79	1258.37	2116.78	2987.38	3868.70	4243.81	4900.24
Pima diabetes	241.17	5.24	33.59	35.51	50.13	182.76	159.74
Post-operative	0.14	0.08	0.08	0.03	0.08	0.08	0.08
Solar flare	4.74	1.39	2.38	2.98	3.42	4.19	4.14
Sonar	11.19	1.74	4.37	4.70	4.81	10.75	6.85
Soybean	6.89	3.09	4.03	4.36	4.91	5.46	5.57
Splice junctions	7043.75	790.24	3522.87	4622.93	5088.37	9785.68	10149.85
Thyroid disease	23767.00	138.92	938.75	1102.65	1461.28	17777.54	16439.19
Wine	4.04	0.31	0.80	1.08	1.30	3.00	2.56
Zoology	0.08	0.03	0.08	0.03	0.08	0.08	0.08

Since SUNRISE presents lower average running time than RISE and achieves good accuracy, the results show that the SUNRISE algorithm seems to be the more indicated choice when working with large data sets. A complete analysis of these experiments can be found in [4].

By adding partitioning [3] to SUNRISE, we expect that the obtained results for large data sets become even better.

3 Conclusions

Besides being faster than RISE, making possible the learning task in data sets intractable with the use of that algorithm, the SUNRISE algorithm showed to be capable to reach an average accuracy as good as that of the RISE algorithm and in some cases superior than that. In 110 tests carried through (not considering $k = 0$), in only 2 of them the SUNRISE algorithm was significantly slower than RISE, achieving a maximum speed-up of 96% in a large data set. In 12 tests the accuracy obtained by SUNRISE was significantly lower than that of RISE, but in 15 tests it was significantly higher.

In an algorithm like RISE, in which the evaluation of the quality of a rule is an expensive process, to generalize a rule more than once and only then to make its evaluation can provide a considerable increase in the algorithm's speed. The use of a more restrictive criterion of acceptance of a generalized rule (turning off the Occam's Razor) provides a final set with less simple rules, but in exchange it can greatly increase the speed of the system. Although these two techniques have been applied to RISE, they could be applied to other bottom-up learning algorithms.

Acknowledgment

We would like to thank Pedro Domingos for providing the source code of the RISE algorithm. The authors are partially financially supported by the Brazilian Research Agency CNPq.

References

1. Blake, C. L., Merz, C. J.: UCI Repository of Machine Learning Databases. Machine-readable data repository. University of California, Department of Information and Computer Science, Irvine, CA (1998) [http://www.ics.uci.edu/~mlearn/MLRepository.html]
2. Domingos, P.: Unifying Instance-Based and Rule-Based Induction. Machine Learning, Vol. 24 (1996) 141–168
3. Domingos, P.: Using Partitioning to Speed Up Specific-to-General Rule Induction. In: Proceedings of the AAAI-96 Workshop on Integrating Multiple Learned Models. AAAI Press, Portland, OR (1996) 29–34
4. de Pina, A. C., Zaverucha, G.: SUNRISE: Improving the Performance of the RISE Algorithm. Technical Report. Federal University of Rio de Janeiro, COPPE/PESC, Department of Systems Engineering and Computer Science, Rio de Janeiro, Brazil (2004) 1–12
5. Mitchell, T. M.: Machine Learning. McGraw-Hill, New York (1997)