Learning Rules From Data

Olcay Taner Yıldız and Ethem Alpaydın

Department of Computer Engineering
Boğaziçi University
İstanbul, 34342, Turkey
yildizol@cmpe.boun.edu.tr, alpaydin@boun.edu.tr

Abstract. We compare two widely-used rule induction methods, C4.5Rules and Ripper, based on three criteria: (1) error rate, (2) number of learned rules; and (3) learning time. Results show that Ripper algorithm has smaller error rate, produces smaller rule sets, and learns faster than C4.5Rules, especially with the improvements we propose.

1 Introduction

A rule contains a conjunction of propositions and a class code which is the label assigned to an instance that is covered by the rule:

$$\text{IF } (x_2 = \text{`red'}) \text{ AND } (x_1 < 1.5) \text{ THEN } C_2$$

The propositions are of the form $x_i = v$ or $x_i < \theta$, depending on respectively whether the input feature x_i is discrete or numeric (Rules can also be extended from propositions to the first-order case, but this is beyond the scope of this paper). A rule set is a list of such rules.

Rule induction algorithms learn a rule set from a training set. Such algorithms have a number of desirable properties:

1. The rule sets they produce are easy to understand, allowing knowledge extraction and validation by application experts. For each class C_i there is a rule set in conjunctive normal form. Each rule set merges rules with OR's, and each rule consists of orthogonal splits related to the features, merged via AND's.
2. These methods are nonparametric in that they assume no a priori form on the model or class densities and fit their complexity to that of data.
3. They learn fast and can be used on very large datasets with a large number of instances.
4. They do their own feature extraction/dimensonality reduction and can be used on datasets with a large number of features.

Because of these reasons, rule induction algorithms are more and more frequently used in many data mining and pattern recognition applications and are preferred over other methods like artificial neural networks.

The two most popular rule induction techniques are C4.5Rules and Ripper. C4.5Rules first trains a decision tree and then converts it to a rule set whereas Ripper learns the

rule set directly. The goal of this paper is to compare these two approaches based on three criteria: (1) error rate, (2) number of learned rules and (3) learning time. This paper is organized as follows: In section 2, we present the algorithms, in section 3 we give the experimental details and results, and in section 4 we conclude.

2 Algorithms

C4.5Rules is based on the C4.5 tree induction algorithm, which is a greedy algorithm that searches for the best split and the best feature at each node in terms of information gain [3]. This corresponds to searching through $N_m - 1$ possible split points for a numeric feature (where N_m is the number of data instances reaching node m) and one split for a discrete feature. If the best feature x_i is numeric, it creates two children ($x_i < \theta$ and $x_i \geq \theta$) and divides the instance space of the parent node into two parts. If the best feature is discrete, it creates k children ($x_i = v_j, j = 1, \ldots, k$), where k is the number of possible values of the feature and divides the instance space of the parent node into k parts. Tree growing continues recursively with the newly generated child nodes until each node has instances from a single class, at which point it is marked as a leaf node and is labelled by the code of that class.

C4.5Rules starts with the C4.5 tree and converts it to a rule set by writing each path from the root to a leaf as a rule. The initial rule set therefore has as many rules as there are leaves in the tree [4]. These generated rules may contain superflous conditions and may cause overfitting. C4.5Rules prunes rules by removing those conditions whose removal do not increase the error rate on the training set. After pruning the conditions in the rules, the best subset of the rules is searched according to a minimum description length criterion (MDL) [5].

Ripper differs from C4.5Rules in that Ripper learns rules from scratch whereas C4.5Rules starts with rules extracted from an unpruned decision tree. Ripper has two phases: In the first phase it builds an initial set of rules and in the second phase it optimizes the rule set k times, typically set to 2 [2].

Ripper builds rules one by one. To each rule, first, propositional conditions are added on at a time, at each step choosing the best one (as in C4.5). The rule is then pruned to alleviate overfitting. Rules are added to a rule set to minimize error and description length. The total description length of a rule set is the number of bits to represent the rules plus the number of bits needed to identify the exceptions to the rules in the training set.

In the second phase, rules in the rule set are optimized. Two alternatives are grown for each rule. The first candidate rule is grown starting with an empty rule (replacement rule), whereas the second candidate rule is grown starting with the current rule (revision rule). These two rules, revision rule and replacement rule, together with the original rule, are compared and the one with the smallest description length is selected and put in place of the original rule.

When adding each condition with a numeric attribute, Ripper (and also C4.5) checks for all possible split positions which takes time when there are many instances. A quicker shortcut is to assume that the two groups are normally distributed and to calculate a

split point analytically in terms of the means and variances of the two groups. Calling these two groups, Left and Right, let us say, m_L, m_R are the means, s_L, s_R are the standard deviations, n_L, n_R are the number of data points of left and right groups respectively, the two candidate best split points are the roots of

$$(s_L^2 - s_R^2)x^2 + 2(m_L s_R^2 - m_R s_L^2)x + (m_R s_L)^2 - (m_L s_R)^2 + 2s_L^2 s_R^2 \log \frac{n_L s_R}{n_R s_L} = 0 \quad (1)$$

If the two groups have the same variance, there is only one root. If the variances are different, there are two roots and the one between the two means is used. If neither of the two roots is between the means or if there are no roots, the midpoint of the two means is chosen as the split point.

To get a more robust estimate of the split point, instances that are further than three standard deviations are considered outliers and are not included in the split point calculation. If the number of examples are small, this is not worth doing; we use the statistical shortcut only if the number of examples is above a certain threshold, typically set to 20. We call this statistical variant the Ripper* method, and as shown in the next section, this makes Ripper much faster without causing any degradation in accuracy or rule set size.

3 Experiments

Table 1. Description of the datasets. K is the number of classes, N is the dataset size, and d is the number of inputs.

Set	K	N	d	Missing	Attr Type	Set	K	N	d	Missing	Attr Type
balance	3	625	5	No	symbolic	ironosphere	2	351	35	No	numeric
breast	2	699	10	Yes	numeric	monks	2	432	7	No	numeric
bupa	2	345	7	No	numeric	mushroom	2	8124	23	Yes	symbolic
car	4	1728	7	No	symbolic	optdigits	10	3823	64	No	numeric
cmc	3	1473	10	No	mixed	pendigits	10	7494	16	No	numeric
credit	2	690	16	No	mixed	pima	2	768	9	No	numeric
cylinder	2	541	36	Yes	mixed	segment	7	2310	19	No	numeric
dermatology	6	366	35	Yes	numeric	spambase	2	4601	58	No	numeric
ecoli	8	336	8	No	numeric	tictactoe	2	958	10	No	symbolic
flare	3	323	11	No	mixed	vote	2	435	17	Yes	symbolic
glass	7	214	10	No	numeric	wave	3	5000	22	No	numeric
haberman	2	306	4	No	numeric	wine	3	178	14	No	numeric
hepatitis	2	155	20	Yes	numeric	yeast	10	1484	9	No	numeric
horse	2	368	27	Yes	mixed	zoo	7	101	17	No	numeric
iris	3	150	5	No	numeric						

Three rule extracting algorithms are tested on 29 datasets from the UCI machine learning repository (Table 1) [1]. We use 10-fold cross validation and use the parametric

Table 2. Error rates and pairwise comparisons of the algorithms on the datasets.

Set	C4.5Rules	Ripper	Ripper*	Set	C4.5Rules	Ripper	Ripper*
bal	52.16±1.85	29.05± 6.16	29.05± 6.16	iro	8.78±4.22	9.92± 5.75	11.60± 5.36
bre	6.71±2.82	4.43± 2.17	4.57± 2.29	mon	0.00±0.00	0.00± 0.00	0.00± 0.00
bup	36.20±5.46	30.99± 5.50	36.79± 8.40	mus	0.00±0.00	0.00± 0.00	0.00± 0.00
car	5.27±1.51	17.02± 3.34	17.02± 3.34	opt	13.92±1.84	11.49± 0.91	10.65± 1.18
cmc	52.07±4.05	48.54± 3.50	49.35± 2.87	pen	5.03±0.90	5.18± 0.52	5.29± 0.74
cre	31.61±3.05	15.07± 3.30	14.47± 3.84	pim	30.21±2.16	25.52± 5.37	24.99± 4.49
cyl	27.62±6.14	30.03± 8.04	32.62± 7.16	seg	6.24±2.03	8.23± 2.36	8.44± 2.80
der	6.24±2.74	7.16± 5.10	6.05± 3.12	spa	10.50±1.09	8.17± 1.56	8.52± 1.42
eco	27.26±8.03	19.34± 7.10	18.65± 5.68	tic	0.00±0.00	1.67± 0.74	1.67± 0.74
fla	10.48±1.92	11.69± 2.92	11.09± 1.85	vot	4.14±2.38	4.37± 2.53	4.37± 2.53
gla	39.07±7.07	38.07± 9.49	40.71±10.55	wav	36.92±1.75	21.38± 1.57	23.94± 2.45
hab	26.48±1.88	26.16± 2.69	27.10± 7.57	win	6.84±7.46	7.42± 7.72	10.74± 7.80
hep	17.52±4.74	19.37± 7.56	22.04± 5.77	yea	53.64±3.49	43.05± 5.06	43.45± 4.70
hor	16.03±3.76	13.81± 4.38	14.14± 4.71	zoo	3.85±5.02	13.93±19.32	11.91± 9.72
iri	6.67±7.03	6.00± 7.34	5.33± 6.13				

	C4.5Rules	Ripper	Ripper*	Total
C4.5Rules	-	4	7	7
Ripper	12	-	2	12
Ripper*	10	1	-	10
Total	12	5	9	

one-sided t test to compare the results for statistically significant difference. We report the average and standard deviations of error rates of the algorithms (Table 2), number of rules they generate (Table 3), and the training time (Table 4).

Since there are more than two algorithms to compare, there can be various possible orderings and we give a second table which contains pairwise comparisons; the entry (i, j) in this second table gives the number of datasets (out of 29) on which method i is statistically significantly better than method j with at least 95 percent confidence. The row and column sums are also given. The row sum gives the number of datasets out of 29 where the algorithm on the row outperforms at least one of the other algorithms. The column sum gives the number of datasets where the algorithm on the column is outperformed by at least one of the other algorithms.

The difference between Ripper and Ripper* occurs in selecting the best split position for numeric attributes. Therefore on the datasets that only contain discrete attributes (*balance*, *car*, *mushroom*, *nursery*, *tictactoe*, *vote*) we have the same results with Ripper and Ripper*.

Table 2 shows that Ripper and its variant Ripper* have better performance than C4.5Rules in terms of error rates (12-4 for Ripper and 10-7 for Ripper*). We see also that the statistical way of calculating the split point used in Ripper* does not degrade the performance of Ripper. Ripper has two significant wins agains Ripper*, whereas Ripper* has one significant win against Ripper. C4.5Rules has the advantage of the exhaustive search of the best rule set when the number of rules for one class is small

Table 3. Number of rules generated by the algorithms on the datasets and pairwise comparisons of the algorithms.

Set	C4.5Rules	Ripper	Ripper*	Set	C4.5Rules	Ripper	Ripper*
bal	1.2±1.0	5.5± 1.4	5.5±1.4	iro	4.6±0.7	2.5± 0.7	3.3±0.8
bre	9.5±1.1	3.5± 0.5	3.9±0.7	mon	8.0±0.0	4.0± 0.0	4.0±0.0
bup	2.0±0.9	1.1± 0.3	1.2±0.4	mus	21.0±0.0	8.0± 0.0	8.0±0.0
car	45.2±1.4	22.5± 4.0	22.5±4.0	opt	55.0±3.9	35.8± 2.1	38.4±1.5
cmc	4.9±1.4	2.1± 0.6	2.4±0.5	pen	75.2±2.7	48.9± 2.6	53.4±2.5
cre	8.9±1.0	2.6± 1.1	1.4±0.5	pim	3.1±0.7	1.6± 0.5	1.7±0.5
cyl	3.8±0.9	2.1± 1.0	2.1±0.3	seg	24.4±1.5	16.7± 1.9	19.6±1.4
der	9.4±1.0	6.0± 0.9	7.2±0.6	spa	27.2±2.3	7.9± 1.1	11.4±1.8
eco	9.3±1.6	5.6± 1.1	6.4±0.7	tic	16.3±0.7	8.0± 0.0	8.0±0.0
fla	2.0±0.7	0.1± 0.3	0.3±0.5	vot	9.0±0.9	2.2± 0.8	2.2±0.8
gla	5.8±0.9	4.3± 1.3	3.7±0.9	wav	29.1±2.2	11.1± 1.7	11.8±2.5
hab	1.5±0.5	0.8± 0.4	1.0±0.0	win	3.9±0.3	2.5± 0.5	2.9±0.3
hep	2.5±0.5	0.7± 0.5	0.8±0.4	yea	19.0±2.6	10.9± 1.9	12.9±1.4
hor	4.1±0.9	1.9± 0.3	2.0±0.0	zoo	7.5±0.5	4.1± 0.3	4.2±0.6
iri	3.4±0.5	2.0± 0.0	2.0±0.0				

	C4.5Rules	Ripper	Ripper*	Total
C4.5Rules	-	1	1	1
Ripper	26	-	10	27
Ripper*	27	2	-	27
Total	27	3	11	

(typically less than 10).

Table 3 shows the complexity of the rule sets generated by the rule induction algorithms in terms of the number of rules. Ripper and its variant Ripper* produce significantly fewer rules with fewer number of conditions than C4.5Rules (26-1 for Ripper and 27-1 for Ripper* in terms of number of rules). This is mainly due to the fact that C4.5Rules starts with the unpruned C4.5 decision tree containing a large amount of superflous rules and conditions. Although C4.5Rules prunes rules and conditions, the results show that growing a large number of rules and then pruning them will get us larger number rules compared to learning rules from scratch.

Table 4 shows that Ripper is significantly faster than C4.5Rules. Since C4.5Rules starts with an unpruned decision tree, the number of rules is significantly larger compared to the end rule set. For example, on *balance*, *car*, *cmc*, *spambase*, *wave*, *yeast*, C4.5Rules starts with 237, 189, 436, 131, 254, 257 rules, ending with 1, 45, 5, 27, 29, 19 rules. C4.5Rules then tries to find best rule set greedily adding and removing rules, which requires substantially large amount of time if we have large amount of rules; this is especially true for noisy datasets [2]. Cohen also mentioned this problem with C4.5Rules — C4.5Rules is said to have an expected learning time of $\mathcal{O}(N^3)$ whereas Ripper has an expected learning time of $\mathcal{O}(Nlog^2N)$.

Ripper* is faster than Ripper on 13 datasets. This is expected because finding the best condition to add a rule using exhaustive search requires Quicksort on instances

Table 4. Time to generate rule sets by the algorithms on the datasets (in seconds) and pairwise comparisons of the algorithms.

Set	C4.5Rules	Ripper	Ripper*	Set	C4.5Rules	Ripper	Ripper*
bal	2534.5± 198.0	0.2± 0.4	0.2± 0.4	iro	1.0± 0.0	0.5± 0.5	0.3± 0.5
bre	4.6± 0.8	1.0± 0.0	0.3± 0.5	mon	3.7± 6.1	0.2± 0.4	0.1± 0.3
bup	16.6± 3.2	0.1± 0.3	0.1± 0.3	mus	91.2± 4.5	1.6± 0.5	1.6± 0.5
car	2012.5± 94.9	3.8± 1.1	3.8± 1.1	opt	551.0± 51.9	1648.9± 97.4	89.2± 4.3
cmc	35103.6±3956.3	1.5± 0.5	0.6± 0.5	pen	837.8± 79.5	403.7± 33.0	47.2± 3.5
cre	217.5± 40.1	0.4± 0.5	0.1± 0.3	pim	142.1± 10.4	0.5± 0.5	0.2± 0.4
cyl	55.9± 13.1	0.7± 0.5	0.3± 0.5	seg	56.4± 7.7	23.3± 2.5	4.5± 0.5
der	0.6± 0.5	1.9± 0.6	0.5± 0.5	spa	6732.6± 788.5	1453.2±118.9	85.9± 5.8
eco	1.5± 0.5	0.5± 0.5	0.1± 0.3	tic	882.6± 129.2	0.3± 0.5	0.3± 0.5
fla	24.6± 4.8	0.1± 0.3	0.0± 0.0	vot	2.0± 0.7	0.1± 0.3	0.1± 0.3
gla	0.8± 0.4	0.2± 0.4	0.2± 0.4	wav	24231.6±1097.6	33.6± 4.8	16.7± 1.8
hab	22.5± 6.2	0.1± 0.3	0.1± 0.3	win	0.0± 0.0	0.0± 0.0	0.0± 0.0
hep	0.6± 0.5	0.2± 0.4	0.0± 0.0	yea	4819.5± 315.3	17.4± 3.0	2.3± 0.5
hor	26.5± 4.0	0.5± 0.5	0.3± 0.5	zoo	0.1± 0.3	0.1± 0.3	0.0± 0.0
iri	0.0± 0.0	0.0± 0.0	0.1± 0.3				

	C4.5Rules	Ripper	Ripper*	Total
C4.5Rules	-	2	0	2
Ripper	23	-	0	23
Ripper*	25	13	-	27
Total	25	13	0	

with an $\mathcal{O}(NlogN)$ expected time, whereas the statistical shortcut of finding the split requires only four passes on instances with an $\mathcal{O}(N)$ expected time. Therefore Ripper* drops the learning time of Ripper* from $\mathcal{O}(Nlog^2N)$ to $\mathcal{O}(NlogN)$. This drop in learning time can be seen especially in larger datasets. For example Ripper* is 17 times faster than Ripper on *optdigits*, 9 times faster on *pendigits*, 15 times faster on *spambase*, 8 times faster on *yeast*, and 5 times faster on *segment* datasets.

4 Conclusion

We compare two rule induction algorithms, C4.5Rules and Ripper. We also give an improvement over Ripper to reduce learning time. Our comparisons in terms of error rates of the algorithms, complexity of the produced rule sets, and the time required in training indicate that Ripper produces more accurate rule sets with smaller rule complexity in less amount of time, especially with the shortcut we propose.

Acknowledgment

This work has been supported by the Turkish Academy of Sciences, in the framework of the Young Scientist Award Program (EA-TÜBA-GEBIP/2001-1-1) and Boğaziçi University Scientific Research Project 02A104D.

References

1. C. Blake, Keogh, E. and Merz, C. J, UCI repository of machine learning databases. Available at http://www.ics.uci.edu/∼mlearn/MLRepository.html.
2. W. W. Cohen, Fast effective rule induction. *The Twelfth International Conference on Machine Learning*, pp. 115–123 San Fransisco, CA. Morgan Kaufmann.
3. J. R. Quinlan, Induction of Decision Trees, *Machine Learning*, Vol. 1, pp. 81-106. (1986)
4. J. R. Quinlan, *C4.5: Programs for Empirical Learning* Morgan Kaufmann, San Francisco, CA, 1993.
5. J. R. Quinlan and Rivest, R. L, Inferring decision trees using the minimum description length principle. *Information and Computation*, 80, pp. 227–248, 1989.