ACTA UNIVERSITATIS LODZIENSIS
FOLIA OECONOMICA 235, 2010

Mariusz Kubus[*]

THE ANALYSIS OF SOME PROPERTIES OF SLIPPER ALGORITHM

Abstract. Rules induction has significant position among nonparametric and adaptive methods of discrimination. Description of classes has a form of conjunctions of features' values. To learn every single rule, heuristic search of class description space and function of criterion are used. The great number of rules induction algorithms follow the separate-and-conquer manner (Michalski 1969). SLIPPER uses boosting instead of separate step (Cohen, Singer 1999). In this paper we discuss the stability of model generated by SLIPPER and we compare the error rate obtained in SLIPPER and classical AdaBoost where classification trees are combined. The comparison with other discrimination methods also is given.

Key words: rules induction, boosting, SLIPPER.

I. INTRODUCTION

With predictive learning one is given a training set of cases with observed values of response variable Y:

$$\{(\boldsymbol{x_1}, y_1),...,(\boldsymbol{x_m}, y_m) : \boldsymbol{x_i} \in \boldsymbol{X} = (X_1,...,X_p), y_i \in Y, i \in \{1,...,m\}\}. \qquad (1)$$

The purpose is to design the model (using training set) so as to predict the value of response variable Y on unseen cases. When Y is nominal we deal with discrimination task.

The rules induction is a nonparametric and adaptive method of discrimination which can be perceived as an alternative to classification trees. The method as well as classification trees can deal with nonmetric data and missing values of variables. Also the assumption about distribution of variables is not required. Classifier has a form of set of conjunctive rules. Let S_j be the set of all possible values for input variable X_j, s_{jt} be a specified subset of those

[*] Ph.D., Department of Mathematics and Computer Science Application, Opole University of Technology.

values ($s_{jt} \subseteq S_j$) and x_j be the realisation of variable X_j for a case $\boldsymbol{x} = (x_1,...,x_p)$. The conditional part of the rule takes the form:

$$R_t(\boldsymbol{x}) = \prod_{j=1}^{p} I(x_j \in s_{jt}) \,, \tag{2}$$

where $I(\cdot)$ is an indicator of the truth of its argument. Note that for the special case when $s_{jt} = S_j$, the corresponding factor can be omitted from the logical product. We can express the formula (2) in the simpler form:

$$R_t(\boldsymbol{x}) = \prod_{s_{jt} \neq S_j} I(x_j \in s_{jt}) \,. \tag{3}$$

Practically, for the purpose of interpretation it is desirable that model consists of "simple" rules each defined by a small number of variables. All possible products (2) will be called class description space and the factors of conjunction (3) will be called conditions. In the simplest case the conditions take one of the following forms: $x_j = v$ for nominal variables (v is a category) or $x_j \geq l$ $(x_j \leq l)$ for metric variables (l is any value which is not necessarily present in training set). Such representation is known as disjunctive normal form (DNF) but other forms are also possible (*see* Fürnkranz 1999).

The great number of rules induction algorithms follow the separate-and-conquer manner that was proposed for the first time by Michalski (1969). Set of rules is generated by learning one rule at a time. After each rule is learned (conquer step), the algorithm removes from the training set the cases covered[1] by the rule (separate step). Usually, only cases from the class which description we are generating are removed. The process is then iterated on the remaining training cases. It can be continued until no cases remain or some stopping criteria are fulfilled. To generate single rule, heuristic search of class description space and function of criterion are used. The function (i.e. entropy or precision) evaluates quality of rules found in the search space. It can be seen as estimation of classification accuracy on unseen data. The most common search strategies in rules induction are hill climbing and beam search. The list of candidate rules is initialized with the most general rule which describes all cases in training set. Then conditions are added to conjunction to optimize the function of criterion. It is so called search from general to specific (or top-down search). Finally, the decision regions corresponding to the rules can overlap and they do not have to

[1] The cases for which the rule is true.

include all cases. These features differentiate this method from classification trees. One of the most effective algorithm representing this approach is RIPPER (Cohen 1995). SLIPPER that uses boosting (Cohen, Singer 1999), is it's modification and is presented below.

II. BOOSTING

Boosting is known mainly from aggregated tree-based models (Breiman 1998; Freund, Schapire 1997; Gatnar 2008). It was proposed as an answer to instability of classification trees. Significant improvement of the model stability and classification accuracy can be obtained by learning a lot of trees and combining them into one aggregated model. Every single tree is learned on different training sample. There are used the system of weights for cases as well as for component models. The weights of cases misclassified by component model are increased. It gives higher probability that such cases will be sampled in the next iteration. One disadvantage of boosted trees is that classifier has a form of black box, so it does not give a possibility of interpretation.

Cohen and Singer (1999) used boosting for inducing rules. In a contradiction to ensemble tree-based classifier the application of boosting in rules induction conducts to comprehensible model. They proposed a replacement of the separation step with boosting. Cases are not removed but their weights are reduced so that to cover rather other cases in the next iteration. In such approach every rule is generated using entire training set. Rules get also weights what gives the possibility of ranking their importance, what is desirable with respect to interpretation.

III. SLIPPER ALGORITHM

The algorithm will be presented in version of binary classification. We are given a training set (1) where categories of the nominal response variable Y are coded with $\{-1,+1\}$. The weights of cases are initialized with equal values $D(i)=1/m.$ The algorithm iteratively repeats two steps. Number of iterations can be set arbitraly or via cross validation. In the first step it learns single rule using procedure from RIPPER (Cohen 1995) and then, in the second step the weights of cases are updated. So boosting is used instead of separation used in classical rules induction. The covered cases are not removed from training set but they obtain weights according to the generalized version of AdaBoost (Schapire and Singer 1998):

$$D_{t+1}(i) = \frac{D_t(i) \cdot e^{-y_i C_{R_t}(i)}}{Z_t}, \tag{4}$$

where:

$$C_{R_t} = \frac{1}{2} \ln \frac{W_+}{W_-} \tag{5}$$

is a weight of rule R_t and

$$Z_t = W_0 + W_+ e^{-C_{R_t}} + W_- e^{C_{R_t}}, \tag{6}$$

is a normalization factor where:

$$W_0 = \sum\nolimits_{i \notin R_t} D(i), \tag{7}$$

$$W_+ = \sum\nolimits_{i \in R_t : y_i = +1} D(i), \tag{8}$$

$$W_- = \sum\nolimits_{i \in R_t : y_i = -1} D(i). \tag{9}$$

Note that symbol $\boldsymbol{x} \in R$ means that $\boldsymbol{x}$ satisfied the rule R. When the rule covers only cases from the class we are learning the description, W_- is equal to 0. Therefore formula (5) is modified as below:

$$\widetilde{C}_{R_t} = \frac{1}{2} \ln \frac{W_+ + \frac{1}{2m}}{W_- + \frac{1}{2m}}. \tag{10}$$

Schapire and Singer (1998) showed that to minimize the resubstitution error the weak learning algorithm (here RIPPER), on each round of boosting, should pick the weak model (here single rule) and its weight which lead to the smallest value of normalization factor Z_t (6). Hence, the weight of the rule is given with formula (5) and we can come to a conclusion that function of criterion in RIPPER procedure of generating a single rule should take a form:

$$\widetilde{Z} = \sqrt{W_+} - \sqrt{W_-}. \tag{11}$$

With respect to original version of algorithm RIPPER, the only differences are applied criteria which base on weights (of cases and rules). Generally RIPPER consists of three steps. First the primary training set (1) is randomly divided into two subsamples: training and test. Training subsample is used to learning rule corresponding to homogeneous region in the feature space. The heuristic search uses hill climbing strategy in direction from general to specific. After single rule is learned it is pruned at once using test subsample. Some final sequences of conditions are deleted so that to minimize (6). Now weights (7–9) are counted on test subsample and weight of rule has a form (10).

Finally, classification is made on the basis of sum of rules weights that cover the case ***x***:

$$H(x) = sign\Big(\sum_{R_t : x \in R_t} \widetilde{C}_{R_t} \Big) . \tag{12}$$

As in many rules based models, the order of class' descriptions can affect on the classification or not. Unordered set of rules in SLIPPER leads to smaller error rate but the complexity of model[2] is much higher (*see* Kubus 2008).

IV. EXPERIMENTS

For empirical experiments we used well known datasets from UCI Repository (Blake, Keogh, Merz 1989). The computations have been done in original Cohen's implementation of SLIPPER and RIPPER and in a few packages of R (adabag,class,MASS).

The first experiment concerns the problem of stability. Using boosting for tree-based models guarantees the improvement of stability, so we ask whether SLIPPER also does it in comparison to its predecessor RIPPER which learns the rules following the separate-and-conquer manner. We chose three datasets: *credit australian*, *ionosphere*, *hepatitis*. Each of them was randomly divided into training and test sample thirty times in proportion 2:1. Then, using training samples, the models were learned running both SLIPPER and RIPPER thirty times and error rates were estimated using test samples. Having thirty errors for each dataset and for two algorithms we compared standard deviations and run the test for variances. Results are shown in Table 1. Two times SLIPPER obtains higher standard deviations then RIPPER and variances differ significantly (for the level of significance equal to 0,05) for *hepatitis* dataset. After this experiment we cannot say that SLIPPER improves stability in comparison to RIPPER.

[2] Number of rules and conditions.

Table 1. Standard deviations and test for variances for thirty estimations of error rate

Datasets	stand.dev. RIPPER	stand.dev. SLIPPER	F	p
credit australian	1,190874	0,417229	8,14671	0,066490
ionosphere	0,667323	1,248807	3,50202	0,252194
hepatitis	1,023269	3,437666	11,28620	0,037591

Source: own research.

The goal of the next experiment was to compare the error rates obtained by SLIPPER and classical AdaBoost in which classification trees are combined. We sampled 20 datasets from UCI Repository. They differ with number of classes, cases and variables. There are both metric and nominal variables. A few of datasets have missing values of variables. When original test set was not available we randomly divided original data into training and test sample. The number of iterations in AdaBoost that we set was 100 (*see* Breiman 1998). In SLIPPER, number of iterations was set via cross validation (precisely, numbers from 1 to 100 were checked and chosen this one for which error rate was the lowest). To obtain lower error rate in SLIPPER we used the version of algorithm that generates unordered set of rules (Kubus 2008). Results are presented in Table 2.

Table 2. Error rates (%) estimated on test samples

Datasets	SLIPPER	AdaBoost	Datasets	SLIPPER	AdaBoost
adult	13,02	15,47	*ionosphere*	9,40	7,69
breast cancer	5,26	2,11	*lymphography*	10,20	6,12
car	4,34	9,38	*nursery*	5,12	2,89
credit australian	12,61	13,91	*pima*	24,22	24,22
credit german	23,72	20,72	*satellite*	11,05	11,60
echocardio	29,55	27,27	*sonar*	25,71	18,57
ecoli	19,64	13,39	*vehicles*	26,60	25,89
glass	39,06	21,88	*vowel*	17,27	6,36
heart-disease C	22,77	20,79	*wine*	0,00	1,69
hepatitis	13,46	13,46	*zoo*	26,47	8,82

Source: own research.

SLIPPER obtains lower error rate only 5 times. AdaBoost outperforms SLIPPER on 13 datasets. To become convinced if the differences between error rates are significant we have done the Wilcoxon test for matched pairs. We obtained the p-value equal to 0,0187 (the value of statistic T was equal to 42). We also compared the time of run for three largest datasets (see Table 3). The times of run were comparable when we used cross validation to set the number of iterations in SLIPPER but it distinctly outperforms AdaBoost when we set number of iterations arbitraly as 40 (in such case error rates are not much higher (*see* Kubus 2008)).

Table 3. The comparison of run time (in seconds) on largest datasets used in research

Datasets	number of cases	number of variables	SLIPPER run time	AdaBoost run time
adult	48842	14	24,74	308,51
nursery	12960	8	2,99	69,51
satellite	6435	36	21,36	174,10

Source: own research.

In the third experiment we compared SLIPPER with other popular discrimination methods like LDA (*linear discriminant analysis*) and kNN (*k nearest neighbours*) in respect to the error rate. Results are presented in Table 4 where 11 metric datasets are exploited. SLIPPER's won-loss-tied record ("won" means that SLIPPER achieved lower error rate) versus LDA is 5-5-1 and versus kNN is 7-3-1. Applying LDA in *ecoli* dataset was impossible because of the presence of the classes with only one case. Three "losses" with kNN were obtained in datasets with many classes (*ecoli*, *satellite*, *vowel*). In binary classification task SLIPPER mostly obtained lower error rates in comparison to kNN.

Table 4. Error rates (%) estimated on test samples

Datasets	SLIPPER	LDA	kNN
breast cancer	5,26	3,2	6,8
echocardio	29,55	27,3	31,8
ecoli	19,64	–	17
glass	39,06	39,1	45,3
ionosphere	9,4	15,4	19,7
pima	24,22	19,9	26,6
satellite	11,05	17,2	10,4
sonar	25,7	24,3	25,7
vehicles	26,60	19,1	41,5
vowel	17,27	35,5	9,7
wine	0	1,7	30,5

Source: own research.

To summarize our results we can say that SLIPPER does not improve stability of model in comparison to RIPPER. It obtains significantly worse error rates then AdaBoost for tree-based models but there is an advantage that weighted ruleset learned by SLIPPER can be interpretable. SLIPPER is also faster if the number of iterations is set arbitraly. The comparison in respect to error rate with popular discrimination methods used in research shows that SLIPPER outperforms kNN in binary classification task but there are no distinct differences with LDA. Note however that rules enable easier interpretation, moreover SLIPPER handles with nonmetric data and missing values of variables (without any pre-processing actions).

REFERENCES

Blake C., Keogh E., Merz C.J. (1998), *UCI Repository of Machine Learning Databases*. Department of Information and Computer Science, University of California, Irvine. www.ics.uci.edu/~mlearn/MLRepository.html

Breiman L. (1998), Arcing Classifiers. *„Annals of Statistics”*, No 26.

Cohen W.W. (1995), Fast effective rule induction. In Prieditis A., Russell S. (Eds.) *Proceedings of the 12th International Conference on Machine Learning*.

Cohen W.W., Singer Y. (1999), A Simple, Fast, and Effective Rule Learner. In *Proceedings of Annual Conference of American Association for Artificial Intelligence* (p.335–342).

Freund Y., Schapire R. E. (1997), A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *„Journal of Computer and System Sciences”*, No 55.

Fürnkranz J. (1999), Separate-and-Conquer Rule Learning. *Artificial Intelligence Review* 13(1).

Gatnar E. (2008), *Podejście wielomodelowe w zagadnieniach dyskryminacji i regresji.* PWN, Warszawa.

Kubus M. (2008), Zastosowanie metody boosting w indukcji reguł. [in:] K. Jajuga, M. Walesiak (red.), *Taksonomia 15, Klasyfikacja i analiza danych – teoria i zastosowania,* Prace Naukowe Uniwersytetu Ekonomicznego we Wrocławiu, No 7 (1207), p.470–477, UE Wrocław.

Michalski R.S. (1969), On the quasi-minimal solution of the covering problem. In *Proceedings of the 5th International Symposium on Information Processing* (FCIP-69), Vol. A3 (Switching Circuits), p.125–128 Bled, Yugoslavia.

Schapire R.E., Singer Y. (1998), Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory,* p. 80–91.

Webb A.R. (2002), *Statistical Pattern Recognition*. 2nd Edition, John Wiley & Sons.

Mariusz Kubus

ANALIZA WYBRANYCH WŁASNOŚCI ALGORYTMU SLIPPER

Indukcja reguł jest nieparametryczną i adaptacyjną metodą dyskryminacji. Może być stosowana dla zmiennych niemetrycznych, regiony decyzyjne nie muszą być rozłączne, a model jest łatwy w interpretacji. Opis klas ma postać koniunkcji wartości cech. Każda pojedyncza reguła generowana jest za pomocą heurystycznego przeszukiwania przestrzeni opisów klas z wykorzystaniem funkcji kryterium. Wiele algorytmów indukcji reguł wykorzystuje schemat separuj-i-zwyciężaj (Michalski 1969). Algorytm SLIPPER (Cohen, Singer 1999) zamiast kroku separuj stosuje metodę boosting, znaną głównie z agregacji drzew klasyfikacyjnych. W artykule zbadana będzie stabilność modelu generowanego przez SLIPPER. Dokonane też będzie porównanie z klasycznym AdaBoost agregującym drzewa klasyfikacyjne oraz z popularnymi metodami dyskryminacji.