Comparison of rule induction, decision trees and formal concept analysis approaches for classification

IOP Conf. Series: Journal of Physics: Conf. Series **1015** (2018) 032068 doi:10.1088/1742-6596/1015/3/032068

Comparison of rule induction, decision trees and formal concept analysis approaches for classification

E V Kotelnikov[1] **and V R Milov**[2]

[1] Vyatka State University, 111, Lenina St., Kirov, 610000, Russia
[2] Nizhny Novgorod State Technical University n.a. R.E. Alekseev, 24, Minin St., Nizhny Novgorod, 603950, Russia

E-mail: kotelnikov.ev@gmail.com

Abstract. Rule-based learning algorithms have higher transparency and easiness to interpret in comparison with neural networks and deep learning algorithms. These properties make it possible to effectively use such algorithms to solve descriptive tasks of data mining. The choice of an algorithm depends also on its ability to solve predictive tasks. The article compares the quality of the solution of the problems with binary and multiclass classification based on the experiments with six datasets from the UCI Machine Learning Repository. The authors investigate three algorithms: Ripper (rule induction), C4.5 (decision trees), In-Close (formal concept analysis). The results of the experiments show that In-Close demonstrates the best quality of classification in comparison with Ripper and C4.5, however the latter two generate more compact rule sets.

1. Introduction

At present, neural networks and deep learning algorithms are widely used [1]. Despite their high efficiency, an important drawback of such algorithms lies in the poor interpretability of the generated models, which makes it difficult to solve descriptive data mining tasks. On the contrary, models that are built using rule-based learning algorithms are transparent and easy to interpret. Within the framework of this approach, one can distinguish rule induction, decision trees and formal concept analysis (FCA). However, there are no studies comparing the effectiveness of these algorithms based on the same datasets. In [2] FCA algorithms are compared, as well as decision trees (C4.5), but it lacks comparison with the rule induction. Apte and Weiss [3] compare decision trees and rule induction, but there is no FCA. The authors' work fills this gap and, on the basis of several datasets, compares three algorithms from different approaches: Ripper (rule induction), C4.5 (decision trees), and In-Close (FCA).

The rest of this paper is organized as follows: the second section describes rule-based learning algorithms. The third section gives the characteristics of used datasets. The fourth section is devoted to the results of experiments and their discussion. The fifth section gives conclusions and prospects for future research.

2. Rule-based learning algorithms

2.1. Ripper

Ripper (Repeated Incremental Pruning to Produce Error Reduction) [4] is a rule induction based on reduced error pruning [5]. Ripper includes three stages of rule processing: building, optimization and clean-up [6]. In the first stage, a training dataset is divided into growing and pruning sets. Rules are constructed on the basis of a growing set. Later these rules are reduced with the help of a pruning set. At the optimization stage, all rules are reconsidered in order to reduce the error on the entire dataset. At the clean-up phase, it is checked whether each Description Length (DL) rule increases the whole rule set and the dataset. The calculation of DL allows one to quantify the complexity of various rule sets, which describe dataset [7]. In case if the rule increases DL, it is deleted.

To date, the Ripper algorithm is considered as the state of the art in rule induction [8] and implemented in the machine learning library WEKA under the name of JRip [6].

2.2. C4.5

C4.5 is one of the most famous algorithms of induction of decision trees [9], which improves the ID3 algorithm [10]. Like ID3, C4.5 recursively builds a decision tree, starting with an empty node, selecting at each step the attribute that has the greatest information gain, when the data are being divided by the values of this attribute.

In C4.5, compared to ID3, the following improvements have been made. First, the algorithm allows one to handle attributes with missing values, assuming that they are statistically distributed in proportion to the known values of this attribute. Second, C4.5 can work with continuous attributes by specifying such a cut-off threshold at which at least 10% of instances would be at the current node, when the data are being divided. Third, C4.5 carries out pruning of the constructed tree.

Based on the generated decision tree, it is easy to turn to the rule set. To do this, one just needs to go through each path from the root of the tree to the leaves. Algorithm C4.5 is implemented in WEKA under the name of J48 [6].

2.3. In-Close

The In-Close algorithm [11] was developed within the FCA framework [12]. This algorithm, based on binary relational data (object-attribute matrix), generates all the formal concepts, with each being a set of attributes *A* and a set of instances *B*, such that each attribute of set *A* belongs to all instances *B*, and each instance from set *B* includes all attributes from set *A*. The In-Close algorithm is based on the Close-by-One algorithm [13], which uses the natural lexical order of combinations.

Concepts, generated by In-Close, can be transformed into rules: the set of attributes *A* forms an antecedent (IF-part of the rule), and a class, for whose instances the concept was formed, is consequent (THEN-part of the rule). The size of set *B* can serve as a support for the rule. The classification based on the created rules can be carried out, for example, by voting (voting) – given instance refers to the class for which there was the largest number of concepts suitable for the instance. The software implementation of the In-Close version 4 algorithm in C++ is available on SourceForge (https://sourceforge.net/projects/inclose).

3. Datasets

For the comparative analysis of the considered algorithms, datasets, taken from the UCI Machine Learning Repository [14], were used: three MONK's problems, SPECT Heart dataset, Hayes-Roth's dataset, Soybean dataset.

MONK's problems are three artificial datasets, created to evaluate machine learning algorithms in the binary classification [15]. In the first set, the division into classes is assigned by simple disjunctive normal forms. In the second set, the classification rule is more complex and similar to the parity problem. In the third set there is again a simple classification rule, but there are 5% of noisy instances.

SPECT Heart is the dataset which describes diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images [16]. Each instance is a patient, which can belong to two classes – normal and abnormal.

Hayes-Roth's dataset was proposed in [17]. It contains descriptions of students, each being represented by four attributes – hobby, age, educational level and marital status. The task is to determine if the student is a member of club 1, club 2 or does not belong to any club.
In Soybean dataset [18], 19 diseases of soybean are described. Each instance is represented by 35 attributes.

Table 1 gives the characteristics of the used datasets: a size of train and test sets, a number of attributes, a type of attributes (binary or nominal) and a number of classes.

Table 1. Characteristics of used datasets

Datasets	Number of instances		Number of attributes	Attribute type	Number of classes
	Train	Test			
MONK's problem 1	124	432	6	Nominal	2
MONK's problem 2	169	432	6	Nominal	2
MONK's problem 3	122	432	6	Nominal	2
SPECT Heart	80	187	22	Binary	2
Hayes-Roth	132	28	4	Nominal	3
Soybean	307	376	35	Nominal	19

When using the In-Close algorithm for all datasets, except SPECT Heart, binarization of the nominal attributes is required. It was implemented using the Python package Pandas and one-hot encoding. Ripper and C4.5 both work with nominal attributes without binarization.

4. Results and discussion

When evaluating the classification quality, metrics Precision (*P*), Recall (*R*) and F1-measure (F1) were used:

$$P_c = \frac{TP_c}{TP_c + FP_c},\ R_c = \frac{TP_c}{TP_c + FN_c},\ F1_c = \frac{2P_cR_c}{P_c + R_c}, \quad (1)$$

where TP_c – a number of test instances correctly classified as belonging to class c (true positive), FP_c – a number of test instances incorrectly classified as belonging to class c (false positive), FN_c – a number of test instances incorrectly classified as not belonging to class c (false negative).

This metrics are averaged by classes with the help of a weighted macro-averaged scheme:

$$P_{wma} = \sum_c P_c \frac{N_c}{N},\ R_{wma} = \sum_c R_c \frac{N_c}{N},\ F1_{wma} = \sum_c F1_c \frac{N_c}{N}, \quad (2)$$

where N_c – a number of test instances belonging to class c, N – a number of all test instances.

The classification results are presented in Table 2.

Table 2. Results of classification

Datasets	Ripper			C4.5			In-Close		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Monks1	100.0%	100.0%	**100.0%**	76.9%	75.7%	75.4%	83.6%	82.9%	82.8%
Monks2	60.2%	62.5%	61.0%	62.9%	65.0%	63.5%	73.5%	73.4%	**69.4%**
Monks3	90.3%	90.3%	90.3%	97.4%	97.2%	**97.2%**	95.0%	94.9%	94.9%
SPECT	90.2%	75.9%	81.1%	90.9%	75.4%	80.7%	92.4%	86.6%	**88.7%**
Hayes-Roth	83.8%	75.0%	73.5%	93.8%	92.9%	**92.8%**	93.8%	92.9%	**92.8%**
Soybean	80.2%	79.8%	79.1%	84.5%	86.7%	**85.0%**	82.5%	79.3%	77.5%
Average	84.1%	80.6%	80.8%	84.4%	82.2%	82.4%	86.8%	85.0%	**84.4%**

In Table 2 the maximum values of F1-measure for each dataset are given in bold type. The last line in Table 2 shows the average value for all datasets.

For MONK's problem 1, the best possible result (100% for all metrics) is reached by Ripper, which found the correct disjunctive normal forms, used to generate the dataset. But in MONK's problem 2, which is the most challenging among the MONK's problems, Ripper takes the last place, with In-Close being in the first place. In MONK's problem 3 with noise, the leader is C4.5, allowing one at the post-processing stage to perform pruning of the constructed tree, thus compensating for the noise effect, with In-Close being behind only by 2.3%.

For SPECT heart the dataset, having only 80 training instances, In-Close still significantly outperforms other algorithms (by 7.6% ahead of Ripper, by 8% ahead of C4.5). This is due to the high recall of In-Close, which is more than 10% higher than the recall of Ripper and C4.5.

When classifying Hayes-Roth's dataset, C4.5 and In-Close showed the same result (92.8%), which significantly exceeds the result of Ripper (73.5%). In this case, Ripper makes mistakes on 7 instances of the first class out of 14, because they do not have suitable rules for them (the *underfitting* effect). On the contrary, algorithms C4.5 and In-Close formed rules for this class as well, which made it possible to obtain high results.

For Soybean dataset, the leader turned out to be C4.5. This dataset has a large number of classes (19) and attributes (35) with an uneven number of instances in each class (from 1 to 40). Algorithm C4.5 built rather a branching decision tree (52 leaves), which successfully recognizes test instances. The high precision (82.5%) of the In-Close algorithm is offset by a relatively low recall (79.3%).

On average, for all datasets, the best results were demonstrated by the In-Close algorithm $(F1 = 84.4\%)$, which is 2% ahead of the C4.5 algorithm and 3.6% of the Ripper algorithm. The In-Close algorithm shows a high quality of classification, especially on small datasets and with a small number of classes, but the number of generated rules can be an order of magnitude greater than that of algorithms C4.5 and Ripper. Algorithm C4.5 describes the data well in the presence of noise and a large number of classes, but it is less accurate than In-Close. The Ripper algorithm copes well with simple datasets, generating 2.3 times fewer rules on average than the C4.5 algorithm, but it falls 1.6% behind C4.5 on the F1-measure.

5. Conclusion

Thus, the leader in the test results is the In-Close algorithm. Furthermore, the simple voting classification scheme for this algorithm can be improved, for example, by using some information about the support of concepts and turn to the weighted voting classification scheme.

A large number of concepts/rules generated by In-Close can be reduced by their ranking according to the explanatory degree of the training instances, as suggested in the JSM method of automatic hypotheses generation [19]. Thus, an effective tool can be obtained to solve descriptive tasks of data mining.

Acknowledgments

The reported study was funded by RFBR according to research project No. 16-07-00342a.

References

[1] Goodfellow I, Bengio Y and Courville A 2016 *Deep Learning* (Cambridge, MA: MIT Press)
[2] Fu H, Fu H, Njiwoua P and Nguifo E M 2004 *LNAI* vol 2961 p 313–320
[3] Apte C and Weiss S 1997 *Future Gener. Comput. Syst*. **13** 197–210
[4] Cohen W W 1995 *Machine Learning: proceedings of the Twelfth International Conference on Machine Learning (ICML 1995)* 115–123
[5] Fürnkranz J and Widmer G 1994 *Machine Learning: proceedings of the Eleventh International Conference on Machine Learning (ICML 1994)* 70–77
[6] Witten I H, Frank E, Hall M and Pal C 2017 *Data Mining. Practical Machine Learning Tools and Techniques* (Cambridge, MA: Morgan Kaufmann)

[7] Quinlan J R 1990 *Mach. Learn*. **5** 239–266
[8] Fürnkranz J 2017 Rule Learning *Encyclopedia of Machine Learning and Data Mining* ed C Sammut, G I Webb (Boston, MA: Springer)
[9] Quinlan J R 1993 *C4.5: Programs for machine learning* (San Mateo, CA: Morgan Kaufmann)
[10] Quinlan J R 1986 *Mach. Learn*. **5** 81–106
[11] Andrews S 2015 *Inform. Sciences* **295(20)** 633–649
[12] Ganter B and Wille R 1998 *Formal Concept Analysis: Mathematical Foundations* (Berlin: Springer-Verlag)
[13] Kuznetsov S O 1996 *Math. Sci*. **80(2)** 1654–1698
[14] Lichman M 2013 *UCI Machine Learning Repository* (http://archive.ics.uci.edu/ml)
[15] Thrun S B, Bala J, Bloedorn E, Bratko I, Cestnik B, Cheng J, De Jong K, Dzeroski S, Fahlman S E, Fisher D et al. 1991 The MONK's Problems – A Performance Comparison of Different Learning Algorithms *Technical Report CS-CMU-91-197* (Carnegie Mellon University)
[16] Kurgan L A, Cios K J, Tadeusiewicz R, Ogiela M. and Goodenday L S 2001 *Artif. Intell. Med*. **23(2)** 149–169
[17] Hayes-Roth B and Hayes-Roth F 1977 *J Verbal Learning Verbal Behav* **16** 321–338
[18] Michalski R S and Chilausky R L 1980 *Int. J. Policy Anal Information Systems* **4(2)** 125–161
[19] Finn V K 2015 *Autom. Doc. Math. Ling*. **49(4)** 122–151