RILL: Algorithm for Learning Rules from Streaming Data with Concept Drift

Magdalena Deckert, Jerzy Stefanowski

Institute of Computing Science, Poznań University of Technology,
60-965 Poznań, Poland
magdalena.deckert,jerzy.stefanowski@cs.put.poznan.pl

Abstract. Incremental learning of classification rules from data streams with concept drift is considered. We introduce a new algorithm RILL, which induces rules and single instances, uses bottom-up rule generalization based on nearest rules, and performs intensive pruning of the obtained rule set. Its experimental evaluation shows that it achieves better classification accuracy and memory usage than the related rule algorithm VFDR and it is also competitive to decision trees VFDT-NB.

1 Introduction

Mining data streams has received a growing research interest. Massive volumes of data, their rapid arrival rate, and changing characteristics impose new computational requirements for algorithms, which are not fulfilled by standard solutions developed for static data repositories. Moreover, the greatest challenge for classifiers learning from data streams is to properly react to *concept drifts*, i.e. changes in definitions of target concepts over time. Depending on the rate of these changes, concept drifts are divided into sudden, gradual or recurrent ones [6]. There exists a need for a new type of classifier, that, besides stream requirements on constrained memory usage, limited learning time and efficient incremental scanning of incoming data, should be able to track drifts and effectively adapt to them.

Most of data stream classifiers are based on either implementing windowing forgetting mechanisms, applying drift detectors or include adaptive ensembles [6]. The most popular single classifiers are Hoeffding Trees [3] and Very Fast Decision Trees which can handle numerical data and concept drifts [6]. However, *decision rules* have not received enough attention in the data stream research community so far. For static data, they are often equivalent to considering trees as they can provide better interpretability and flexibility for applying them in various systems [5]. Moreover, individual rules can be considered independently and in case of concept drift the single outdated rules can be adapted more easily than rebuilding the complete classifier or even changing the structure of the tree [7]. Besides the pioneering work on FLORA [11], only three other rule approaches have been introduced: AQ11-PM-WAH [9], FACIL [4], and VFDR [7].

We claim that there is still need for a new rule algorithm that fulfills stream computational requirements and has better reaction to different types of drifts.

In our proposal we want to consider two-fold knowledge representation—*rules and single instances*. It can be more appropriate for dealing with difficult decision boundaries and decomposition of the concepts into many sub-part with outlying examples which may occur in changing data. Here we are inspired by positive experiences of BRACID algorithm for handling static imbalanced data [10]. Unlike divide and conquer general to specific strategy from VFDR we propose to consider stepwise *bottom-up generalization* based on the *nearest rule* idea. We will also promote other solutions for coping changes.

The main aim of our paper is to introduce this algorithm, called RILL, and to evaluate it in the comparative study on several data containing different types of concept drifts.

2 Related Works

Fundamental works on incremental rule learning with concept drift has been started with FLORA family of algorithms [11]. Their main idea is based on successive modifications of nominal attribute conditions in positive rules (having the same label as the new incoming example), *boundary*, and negative rules with each incoming example. FLORA was also equipped in forgetting mechanism using global window with learning examples. The next proposal is the AQ11-PM-WAH system [9], which stores positive examples from the boundaries of the current rule set inside the partial memory. When new examples arrive, the AQ11 learner combines them with those from the partial memory to modify the rules. Moreover, it dynamically tune the period for keeping examples in the partial memory using the forgetting heuristic from FLORA2.

However, according to [4] these algorithms were not effective for larger data streams, especially with numerical attributes, and the authors introduced the FACIL algorithm. Conditions in FACIL's rules are expressed as intervals over numerical attributes defining a hyper-rectangle in the normalized real numbered space. The algorithm starts from very specific rules. When a new incoming example is not covered by any rule, FACIL tries first to generalize positive rules by looking for the smallest extensions of intervals inside rules. However, it is accepted if each extension is smaller the user-defined threshold and the candidate for generalization does not intersect with any negative rules. The core of FACIL is to store with each rule both negative and selected positive examples (two positive per one negative example covered). This local set with examples is updated when the rule covers the new example. FACIL's rules may be inconsistent (covering both positive and negative examples), however the possible purity is controlled by the user defined threshold. When this threshold is reached, the rule and its generalizations are blocked, and the examples associated with the rule are used to induce new positive and negative rules.

The completely different induction mechanism is applied in Very Fast Decision Rules (VFDR) learner, which generates either ordered or unordered set of rules [7]. Following divide and conquer strategy [5] it starts from the most general rules and successively specialize them by adding new conditions. The

specialization is based on the Hoeffding bound adapted from VFDT [3]. VFDR has been extended in [8] to cope with concept drifts by incorporation of the drift detection method (DDM [6]) with each individual rule. This improves the adaptation to changes and enables pruning of the rule set. Worth of notice is fact that only VFDR was more extensively evaluated on massive data streams with concept drift showing that it is competitive to VFDT.

As our proposal follows bottom-up rule induction and integrates rules with single instances it is more similar to FACIL than to VFDR. However, we identify several critical issues and differences in comparison to FACIL which motivate our proposal. Firstly, FACIL does not directly track drifts (only in a limited range by updating stored examples, although in a too computationally complicated way). It also insufficiently prunes rules (which can be critical for sudden drifts). Furthermore, its rule generalization is very limited and it can favor quite pure rules with small supports. Finally, its classification strategy seems to be too complex and not intuitive.

We hypothesize that it is beneficial to: (1) use single instances besides rules to better model complex concepts and their changes; (2) allow much stronger and simpler generalization of rule conditions based on the well known distance measure; (3) keep a simple forgetting mechanism with the global sliding window; (4) implement more aggressive rule pruning depending on monitoring their predictive abilities and (5) use the nearest rule / instance classification strategy.

3 Rule-based IncrementaL Learner

RILL is an acronym of words **R**ule-based **I**ncrementa**L** **L**earner. It incrementally processes learning instances described by nominal and numerical conditional attributes and the decision class label. The RILL algorithm induces unordered set of decision rules and single instances. Each rule is represented in a form:

$$\textbf{if } (attr_{num} \text{ in } [b_l; b_u]) \text{ and } (attr_{nom} = value) \textbf{ then } \text{class},$$

where $attr_{num}$ is a numerical attribute, b_l is its lower bound, b_u is its upper bound, $attr_{nom}$ is a nominal attribute, $value$ is its value, and $class$ is the value of the decision class indicated by the rule.

The pseudocode of the RILL algorithm is presented as Algorithm 1. It operates as follows. When a new learning example e_i is available, index of currently processed instance is incremented and the example e_i is added to the sliding window sw. Moreover, the distribution d of classes for learning examples in the window sw is updated according to the label of e_i (Alg. 1, line 2).

Next, RILL checks if e_i is covered by any positive rule with the same class label as e_i (Alg. 1, line 3). For every rule covering e_i, statistics like the number of covered positive examples in the window sw and the timestamp of its last usage are updated. Next, negative rules are checked. If they also cover e_i, their respective statistics are updated (Alg. 1, line 4).

If no positive rule covers e_i, the generalization procedure is fired (presented as Algorithm 2). First, this procedure looks for the nearest rule nr to the current

Algorithm 1: RILL (Rule-based IncrementaL Learner)

Input : S: data stream of examples;
w: maximum size of the sliding window;
a: rule's maximum age
Output: RS: updated set of decision rules;
sw: sliding window with learning examples;
d: distribution of the learning examples in the sliding window

1 **foreach** *(learning example* $e_i \in S$*)* **do**
2 add example e_i to sw and update d;
3 $positiveCoverage$ = `PositiveCoverage` (e_i);
4 $negativeCoverage$ = `NegativeCoverage` (e_i);
5 **if** *(*$positiveCoverage = false$*)* **then**
6 $generalization$ = `Generalization` (e_i, sw, d);
7 **if** *((*$positiveCoverage = false$*) and (*$generalization$=*false))* **then**
8 r = full description of the example e_i;
9 $RS_c \leftarrow RS_c \cup \{r\}$;
10 **if** *(*$swSize > w$*)* **then**
11 remove the oldest example from sw and update d;
12 RS = `DeleteOldRules` (RS, a);
13 RS = `DeleteImpureRules` $(RS, swSize, d)$;
14 RS = `DeleteErroneousRules` $(RS, swSize, d)$;

example e_i (Alg. 2, line 1), i.e. the rule with the smallest distance calculated using modified HVDM measure [12]. The value of this distance is defined as:

$$distance = \sum_{a \in attributes} \begin{cases} 0 \vee 1 & \text{if a is nominal} \\ 0 \vee (b_l - value_a) \vee (value_a - b_u) & \text{if a is numerical} \end{cases}.$$

This formula expresses the distance between the learning example and the decision rule as a sum of distances for each rule's conditional attribute. In case when rule's elementary condition for given attribute matches the example's value, then the distance equals 0. When the rule does not match the example's value for given attribute, the distance depends on the type of the attribute. In case of nominal attributes, the distance equals 1. On the other hand, if the type of the attribute is numerical, the distance is calculated to the nearest bound of the elementary condition in the rule. Next, the nearest rule nr is generalized to cover e_i (Alg. 2, line 2). It is enhanced by dropping condition in case of nominal attributes or extending boundaries of numerical attributes $((b_l = value_a) \vee (b_u = value_a))$ to include attribute value describing the example e_i. After obtaining generalized rule gr, its statistics are updated and its purity is calculated (Alg. 2, line 4). If the obtained purity value, calculated as $\frac{\text{positive examples from the sliding window covered by the rule}}{\text{total number of covered examples from the sliding window}}$, is higher than the relative frequency of the rule's class calculated from the window sw (Alg. 2, line 5), then the procedure looks for the nearest negative example not covered by gr (Alg. 2, line 6). The motivation is to consider more general rule conditions. If such an example exist, the rule gr is extended on numerical attributes to the half of the distance to the negative example and the new rule

er replaces the old rule nr in the current set of rules RS (Alg. 2, line 7—9). Otherwise, the rule nr is removed from the rules' set RS and the rule gr is added to the current set of rules RS (Alg. 2, line 11).

Algorithm 2: Generalization procedure

Input : e_i: current learning example;
sw: sliding window with number of recent learning examples;
d: distribution of learning examples in the sliding window
Output: *generalization*: flag indicating whether generalization was performed

```
1  nr = find nearest rule to e_i;
2  gr = generalize nr to cover e_i;
3  if (length of gr > 0) then
4  |  rule's gr purity = positiveCoverage / (positiveCoverage+negativeCoverage);
5  |  if (purity >= d(class)/swSize) then
6  |  |  neg = find the nearest negative example not covered by gr;
7  |  |  if (neg ≠ null) then
8  |  |  |  er = extend gr on numerical attributes to the half distance to neg;
9  |  |  |  RS_c ← {RS_c \ {nr}} ∪ {er};
10 |  |  else
11 |  |  |  RS_c ← {RS_c \ {nr}} ∪ {gr};
12 |  |  generalization = true;
```

In case when both procedures: finding positive coverage and generalization attempt fail, then full description of the learning example e_i is added to the set of rules RS as the most specific rule (Alg. 1, lines 7—9). Next, if the number of stored learning examples exceeds the maximum size of the sliding window sw, the oldest example from the sliding window sw is removed and the distribution d is updated (Alg. 1, lines 10—11).

The set of rules RS is pruned using three criteria. First, rules that are not used for more than a maximum age threshold are removed (Alg. 1, line 12). Secondly, rules with too low purity (below the relative frequency of its class) are deleted (Alg. 1, line 13). Finally, rules making too many prediction errors are removed (Alg. 1, line 14). This criterion uses the confidence intervals related to class probabilities, which are constructed for both, the rule classification accuracy and its class relative frequency observed over the sliding window. If the accuracy interval's higher endpoint is less than its class frequency interval's lower endpoint, then the rule is deleted [1].

Finally, the RILL's classification strategy assigns the new examples according to the class labeled of its nearest rule, which is consistent with the idea used in generalization process.

4 Experiments

Experimental Setup. The aims of this experiments are to evaluate the RILL algorithm and compare it with related classifiers. We consider the only available rule-based classifier in MOA—Very Fast Decision Rules (VFDR). Unfortunately,

despite all our efforts, FACIL's code was inaccessible for the public use. Moreover, in order to be consistent with former experiments of VFDR [7], we chose other incremental tree classifier: Very Fast Decision Trees (VFDT) and Very Fast Decision Trees with Naïve Bayes leaves (VFDT-NB). Additionally, we also checked a single Naïve Bayes (NB). All classifiers, except RILL, are not adjusted to directly deal with concept drift, that is why we combined them with the sliding window of the same size as used in RILL. All algorithms are implemented in Java, including our implementation of RILL, and are embedded into the Massive Online Analysis framework for mining streams of data[1]. All classifiers were run with default values of their parameters. As for RILL, the size of the sliding window was set to 1000. We tested other sizes but results obtained for chosen value stated the best compromise between achieved accuracy of classification and computational demands. RILL's maximum age threshold was chosen experimentally and finally was set to 3000. Lower values caused removing rules, which were still up-to-date. On the other hand, higher values of age threshold resulted in decreasing value of classification accuracy.

Dataset	#Examples	#Attributes	#Classes	Change type
AgrawalGradual	100000	9	2	gradual
CovType	581012	54	7	unknown
Crash[2]	999900	8	4	gradual
Electricity	45312	8	2	unknown
HyperplaneFaster	100000	10	4	gradual
HyperplaneSlow	100000	10	4	gradual
PAKDD09	50000	30	2	unknown
Poker	829201	11	10	unknown
Power	29928	2	24	unknown
RBFBlips	100000	20	4	blips
RBFGradualRecurring	100000	20	4	gradual
RBFNoDrift	100000	10	2	N/A
RBFSudden	100000	20	4	sudden
SEAGradual	100000	3	2	gradual
STAGGERGradual	100000	3	2	gradual
STAGGERSuddenFaster	100000	3	2	sudden

Table 1. Characteristics of datasets

To estimate classification performance we used the Evaluate Prequential method from MOA [6]. It first uses each example in the stream to assess a classification accuracy and then this example can be used to update the classifier. Moreover, this method uses a sliding window or a fading factor as a forgetting mechanism. Besides the total classification accuracy, we also recorded values of accumulated processing time from the beginning of the learning phase and the size of current model (expressed by its used memory size).

We considered several datasets involving different types of changes, such as gradual drifts, sudden drifts, blips (representing rare events—outliers in a stable

[1] For more about MOA project see http://moa.cs.waikato.ac.nz/

[2] We would like to thank Radosław Ziembiński, who provided us this dataset.

period, which a good classifier should not treat as real drifts), stability periods (no drifts for which a classifier should not be updated) and complex/mixed changes. To model precisely these drifts we used data stream generators available in the MOA framework to construct 11 synthetic datasets. To extend the study on more real world scenarios, we decided to additionally consider 5 publicly available real datasets previously used to test the related ensemble algorithms in several papers. However, for some of them there is no precise information about type of drifts. Detailed characteristics of these datasets are given in Table 1.

Experimental Results. Although we carried out more experiments, due to the page limits, we can present only the most representative results showing the general tendency. The accuracy values were averaged over recording time points (every 1000 examples, more frequent records did not influence the results). They are presented in Table 2, where the best results are denoted in bold.

Dataset	VFDT	NB	VFDT-NB	VFDR	RILL
AgrawalGradual	68.09	**73.22**	71.71	66.11	67.91
CovType	60.06	79.41	78.91	61.39	**88.13**
Crash	50.97	53.23	50.97	30.91	**81.91**
Electricity	68.12	65.41	68.76	67.47	**76.02**
HyperplaneFaster	61.36	**86.35**	85.25	69.61	68.55
HyperplaneSlow	50.35	**88.78**	88.76	52.80	66.55
PAKDD09	**80.19**	53.75	53.31	80.10	68.99
Poker	59.25	56.67	57.22	58.42	**79.22**
Power	13.77	13.77	13.77	5.24	**14.08**
RBFBlips	39.52	76.19	76.02	51.59	**80.34**
RBFGradualRecurring	33.92	67.18	67.19	42.86	**78.34**
RBFNoDrift	49.99	71.24	71.24	70.33	**80.59**
RBFSudden	33.90	67.14	67.16	43.26	**78.81**
SEAGradual	64.22	**91.25**	**91.25**	87.96	87.88
STAGGERGradual	72.52	85.99	85.99	64.32	**92.15**
STAGGERSuddenFaster	59.82	78.32	78.32	60.85	**88.68**

Table 2. Average values of classification accuracy [%]

To globally compare classifiers we carried out the ranked Friedman test with the significance level $\alpha = 0.05$. According to it (p-$value$=0.00008) we claim that there is a significant difference in the results of compared classifiers. Notice that RILL achieved the highest value of average ranks equal 4.19. The second and third were NB (3.47) and VFDT-NB (3.31), which were very near one another. The post-hoc analysis showed that the differences in average ranks between these methods are too small to conclude that they are significant. However, we can resume that each of the top three methods was significantly better than VFDR and VFDT. We additionally carried out the paired Wilcoxon tests for comparing RILL against NB or VFDT-NB. Results of this test were nearly at the significance level (in both cases p-$value$ equals 0.07), so we were very close to confirm that RILL was significantly better. However, making win-loss analysis for single data, we noticed that RILL won 11 times, was third 2 times, and fourth—3 times.

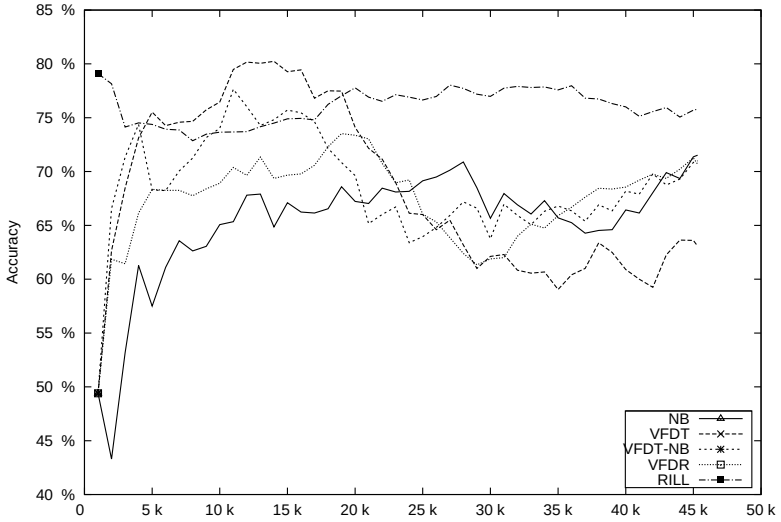

Fig. 1. Prequential classification accuracy for Electricity datasets

Better insight into dynamics of learning is available by studying figures of prequential classification accuracy with respect to processing every learning example. Again, due to the space limits, we present only the representative figures for real dataset Electricity and artificial data with sudden drift RBFSudden—see Figures 1, 2. For most of other datasets (except PAKDD09) RILL achieved also the best trend of classification accuracy. In case of artificial data with STAGGER concepts, Radial Basis Function and crashes, RILL was definitely the best classifier in terms of the accuracy of classification. Moreover, results obtained by RILL are quite stable—without any drastic rises and falls. Only for Hyperplane datasets and SEA functions, RILL was not able to outperform Very Fast Decision Tree with Naïve Bayes leaves and the single Naïve Bayes classifier.

In terms of memory consumption, the biggest amount of memory needs VFDR classifier. RILL have lower demands. However, the least memory consuming are VFDT, NB, and VFDT-NB. This tendency is also visible for most of the datasets and algorithms. Only for 2 datasets with STAGGER concepts, RILL uses more memory than VFDR classifier. In case of time, the fastest algorithms are VFDT, NB, and VFDT-NB. Other algorithms, RILL and VFDR, need more time. For majority of datasets VFDR is the slowest one. For some real problems, RILL needs more time however it becomes the most accurate classifier.

5 Discussion of Results and Final Remarks

In this paper we presented a new incremental algorithm called RILL, which induces an unordered set of decision rules and single instances. It processes data streams and attempts to adapt rules to concept drift by rule pruning and

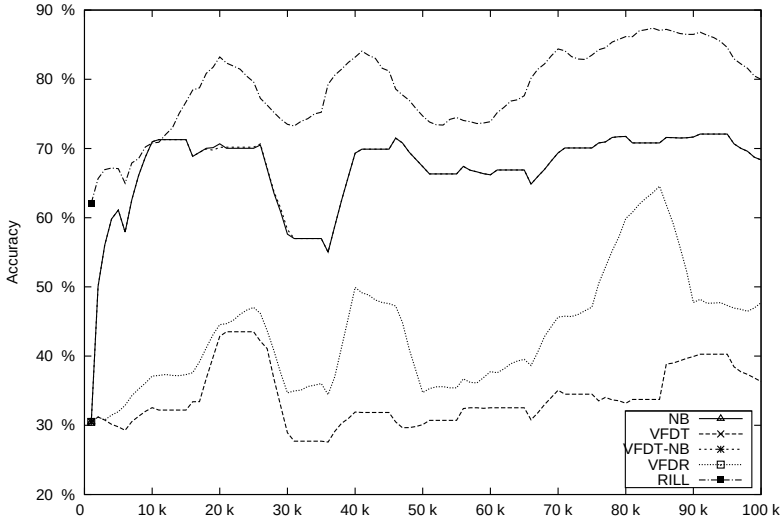

Fig. 2. Prequential classification accuracy for RBFSudden datasets

sliding window. Unlike related algorithms, it uses bottom-up rule generalization based on nearest rules, performs their intensive pruning and is integrated with the classification strategy also based on looking for the nearest rules. RILL was evaluated in a comparative study with available in MOA incremental classifiers on several data containing different types of concept drifts.

In terms of the total accuracy of classification, RILL was the best on most of the datasets. It won 11 times, was third 2 times, and fourth—3 times. With respect to more precise analysis we can say that it is competitive (according to statistical test) to the best decision trees VFDT-NB. Nevertheless, RILL outperformed the other rule-based classifier VFDR on 13 datasets and this difference is statistically significant. RILL left its opponents far behind especially on datasets with Radial Basis Function. The reason for this behavior is that these datasets represent the complex concept decomposed in many changing sub-concepts, which are not straightforward to approximate with a decision tree model. However, we have to admit, that RILL is not doing so well on datasets with moving hyperplane, SEA functions and loan functions introduced by Agrawal. We suspect that for the last two datasets, the problem lies in a definition of the concept as: (1) it is modeled as a quite difficult mathematical function with conditions over a subset of numerical attributes; (2) datasets also contain irrelevant numerical attributes. Notice that the current version of RILL generalization procedure does not allow to discard numerical attributes.

Furthermore, we experimentally showed that RILL is less computationally demanding, mainly from the memory usage, than VFDR rule classifiers.

Comparing RILL to FACIL algorithm, we were unable to do it experimentally. However, we could evaluate their differences with respect to computational

costs. From the memory usage perspective, FACIL may be more demanding due to the fact that it stores examples for every rule. Moreover, its pruning is quite limited, so it still may keep many rules. On the other hand, RILL stores always the same number of examples in the global window and offers stronger pruning. As for processing time, RILL may operate longer during the rule generalization phase, because it calculates rule statistics from the complete window.

During experiments we also analyzed the number of rule generalizations performed by RILL. For most of the datasets almost half of the generalization attempts are successful. However, in case of the hyperplane datasets only 3% of generalizations are performed. Therefore in our future research we will consider modifications of these generalizations and dropping numerical conditions for such difficult datasets.

Acknowledgments. The research has been supported by internal grant no. 09/91/DSPB/0543.

References

1. Aha D.W., Kibler D., Instance-based learning algorithms, Machine Learning, vol. 6, pp. 37-66, 1991.
2. Deckert M., Incremental Rule-based Learners for Handling Concept Drift: an Overview, Foundations of Computing and Decision Sciences, vol. 38(1), pp. 35-65, 2013.
3. Domingos P., Hulten G., Mining high-speed data streams, Proceedings of the 6th ACM SIGKDD International Conference, KDD, pp. 71-80, 2000.
4. Ferrer-Troyano F.J., Aguilar-Ruiz J.A., Riquelme J.C., Data Streams Classification by Incremental Rule Learning with Parametrized Generalization, Proceedings of ACM Symposium on Applied Computing, SAC 2006, pp. 657-661, ACM, 2006.
5. Fürnkranz J., Gamberger D., Lavrac N., Foundations of Rule Learning, Springer Verlag, 2012.
6. Gama J., Knowledge Discovery from Data Streams, CRC Publishers, 2010.
7. Gama J., Kosina P., Learning Decision Rules from Data Streams, Proceedings of the 22th International Joint Conference on Artificial Intelligence, IJCAI 11, vol. 2, pp. 1255-1260, AAAI Press, 2011.
8. Kosina P., Gama J., Handling time changing data with adaptive very fast decision rules, Proceedings of the 2012 European conference on Machine Learning and Knowledge Discovery in Databases, ECML/PKDD 2012, Bristol, United Kingdom, vol. 1, pp. 827-842, 2012.
9. Maloof M., Incremental Rule Learning with Partial Instance Memory for Changing Concepts, Proceedings of the International Joint Conference on Neural Networks 2003, IJCNN-03, vol. 4, p. 2764-2769, IEEE Press, 2003.
10. Napierala K., Stefanowski J., BRACID: a comprehensive approach to learning rules from imbalanced data, Journal of Intelligent Information Systems, vol. 9(2), pp. 335-373, 2012.
11. Widmer G., Kubat M., Learning in the presence of concept drift and hidden contexts, Machine Learning, vol. 23, pp. 69-101, 1996.
12. Wilson D.R., Martinez T.R., Improved Heterogeneous Distance Functions, Journal of Artificial Intelligence Research, vol. 6(1), pp. 1-34, 1997.