sciendo

JAISCR, 2020, Vol. 10, No. 4, pp. 287 – 298
10.2478/jaiscr-2020-0019

A NOVEL DRIFT DETECTION ALGORITHM BASED ON FEATURES' IMPORTANCE ANALYSIS IN A DATA STREAMS ENVIRONMENT

Piotr Duda[1,*], Krzysztof Przybyszewski[2], Lipo Wang [3]

[1]*Department of Computer Engineering, Czestochowa University of Technology, Częstochowa, Poland*

[2]*Information Technology Institute, University of Social Sciences, 90-113 Łódz and Clark University Worcester, MA 01610, USA*

[3]*Nanyang Technological University, School of Electrical and Electronic Engineering, Singapore*

[*]*E-mail: piotr.duda@pcz.pl*

Submitted: 5th November 2019; Accepted: 18th May 2020

Abstract

The training set consists of many features that influence the classifier in different degrees. Choosing the most important features and rejecting those that do not carry relevant information is of great importance to the operating of the learned model. In the case of data streams, the importance of the features may additionally change over time. Such changes affect the performance of the classifier but can also be an important indicator of occurring concept-drift. In this work, we propose a new algorithm for data streams classification, called Random Forest with Features Importance (RFFI), which uses the measure of features importance as a drift detector. The RFFT algorithm implements solutions inspired by the Random Forest algorithm to the data stream scenarios. The proposed algorithm combines the ability of ensemble methods for handling slow changes in a data stream with a new method for detecting concept drift occurrence. The work contains an experimental analysis of the proposed algorithm, carried out on synthetic and real data.

Keywords: data stream mining, random forest, features importance

1 Introduction

The crucial stage in creating machine learning models is to gather a training set that reflects the considered issue in the best possible way. On the other hand, the nature of many real-world problems changes over time. As a result, it is not possible to access in one moment the data corresponding to all possible scenarios, and in consequence to prepare one model, able to handle every change. This problem has become particularly important in recent years as the number of collected data has increased. Therefore, the researchers paid special attention to a field of artificial intelligence called data streams mining (DSM) [1–13]. In the data stream scenario, instead of the static training set, we assume that the data come to the system continuously, one after the other. The DSM is focused on models that can adapt to changes in incoming data. Moreover, these algorithms should minimize two additional criteria: the number of data stored in a system and learning time. The model must be able to

provide the output at any time, and the resources used by the model should be strictly limited. The algorithms of DSM have found many applications, e.g. in network traffic analysis [14], financial data analysis [15], or credit card fraud detection [16]. Recently, the possibilities of combining stream processing methods with deep learning techniques are being explored [13, 17].

One of the most popular techniques for data stream mining are ensemble algorithms [18, 19]. In the classic approach, their main idea is to combine the outputs of models built only on a part of data. This allows for achieving better results with respect to a single component. A simple modification of a classic ensemble algorithm allows us to effectively adapt the model to the changes observed in incoming data. Training new components on chunks of recent data can keep the model up-to-date. The appropriate criterion for including a new component into the ensemble is an important factor that affects the performance of the model. This issue is currently the subject of many studies [20–23].

A non-stationarity phenomenon in the context of data streams is called a concept-drift. In the literature two types of concept-drifts are distinguished: virtual, when changes in the distribution of data do not affect the decision boundaries, and the real, when the decision boundaries are changed. There are a few approaches that allow updating the algorithms to operate in a new environment. One of them is the passive approach [24, 25]. It is based on the continuous adaptation of the model to current data, it is used, inter alia, in ensemble algorithms. Another method, the so-called active approach, is based on the permanent monitoring of the stream itself and to indicate in which moment the concept-drift took place. The methods that indicate the moments of significant change in data distribution are called drift detectors (DDs). In this approach, the model is updated only if DD signalizes that the concept-drift occurred.

The most popular techniques for creating ensembles of classifiers are bagging and random forests (RF). These methods allow the creation of many different models from one training set. For this purpose, they use the bootstrap samples technique. This method consists in generating several subsets by sampling with replacement from the training set. The idea of bagging is to learn independent models based on these subsets. The random forests algorithm also uses bootstrap samples, but additionally, different features are excluded in different subsets. It should be noted that decision trees are used as weak classifiers in the RF algorithm. The adaptation of these algorithms to work in the case of data streams requires some modifications due to the need for minimizing the time for processing available data.

The motives mentioned above inspired us to propose a new algorithm for data streams classification using the RF method. Our algorithm combines the ability of ensemble methods for handling slow changes in a data stream (passive reacting) with a new method, based on FI, developed herein to detecting (active reacting) concept drift occurrence. Developed methods potentially can be applied to monitor various industrial processes, see e.g. [26–31].

The rest of the paper consists of the following sections. Section 2 presents the main trends in the area of ensemble classifiers, random forest, and features importance. In Section 3, the descriptions of the RF algorithm and the method of computing FI are shown. The proposed ensemble algorithm and new drift detector are presented in Section 4. Section 5 depicts results obtained in simulations performed on synthetic and real data. The article ends with the conclusions presented in Section 6.

2 Related works

In this Section, we recall the most significant and the most recent papers about ensemble methods, features importance, and drift detectors.

Ensemble methods are popular techniques of data mining in a static environment. They owe their popularity to the possibility of using them to solve many real-world problems, see e.g. [32]. The most significant features that distinguish various ensemble methods are the method of creating new components and the methods for aggregating outputs. However, their adaptation to operate in the data stream scenario also requires important modifications, in particular, a special approach to data preprocessing.

In the Streaming Ensemble Algorithm (SEA) [18] the authors proposed to create the new clas-

sifier based on chunks of data (subsequently gathered from the stream and forgotten after processing). To decide about classifying a new instance, a major voting strategy was applied. In [33] the authors proposed the Accuracy Weighted Ensemble (AWE) algorithm, which is an improvement of the SEA algorithm by weighting the power of a vote of each component according to its accuracy. Additionally, the authors proved that the decision made by the ensemble will always be at least as good as made by a single classifier. The resampling method inspired by AdaBoost was proposed in the Learn++ algorithm [34], originally in a static environment. Additionally, the authors proposed a new way to establish the weights for the base classifiers. This idea was adapted to the data stream scenario in [35]. In [22] the authors proposed a method to include newly create components only if it ensures increasing the accuracy not only for a current chunk of data but also for a whole data stream. In [23] a new weighting strategy was proposed, assuming that the weak learners are decision trees. Instead of assigning a weight to the whole tree, the authors propose to establish weights in the leaves. The online version of Bagging and Boosting was proposed in [19], and this approach was extended in [36]. For more recent information about ensemble algorithms, the reader is referred to [37] and [38].

In the paper [39], the author presents a procedure of random forest in a static environment, by introducing randomness both on a training set and on a feature set. This idea was tailored to the data stream scenario in several ways. The Dynamic Streaming Random Forest (DSRF) was proposed in [40]. In this approach, after the initial phase of the subsequently generating a finite number of trees, the algorithm update statistics defining thresholds for decision trees construction. Then the algorithm update forest with a fixed percentage of the trees. In the DSRF algorithm, the entropy of incoming data is measured for drift detecting. If the drift is detected, all the parameters of the algorithm are reset to initial values, and the algorithm replaces a specific number of trees in the forest, which number depends on the value of measured entropy. Its ideas are extended in the paper [41]. In [42] the authors propose the Adaptive Random Forests algorithm, which combines classical random forest procedure with Hoeffding's decision trees [43]. To react to changes in data stream, a procedure based on the ADWIN algorithm [44] and the Page-Hinkley test [45] is applied.

As a consequence of the model training based on nonstationary data, the significance of particular features can change over time. Such type of drift is called contextual concept drift [46], or feature drift [47]. In [48] the authors adopt the off-line evaluating feature importance procedure to operate in the on-line scenario with classification models. They proposed two models based on a mean decrease in Gini impurity and a mean decrease in accuracy, respectively. In [49] the authors investigate the statistical properties of feature importance measure to propose a novel algorithm called Reinforcement learning trees. The method called Iterative Subset Selection was proposed in [50]. This method, first ranking the features, and then iteratively selecting the best features from the ranking. More about feature selection on the static and streaming environments can be found in [51, 52], and [53].

In literature, there exist many drift detecting methods. One of the most popular DD is the CUSUM algorithm [45]. It is based on tracing a performance measure (e.g., the accuracy of classifier). If this measure in the consecutive steps exceeds a fixed threshold, then the cumulative sum starts to grow. If it is higher than a certain threshold, then the algorithm indicates concept-drift. The Page-Hinkley test examines differences between current observations and means of previously analyzed data in a similar way to the CUSUM algorithm. The DDM [54] (Drift Detection Method) algorithm treats data from a stream as Bernoulli trials (assigning them values of 0 or 1 depending on whether they were correctly classified by the current model). The final decision is based on a test that takes into account the means and standard deviations of the previous trials. It was enhanced to deal with abrupt concept drift as the EDDM algorithm [55]. The Adwin algorithm [44] is based on a sliding window. It searches the point on current windows to obtain two sub-windows with significantly different means values. The decision is taken on a base of the Hoeffding's bound. Moreover, as drift detectors, different methods of tracing moving averages can be used. One of the most popular is GMADM (geometric moving average detecting method) [56].

3 The formalisms

Let us consider a training set $D = \{(X_1,Y_1),(X_2,Y_2),\ldots,(X_n,Y_n)\}$, where X_i is a d-dimensional feature vector and $Y_i \in Y = \{1,\ldots,l\}$ is a class label, for $i = 1,\ldots,n$. From the classic machine learning point of view, the task is to create the mapping $f : X \to Y$, where $X = f_1 \times f_2 \times \cdots \times f_d$, and f_j corresponds to a single feature, for $j = 1,\ldots,d$.

In a case of ensemble models, the mapping f consists of many so-called, weak classifiers h_i, for $i = 1,\ldots,T$, where T is the size of the ensemble. The decision about predicted class, for vector $\mathbf{x}$, is indicated by the majority voting

$$\hat{y}=\mathrm{f(x)}=$$

$$\arg\max_{c\in Y}[card(\{h_i(x)=c|i=1,...,T\})], \qquad (1)$$

where $card(A)$ denotes the cardinality of the set A.

In the random forest algorithm, the ideas of bagging [57] and random subspaces [58] are combined to create a family of classifiers. Based on the training set D, a new (smaller) training sets are created by sampling with replacement. New weak classifiers are trained on those newly created training sets but restricted only to some subspace of feature.

The different features have different importances. We can say that the i-th feature is unimportant if

$$P(f_A(\mathbf{x}) = y) = P(f_{A\setminus f_i}(\mathbf{x}) = y), \qquad (2)$$

for every $\mathbf{x}$ and y, where f_A is a mapping from the set $A \subseteq X$ into the set Y, $f_A : A \to Y$. From the other side, we say that the feature is important if $P(f_X(\mathbf{x}) = y) \neq P(f_{X\setminus f_i}(\mathbf{x}) = y)$. To measure a level of feature importance is not a trivial task. In particular, the following procedure can be applied (see [39])

1. Calculate the outputs for every tree based on the testing set (B).
2. For each of the features separately
3. Perform the permutation of the considered feature values based on the test set.
4. Calculate the outputs for every tree based on the changed testing set ($\tilde{B}$)
5. Set the value of feature importance as the difference between accuracy obtained on the original and permuted test set, divided by the standard deviation of the outputs.

By applying such an approach, the accuracy is calculated on two different sets for which marginal distributions are identical. This idea, in the next Section, will be extended to deal with data streams.

4 Random Forest with Feature Importance Drift Detector

In this Section, the proposed method of adaptation of random forest to stream data is described. It uses a chunk-based approach, popular pre-processing method. Let say we have stream od data $S = \{(X_1,Y_1),(X_2,Y_2),\ldots,\}$, and data come to the system continuously one after the other. In the chunk-based approach, we try to gather a fixed number of data. After obtaining the first n data from the stream (chunk B_0), one can call out a standard RF algorithm which generates M_0 trees. In the next step, a new chunk of data (B_t), $t = 1,2,\ldots$, is gathered. Before creating a new component, a current chunk of data is used to assess a formerly established ensemble (such procedure is called sequential evaluation). Aside from computing an accuracy, we can also use this chunk of data to obtain the values of features' importance. We can assume that every particular chunk of data is a set of independent random variables. Let f_t be the RF after processing t chunks, $t = 1,2,...$, then values of the function ϕ comparing predictions with actual values are also random variables defined as

$$\phi(X_i) = \begin{cases} 1, & \text{if } f_t(X_i) = Y_i \\ 0, & \text{if } f_t(X_i) \neq Y_i, \end{cases} \qquad (3)$$

for $X_i \in B_t$, $i = 1,2,\ldots,card(B_{t+1})$. Then the accuracy of the classifier given by the following formula

$$Acc(B_{t+1}) = \frac{\sum_{X\in B_{t+1}} \phi(X)}{card(B_{t+1})} \qquad (4)$$

is a mean of random variables taking values from a binomial distribution.

Comparing values of accuracy in the original (B_{t+1}) and permuted ($\tilde{B}^j_{t+1}$) training sets we can define values of feature importance

$$VI_j = Acc(B_{t+1}) - Acc(\tilde{B}^j_{t+1}), \qquad (5)$$

for $j = 1, \ldots, d$.

The initial values of FI (VI_j^0) are computed based on chunk B_0. Every next value of FI will be computed on unseen testing set B_{t+1}. The idea of the proposed drift detection method is to compare, new values of VI_j, with the previously obtained. The significance of the changes is tested by application of the Hoeffding's bound [59]

$$VI_j^0 - VI_j < \sqrt{\frac{R^2 \ln 1/\alpha}{2n}}, \tag{6}$$

where R is a range of considered random variable (in this particular case equal 2), and α is a fixed parameter. If inequality (6) is satisfied, then any changes are not made in the ensemble. New trees can be trained on B_{t+1}, to add them to the ensemble. The number of additional trees, equal to M, is fixed by the user. In the other case, we replace the forest by a new one trained on the current chunk of data and replace VI_j^0 values by the lastly obtained.

In this paper, we will examine three different strategies for computing VI_j values.

FP - Fixed Permutation. In this approach, one permutation is used for every feature and every chunk of data during the whole data stream processing.

SP - Single Permutation. In this approach, one permutation is used for every feature, but with every chunk of data, a new permutation is chosen.

MP - Multiple Permutations. The specific choice of permutation can result in different values of feature importance. In this approach, we will average results obtained by many permutations. This approach allows as to obtain more robust results, however, the number of considered permutations is a bottleneck in terms of the speed of coming data.

The proposed algorithm is called RFFI (Random Forest with Features Importance), and its pseudo-code is given below.

Algorithm 1. The RFFI algorithm

Data: Data stream S in a form of data chunks B_0, B_1, B_2, ...; Number of initial trees M_0, Number of addiitional trees M
Result: Ensemble of classifiers
$t = 0$;
Take the first chunk B_t from the stream ;
Train Random Forest on B_t (M_0 trees);
Compute VI_j^0 on B_t by the equation (5), for $j = 1, 2, \ldots, d$;
while *new data chunks are available* **do**
  $t = t + 1$;
  Take next chunk B_t from the stream ;
  Compute VI_j on B_t by the equation (5), for $j = 1, 2, \ldots, d$;
  Compute differences $VI_j^0 - VI_j$ for $j = 1, \ldots, d$;
  Choose feature F which maximize the computed differences;
  if *inequality* (6) *is not satisfied for feature* F **then**
    $VI_j^0 = VI_j$;
    Train new Random Forest on B_t (M_0 trees);
  else
    Do not make any changes;
    Add new M random trees, trained on B_t, to the forest;
  end
end

5 Experimental results

In this Section, the results of the simulation experiments are presented. Even though many parameters can have an important influence on the RFFI algorithm performance, because of the lack of space, the experiments are focused on reacting on different concept-drift types. The issue of determining the permutation of the values for a given feature is a key issue for the speed of learning of the ensemble. It is worth noting that accuracy can be computed in a parallel way on a new data chunk and its permutations. On the other hand, different permutations can result in a different output of the drift detector. In order to evaluate the repeatability

of the FI values, synthetic data was generated using the Random Tree Generator implemented in the MOA software [60]. The data were described by 25 numerical features and one of a five-class label. The first chunk of stream, containing 2000 data elements, was used to train the RFFI algorithm. The second one, obtained from the same stream (without concept-drift), was used to compute FI. The values obtained by SP, FP, and MP methods (see Section 4), were computed 124 times independently. The results for one feature, in the form of boxplot, are presented in Figure 1. The values of MP are the results of averaging 25 samples.

One can see that the widest variety of values is in the case of the FP method. The remaining methods seem to provide similar values. The MP method gives slightly more stable outputs, but at the expense of an increasing number of computations, it seems to be the worse choice for data stream processing.

In the following subsections, the SP method is applied to investigate various types of concept-drift.

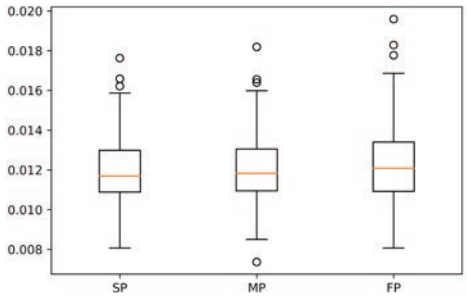


Figure 1. FI values, after 124 iterations, computed by FP (Fixed Permutation), SP (Single Permutations) and MP (Multiple Permutations) methods.

5.1 Abrupt concept drift

The Random Tree Generator was applied to generate 100000 data. The first 50000 was taken from the first concept and the rest of the data from the second one. Both concepts have 25 features, and each element was assigned to one of two classes. The maximal depths of the trees (to generating data) were set for considered concepts as 20 and 15, respectively. In the first experiment, the performance of the RFFI algorithm was compared with different Random Forest-based algorithms, in particular, the Adaptive Random Forest (ARF) algorithm equipped with various drift detection methods. In the simulations, the following DD methods, described in Section 2 were used: Adwin, CUSUM, DDM, EDDM, GMADM, and no-change (NoCh). Those algorithms use VFDT as a weak classifier. The number of components in the RFFI algorithm was set to 10, $M = 0$, and the level of confidence for the drift detector was equal to $\alpha = 0.9$. The ID3 algorithm was applied as the weak classifier. The results obtained after each 1000 data elements are presented in Figure 2.

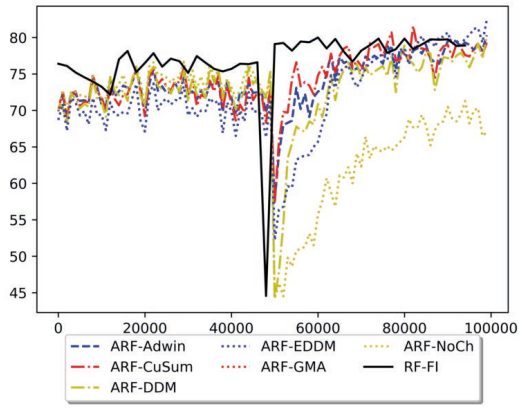


Figure 2. The accuracies (in percent) for ARF algorithm with various DD and RFFI, computed after every 1000 data elements on the synthetic dataset with abrupt concept drift.

One can see that most of the considered algorithms give similar results. This should not come as a surprise due to the fact that all algorithms are based on the same approach (RF). However, it is worth noticing that, the RFFI algorithm detects the change of concept at the earliest. It reacts just like the first chunk of data from the new concept came. All the other DDs require more data.

To compare with different state-of-the-art data streams classifiers, the results were computed for the following algorithms:

- Oza algorithm with 10 components and VFDT as weak classifier [Oza-HT-10]
- Oza algorithm with 50 components and VFDT as weak classifier [Oza-HT-50]
- Oza algorithm with 10 components and VFDT as weak classifier and ADWIN [Oza-B-10]
- VFDT algorithm with entropy as impurity mea-

sure and confidence level of equal to $\delta = 0.05$ [HT-E-0.05]

– VFDT algorithm with entropy as impurity measure and confidence level of equal to $\delta = 0.01$ [HT-E-0.01]

– VFDT algorithm with Gini index as impurity measure and confidence level of equal to $\delta = 0.05$ [HT-G-0.05]

– VFDT algorithm with Gini index as impurity measure and confidence level of equal to $\delta = 0.01$ [HT-G-0.01]

The results are depicted in Figure 3. One can see that RFFI turned out to be better than most of the other algorithms. Only the accuracy of Oza-B-10 was better than RFFI at the end of the stream. This is due to the type of the used weak classifier. Oza-B-10 uses the VFDT algorithm, which allows training components during stream processing. In the case of the proposed algorithm, the static components were used. However, it is worth noticing that the proposed DD reacts to the change earlier than other algorithms.

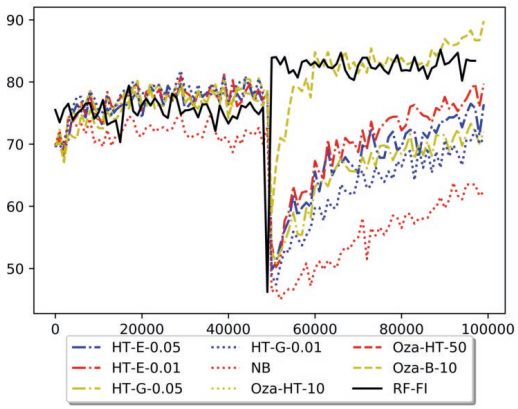


Figure 3. The accuracies (in percent) for RFFI and state of the art algorithms, computed after every 1000 data elements on the synthetic dataset with abrupt concept drift.

5.2 Recurring concept drift

The abilities of the proposed algorithm to react to the recurring concept-drift were compared with the same algorithms as in subsection 5.1. Also the data was generated as in the previous Section. The only change was the number of concept changes. For an initial 50,000 data, they were generated from one concept. Then, data from two different concepts began to be appended to the stream, alternately, each in a package of 2000 elements. Ultimately, the stream contains 100000 elements. The comparisons with other random forest-based and stare-of-the-art algorithms are presented in Figures 4 and 5, respectively.

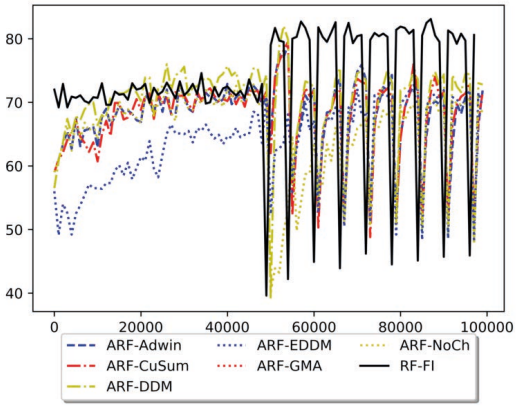


Figure 4. The accuracies (in percent) for ARF algorithm with various DD and RFFI, computed after every 1000 data elements on the synthetic dataset with recurring concept drift.

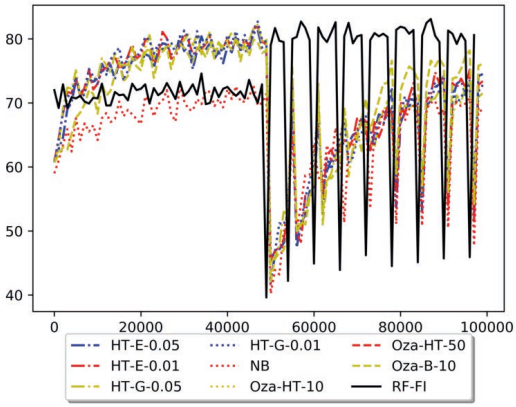


Figure 5. The accuracies (in percent) for RFFI and state-of-the-art algorithms, computed after every 1000 data elements on the synthetic dataset with recurring concept drift.

On a background of the ARF-based algorithm, it is clearly seen that the type of change and its frequency did not allow any of the considered drift detectors to indicate the moment of drift correctly. All algorithms re-create the same components for one concept. The use of static trees instead of VFDT allowed to achieve better accuracy of the proposed method. However, this does not change the fact that

all these methods cannot cope with the detection of rapidly changing abrupt drifts. Comparison with other classifiers shows that they also have a problem with analyzing such frequently changing data.

5.3 Incremental concept drift

To illustrate the performance of algorithms on data with slow incremental changes in the concept, the Hyperplane Generator from the MOA software was applied. The data consist of 25 features, and 10 of them were subject to concept-drift. The magnitude of changes after each data element was set to 0.02 and the probability of reversing the direction of changes was set to 0.1.

Compared to the algorithms in subsection 5.1, only the setting of RFFI was changed. Basing only on the drift detector does not bring satisfactory results in the case of such changes in the distribution. For efficient operation, it is necessary to use the passive property of ensemble algorithms. For this purpose, each time when the drift was not detected, four new components, generated from the last arrived data, were attached to the forest ($M = 4$). Other parameters remained unchanged. The results obtained by RFFI and ARF based algorithms are presented in Figure 6.

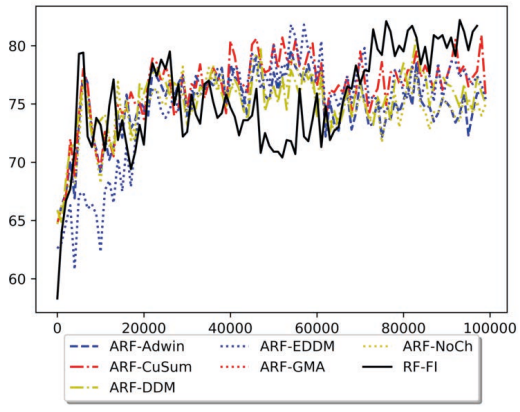


Figure 6. The accuracies (in percent) for ARF algorithm with various DD and RFFI, computed after every 1000 data elements on the synthetic dataset with incremental concept drift.

The results are similar in all methods. This shows that DDs do not play a key role. The comparison with the other classifiers is presented in Figure 7.

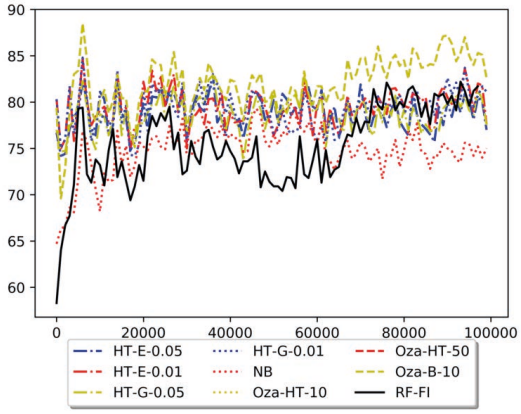


Figure 7. The accuracies (in percent) for RFFI and state-of-the-art algorithms, computed after every 1000 data elements on the synthetic dataset with incremental concept drift.

One can see that only the Oza-B-10 algorithm outperforms the others. It is a consequence application of the Adwin algorithm. The application of sliding windows seems to be especially beneficial in the case of such changes.

5.4 Real-world data

To perform simulation on real-world data, the popular benchmark dataset, called electricity, was applied [54]. The data contains eight features and belongs to one of two classes. The number of data is equal to 45312. The obtained results after processing data chunks, each consisting of 1000 elements, are depicted in Figure 8. The RFFI algorithm was applied with the same values of parameters as in subsection 5.3.

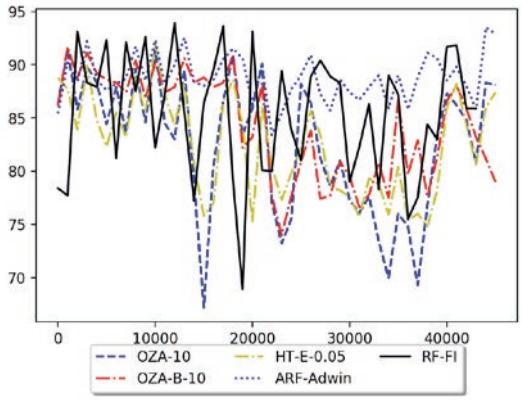


Figure 8. The accuracies (in percent) for RFFI and state-of-the-art algorithms, computed after every 2000 data elements on the real-world dataset.

In the following experiment, the performance of the RFFI algorithm is compared with the best operating algorithm from the previous subsection, i.e. $OZA-10$, $OZA-B-10$, $HT-E-0.05$, $ARF-CUSUM$. The actual moments of the concept-drift occurrence are not known in the case of real data. As the prequential evaluation was used, during the learning process the algorithms obtain various accuracies for the subsequent data-chunks. The proposed algorithm allows for achieving the best result, **93.9**, of all algorithms. However, one can see that the most stable accuracy was provided by the AFT-Adwin, which is probably due to the use of sliding windows. None of the data chunk-based approaches have given better results.

The aggregated values of accuracy and standard deviations obtained after processing the whole stream, and the maximal values of accuracies are depicted in Table 1.

Table 1. Average accuracies (Aa) and standard deviations (Sd), in percents, and the maximum value obtained by the algorithms on real dataset

Algorithm	Aa	Sd	max
Oza-10	82.5	6.25	91.4
Oza-B-10	84.28	4.89	91.7
HT-E-0.05	82.54	4.7	92
ARF-Adwin	**88.8**	**2.22**	93.5
RFFI	85.37	5.79	**93.9**

The presented results demonstrate that the RFFI algorithm can be effectively used to analyze real data.

6 Conclusions

In this article, we proposed a new algorithm for classification in the data stream scenario. Our proposal is based on the random forest algorithm. To enable the algorithm to adapt to changes in the environment, two approaches were used: the mechanism of incorporating newly learned trees into the ensemble and an innovative drift detector. By combining both these techniques, the algorithm can operate in environments of different types of non-stationarity. The proposed drift detector works no worse than other commonly used methods. Besides, when catching changes, it provides information about which feature (or features) has the most significant impact on the detected drift.

As part of further work, we will improve the drift detector to enhance its work in a rapidly changing environment. In the presented version detector operates on data chunks. In the future, we will try to develop its fully on-line version.

References

[1] P. Duda, M. Jaworski, L. Pietruczuk, and L. Rutkowski, A novel application of Hoeffding's inequality to decision trees construction for data streams, in Neural Networks (IJCNN), 2014 International Joint Conference on. IEEE, 2014, pp. 3324–3330.

[2] L. Rutkowski, L. Pietruczuk, P. Duda, and M. Jaworski, Decision trees for mining data streams based on the McDiarmid's bound, IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 6, pp. 1272–1279, 2013.

[3] L. Rutkowski, M. Jaworski, L. Pietruczuk, and P. Duda, Decision trees for mining data streams based on the Gaussian approximation, IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 1, pp. 108–119, 2014.

[4] L. Rutkowski, M. Jaworski, L. Pietruczuk, and P. Duda, The CART decision tree for mining data streams, Information Sciences, vol. 266, pp. 1–15, 2014.

[5] L. Pietruczuk, L. Rutkowski, M. Jaworski, and P. Duda, The parzen kernel approach to learning in non-stationary environment, in Neural Networks (IJCNN), 2014 International Joint Conference on. IEEE, 2014, pp. 3319–3323.

[6] L. Rutkowski, M. Jaworski, L. Pietruczuk, and P. Duda, A new method for data stream mining based on the misclassification error, IEEE Transactions on Neural Networks and Learning Systems, vol. 26, no. 5, pp. 1048–1059, 2015.

[7] P. Duda, M. Jaworski, and L. Rutkowski, Knowledge discovery in data streams with the orthogonal series-based generalized regression neural networks, Information Sciences,, 2017.

[8] M. Jaworski, P. Duda, and L. Rutkowski, New splitting criteria for decision trees in stationary data streams, IEEE Transactions on Neural Networks and Learning Systems, vol. PP, no. 99, pp. 1–14, 2017.

[9] M. Jaworski, P. Duda, L. Rutkowski, P. Najgebauer, and M. Pawlak, Heuristic regression function estimation methods for data streams with concept drift, in Lecture Notes in Computer Science. Springer, 2017, pp. 726–737.

[10] M. Jaworski, P. Duda, and L. Rutkowski, On applying the restricted boltzmann machine to active concept drift detection, in Computational Intelligence (SSCI), 2017 IEEE Symposium Series on. IEEE, 2017, pp. 1–8.

[11] M. Jaworski, Regression function and noise variance tracking methods for data streams with concept drift, International Journal of Applied Mathematics and Computer Science, vol. 28, no. 3, pp. 559–567, 2018.

[12] P. Duda, M. Jaworski, and L. Rutkowski, Convergent time-varying regression models for data streams: Tracking concept drift by the recursive parzen-based generalized regression neural networks, International Journal of Neural Systems, vol. 28, no. 02, p. 1750048, 2018.

[13] P. Duda, M. Jaworski, A. Cader, and L. Wang, On training deep neural networks using a streaming approach, Journal of Artificial Intelligence and Soft Computing Research, vol. 10, no. 1, 2020.

[14] A. Lall, V. Sekar, M. Ogihara, J. Xu, and H. Zhang, Data streaming algorithms for estimating entropy of network traffic, in ACM SIGMETRICS Performance Evaluation Review, vol. 34, no. 1. ACM, 2006, pp. 145–156.

[15] C. Phua, V. Lee, K. Smith, and R. Gayler, A comprehensive survey of data mining-based fraud detection research, arXiv preprint arXiv:1009.6119, 2010.

[16] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, Credit card fraud detection: A realistic modeling and a novel learning strategy, IEEE transactions on neural networks and learning systems, vol. 29, no. 8, p. 3784–3797, August 2018.

[17] S. Disabato and M. Roveri, Learning convolutional neural networks in presence of concept drift, in 2019 International Joint Conference on Neural Networks (IJCNN), 2019, pp. 1–8.

[18] W. N. Street and Y. Kim, A streaming ensemble algorithm (sea) for large-scale classification, in Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2001, pp. 377–382.

[19] N. C. Oza, Online bagging and boosting, in Systems, man and cybernetics, 2005 IEEE international conference on, vol. 3. IEEE, 2005, pp. 2340–2345.

[20] P. Duda, On ensemble components selection in data streams scenario with gradual concept-drift, in International Conference on Artificial Intelligence and Soft Computing. Springer, 2018, pp. 311–320.

[21] P. Duda, M. Jaworski, and L. Rutkowski, On ensemble components selection in data streams scenario with reoccurring concept-drift, in 2017 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 2017, pp. 1–7.

[22] L. Pietruczuk, L. Rutkowski, M. Jaworski, and P. Duda, A method for automatic adjustment of ensemble size in stream data mining, in Neural Networks (IJCNN), 2016 International Joint Conference on. IEEE, 2016, pp. 9–15.

[23] L. Pietruczuk, L. Rutkowski, M. Jaworski, and P. Duda, How to adjust an ensemble size in stream data mining? Information Sciences, vol. 381, pp. 46–54, 2017.

[24] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, Learning in nonstationary environments: A survey, IEEE Computational Intelligence Magazine, vol. 10, no. 4, pp. 12–25, 2015.

[25] P. Duda, L. Rutkowski, M. Jaworski, and D. Rutkowska, On the Parzen kernel-based probability density function learning procedures over time-varying streaming data with applications to pattern classification, IEEE transactions on cybernetics, vol 50, no. 4, pp. 1683-1696, 2020.

[26] E. Rafajlowicz, W. Rafajlowicz, Testing (non-) linearity of distributed-parameter systems from a video sequence, Asian Journal of Control, Vol. 12, no. 2, pp. 146–158, 2010.

[27] E. Rafajlowicz, H. Pawlak-Kruczek, W. Rafajlowicz, Statistical Classifier with Ordered Decisions as an Image Based Controller with Application to Gas Burners , Springer, Lecture Notes in Artificial Intelligence, vol. 8467, pp. 586–597, 2014.

[28] E. Rafajlowicz, W. Rafajlowicz, Iterative learning in optimal control of linear dynamic processes , International Journal Of Control, vol. 91, no. 7, pp. 1522–1540, 2018.

[29] P. Jurewicz, W. Rafajlowicz, J. Reiner, et al., Simulations for Tuning a Laser Power Control System of the Cladding Process , Lecture Notes in Computer Science, vol. 9842, pp. 218–229, Springer, 2016.

[30] E. Rafajlowicz, W. Rafajlowicz, Iterative Learning in Repetitive Optimal Control of Linear Dynamic Processes , 15th International Conference on Artificial Intelligence and Soft Computing (ICAISC), 2016, Springer, vol. 9692, pp. 705–717, 2016.

[31] E. Rafajlowicz, W. Rafajlowicz, Control of linear extended nD systems with minimized sensitivity to parameter uncertainties , Multidimensional Systems And Signal Processing, vol. 24, no. 4, pp. 637–656, 2013.

[32] S. A. Ludwig, Applying a neural network ensemble to intrusion detection , Journal of Artificial Intelligence and Soft Computing Research, vol. 9, no. 3, pp. 177–188, 2019.

[33] H. Wang, W. Fan, P. S. Yu, and J. Han, Mining concept-drifting data streams using ensemble classifiers, in Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. AcM, 2003, pp. 226–235.

[34] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, Learn++: An incremental learning algorithm for supervised neural networks, IEEE transactions on systems, man, and cybernetics, part C (applications and reviews), vol. 31, no. 4, pp. 497–508, 2001.

[35] R. Elwell and R. Polikar, Incremental learning of concept drift in nonstationary environments, IEEE Transactions on Neural Networks, vol. 22, no. 10, pp. 1517–1531, 2011.

[36] A. Beygelzimer, S. Kale, and H. Luo, Optimal and adaptive algorithms for online boosting, in Proceedings of the 32nd International Conference on Machine Learning (ICML-15), 2015, pp. 2323–2331.

[37] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, A survey on ensemble learning for data stream classification, ACM Computing Surveys (CSUR), vol. 50, no. 2, p. 23, 2017.

[38] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Wozniak, Ensemble learning for data stream analysis: A survey, Information Fusion, vol. 37, pp. 132–156, 2017.

[39] L. Breiman, Random forests, Machine learning, vol. 45, no. 1, pp. 5–32, 2001.

[40] H. Abdulsalam, D. B. Skillicorn, and P. Martin, Classifying evolving data streams using dynamic streaming random forests, in International Conference on Database and Expert Systems Applications. Springer, 2008, pp. 643–651.

[41] H. Abdulsalam, P. Martin, and D. Skillicorn, Streaming random forests, 2008.

[42] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, Adaptive random forests for evolving data stream classification, Machine Learning, vol. 106, no. 9-10, pp. 1469–1495, 2017.

[43] P. Domingos and G. Hulten, Mining high-speed data streams, in Proc. 6th ACM SIGKDD Internat. Conf. on Knowledge Discovery and Data Mining, 2000, pp. 71–80.

[44] A. Bifet and R. Gavaldà, Adaptive learning from evolving data streams, in International Symposium on Intelligent Data Analysis. Springer, 2009, pp. 249–260.

[45] E. S. Page, Continuous inspection schemes, Biometrika, vol. 41, no. 1/2, pp. 100–115, 1954.

[46] J. P. Barddal, H. M. Gomes, F. Enembreck, and B. Pfahringer, A survey on feature drift adaptation: Definition, benchmark, challenges and future directions, Journal of Systems and Software, 07 2016.

[47] H.-L. Nguyen, Y.-K. Woon, W.-K. Ng, and L. Wan, Heterogeneous ensemble for feature drifts in data streams, in Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, 2012, pp. 1–12.

[48] A. P. Cassidy and F. A. Deviney, Calculating feature importance in data streams with concept drift using online random forest, in 2014 IEEE International Conference on Big Data (Big Data). IEEE, 2014, pp. 23–28.

[49] R. Zhu, D. Zeng, and M. R. Kosorok, Reinforcement learning trees, Journal of the American Statistical Association, vol. 110, no. 512, pp. 1770–1784, 2015.

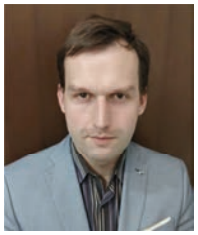
[50] L. Yuan, B. Pfahringer, and J. P. Barddal, Iterative subset selection for feature drifting data streams, in Proceedings of the 33rd Annual ACM Symposium on Applied Computing. ACM, 2018, pp. 510–517.

[51] L. C. Molina, L. Belanche, and À. Nebot, Feature selection algorithms: A survey and experimental evaluation, in 2002 IEEE International Conference on Data Mining, 2002. Proceedings. IEEE, 2002, pp. 306–313.

[52] G. Ditzler, J. LaBarck, J. Ritchie, G. Rosen, and R. Polikar, Extensions to online feature selection using bagging and boosting, IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 9, pp. 4504–4509, 2018.

[53] J. P. Barddal, H. M. Gomes, F. Enembreck, and B. Pfahringer, A survey on feature drift adaptation: Definition, benchmark, challenges and future directions, Journal of Systems and Software, 07 2016.

[54] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, Learning with drift detection, in Brazilian symposium on artificial intelligence. Springer, 2004, pp. 286–295.



Piotr Duda received the M.Sc. degree in mathematics from the Department of Mathematics, Physics, and Chemistry, University of Silesia, Katowice, Poland, in 2009. He obtained the Ph.D. degree and Sc.D. in computer science from Częstochowa University of Technology, Częstochowa, Poland in 2015 and 2019, respectively. His current research interests include deep learning and data stream mining.

Krzysztof Przybyszewski is a professor at the University of Social Sciences in Łódź. His adventure with applied computer science began in the 1980s with a simulation of non-quantum collective processes (the subject of a Ph.D. dissertation). At present, he is involved in research and applications of various artificial intelligence technologies and soft computing methods in selected IT problems (in particular, in expert systems supporting the management of education quality in universities - the use of fuzzy numbers and sets). As a deputy dean at the University of Social Sciences, he is the designer and organizer of the on-Computer Science Faculty education program. He is the author of over 80 publications in the field of computer science and IT applications.

Dr. **Lipo Wang** received the Bachelor degree from National University of Defense Technology (China) and Ph.D. from Louisiana State University (USA). His research interest is artificial intelligence/machine learning with applications to communications, image/video processing, biomedical engineering, and data mining. He has authored 320 papers, of which 110 are in journals. He has authored 2 monographs and edited 20 books. His work has been cited 7,800 times in Google Scholar. He was/will be keynote speaker for 40 international conferences. He was President of the Asia-Pacific Neural Network Assembly (APNNA) and received the APNNA Excellent Service Award.