Adaptive Bagging Methods for Classification of Data Streams with Concept Drift

Martin Sarnovsky, Jan Marcinko

Department of Cybernetics and Artificial Intelligence
Technical University of Košice
Letná 9, 042 00 Košice, Slovakia
martin.sarnovsky@tuke.sk, jan.marcinko@student.tuke.sk

Abstract: Data streams represent a continuous stream of data, in many forms, coming from different sources. Streams are often dynamic and its underlying structure usually changes over time. When solving predictive tasks on the streaming data, traditional models, trained on historical data, may become invalid, when such change occurs. Therefore, adaptive models, equipped with mechanisms to reflect the changes in the data, are suitable to solve these tasks. Adaptive ensemble models represent a popular group of such methods used in classification tasks on data streams. In this paper, we designed and implemented the modifications of the adaptive bagging methods, which utilizes internal class-weighting schemes for the model adaptation. Implemented models were evaluated on two simulated real-world data streams and compared with base classifiers and other adaptive methods. In addition to the performance evaluation, we also analyzed other models' characteristics, such as the duration of model update and memory requirements.

Keywords: concept drift; classification; data streams; ensemble learning

1 Introduction

Nowadays, the size of data is growing much faster, than in the past. Information is being collected from household appliances, tools, mobile devices, GPS, vehicles, various sensors, websites and many other sources. An increasingly large number of organizations are starting to analyze this big data, as the information obtained from these data can provide a competitive advantage over other businesses. Data collection from devices is often continuous, and the data comes in the form of data streams [1]. The data stream can be defined as potentially unlimited, ordered sequence of data items coming in over time. The data streams can be divided according to whether they provide data incrementally (sequentially, one item by one) or in blocks of data. The blocks usually have the same length, and their processing, evaluation or updating is done when all the examples in the new block are available.

There are two basic types of data streams: stationary and non-stationary [2]. For stationary streams, examples are drawn from a fixed, albeit unknown probability distribution. A significant change in attributes is not expected, so in case of predictive tasks applied on this type of stream, it is possible to use a model trained on historical data as expected to perform in constant accuracy over the time. Non-stationary data streams are characterized by changing data over time [2]. This results in a gradual or sudden fall of the models performance. In other words, the concept, which generates a data stream moves after a minimum period of stability. This phenomenon of shifting is called a concept drift or a covariant shift [3].

When solving predictive or classification tasks on streaming data, the data generation process is not strictly stationary and its underlying structure may change over time. From the model training perspective, one of the essential requirements is the ability to adapt and incorporate new data into the model to react to the potential drift occurrence [4]. In that field, the adaptive learning algorithms are advanced machine learning methods that can adapt to new data streams in real-time. There are multiple types of adaptive learning models available, including ensemble methods. Ensemble methods are based on a combination of several models and a combination of their individual predictions into a final one. Based on a different ensemble method types, those models are trained on different training set subsets, using different subsets of predictors. When classifying of a new instance, voting mechanisms are usually applied. Often, ensemble methods have superior performance. Also, due to the method's nature, it is relatively easy to scale such approaches to handle the big data. Among the most popular ensemble methods are boosting and bagging methods [5].

This paper describes the design and implementation of modifications of adaptive ensemble bagging algorithm with different mechanisms of weight-based individual models update. Two modifications of such mechanism are presented and evaluated on two different real-world data streams. The paper is organized as follows: the second section provides basic definitions related to concept drift in data streams and description of various drift types. The following section describes the actual state of the art in the area of the adaptive models used to handle the concept drift and defines the motivation for the presented approach. Next section describes the designed and implemented adaptive ensemble methods. The evaluation section, then describes experiments and results on two datasets. The last section concludes and summarizes the experimenal results and lastly, addresses some possible directions for future research.

2 Drifting Data Streams

For each point at time t, example x is generated, which has a common probability distribution $P_t(x, y)$. The concepts in the data are stable or stationary if all the examples are generated by the same distribution. However, the components in

$P_t(x, y)$ may change. If there is x between the two points t and $t + \Delta$, which holds the expression $P_t (x, y) \neq P_t + \Delta (x, y)$, then a concept is drift present. Some real-world examples of the drifting data may include:

- Traffic monitoring, where traffic may vary over time,
- Weather forecast where climate change and other natural anomalies can affect the final prognosis,
- Systems for tracking individuals' interests, such as personalized ads, where people can gradually change interests.

There are two different types of concept drift. The real drift represents a probability change $P(y/x)$. This can occur with or without changes in the probability $P(x)$. Virtual drift is defined as the change in the value of $P(x)$, or class distribution $P(s)$ that do not affect the decision boundary of the classifiers. Sometimes a virtual drift is defined as a change that does not affect later probabilities. Virtual drift is also referred to as temporary drift or sampling shift.

In addition to the differences between the cause and the effect of changes in concepts, we distinguish several ways these changes have occurred. According to the frequency and velocity of the concept drift, we can recognize four different categories [6]: incremental, gradual, sudden and reoccurring drift. A sudden drift occurs when, at time t, the target distribution S_t , is suddenly replaced by another distribution $S_{(t + 1)}$. For example, when city crime data is gathered, and the classifier tries to predict the development of crime based on these data, and a change in legislative may cause, that some crime types can be classified under a different kind or some cases are no longer considered as offences. With gradual drift, data transformations are not so radical and are associated with a slower rate of change that can only be tracked after a long-term view of the data stream. The gradual drift refers to the transition phase in which the probability of the sample from the first distribution P_j is reduced, while the probability of obtaining the examples from the following distribution $P_{(j + 1)}$ increases. The incremental drift consists of a sequence of small changes. If the difference is slight, the drift can be only captured during observation of a more extended period of data, for example, technological developments, where the gradual development of new technologies is beginning to replace the older ones. Such development is not initially visible but becomes evident in a longer time horizon. In some cases, the concepts can revert over time. A return to previous concepts (repeated in some cycles), represents a seasonal or reoccurring drift. For example, we can use data representing an offering of seasonal jobs, where their number rises significantly at a certain point in time but returns to the original numbers after then [7].

When processing the non-stationary drifting streams, the necessary feature of the predictive algorithms is their ability to adapt. Some of the algorithms are naturally incremental (e.g. Naive Bayes), while other ones require significant changes in the algorithm structure to enable incremental processing. Therefore, the learning

algorithms applied on the drifting streams are usually modified with the mechanisms to update the model with the newly appearing concepts and on the other hand, mechanisms able to forget the obsolete ones. Drift detectors are used to detect the concept drift in the data streams. These methods can detect the possible drift occurrence by analyzing the incoming data or monitoring the classifier performance. Drift detectors then usually trigger the update of the classification model. There are several drift detection methods, Drift Detection Method (DDM) [8], Early Drift Detection Method (EDDM) [9] as the most popular ones. Methods which utilize any type of drift detection are often called as active ones. Another group of adaptive models, also called passive methods, periodically update the model, without any prior knowledge about the drift occurrence.

A very popular group of adaptive models for the drifting data classification, are ensemble models. The ensemble model is in general, composed of a collection of classifiers, also called base learners or experts. The composed ensemble model then combines individual decisions to classify the new examples [10]. The primary motivation behind the ensemble models is the assumption that a set of "weak" classifiers together can achieve better performance than individual classifiers. Bagging (or bootstrap aggregation) represents a popular ensemble method. The basic principle of bagging is in the generation of the m training sets D_i (each of the same size) of the training set D by sampling with replacement. Sampling with replacement causes that some examples from the training set may be repeated in D_i. Then, m classifiers are trained on the created training sets. Outputs of the partial classifiers are combined using voting. Usually, decision trees are applied as base learners in the bagging approach, but it can be used with any kind of classification method. Bagging models are also suitable for data streams classification where target concepts change over time. The following section summarizes the use of ensembles (including bagging methods) in the classification of drifting streams.

3 Related Work

In this section, we describe the current state of the art in the area of adaptive ensemble classifiers used to classify the drifting data [11]. Various types of different adaptive models are available, one of the frequently used groups of such models are ensemble methods. There are several versions of bagging methods implemented for data streams processing with adaptive behaviors, e.g., Online Bagging (or OzaBagging) and Leveraging Bagging [12] [13]. The advantage of these methods is that they can be used not only for processing data streams but also for static data when there is a lack of memory and computing capacity for the processing in a single iteration as evaluation and possibly update of the models on relatively small data sets is less demanding for computing performance [14].

OzaBagging [15], on the other hand, which does not use random sampling from the data, but uses the Poisson distribution, to mimic the bootstrapping. These methods are also capable of handling continuously incrementing data and can adapt to different types of drift based on different weighting rules of individual classifiers. An interesting combination is also using ASHT Bagging (trees of different size) with the ADWIN (Adaptive Windowing) approach. OzaBagging can also be combined with ADWIN when ADWIN can detect the drift and reset the worst classifier in the ensemble [16].

Adaptive bagging was successfully used in the classification of the imbalanced data streams [17]. Moreover, by adapting scalable technologies, online bagging ensembles were successfully used to tackle with the big data [11].

Most of the mentioned ensemble methods are based on different modifications of adaptation rules and different variations of voting mechanisms. The main objective of work presented in this paper is to focus on model re-training mechanisms. We used an adaptive bagging algorithm as a basis and designed and implemented several different mechanisms of partial models re-training. Our idea was based on the evaluation of partial models quality within the bagging ensemble and specifying of rules which of the partial models and how they should be re-trained. For models re-training, we used different approaches, combining re-training using both newly arrived instances and historical data. As a base model, we used tree classifiers. We evaluated the designed and implemented algorithms on two data sets – network intrusion detection and energy consumption prediction data. For evaluation purposes, we used standard model quality metrics (e.g., precision, recall, F1). On the other hand, an essential aspect of adaptive models able to handle a concept drift is also a time and resources needed to re-train and deploy the updated model to react quickly as possible to the drift occurrence. Therefore, metrics describing the resources spent on re-training of the models were also considered.

4 Proposed Adaptive Bagging Methods

We designed and implemented two different variations of the basic adaptive bagging method, each with a different way of updating individual partial classifiers. The differences between them concerned the frequency of updates of the individual ensemble members as well as the number of updated models. Another factor was the way of combining newly arriving data with historical data when updating the ensemble. Either a new sample of data was added to the historical data, and a random set was chosen from this combined sample, based on which a particular classifier was trained, or an entirely new classifier was created using the most up-to-date data and replaced the older one.

Parameter description:

- *D* - a set of trained classification models
- *IW* (iterative window) – represent the data in the current batch
- L_{IW} – the length of the iterative window
- *M* - a list containing the results of metrics (precision, recovery, F1 score)
- *N* – the number of classifiers in the ensemble
- *NU* - the number of models to be updated in each iteration
- *PL* - a set of lists containing predictors for individual classifiers
- *R* - the difference between the *TW* and *IW* sizes
- *TW* (train window) - list consists of lists containing data that are used for classifiers training
- w_i - the weight of i^{th} classifier in the ensemble
- *YP* – list of predicted class values

As the simplest variant of the adaptive bagging algorithm, we used a bagging model which updates all of its ensemble members periodically in each iteration. Its main goal is to evaluate the data in an iteration of a specified length and then update all partial models in the ensemble using these newly obtained data. Initially, the method creates multiple equal subsets of TW_x data, which serve as training data for individual partial models. In the case of working with streaming data, we could simply adjust the processing queue to the required length. In each iteration, the individual D_x classifiers are first trained on the TW_x data. Each partial model in the ensemble then individually evaluates a set of newly arriving data from the last iteration window (*IW*). The results of partial models are sets of evaluated, and the most frequent predicted value is selected. These predicted values are stored and compared to the actual values obtained from the dataset. The last step of the process is to update the *TW* training set with currently received records from the previous iteration window. If the training window *TW* is larger than *IW*, the entire set of data that from the last iteration is included in the updated training set. The rest of the data are selected from the original training data.

4.1 Adaptive Bagging – Weight Update Classifiers (WUC)

This modification of the adaptive bagging method does not update all partial classifiers in each iteration but based on the calculated weights it takes the worst-performing *N* classifiers (with the highest error rate) and updates only those. Other ensemble members remain unchanged.

In the first step, the goal is to predict IW_x data using classifiers *D*. The results are compared to the actual real IW_y values and compute the *F1 metric* for each partial classifier. The *NB* weights, with the highest values, (e.g. *NB*=3, shown in Fig. 1),

represent the ensemble members who are worst-performing members. These classifiers will be updated by re-training them using new data from the stream. The predicted class will be chosen as a majority vote of remaining classifiers (in the current iteration).

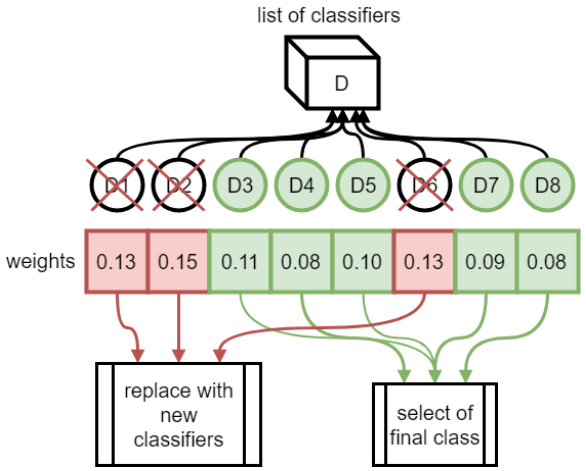

Figure 1

Determining the worst-performing members of the ensemble

This algorithm has not updated all classifiers in each iteration. It does so only with the specified number of worst-performing partial classifiers at a given time as well as it doesn't compute the prediction voting of all partial predictions but only of those that were not updated in the current iteration.

Adaptive Bagging – Weight Update Classifiers
Inputs: *N, LT, L_{IW}, NB*
Create random data samples
For $i = 1, ..., N$
  TW_i = A random sample of the training set
  $TW = TW_i \cup TW$
  Train the classifier D_i on the TW_i data

Return *TW, D*
For each new record in *IW*
  Prediction and evaluation of individual classifiers
  $PL = Ø$
  All weights w set to 1
  For $j = 1, ..., N$
    Add the predictions of D_j classifier on the IW_x to PL_j
    Calculate the *F1 Score* based on the true IW_y class values and predicted class values PL_j
    Weights $w_j = 1 - F1\ Score$

14: *worst_pos* = position *NB* worst performing ensemble members
15: *PLF* = the most common values only for those PL_w, that are not in the list *worst_pos*
Removing *NB* of the worst classifiers and replacing them with new ones
16: **For each** position *p* from the list *worst_pos*
17: **If** *part_fit* = true
18: Merge the lists TW_p and *IW*
19: TW_p = replace the original list of the random data on the size of the original data from the TW_p
20: **Else**
21: *R* = length of *TW* - *LIW*
22: **If** *R* <= *0*
23: TW_p = replace the original list of random data on the size of the original TW_p from the *IW* list
24: **Else**
25: TW'_p = select the last *R* data from TW_p
26: TW_p = merge the lists TW'_j and *IW* and then randomly select the specified % of these data
27: Train the D_p classifier on the TW_p data
28: Replace D_p in the list of classifiers *D* with new D_p
29: **Return** *TW, D*

4.2 Adaptive Bagging – Weight Update Classifiers Parameters (WUCP)

This modification of the adaptive bagging method is an improved version of the WUC adaptive bagging. The initial creation of training sets and initial model training are identical with WUC. WUCP method then works with recovery thresholds, which means that if the performance of a partial ensemble member drops below a certain threshold, then this particular member is updated. The determination of the final prediction is realized in the same way as for the WUC method. Based on the comparison of predicted *PW* data with actual *IW* data, the individual ensemble members are evaluated, and their error rate is transferred to the respective weights. Updates of individual ensemble members occur when their weights reach or exceed the recovery limit. The method works with two recovery thresholds which determine, how the particular ensemble member will be updated, e.g., what data will be used for re-training of the ensemble member.

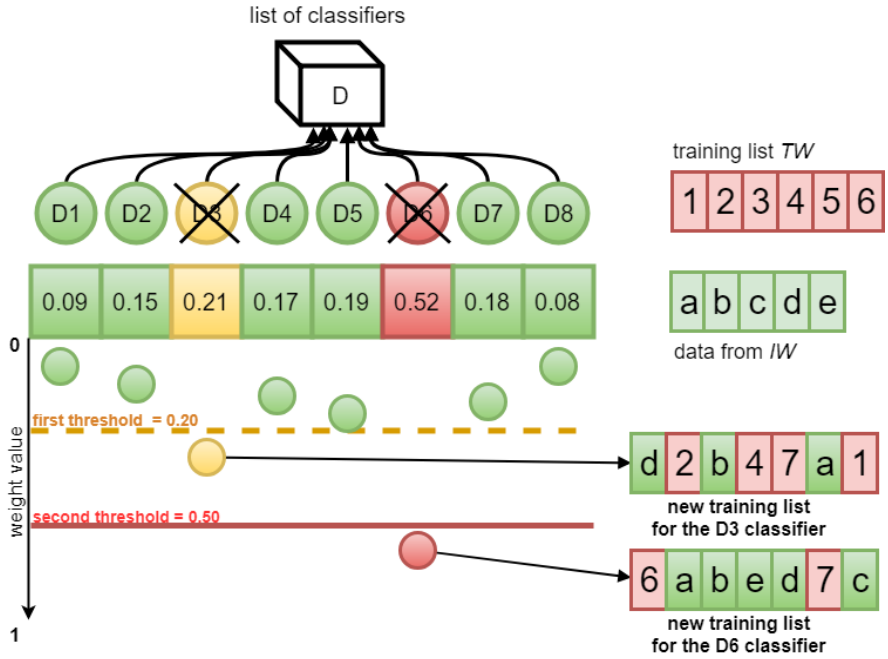

Figure 2

Adaptive ensemble with updated weights of base classifiers parameters

If a partial classifier's weight falls under the first threshold (yellow line depicted on Fig. 2), the ensemble member will be updated by re-training of the expert on a random selection from the dataset which is composed of the training set of the given classifier TW_i of the current *IW* data window. If the ensemble members weight exceeds the second threshold (red line on Fig. 3), it will be re-trained using just the latest *IW* data. If the training set *TW* were larger than *IW*, all *IW* data and the last data of the original TW_i would be included in the creation of the new classifier, which will help us achieve the required size of the training set.

Adaptive Bagging – Weight Update Classifiers Parameters
Inputs: *N*, L_{IW}, *THR_1*, *THR_2*

Create random data samples
1: **For** $i = 1, ..., N$
2: TW_i = random sample of data representing from the training set
3: $TW = TWi \cup TW$
4: Train the classifier D_i on the TW_i data
5: Save $D = D_i \cup D$
7: **Return** *TW, D*
8: **For Each** record in *IW*
Prediction and evaluation of individual classifiers
9: *PL, best_pos, worst_pos* = Ø

```
10:             All weights w set to 1
11:             For j = 1, ..., N
12:                 Add D_j predictions of IW_x records to PL_j
13:                 Calculate the F1 metric based on the actual IW_y classes
                    and predicted classes PL_j
14:                 Weights w_j = 1 – F1 Score
15:             End For
16:             best_pos – position NB of the best weights

                Removing NB of the worst classifiers and replacing them with
                new ones
17:             For Each weight w_x from w
18:                 If w_x > THR_1
19:                     R = length TW - LIW
20:                     If R <= 0
21:                         TW_p = replace random data from IW
22:                     Else
23:                         TW'_p - select the last R records from
                            TW_p
24:                         TW_p = merge the lists TW'j and IW
                            and randomly select the data sample
25:                     End If
26:                     Train the D_p classifier on the TW_p data
27:                     Update D_p members in the ensemble
28:                 Else If w_x > THR_2
29:                     Merge the lists TW_p and IW
30:                     TW_p = replace the random data of the size of
                        the original data from the TW_p
31:                     Train the D_p classifier on the TW_p data
32:                     Update D_p members in the ensemble
33:                 End If
34:             End For
35:     End For
```

5 Experiments

We used two datasets, from two different application domains, to conduct the experiments. Instead of using synthetic data, we focused on using real-world datasets. From the existing available real-world data used frequently in benchmarks, we considered two: network intrusion detection and electricity datasets. Both of the datasets contain a different type of concept drift.

The first dataset is from the KDD Cup competition in 1999 [18]. This data file is a listing of device logs in a LAN network collected over nine weeks. The sample

contains 494,021 rows. The class label is a binary one and specifies if a particular log represents any kind of network attack or normal connection. This dataset contains a sudden concept drift.

The second dataset considered in the experiments was the New South Wales Electricity Market dataset (commonly referenced as the ELEC dataset) [19]. In this case, we used the normalized version of the data. The dataset contains 45,312 instances. The class label identifies the change of the price relative to a moving average of the last 24 hours. These data contain a balanced class distribution and contains incremental concept drift.

On both datasets, we have performed two experiments. During the first set of the experiments, we compared the implemented adaptive bagging models them with baseline classifier (simple decision tree mode), non-adaptive bagging algorithm and basic adaptive bagging model with no weight adaptation mechanisms implemented. As a base classifier in the bagging ensemble, we used decision tree models. Our main objective was to measure the model quality metrics (precision, recall, F1 measure) as well as the time and resources needed to build and update the adaptive models. Lastly, we compared the best adaptive bagging algorithm with other similar adaptive models. Following sub-sections summarize the achieved results. The experiments were performed on a standard computer, equipped with an Intel processor, 8 Gigabytes of RAM running a Windows operating system.

5.1 Experiments on the ELEC Dataset

Table 1

Comparison of model performance on the ELEC data. Time is measured in seconds; performance metrics are calculated as the final percentage of examples over the complete data.

	Base classifier	Bagging	Adaptive Bagging	Adaptive Bagging - WUC	Adaptive Bagging - WUCP
Precision	70.39%	70.58%	82.05%	85.70%	88.22%
Recall	66.91%	66.39%	75.29%	81.58%	84.22%
F1	61.13%	61.14%	73.12%	79.89%	82.90%

Based on the results summarized in Tab. 1, the WUCP method achieved superior performance. Using the decision tree as a base classifier, it scored the highest score in all metrics. The optimal setting (also used in other experiments) of the offset size and the training set size is the value 1000 for both parameters.

During the second set of the experiments, we compared the performance of the adaptive bagging models with the other popular adaptive models. We used the following algorithms: DDM, ADWIN and Page-Hinkley method. Adaptive bagging models were set using the optimal parameters identified from the first set

of the experiments. Comparison of the model performance on the ELEC dataset visualized on Fig. 3 depicts, how the selected metrics evolved during the stream processing. The performance of basic methods decreases faster than adaptive, which makes me aware of the fact that these methods do not update over time and if the concept drift starts to occur in the data, so they have no way to find and further predict values on based on the classifiers learned at the beginning of the process. Adaptive methods also show a decrease in classifier accuracy, but it is not as significant, as static methods.

In the following experiments, our primary goal was to compare the designed adaptive ensemble methods with other adaptive methods. We chose the implementations based on Page-Hinkley, ADWIN, and DDM algorithms for comparison.

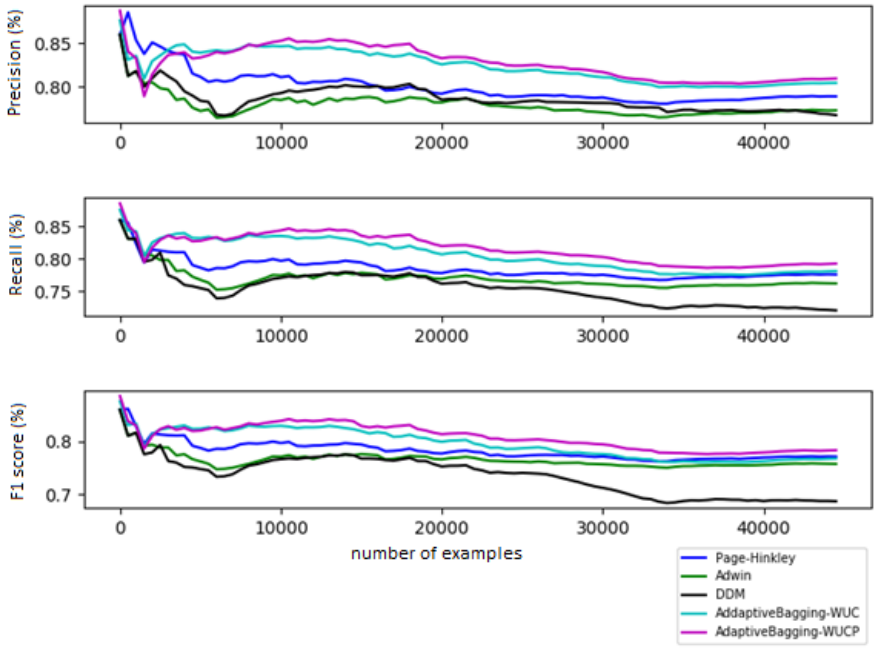

Figure 3
Performance metrics evolution comparison of the implemented models with the other adaptive models on the ELEC dataset

Fig. 3 depicts the performance evaluation of the WUC and WUCP Bagging methods with other adaptive methods. The experiments show that the adaptive bagging methods proved to be more efficient, especially when comparing to the DDM method.

5.2 Experiments on the KDD 99 Dataset

In the first series of the experiments, we compared the performance of the base classifier, standard bagging method, adaptive bagging with a constant update of the ensemble members with proposed adaptive bagging methods. The results are summarized in the Tab. 2. From the performance metrics, we can observe superior performance achieved by the WUCP adaptive bagging method.

Table 2
Comparison of the methods performance on the KDD 99 dataset

	Base classifier	Bagging	Adaptive Bagging	Adaptive Bagging - WUC	Adaptive Bagging - WUCP
Precision	82.95%	93.56%	91.87%	96.24%	99.05%
Recall	76.25%	93.95%	92.79%	96.37%	98.96%
F1	75.46%	93.51%	92.08%	95.76%	98.86%

In a similar fashion as during the experiments on the ELEC data, we compared the performance of the proposed methods to other adaptive models. We focused on the evolution of the F1 metric on the entire simulated data stream. Fig. 4 visualizes the F1 ratio on the simulated stream of the KDD 99 data. As it can be seen from the performance visualization, the adaptive bagging methods (both WUC and WUCP) recover faster from the drift occurrence when comparing with the other adaptive methods. On this dataset, the WUC bagging method proved to be the more efficient one.

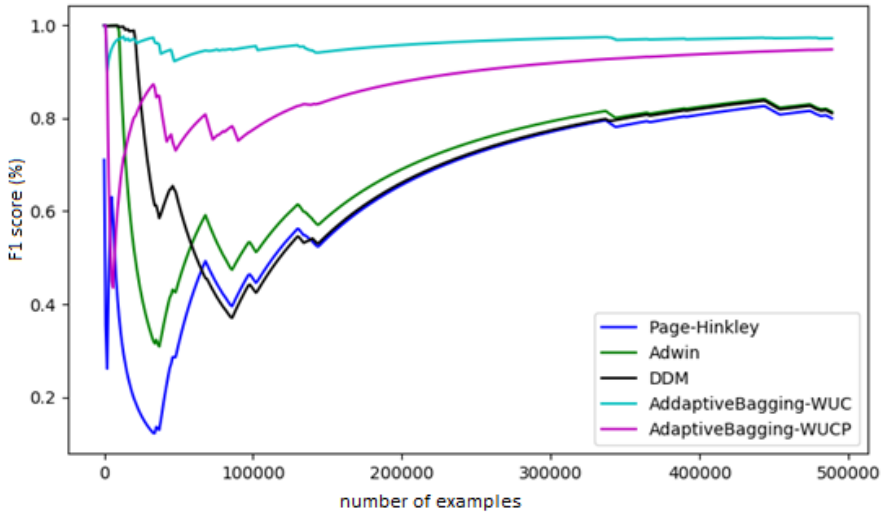

Figure 4
F1 metric evolution comparison of the implemented models with the other adaptive models on the KDD 99 dataset

When comparing adaptive methods on KDD CUP data, we can observe that all methods have a problem when drift occurs suddenly. The success of the method is based on how quickly they are able to adapt to the change. Adaptive bagging methods are able to recover faster when compared to other methods.

5.3 Memory and Computation Requirements Comparison

In this section, we compared how the implemented models performed in terms of their requirements on computational resources. An important aspect of the adaptive models is the fast recovery time - the time needed to update the classification model. In real-world scenarios, the aim is to minimize the recovery time in order to deploy the updated model as fast as possible. In this experiment, we observed how the implemented models utilized the RAM memory and measured the time to update the model during the process. Fig. 5 summarizes the results of the experiments. We can observe that the adaptive bagging methods with implemented class-weighting required slightly more memory. Update time also increased when comparing to the adaptive bagging with no class weighting scheme implemented.

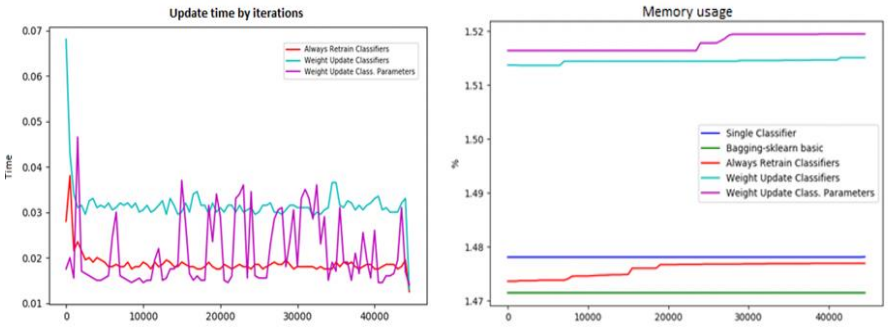

Figure 5

The update time of the adaptive bagging methods and memory usage of the classifiers during the stream processing (Time measured in ms)

The experiment was performed on the ELEC dataset. It is important to note that the observed metrics are highly dependent on the processed data. Data streams with a more complex structure (e.g. with large feature space) require more resources to process. Another important aspect is the actual real stream velocity (the number of items arriving per second), which determines the requirements for the recovery times.

Conclusions

The aim of the presented paper, is to propose weighted modifications, of adaptive bagging classification methods. We evaluated the models on two different real-world datasets that contain a different type of concept drift. In both experiments,

we compared the adaptive bagging methods with the baseline classifier as well as with other incremental od adaptive models. The results prove that the absence of classifier ability to update, when concept drift occurs, results in a gradual decrease of model performance. Adaptive bagging methods achieved relatively good performance results on both of the datasets. Their accuracy and speed of update depended on how large a set of data was used in each upgrade iteration and how many partial classifiers were needed to be retrained. When comparing concrete approaches to retrain the classifiers, we can conclude that it is better to use the partial update method, which is a random selection of historical data and recent examples of the current iteration. Performance-wise, the WUCP adaptive bagging algorithm achieved the best results. When compared to other adaptive methods, those methods performed fast enough and again yielded the best results on both datasets. Based on these findings, we can say that adaptive ensemble methods are well able to be adapted on the data in which the concept drift occurs and can be very useful to the classification method for processing of data streams in a dynamic and ever-changing environment. For future work, we aim to enhance the adaptive ensemble models with a semantic model of the application domain. Such a knowledge model, should be used to improve the classification by capturing the expert domain knowledge, which may be related to drift occurrences. These models could be used to detect the patterns leading to the drift and therefore, be used in drift detection or adaptation rules.

Acknowledgement

This work was partially supported by the Slovak Research and Development Agency under the contracts No. APVV-16-0213.

References

[1] KRAWCZYK, Bartosz, MINKU, Leandro L., GAMA, João, STEFANOWSKI, Jerzy, WOŹNIAK, Michał: Ensemble learning for data stream analysis: A survey, Information Fusion, 37, pp. 132-156, DOI: 10.1016/j.inffus.2017.02.004. ISSN 15662535, 2017

[2] GAMA, João: Knowledge discovery from data streams. Boca Raton, FL: Chapman & Hall/CRC, Chapman & Hall/CRC data mining and knowledge discovery series, ISBN 9781439826119, 2010

[3] ZLIOBAITE, Indre: Adaptive training set formation. Vilnius, PhD thesis. Vilnius University, 2010

[4] BREIMAN, Leo: Bagging predictors. Machine Learning. https://doi.org/10.1007/bf00058655, 1996

[5] GIRAUD-CARRIER, Christophe: A note on the utility of incremental learning, AI Communications, Netherlands: IOS Press Amsterdam, 13(4), pp. 215-223, 2000

[6] TRAJDOS, Pawel, KURZYNSKI, Marek: Multi-label Stream Classification Using Extended Binary Relevance Model. In: 2015 IEEE Trustcom/BigDataSE/ISPA. IEEE, pp. 205-210, DOI: 10.1109/Trustcom.2015.584, ISBN 978-1-4673-7952-6, 2015

[7] TSYMBAL, Alexey: The problem of concept drift: definitions and related work, Technical Report TCD-CS, Trinity College Dublin, Ireland, 2004

[8] GAMA, João, MEDAS, Pedro, CASTILLO, Gladys, RODRIGUES, Pedro: Learning with drift detection, Lecture Notes in ComputerScience (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 2004

[9] BAENA-GARCÍA, Manuel, DEL CAMPO-ÁVILA, José, FIDALGO, Raúl, BIFET, Albert, GAVALDÀ, Ricard, MORALES-BUENO, Rafael: Early Drift Detection Method, In: 4th ECML PKDD International Workshop on Knowledge Discovery from Data Streams, 2006

[10] SAGI Omer, ROKACH Lior: Ensemble learning: A survey, 2018

[11] LV, Y., PENG, S., YUAN, Y., WANG, C., YIN, P., LIU, J., WANG, C.: A classifier using online bagging ensemble method for big data stream learning. Tsinghua Science and Technology. https://doi.org/10.26599/TST.2018.9010119, 2019

[12] OZA, Nikunj C., RUSSELL, Stuart: Online bagging and boosting. In Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics (AISTATS'01) p. 105112, Key West, USA. Morgan Kaufmann, 2001

[13] BIFET, Albert, HOLMES, Geoffrey, PFAHRINGER, Bernhard: Leveraging bagging for evolving data streams. In ECML/PKDD (1), pp 135-150, 2010

[14] ABDALLAH, Zahraa Said, GABER, Mohamed Medhat, SRINIVASAN, Bala, KRISHNASWAMY, Shonali: Anynovel: detection of novel concepts in evolving data streams, Evolving Systems, 7(2), pp. 73-93, 2016

[15] BIFET, A., HOLMES, G., PFAHRINGER, B., GAVALDÀ, R.: Improving adaptive bagging methods for evolving data streams. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) https://doi.org/10.1007/978-3-642-05224-8_4, 2009

[16] READ, J., BIFET, A., HOLMES, G., PFAHRINGER, B.: Scalable and efficient multi-label classification for evolving data streams, Machine Learning, Vol. 88, No. 1-2, pp. 243-272, 2012

[17] WANG, B., PINEAU, J.: Online Bagging and Boosting for Imbalanced Data Streams. IEEE Transactions on Knowledge and Data Engineering. https://doi.org/10.1109/TKDE.2016.2609424, 2016

[18] KDD Cup 1999 Data: Abstract [online] [2018-01-14] Available from: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[19] MOA: Datasets [online] [2018-01-20] Available from: https://moa.cms.waikato.ac.nz/datasets/