Weighted Naïve Bayes Classifier with Forgetting for Drifting Data Streams

Bartosz Krawczyk
Department of Systems and Computer Networks
Wrocław University of Technology
Wrocław, Poland
bartosz.krawczyk@pwr.edu.pl

Michał Woźniak
Department of Systems and Computer Networks
Wrocław University of Technology
Wrocław, Poland
michal.wozniak@pwr.edu.pl

***Abstract*—Mining massive data streams in real-time is one of the contemporary challenges for machine learning systems. Such a domain encompass many of difficulties hidden beneath the term of Big Data. We deal with massive, incoming information that must be processed on-the-fly, with lowest possible response delay. We are forced to take into account time, memory and quality constraints. Our models must be able to quickly process large collection of data and swiftly adapt themselves to occurring changes (shifts and drifts) in data streams. In this paper, we propose a novel version of simple, yet effective Naïve Bayes classifier for mining streams. We add a weighting module, that automatically assigns an importance factor to each object extracted from the stream. The higher the weight, the bigger influence given object exerts on the classifier training procedure. We assume, that our model works in the non-stationary environment with the presence of concept drift phenomenon. To allow our classifier to quickly adapt its properties to evolving data, we imbue it with forgetting principle implemented as weight decay. With each passing iteration, the level of importance of previous objects is decreased until they are discarded from the data collection. We propose an efficient sigmoidal function for modeling the forgetting rate. Experimental analysis, carried out on a number of large data streams with concept drift prove that our weighted Naïve Bayes classifier displays highly satisfactory performance in comparison with state-of-the-art stream classifiers.**

***Index Terms*—machine learning, data stream, concept drift, big data, incremental learning, forgetting.**

I. Introduction

Contemporary computing systems need to deal with massive and complex data, that need to be efficiently transmitted stored and analyzed. Due to nature of such Big Data, we require novel algorithms for efficient data mining on such enormous collections of information. Additionally, designing such analytical tools should take into a consideration that most of data arrives continuously in the form of so-called data stream [8]. Furthermore, the relation within the data, i.e., statistical dependencies characterizing a given phenomenon (such as client behavior) may change [9]. Such a phenomenon is known as concept drift. Important aspect of such shifts ad drifts in data is their impetuosity. According to it, we can distinguish slower changes (gradual or incremental drift) or rapid ones (sudden drift). Such non-stationary nature of analyzed environment can significantly affect the data mining process, as used models can quickly become outdated. Therefore, there is a need for proposing adaptive and flexible classifiers, that can swiftly adjust their structure with an acceptable time constraints [19].

Naïve Bayes classifier has become a popular reference method for data stream mining, due to its online nature, low complexity and reduced memory consumption. however, it performance on changing streams is often far from being satisfactory. It has no mechanism for detecting and handling concept drift, thus making it prone to non-stationary situations. That is why currently it is more often used as a reference method, than as an effective tool for stream mining.

In this paper, we introduce a novel modification of the Naïve Bayes classifier that is able to automatically and swiftly adapt to changes in data. We propose a weighted Naïve Bayes scheme, that benefits from assigning importance factor to training objects. Weighted Naïve Bayes proposed so far in the literature was focused mainly on feature weighting [21] or weighting local neighborhood of data [7]. We show, how to assign weights to all of objects and process them as the stream progresses. The higher the weight, the more significant becomes the object in the classifier training procedure. This way, we can directly control which objects should affect the classifier in the current moment of time. This can be easily translated to data stream analytics. We propose two complementary weight calculation schemes. One is used to boost the importance of new, incoming data - as they reflect the current state and properties of the data stream. The second one is responsible for controlling the objects from previous chunks of data. We propose a smooth forgetting function, that keeps a balance between extracting useful information from older objects and not storing outdated data. This methods allow for our classifier to flexibly adapt to shifts and drifts in streams, without the need for using any kind of drift detector. We show how to directly input these weights into *a posteriori* probabilities calculation during Naïve Bayes training procedure.

We compare our classifier with the canonical Naïve Bayes and two models based on Hoeffding Trees. Experimental analysis carried out on six data stream datasets with different types of concept drift and backed-up by the statistical analysis

978-1-4799-8697-2/15 $31.00 © 2015 IEEE
DOI 10.1109/SMC.2015.375

proves the usefulness of the proposed method. Not only we are able to boost the classification accuracy and improve the recovery rate in the drift presence, but also we maintain high speed and low memory requirements of the original Naïve Bayes classifier.

II. Learning From Non-Stationary Data Streams

There are three main approaches on handling streaming and drifting data:

- Training new classifier every time when new data is available. Such an approach is impractical and prohibitive due to its high costs (especially for sudden drifts).
- Detecting changes in incoming data, and if they are significant enough to be considered as a concept drift then train the classifier on the basis of new data gathered after a drift occurred.
- Utilizing an incremental learning paradigm for the classification model to smoothly adapt to changing nature of incoming objects.

There is a plethora of classifiers designed specifically for stream analysis (both in cases of stationary and drifting ones) [10], [18]. They can be divided into the following types: drift detection algorithms [22], online learners [5], sliding window-based solutions [1], and ensemble approaches [4], [13].

The drift detectors try to alarm the recognition system in case of changes occurrence. The detector can be based on changes in the probability distribution of the instances or classification accuracy. This is known as explicit change detection. One should note, that there is a family of evolving classifiers that continuously adjust the model to incoming data [14], [15]. This in known as implicit drift detection.

Online learners update their structure while processing the incoming data. Here each object is being processed only once, and a strong emphasis is put on limiting the time and memory consumption. Such classifiers work fast and are able to adapt their model in a very flexible manner.

Algorithms based on sliding windows incorporate the forgetting mechanism. This approach is based on the assumption that the recently arrived data have higher relevancy, because they contain characteristics of the current context. Usually sliding window or instance weighting approaches are used here.

The last group consists of algorithms that incorporate a set of classifiers. It has been shown, that a collective decision can increase classification accuracy because the knowledge that is distributed among the classifiers may be more comprehensive. This premise is true if the set consists of diverse members [3]. In a changing environment diversity can also refer to the context. This makes ensemble systems interesting for researchers dealing with concept drift.

III. Weighted Naïve Bayes for Mining Data Streams

Among a plethora of batch and online learning algorithms dedicated to data streams Naïve Bayes classifier is an attractive method due to its simplicity, online nature and low time and memory complexity [17]. One should note, that more work in this field is devoted to proposals of compound models (like ensembles), of which Naïve Bayes is the component [16], than to adapting the nature of this classifier for dealing with the specifics of data streams. In this paper, we focus on introducing a modification of the Naïve Bayes learning procedure that will make it more suitable for evolving data.

When dealing with data streams one needs to take into consideration, that we may work in a non-stationary environments. In such a case it is no longer sufficient to simply process the incoming instances - one needs also to detect the change and properly react to it. Naïve Bayes classifier has no ability to deal with drifting and shifting data, thus loosing effectiveness in such a scenario [2].

We propose to adapt the learning scheme of Naïve Bayes classifier by adding a weighting factor to each of the processed samples. By this, we will be able to control the level of influence a given training object has on the *a posteriori* probabilities computed for each class. Objects that are more representative for a given class (concept) should have assigned a higher weight than others, in order to boost their impact on the classification procedure. One may easily relate this to the evolving data streams. In such a case, the highest weights should be assigned to the current state of the concept in order to force the classifier to adapt to it. Each incoming chunk of data will change the weights used for the computation and, in case of drift presence, will promote the objects from the altered distribution. We call our method Weighted Naïve Bayes for Concept Drift (WNB-CD)

Our weighting concept aims at promoting the objects from the most recently arriving chunk of data. Thus for all of objects from the current stream chunk $\mathcal{DS}_k$ we assign the maximum weights:

$$\forall_{x_i \in \mathcal{DS}_k} \quad w_i = 1. \tag{1}$$

However, by applying only this weighting scheme for new examples, we will get a standard Naïve Bayes method as all of objects will have an identical weight. Additionally, by storing all of the examples extracted from the data stream we enlarge our data set and the memory requirements, what is impractical and leads to a poor generalization ability of our classifier. We can solve this by removing unnecessary and outdated examples [12] that no longer represent the present state of the analyzed data stream. It seems natural that the degree of importance of stored objects reduces as the time passes, especially in case of non-stationary environments (as the current characteristics of objects can differ significantly from those of previous iterations). That is why we propose to add a forgetting principle to our weighted Naïve Bayes classifier.

The simplest way to is a removal of objects coming from the previous (or oldest iteration). Yet in this way we discard all the information they carried - while they still may have some valuable influence on the classification boundary (e.g.,

in case of a gradual concept drift where the changes of the data distribution are not rapid). A method that allows for a gradual decrease of the object influence over time seems a far more attractive idea.

To achieve a forgetting mechanism, we propose to directly modify the weights assigned to objects stored in our system. As we assign the maximum weights to newest examples, we need to reduce the weights of older examples to reduce their importance level. With each iteration, we penalize the objects from previous chunks by reducing their weights according to some forgetting function. With this, we gradually reduce the influence of older objects on the process of calculating *a posteriori* probabilities for each class.

We propose a forgetting mechanism for weight decay implemented as the following sigmoidal function:

$$w_i^{k+1} = w_i^k 2\exp(-\beta(k+1))(1+\exp(-\beta(k+1)) \quad (2)$$

where β is responsible for the forgetting rapidity. Choosing a proper value of $/beta$ is a challenging task, because it is not possible to set its value arbitrary. It depends inverse proportionaly to the rapidity of the model's paramete change. Therefore, its value should be determined experimentally. This method allows for a smooth forgetting of previous data with the rapidity of changes controlled by user.

To further reduce the complexity of our classifier, we propose to introduce a weight threshold ε. All objects with weights lower than this threshold will be discarded from our set stored in the memory. This will speed-up the computation and reduce the memory requirements for our model.

Having these methods for establishing weights for objects from the data stream, we need to embed them into the Naïve Bayes classifier. Let us start from a canonical definition of this classifier.

Naïve Bayes calculates the *posterior* probability of class c_m for a given test object (in our case an instance extracted from the data stream) with given f features as follows:

$$p(c_m|a_1,a_2,\cdots,a_f) = \frac{p(c_m)\prod_{j=1}^{f} p(a_j|c_m)}{\sum_{q=1}^{M}\left(p(c_q)\prod_{j=1}^{f} p(a_j|c_q)\right)}, \quad (3)$$

where M stands for a number of classes in the considered pattern classification problem.

To apply the proposed weighting schemes from Eq. (1) and (2), we may change the way of calculating the individual probabilities used in Eq. (3). We propose to calculate them with the usage of weighted training objects in form of:

$$p(c_m) = \frac{1+\sum_{i=1}^{N} I(c_{x_i}=c_m)w_i}{M+sum_{i=1}^{N} w_i}, \quad (4)$$

where N is the number of training objects in the current state of our data streams $\mathcal{DS}$, c_{x_i} is the class of the i-th training example, and $I(\cdot)$ is the indicator function (assuming value of 1 if the condition is fulfilled, and value of 0 otherwise).

The conditional probability of j-th feature (assuming, that we deal with nominal attributes) is given as:

$$p(a_j|c_m) = \frac{1+\sum_{i=1}^{N} I(a_j=a_{ij})I(c_{x_i}=c_m)w_i}{n_j+sum_{i=1}^{N} I(a_j=a_{ij})w_i}, \quad (5)$$

where n_j is the number of possible values of j-th feature, and a_{ij} is the value of j-th feature for i-th training example.

If the data contains numeric attributes, we discretize them using Fayyad and Irani's MDL-based scheme [6], and from this point process them like nominal attributes.

This allows us to compute a Naïve Bayes classifier for streaming data with weighted objects according to their importance. By combining methods for treating samples from incoming data and old examples, we get an adaptive classifier that can quickly change its characteristics in the presence of shifts and drifts (as it is influenced most by new examples and forgets the outdated ones). Please note, that our method adapt to changes automatically and does not require any explicit drift detector. This is a significant advantage of the proposed model, as it lifts the problem of proper drift detector selection [20] from the user.

In this paper, we propose to run our algorithm for chunk-based data streams in train-then-test manner. Here, objects from new data chunk are firstly classified, then we obtain true labels for them and use them to update our classifier.

To sum it all up, let us present the pseudo-code of our WNB-CD method in Algorithm 1.

Algorithm 1 Weighted Naïve Bayes for Concept Drift (WNB-CD)

Require: input data stream,
Weighted Naïve Bayes classifier Ψ
data chunk size,
Weighted Naïve Bayes incremental training procedure()
threshold ε

1: $i \leftarrow 0$
2: **repeat**
3: collect new data chunk $\mathcal{DS}_i$
4: classify data from $\mathcal{DS}_i$ using Ψ
5: obtain true labels of $\mathcal{DS}_i$
6: calculate the weights of the examples from $\mathcal{DS}_i$ according to Eq. (1)
7: correct the weights of the old data from DS according to Eq. (2)
8: $\mathcal{DS} = \mathcal{DS} \cup \mathcal{DS}_i$
9: remove from DS all objects with weights smaller than ε
10: $\Psi \leftarrow$ Weighted Naïve Bayes incremental training procedure (DS, weights assigned to objects in DS)
11: $i \leftarrow i+1$
12: **until** end of the input data stream

IV. Experimental Study

The aims of our experiments were to establish if the proposed weighting schemes in WNB-CD classifier will allow it to handle the concept drift efficiently and make the Naïve Bayes

learning scheme competetive to other classifiers dedicated to mining data streams.

A. Datasets

There is an abundance of benchmarks for comparing machine learning algorithms working in static environments. However, for non-stationary data streams there is still just a few publicly available data sets to work with [1]. Most of them are artificially generated ones, with only some real-life examples. Following the standard approaches found in literature, we decided to use both artificial and real-life data sets. Details of used benchmarks can be found in Table I.

TABLE I
DETAILS OF DATA STREAM BENCHMARKS USED IN THE EXPERIMENTS.

Data set	Objects	Features	Classes	Drift type
AIR	539 383	7	2	unknown
COV	581 012	53	7	unknown
ELEC	45 312	7	2	unknown
HYP	1 000 000	10	2	incremental
LED	1 000 000	24	10	gradual
SEA	1 000 000	3	4	sudden

B. Methods and Set-up

For the purpose of the experiments, we use our proposed WNB-CD with the following parameters: forgetting factor $\beta = 9$ and the threshold for discarding old objects $\varepsilon = 0.1$.

As reference methods, we use a standard Naïve Bayes classifier (NB), Hoeffding Tree with a static window (HT), and Drift Detection Method with a Hoeffding Tree (DHT).

The data block size used for creating data chunks was $d = 2500$ for all the data sets.

We use the Shaffer post-hoc test [11] to find out which of the tested methods are distinctive among an n x n comparison. The post-hoc procedure is based on a specific value of the significance level α. Additionally, the obtained p-values should be examined in order to check how different given two algorithms are. We set significance level $\alpha = 0.05$.

The experiments were performed on a machine equipped with an Intel Core i7-4700MQ Haswell @ 2.40 GHz processor and 8.00 GB of RAM.

All experiments were conducted in R[2] environment, with the usage of RMOA[3] package. The proposed WNB-CD method was implemented in R by authors.

C. Results and Discussion

The average accuracies achieved by tested algorithms are presented in Table II, their average training times on data chunk are presented in Table III, their average classification (response) times on a new data chunk are presented in Table IV, and their average memory usage is presented in Table V. The results of Shaffer post-hoc test are given in Table VI. Additionally, we present detailed accuracy behavior of the examined methods for one exemplary dataset in Figure IV-C to allow for a visual inspection of their performance.

TABLE II
AVERAGE CLASSIFICATION ACCURACIES [%].

Data set	NB	HT	DHT	WNB-CD
AIR	66.81	64.97	66.13	**68.42**
COV	66.03	**77.24**	59.23	72.14
ELEC	73.12	70.44	65.05	**75.02**
HYP	81.00	87.20	**87.94**	87.20
LED	67.15	65.48	67.30	**69.56**
SEA	86.24	86.88	88.30	**89.37**

TABLE III
AVERAGE CHUNK TRAINING TIME IN CENTISECONDS [CS].

Data set	NB	HT	DHT	WNB-CD
AIR	0.02	0.76	0.58	0.03
COV	0.11	0.42	1.30	0.13
ELEC	0.10	1.04	1.28	0.13
HYP	0.04	0.21	0.37	0.05
LED	0.05	0.30	0.69	0.08
SEA	0.04	0.16	0.19	0.06

TABLE IV
AVERAGE CHUNK CLASSIFICATION TIME IN CENTISECONDS [CS].

Data set	NB	HT	DHT	WNB-CD
AIR	0.21	0.25	0.27	0.21
COV	1.16	0.93	0.64	1.16
ELEC	0.33	0.28	0.24	0.33
HYP	0.19	0.22	0.23	0.19
LED	0.39	0.44	0.49	0.39
SEA	0.08	0.10	0.12	0.08

TABLE V
AVERAGE MEMORY CONSUMPTION IN MEGABYTES [MB].

Data set	NB	HT	DHT	WNB-CD
AIR	0.06	0.05	13.28	0.07
COV	0.05	0.03	0.10	0.05
ELEC	0.02	0.01	0.04	0.02
HYP	0.02	0.01	0.24	0.03
LED	0.03	0.02	0.18	0.04
SEA	0.01	0.01	0.16	0.01

TABLE VI
SHAFFER TEST FOR THE PROPOSED WNB-CD VERSUS REFERENCE METHODS, ACCORDING TO THEIR ACCURACIES. SYMBOL '=' STANDS FOR CLASSIFIERS WITHOUT SIGNIFICANT DIFFERENCES, '+' FOR SITUATION IN WHICH THE METHOD ON THE LEFT IS SUPERIOR AND '-' VICE VERSA.

hypothesis	p-value
WNB-CD vs NB	+ (0.0084)
WNB-CD vs HT	+ (0.0257)
WNB-CD vs DHT	+ (0.0326)

When analyzing the results, one may reach several interesting conclusions. Let us firstly concentrate on comparison between canonical NB and the proposed WNB-CD models. Considering accuracy the proposed modification outperforms standard NB for all six analyzed stream benchmarks. We should emphasize, that introducing our weighting schemes did not lead to any potential decrease of the classification accuracy in comparison to NB model. What is more, for all of datasets we are able to significantly boost the obtained accuracy by

[1] http://en.wikipedia.org/wiki/Concept_drift
[2] http://www.r-project.org/
[3] https://github.com/jwijffels/RMOA

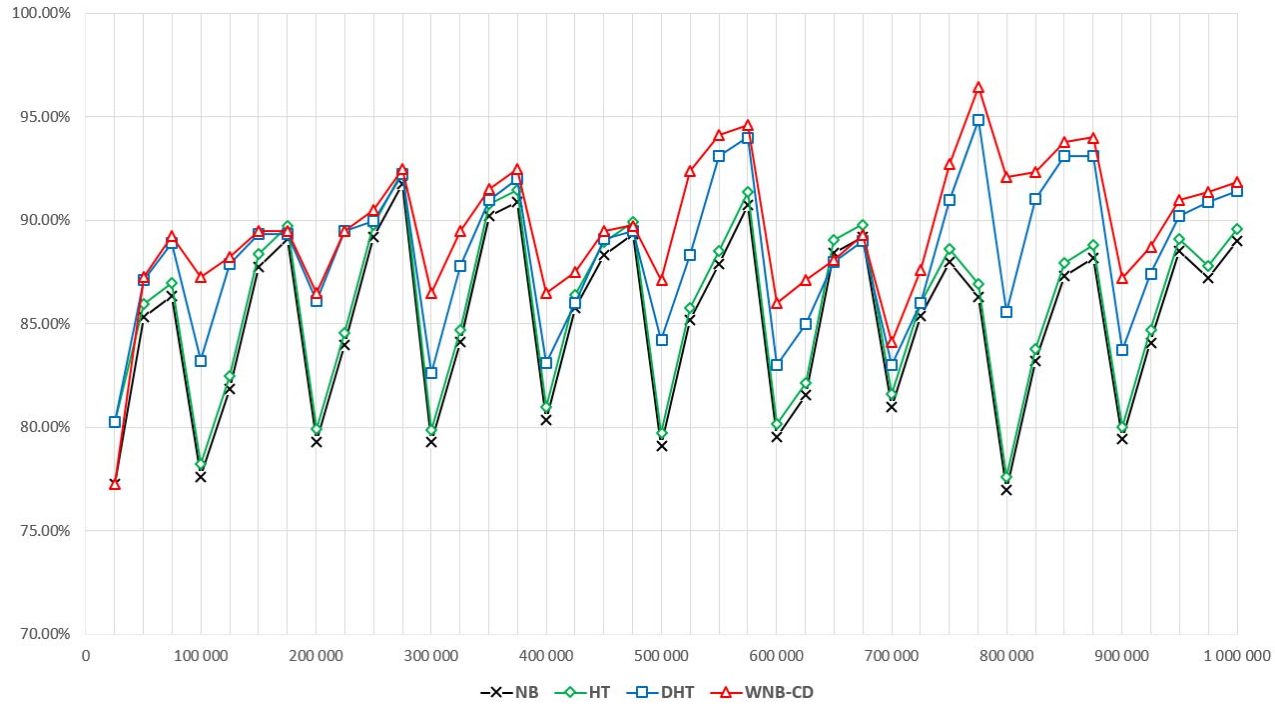

Fig. 1. Accuracies of the analyzed classifiers for SEA data set.

using our proposed weighting schemes. When considering the time and memory complexities of our model, we can see from Tables III- V that our WNB-CD has very similar requirements as NB method. The classification times of both classifiers is identical, as they follow the same procedure. We may observe a marginal increase in memory consumption and training time, due to the need of storing and processing weights assigned to objects. However, the additional cost is practically negligible for modern computing systems. This shows, that it is possible to adapt original NB scheme to work better with drifting data streams without imposing a more resource-consuming training scheme.

When comparing to reference methods based on Hoeffding Trees, one may see that our method returns superior performance in four out of six benchmarks. Only for COV and HYP datasets tree-based classifiers can outperform WNB-CD. In remaining cases, we are able to return more accurate predictions than any examined model in this paper. Shaffer post-hoc test further confirms that the proposed classifier is significantly better than reference methods, when considering their performance over all of datasets. We obtain good p-values (see Table VI), that are much lower than the used significance level $alpha$. Additionally, our classifier maintains low memory and time requirements, making it suitable for many real-life applications. The biggest difference in memory resource requirements can be seen when comparing WNB-CD to DHT.

When analyzing the detailed results of SEA dataset presented in Figure IV-C, one may see that both NB and HT does not perform very well in the presence of changes in data. When a shift occurs they suffer a significant lost of accuracy. After processing a number of new chunks they are able to recover, but time taken makes them almost prohibitive to use in cases of rapidly and frequently changing data streams. HDT and WNB-CD are able to keep the track of changes much more efficiently and quickly adapt their model to novel characteristics of incoming data. Here one should notice, that HDT owns this property to the external drift detector. It is useful, but consumes additional training time and a significant amount of memory. Additionally, it requires labels to detect changes - so may detect a drift with a delay, or even be impossible to use when we have a limited access to true class labels. Our WNB-CD does not require any drift detector. By weighting the importance of objects and smoothly forgetting the outdated ones WNB-CD is able to flexibly adjust itself to the current state of the concept. Therefore, we do not need to explicitly use any kind of drift detector, as our method will evolve its properties (dictated by the weights) according to the nature of incoming data. When looking at the Figure IV-C, we can see that the recovery rate of WNB-CD after the concept drift is very high - we are able to quickly return to a high accuracy on the classified streaming data.

V. Conclusion

In this paper, we have proposed a novel modification of Naïve Bayes classifier adapted for mining data streams with concept drift. We proposed to utilize weights assigned to each training object, in order to reflect its importance. As we deal

with objects coming from different chunks of data (extracted in different moments), it is only natural to control the level of influence they have over the classifier training step. Our WNB-CD classifier incorporates weights into the *a posteriori* probabilities calculation. We proposed two complementary approaches for calculating weights for data. The first one was related to incorporating new objects from the stream. We proposed to assign maximal weights to new objects, as they reflect the current state of the stream. Second method was related with forgetting mechanism, and penalized objects from previous iterations by smoothly decreasing their weights. When a weight of an object has fallen below a given threshold, this object was discarded. This way, we optimized the memory consumption of our method. Our classifier did not needed an explicit drift detector, as it was able to naturally adapt o changes in data.

Experimental analysis on six different data streams and comparison with three reference methods. We have showed, backed-up with the statistical analysis, that WNB-CD could outperform all of them, while keeping very low time and memory complexities.

In our future work, we will examine the forgetting mechanism in order to make it self-adaptable to the type of concept drift in data. We will also examine the possibility of creating online ensembles with the usage of our WNB-CD as a base model for mining high-speed streams.

ACKNOWLEDGMENT

This work was supported partially by the Polish National Science Center under the grant no. DEC-2013/09/B/ST6/02264. The work was also supported by EC under FP7, Coordination and Support Action, Grant Agreement Number 316097, ENGINE European Research Centre of Network Intelligence for Innovation Enhancement (http://engine.pwr.wroc.pl/).

REFERENCES

[1] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà. New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 139–148. ACM, 2009.

[2] D. Brzezinski and J. Stefanowski. Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Trans. Neural Netw. Learning Syst.*, 25(1):81–94, 2014.

[3] B. Cyganek. One-class support vector ensembles for image segmentation and classification. *Journal of Mathematical Imaging and Vision*, 42(2-3):103–117, 2012.

[4] I. Czarnowski and P. Jedrzejowicz. Ensemble online classifier based on the one-class base classifiers for mining data streams. *Cybernetics and Systems*, 46(1-2):51–68, 2015.

[5] P. Domingos and G. Hulten. A general framework for mining massive data streams. *Journal of Computational and Graphical Statistics*, 12:945–949, 2003.

[6] U.M. Fayyad and K.B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993*, pages 1022–1029, 1993.

[7] E. Frank, M.A. Hall, and B. Pfahringer. Locally weighted naive bayes. In *UAI '03, Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence, Acapulco, Mexico, August 7-10 2003*, pages 249–256, 2003.

[8] J. Gama. *Knowledge Discovery from Data Streams*. Chapman & Hall/CRC, 2010.

[9] J. Gama. A survey on learning from data streams: current and future trends. *Progress in AI*, 1(1):45–55, 2012.

[10] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, 2014.

[11] S. García, A. Fernández, J. Luengo, and F. Herrera. Advanced non-parametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Inf. Sci.*, 180(10):2044–2064, 2010.

[12] I. P. Gent, I. Miguel, and N. C. A. Moore. An empirical study of learning and forgetting constraints. *AI Communications*, 25(2):191–208, 2012.

[13] K. Jackowski. Fixed-size ensemble classifier system evolutionarily adapted to a recurring context with an unlimited pool of classifiers. *Pattern Anal. Appl.*, 17(4):709–724, 2014.

[14] D. Jankowski and K. Jackowski. Evolutionary algorithm for decision tree induction. In *Computer Information Systems and Industrial Management - 13th IFIP TC8 International Conference, CISIM 2014, Ho Chi Minh City, Vietnam, November 5-7, 2014. Proceedings*, pages 23–32, 2014.

[15] B. Krawczyk and M. Woźniak. One-class classifiers with incremental learning and forgetting for data streams with concept drift. *Soft Computing*, pages 1–14, 2014.

[16] H. K. H. Lee and M. A. Clyde. Lossless online bayesian bagging. *J. Mach. Learn. Res.*, 5:143–151, 2004.

[17] J. Ren, S.D. Lee, X. Chen, B. Kao, R. Cheng, and D. Cheung. Naive bayes classification of uncertain data. In *Data Mining, 2009. ICDM '09. Ninth IEEE International Conference on*, pages 944–949, 2009.

[18] L. Rutkowski, J. Jaworski, L. Pietruczuk, and P. Duda. A new method for data stream mining based on the misclassification error. *IEEE Trans. Neural Netw. Learning Syst.*, 26(5):1048–1059, 2015.

[19] A. Shaker and E. Hüllermeier. Recovery analysis for adaptive learning from non-stationary data streams: Experimental design and case study. *Neurocomputing*, 150:250–264, 2015.

[20] P. Sobolewski and M. Woźniak. Concept drift detection and model selection with simulated recurrence and ensembles of statistical detectors. *Journal of Universal Computer Science*, 19(4):462–483, feb 2013.

[21] H. Zhang and Shengli Sheng. Learning weighted naive bayes with accurate ranking. In *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 2004), 1-4 November 2004, Brighton, UK*, pages 567–570, 2004.

[22] Indre Zliobaite. Change with delayed labeling: When is it detectable? In *Proceedings of the 2010 IEEE International Conference on Data Mining Workshops*, ICDMW '10, pages 843–850, Washington, DC, USA, 2010. IEEE Computer Society.