ACCEPTED FOR PUBLICATION IN INFORMATION SCIENCES, vol. 381, pp. 46-54, 2017

How to Adjust an Ensemble Size in Stream Data Mining?

Lena Pietruczuk[a], Leszek Rutkowski[a], Maciej Jaworski[a], Piotr Duda[a]

[a] *Institute of Computational Intelligence, Czestochowa University of Technology, Al. Armii Krajowej 36, 42-200 Czestochowa, Poland*

Abstract

In this paper we propose a new approach for designing an ensemble applied to stream data classification. Our approach is supported by two theorems showing how to decide whether a new component should be added to the ensemble or not, based on the assumption that such an action should increase the accuracy of the ensemble not only for the current portion of observations but also for the whole (infinite) data stream. The conclusions of these theorems hold with a certain probability (confidence) set by the user. Through computer simulations, among others, we show that decreasing the confidence that decision based on the finite portion of the stream is the same as based on the whole (infinite) data stream only slightly improves the accuracy at the expense of significant memory consumption. Moreover, we will introduce a novel procedure of weighting ensemble components, i.e. decision trees, by assigning a weight to each leaf of the tree. In previous approaches a weight was assigned to the whole ensemble component. The new approach is based on the observation that probability of the correct tree outcome is different in various tree sections.

Keywords: stream data, data mining, classification, ensemble methods

1. Introduction

Data mining is a subject widely investigated by many scientists all over the world. However, most of developed methods were designed to analyze databases of a finite size. Nowadays the technological development allows to obtain and process huge amount of various data. The number of data elements that is available can not be established in advance. Therefore a new field of study emerged called stream data mining [1] [6] [12] [16] [24] [25] [28] [29] [33] [48] [50]. The problem is to obtain methods that can fast and efficiently extract information from constantly incoming data. Because of the high rate and great complexity of incoming data, previously created algorithms are not applicable. Depending on the particular problem tasks can be very different, e.g. classification [23] [31] [32] [37] [42] [45] [46] [47], clustering [4] [19], [49], regression [35], summarization [10], association rule learning [9] and anomaly detection [11]. In this paper we focus on the first task, i.e. the problem of assigning new data element to one of predefined classes. In the literature there exists a wide range of algorithms designed for static data classification. Most commonly known methods are based on decision trees, neural networks, K-NN, Naive Bayes classifier, support vector machines and ensemble methods. These methods are inspirations for new algorithms developed and dedicated to stream data mining, like the McDiarmid Decision Tree and Gaussian Decision Tree [38] [39] [40] [41], the Very Fast Decision Tree (VFDT) [13] and the Concept-adapting Very Fast Decision Tree (CVFDT)

[21], On Demand Classification [2], ANNCAD Algorithm [26], the Online Information Network (OLIN) [27], the Streaming Ensemble Algorithm (SEA) [43], the LWClass Algorithm [17], the SCALLOP Algorithm [3], the Accuracy Updated Ensemble algorithm (AUE) [7] and the Online Accuracy Updated Ensemble algorithm (OAUE) [8]. Many of these methods exploit the particular advantages of the algorithms previously developed for static problems of data mining.

One of the most promising ideas for designing an efficient system for data stream classification is to create an ensemble of components. Despite the fact that many ensemble methods have been already introduced in the literature, there are still some issues that have to be addressed. One of the most important problems in designing an ensemble method is addressing the question: 'How many components should be put into the ensemble?'. Too small ensemble can not be sufficient enough to cover the whole decision space and therefore the properties of this method could not be fully exploited. On the other hand, too large set of ensemble components may not satisfy memory restrictions and can significantly slow down the decision making process. In most ensemble methods the maximum pool of components that will take part in the classification process has to be defined in advance by the user. This approach was used successfully in many static data problems. However, when we are faced with the stream data challenges, it turns out not to be suitable. The problem with this approach is that it is impossible to establish in advance the proper number of ensemble components that will be optimal in some sense for the whole data stream. Because the number of training data elements is growing constantly, therefore it is potentially possible to create an ensemble of infinite size.

Despite of the drawbacks, many algorithms use a fixed size of ensemble, see e.g. [22] or [36]. In other methods, a pruning mechanism was introduced to reduce the number of ensemble members [15]. However, none of those methods solve the problem on how to decide if the ensemble component should be added, by considering its usefulness to the whole ensemble according to infinite data stream. Even for static problem not much has been done to develop a method of finding the optimal number of ensemble components (see [20] and references cited therein).

In this paper the issue of establishing the proper size of ensemble that will be optimal for the whole (infinite) data stream, by taking into account the ensemble performance and memory requirements, will be considered. To our best knowledge such procedure has never been previously studied in literature.

The main novelty and characteristics of our method can be summarized as follows:

1. We will prove the theorem (see Theorem 1) showing how to decide whether a new component should be added to the ensemble or not, based on the assumption that such an action should increase the accuracy of the ensemble not only for the current portion of observations but also for the whole (infinite) data stream. The conclusion of this theorem will be valid with probability at least $1 - \gamma_1$, where $\gamma_1 > 0$ is a parameter set by the user.
2. We will generalize Theorem 1 (see Theorem 2) by introducing a parameter $\Psi > 0$ describing a minimal increase of the accuracy required in order to add a new component to the ensemble. The conclusion of this theorem will be valid with a probability at least $1 - \gamma_2$, where $\gamma_2 > 0$ is a parameter set by the user.
3. We will introduce a novel procedure for weighting (see formulas (18) and (19)) ensemble components, i.e. for decision trees, by assigning a weight to each leaf of the tree. In previous approaches a weight was assigned to the whole ensemble component. The new approach is based on the observation that probability of the correct tree outcome is different in various tree sections. Therefore, we can use this additional knowledge to increase the accuracy of the whole ensemble.

4. We will show in several experiments that our method of the ensemble size adjustment leads to the best performance for the three real datasets (see Table 2), comparing with well-known 10 algorithms studied by previous authors, and moreover our method is mathematically justified. The simulations for synthetic data (see Table 1) also confirm the usefulness of our method.
5. We will show (see Table 3) that decreasing the confidence (increasing parameter γ_2) that decision based on the finite portion of the stream is the same as based on the whole (infinite) data stream only slightly improves the accuracy at the expense of significant memory consumption (number of trees in the ensemble).
6. We will show through computer simulations that by increasing the value of parameter Ψ (see point 2 above) we significantly reduce the ensemble size, while accuracy is less effected.

The rest of the paper is organized as follows. In section 2 a new method for determining the proper size of the ensemble is shown and in section 3 the performed experimental investigation is described. Section 4 contains the conclusions.

2. The main result

In the stream data scenario we consider a sequence of chunks S_n^t, $t = 1, 2, ...$, each having n elements. In the sequel $S_n = S_n^t$ denotes a current investigated chunk and for clarity of paper the index t will be omitted. Let Γ denote the ensemble of components τ_j, $j = 1, \ldots, |\Gamma|$ and Γ^+ denote the ensemble expanded by one additional component ($\Gamma^+ = \Gamma \cup \{\tau_{|\Gamma|+1}\}$). The problem is to decide if the new created classifier $\tau_{|\Gamma|+1}$ should be added to the ensemble Γ. The added component should provide an increase of accuracy of the ensemble not only for S_n, but also for the whole (infinite) data stream S_∞. Let us define the function $G_\Gamma(X_i)$ as

$$G_\Gamma(X_i) = \begin{cases} 1 & \text{if } X_i \text{ is correctly classified,} \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where X_i, $i = 1, ..., n$, is the i-th data element in the investigated data chunk S_n. Now the accuracy of the whole ensemble can be written as follows

$$P_\Gamma(S_n) = \frac{\sum_{i=1}^n G_\Gamma(X_i)}{n}, \tag{2}$$

$$P_{\Gamma^+}(S_n) = \frac{\sum_{i=1}^n G_{\Gamma^+}(X_i)}{n}, \tag{3}$$

where $G_\Gamma(X_i)$, $i = 1, \ldots, n$, are from the Bernoulli distribution with mean μ and variance $\mu(1-\mu)$ and $G_{\Gamma^+}(X_i)$, $i = 1, \ldots, n$, are from the Bernoulli distribution with mean μ_+ and variance $\mu_+(1-\mu_+)$.

Let us present the following theorem to resolve problem of deciding if the size of the ensemble should increase:

Theorem 1. *Let S_∞ be an infinite stream of data and $S_n = \{X_1, X_2, ..., X_n\} \subset S_\infty$ be a set of n independent random variables. Moreover, let Γ and Γ^+ denote two ensembles where $\Gamma = \{\tau_1, \tau_2, ...\tau_{|\Gamma|}\}$ and $\Gamma^+ = \Gamma \cup \{\tau_{|\Gamma|+1}\}$. If the following inequality holds*

$$P_{\Gamma^+}(S_n) - P_\Gamma(S_n) > z_{1-\gamma_1} \frac{1}{\sqrt{n}}, \tag{4}$$

where $z_{1-\gamma_1}$ is the $(1-\gamma_1)$ quantile of the standard normal distribution $\mathcal{N}(0,1)$, $P_\Gamma(S_n)$ and $P_{\Gamma^+}(S_n)$ are defined by eq. (2) and (3), respectively, then with probability at least $1-\gamma_1$

$$P_{\Gamma^+}(S_\infty) > P_\Gamma(S_\infty). \tag{5}$$

Therefore, if the conditions of Theorem 1 are satisfied, then, with at least $1-\gamma_1$ level of confidence, we can say that by adding a new component $\tau_{|\Gamma|+1}$ to the ensemble we obtain the increase of the accuracy, not only for S_n, but also for the whole data stream S_∞.

Proof of Theorem 1. According to the Multivariate Central Limit Theorem [44] the distribution of the pair $(P_\Gamma(S_n), P_{\Gamma^+}(S_n))$ for sufficiently large n is well approximated by the bivariate normal distribution with mean vector (μ, μ_+), and covariance matrix

$$\Sigma = \begin{bmatrix} Var(G_\Gamma)/n & cov(G_\Gamma, G_{\Gamma^+})/n \\ cov(G_\Gamma, G_{\Gamma^+})/n & Var(G_{\Gamma^+})/n \end{bmatrix} = \begin{bmatrix} \mu(1-\mu)/n & cov/n \\ cov/n & \mu_+(1-\mu_+)/n \end{bmatrix}, \tag{6}$$

where $cov := cov(G_\Gamma, G_{\Gamma^+})$. In consequence both marginal distributions are normal: $P_\Gamma(S_n)$ is from $\mathcal{N}(\mu, \mu(1-\mu)/n)$ and $P_{\Gamma^+}(S_n)$ is from $\mathcal{N}(\mu_+, \mu_+(1-\mu_+)/n)$. Since $P_\Gamma(S_n)$ and $P_{\Gamma^+}(S_n)$ have normal distributions and $(P_\Gamma(S_n), P_{\Gamma^+}(S_n))$ has a bivariate normal distribution then, according to the proof presented in [34], the difference $P_{\Gamma^+}(S_n) - P_\Gamma(S_n)$ has the following normal distribution

$$\mathcal{N}\left(\mu_+ - \mu, \frac{\mu(1-\mu) + \mu_+(1-\mu_+) - 2cov}{n}\right). \tag{7}$$

Obviously $P_\Gamma(S_\infty) = \mu$ and $P_{\Gamma^+}(S_\infty) = \mu_+$. Then, with probability at least $1-\gamma_1$ the following inequality is true

$$P_{\Gamma^+}(S_n) - P_\Gamma(S_n) \leq P_{\Gamma^+}(S_\infty) - P_\Gamma(S_\infty) + z_{(1-\gamma_1)} \sqrt{\frac{\mu(1-\mu) + \mu_+(1-\mu_+) - 2cov}{n}}, \tag{8}$$

where $z_{(1-\gamma_1)}$ is the $(1-\gamma_1)$-th quantile of the standard normal distribution. Therefore, if

$$P_{\Gamma^+}(S_n) - P_\Gamma(S_n) > z_{(1-\gamma_1)} \sqrt{\frac{\mu(1-\mu) + \mu_+(1-\mu_+) - 2cov}{n}}, \tag{9}$$

then, with probability $(1-\gamma_1)$,

$$P_{\Gamma^+}(S_\infty) > P_\Gamma(S_\infty). \tag{10}$$

The values of μ, μ_+ and cov are unknown, however they are bounded as follows:

$$\mu(1-\mu) \leq \frac{1}{4}, \tag{11}$$

$$\mu_+(1-\mu_+) \leq \frac{1}{4}, \tag{12}$$

$$|cov| \leq \sqrt{\mu(1-\mu)\mu_+(1-\mu_+)} \leq \frac{1}{4}. \tag{13}$$

Using (11)-(13) the value of the right hand side of inequality (9) can be limited in the following way

$$z_{(1-\gamma_1)}\sqrt{\frac{\mu(1-\mu)+\mu_+(1-\mu_+)-2cov}{n}} \leq z_{(1-\gamma_1)}\frac{1}{\sqrt{n}}. \tag{14}$$

Therefore, if

$$P_{\Gamma^+}(S_n) - P_{\Gamma}(S_n) > z_{(1-\gamma_1)}\frac{1}{\sqrt{n}}, \tag{15}$$

then (9) is also satisfied. From this we can conclude that if (15) is true, then with probability at least $1-\gamma_1$, the addition of classifier $\tau_{|\Gamma|+1}$ will result in an increase of accuracy.

Let us now describe the procedure of making a decision whether a particular classifier should be added to the ensemble. First, an ensemble Γ^+ is created by combining classifiers from Γ and temporary classifier τ_{temp} learned on the last data chunk. Next, a new chunk of data elements is collected. Based on the new data chunk the values of P_Γ and P_{Γ^+} are calculated using (2) and (3), respectively. Then, if the conditions of Theorem 1 are satisfied, the classifier τ_{temp} is added to the ensemble. However, if condition (4) is not fulfilled, then the investigated classifier is discarded. Next a new classifier is created based on that chunk and is labeled by τ_{temp}.

Sometimes the component, that is found to be useful as a new ensemble member according to Theorem 1, can only slightly increase the accuracy of the whole ensemble. This is an undesirable property because that component uses up memory resources with a little performance increase of the whole ensemble. Therefore, a new theorem with an additional parameter Ψ is proposed. This parameter defines the minimal increase of the accuracy that has to be gained in order to add the component $\tau_{|\Gamma|+1}$ to the ensemble.

Theorem 2. *Let S_∞ be an infinite stream of data and $S_n = \{X_1, X_2, ..., X_n\} \subset S_\infty$ be a set of n independent random variables. Moreover, let Γ and Γ^+ denote two ensembles where $\Gamma = \{\tau_1, \tau_2, ...\tau_{|\Gamma|}\}$ and $\Gamma^+ = \Gamma \cup \{\tau_{|\Gamma|+1}\}$. If the following condition is satisfied*

$$P_{\Gamma^+}(S_n) - P_{\Gamma}(S_n) - z_{1-\gamma_2}\frac{1}{\sqrt{n}} > \Psi, \tag{16}$$

where $P_\Gamma(S_n)$ and $P_{\Gamma^+}(S_n)$ are defined by eq. (2) and (3), respectively, and $\Psi > 0$, then with probability at least $1-\gamma_2$

$$P_{\Gamma^+}(S_\infty) - P_{\Gamma}(S_\infty) > \Psi. \tag{17}$$

Proof of Theorem 2. The proof of this theorem can be conducted in the same manner as the proof of Theorem 1.

Based on the presented theorems we propose a new procedure of obtaining the optimal number of components in the ensemble. The procedure is presented in the form of the pseudocode in Algorithm 1.

Algorithm 1 The Dynamically Expanded Ensemble Algorithm (DEEA)

Require: $S_n = \{X_1, X_2, ..., X_n\}$ - set of data elements, $\Psi \geq 0$, Γ - ensemble of components, τ_{temp} - new classifier, $\Gamma^+ = \Gamma \cup \{\tau_{temp}\}$

1: **for** $i = 1, ..., n$ **do**
2: obtain value of $G_{\Gamma}(X_i)$ and $G_{\Gamma^+}(X_i)$
3: **end for**
4: calculate $P_{\Gamma}(S_n)$ and $P_{\Gamma^+}(S_n)$ using (2) and (3)
5: **if** $P_{\Gamma^+}(S_n) - P_{\Gamma}(S_n) - z_{1-\gamma_2}\frac{1}{\sqrt{n}} > \Psi$ **then**
6: $\Gamma = \Gamma \cup \tau_{temp}$
7: **else**
8: remove τ_{temp}
9: **end if**

3. Experimental Results

3.1. Experiments settings

We will investigate the performance of the proposed method on 7 synthetic and 3 real data bases. Synthetic data bases were obtained with the use of the MOA (Massive Online Analysis) [5] free open-source software (version: MOA Pre-Release 2015.10). This framework is used to compare the performance of our method with the results obtained by other known in the literature algorithms. The synthetic databases consisted of 1 million data elements generated using the following settings:

Agrawal Generator (denoted by AgrGen) - generates the data from some loan function. These data were generated with default settings of the MOA framework (function 1, instance random speed 1, peturb fraction 0.05). The generated dataset consisted of six numeric (salary, commission, age, hvalue, hyears, loan) and three nominal (elevel, car, zipcode) attributes. Each data element belongs to one of two classes: groupA or groupB.

Hyperplane Generator (denoted by HypGen) - simulates the problem of predicting the class to data element where the decision class boundary is described by the rotating hyperplane. Obtained dataset was generated with the default settings of the MOA framework (instance random seed 1, number of classes 2, number of attributes 10, magnitude of change 0, noise percentage 5, sigma percentage 10). Each data element consisted of 10 values of numerical attributes and the assigned class label (class1 or class2).

LED Generator (denote by LED) - generator produces a data to analyze the problem of predicting the digit displayed on a 7-segment LED display. These data were generated with default settings of the MOA framework (instance random seed 1, noise percentage 10). We have to our disposal data elements with 24 binary attributes. Each data element is assigned to one of ten classes (0, 1, 2, 3, 4, 5, 6, 7, 8, 9).

Random Decision Tree (denoted by RDT) - stream of data is generated based on randomly generated tree. Obtained data were generated by using the MOA framework with default settings (tree random seed 1, instance random seed 1, number of classes 2, number of nominal attributes 5, number of numerical attributes 5, number of different values of nominal attributes 5, maximal tree depth 5, first leaf level 3, leaf fraction 0.15).

Random RBF Generator (denoted by rRBF) - generates a stream based on random radial function. Those data were also generated with default settings of the MOA framework (model random seed 1, instance random seed 1, number of classes 2, number of attributes 10, number of centroids 50). Each data element is a ten dimensional vector of real values from [-1.0,1.0] and is assigned to one of two classes (class1 or class2).

SEA Generator (denoted by SEA) - the procedure for data generating is described in [43]. Obtained dataset was generated with the default settings of the MOA framework (function 1, instance random seed 1, noise percentage 10). Each data element was described by three attributes taking values in the interval [0.0,10.0] and a class label (groupA or groupB).

Waveform Generator (denoted by WavGen) - this generator allows to analyze the problem of how to predict one of three waveform types. The data were generated with the instance random seed 1 in MOA settings. The generated dataset consists of 21 numeric attributes with values in [-5.0,5.0] and class value (class1, class2 or class3).

For the purpose of this experiments we use also three real datasets obtained from UCI Machine Learning Repository [30]:

Covertype Data Set - This dataset consists of data denoting the parameters of forest cover. It was introduced to obtain a prediction of the cover type only from cartographic variables. This dataset consists of 581 012 data elements, each of which is a 54 dimensional vector (10 quantitative variables, 4 binary wilderness areas and 40 binary soil type variables). Each data element is labeled by one of seven classes describing the types of forest cover.

Abalone Data Set - This dataset was created to predict the age of abalone from physical measurements. Each data element in this set consists of the 8 attribute values (one nominal and 7 numerical) and one integer value (class) describing the abalone age. There are 29 classes and 4 177 instances in this dataset.

Connect-4 Data Set - This dataset consists of data describing the possible moves in game connect-4. Only not forced moves are taken into account. In this dataset there are 42 attributes with possible values 'x', 'o', 'b' denoting 'player x has taken', 'player o has taken' and 'blank', accordingly, where 'x' is the first player and 'o' i the second player. There are three class labels (win, loss, draw) denoting the outcome regarding the first player. There are 67 557 instances in this dataset.

For comparison of the proposed method with known in the literature algorithms we used the MOA framework. We obtained results for the following algorithms:

- the Accuracy Updated Ensemble classifier (denoted by AUE),
- the Accuracy Weighted Ensemble classifier (denoted by AWE),
- the Hoeffding Tree algorithms (denoted by Hoef.Tree),
- The Hoeffding Adaptive Tree (denoted by Hoef.Ada.Tree),
- the Hoeffding Option Tree (denoted by Hoef.Opt.Tree),
- the Naive Bayes incremental learner (denoted by Naive Bayes),
- the Incremental On-line Bagging algorithm (denoted by Bagging),

- the Bagging Adwin algorithm (denoted by BagAdwin),
- the Incremental On-line Boosting (denoted by Boosting),
- the Weighted Majority algorithm (denoted by WMA).

For the purpose of simulations in our method we used the classic procedure of decision tree construction to create ensemble components. Three most commonly known split measures were used, that is Gini gain (based on Gini index), information gain (based on information entropy) and split measure based on misclassification error. We use the following notation for different settings of our method

- DEEAg - Gini gain,
- DEEAi - information gain,
- DEEAm - misclassification error.

We also investigated the performance of our new method with the use of chunks of different sizes. In the sequel the chunk size is denoted by n, where $n = 500$ and $n = 1000$.

For the purpose of the experiments we used only decision trees as ensemble components. We developed a new way to measure the usefulness of components (new method of assigning weights to each component). Based on the observation of the decision trees performance we can notice that different tree leaves have different accuracy. In previous approaches one weight was calculated as a performance measure of the whole decision tree. Contrary to that we propose to assign a weight to each leaf of the decision tree rather than one weight to the whole tree. In this case we obtain a decision of the tree in the form of the weight calculated for a leaf to which investigated data element was sorted. Then the ensemble outcome is a class for which the highest value of a sum of outcomes (appropriate leaves weights) of all decision trees (ensemble components) was obtained:

$$\Gamma(X) = \underset{C \in \xi}{\operatorname{argmax}} \sum_{j=1}^{|\Gamma|} \mathbb{1}_{\{\tau_j(X)=C\}} \mathcal{W}\left(l_j(X)\right) \tag{18}$$

where $l_j(X)$ denotes the leaf in τ_j to which data element X was sorted, $\tau_j(X)$ denotes the class assigned to X by τ_j, $\mathcal{W}(l_j(X))$ denotes the value of the weight for τ_j in the leaf $l_j(X)$ and ξ is a set of all possible classes. The weight for each leaf is updated with time in the following form

$$\mathcal{W}_t(l_j(X)) = \alpha \cdot \mathcal{W}_{t-1}(l_j(X)) + (1 - \alpha) \cdot Acc_t, \tag{19}$$

where t is a time stamp, $\alpha \in [0; 1)$ is a parameter set by the user and Acc_t denotes the accuracy of the investigated leaf obtained for current data chunk $S_n = S_n^t$. If the value of Acc_t in the considered leaf is unknown, because no data elements were seen in it so far, then we take the probability of random guess $1/|\xi|$ as a weight value where $|\xi|$ is the number of classes. It is also worth noticing that for $\alpha = 0$ the weight in a leaf is equal to its most recent accuracy value.

3.2. The performance analysis

First we will compare the performance of our method with the mentioned above algorithms. The obtained results are presented in Tab. 1 and Tab. 2, where $n = 500$ and $n = 1000$, $\gamma_2 = 0.2$, $\alpha = 0.2$ and $\Psi = 0$. The best obtained accuracies for a particular database is marked by coloring

the background of a table cell. In the first table we obtained the results for all synthetic data bases. As we can see our method gives better results with all investigated settings for the database ArgGen. A better accuracy was also obtained for databases LED and WawGen. In the second table (see tab. 2) we obtain the results for real data sets. For Covertype data all investigated settings of DEEA give better results than obtained for other methods. For databases Abalone and Connect-4 we obtained better accuracy for particular settings. Although it is surprising that the MOA framework in three cases (see tab. 2) provided a very low performance for the Hoef.Ada.Tree (6.17%), BagAdwin (6.36%) and Boosting (2.40%) algorithms, despite of many trials, we include the results of these simulations for the sake of the experiment completeness.

We have investigated the performance of our method (DEEAg, $n = 1000$, $\alpha = 0.2$ and $\Psi = 0$) for different values of parameter γ_2 (see tab. 3). As we can see in most cases along with the growth of the parameter value the accuracy increases. This is the result of the ensemble size growth. The number of components in some cases grows to a great size, e.g. for the Covertype dataset with $\gamma_2 = 0.2$, there are 54 trees stored in the ensemble. Moreover, let us notice that in most cases the accuracy of the DEEA changes a little along with the growth of parameter γ_2. However, the size of the ensemble grows very fast for most databases. In the case of the AgrGen and Covertype databases even for $\gamma_2 = 0.01$ the results are better compared to other algorithms.

Finally let us compare the performance of our method depending on the value of parameter Ψ (see Theorem 2). For this experiment $n = 1000$, $\gamma_2 = 0.2$ and $\alpha = 0.2$. The results are shown in tab. 4. By analyzing the first two rows of this table we can observe, in most cases, the biggest difference in performance of our algorithm. The change of Ψ from 0 to 0.1 causes the decrease of accuracy (from 0.99% for the SEA dataset to 7.58% for the LED database). However, the most significant change can be observed by analyzing the ensemble size. The use of the parameter Ψ causes a significant reduction of the number of ensemble components - for the Covertype dataset the size decreases by 40 components and for the WavGen 32 components less were added to the ensemble. Based on this experiment we can conclude that with the use of parameter Ψ we can significantly reduce the ensemble size with some accuracy loss.

4. Conclusions

In the paper a new method for deciding if a new component should be added to the ensemble was proposed. Two theorems were presented that give the foundations for this method. These theorems allow determining, with defined by the user confidence, if by adding the additional component we will obtain an increase of accuracy of the whole ensemble for the entire infinite data stream. Moreover, a new method for increasing the ensemble performance, by assigning new weights to the ensemble component, was proposed. The series of experiments shown that obtained ensemble method gives satisfactory results and can outperform other methods existing in the literature.

5. Acknowledgements

This work was supported by the Polish National Science Center under Grant No. 2014/15/B/ST7/05264.

[1] C. Aggarwal, Data Streams: Models and Algorithms, New York: Springer, LLC, 2007
[2] C. Aggarwal, J. Han, J. Wang, P. S. Yu, On Demand Classification of Data Streams, Proc. 2004 Int. Conf. on Knowledge Discovery and Data Mining (KDD04), Seattle, WA, 2004
[3] G.T. Balls, S.B. Baden, P. Colella, SCALLOP: A Highly Scalable Parallel Poisson Solver in Three Dimensions, Proceedings of the 2003 ACM/IEEE Conference on Supercomputing, SC '03, ACM, (2003) 23

Table 1: Accuracy of synthetic data

	AgrGen	HypGen	LED	RDT	rRBF	SEA	WavGen
AUE	95,05%	91,52%	73,96%	98,61%	95,57%	89,92%	85,57%
AWE	94,83%	93,48%	73,96%	80,26%	71,61%	87,79%	81,76%
Hoef.Tree	67,20%	90,10%	74,11%	96,78%	92,65%	89,70%	84,69%
Hoef. Ada.Tree	94,77%	90,16%	74,09%	97,39%	92,87%	89,71%	84,72%
Hoef. Opt.Tree	95,06%	90,10%	74,11%	96,78%	93,62%	89,70%	84,66%
Naive Bayes	88,61%	94,22%	74,07%	73,67%	71,98%	88,23%	80,45%
Bagging	95,06%	91,15%	74,14%	97,47%	95,06%	89,86%	86,02%
BagAdwin	95,07%	91,30%	73,99%	97,47%	95,06%	89,86%	86,00%
Boosting	93,23%	90,45%	73,99%	98,81%	94,89%	89,69%	85,04%
WMA	95,07%	94,18%	74,10%	96,87%	92,68%	89,79%	84,68%
DEEAg n=1000	97,22%	88,28%	71,07%	90,71%	89,53%	89,22%	87,26%
DEEAg n=500	95,87%	88,51%	69,21%	89,35%	89,15%	88,91%	87,89%
DEEAi n=1000	97,28%	88,34%	80,73%	90,76%	89,53%	89,22%	87,10%
DEEAi n=500	95,84%	88,72%	69,22%	89,29%	89,17%	88,91%	86,92%
DEEAm n=1000	96,74%	89,96%	77,82%	88,71%	92,60%	89,46%	89,52%
DEEAm n=500	95,33%	90,35%	77,87%	88,69%	91,81%	89,13%	89,17%

Table 2: Accuracy of real data

	Covertype	Abalone	Connect
AUE	36,46%	24,30%	61,72%
AWE	36,46%	24,90%	43,93%
Hoef.Tree	66,04%	23,99%	72,94%
Hoef. Ada.Tree	6,17%	23,82%	60,28%
Hoef. Opt.Tree	73,88%	23,99%	72,58%
Naive Bayes	65,43%	23,99%	72,20%
Bagging	74,28%	24,44%	74,74%
BagAdwin	6,36%	24,76%	55,62%
Boosting	72,18%	2,40%	74,98%
WMA	23,99%	23,99%	72,63%
DEEA n=1000	79,73%	20,90%	72,49%
DEEAg n=500	80,60%	20,77%	74,06%
DEEAi n=1000	81,51%	27,20%	71,96%
DEEAi n=500	83,14%	35,63%	75,65%
DEEAm n=1000	81,81%	24,93%	72,82%
DEEAm n=500	81,58%	29,21%	74,82%

Table 3: Dependence of the performance on the value of γ_2

	0,01		**0,05**		**0,1**		**0,15**		**0,2**	
γ_2	Accuracy	Trees count	Accuracy	Trees count	Accuracy	Trees count	Accuracy	Trees count	Accuracy	Trees count
AgrGen	95,32%	2	96,18%	3	96,54%	4	96,98%	5	97,22%	6
HypGen	86,12%	6	87,33%	10	87,87%	15	87,89%	17	88,28%	24
LED	69,58%	4	70,06%	3	69,80%	4	70,65%	5	71,07%	7
RDT	89,83%	4	89,91%	5	90,28%	8	90,73%	8	90,71%	11
rRBF	88,87%	5	89,06%	7	89,37%	9	89,71%	13	89,53%	16
SEA	88,23%	2	89,12%	3	89,37%	4	89,24%	4	89,22%	5
WavGen	85,44%	10	86,05%	16	87,64%	23	87,57%	31	87,26%	37
Covertype	81,81%	20	80,15%	31	79,52%	40	80,10%	50	79,73%	54
Abalone	19,83%	1	19,83%	1	19,83%	1	20,90%	2	20,90%	2
Connect	72,07%	13	72,87%	15	72,25%	18	72,53%	20	72,49%	20

Table 4: Dependence of the performance on the value of Ψ

	AgrGen		HypGen		LED		RDT		rRBF	
Ψ	Accuracy	Trees count	Accuracy	Trees count	Accuracy	Trees count	Accuracy	Trees count	Accuracy	Trees count
0	**97,22%**	**6**	**88,28%**	**24**	**71,07%**	**7**	**90,71%**	**11**	**89,53%**	**16**
0,1	**91,73%**	**1**	**85,13%**	**4**	**63,69%**	**2**	**87,64%**	**2**	**87,13%**	**2**
0,15	91,73%	1	82,80%	2	63,69%	2	87,64%	2	87,13%	2
0,2	91,73%	1	82,80%	2	67,72%	2	87,64%	2	87,13%	2
0,25	91,73%	1	82,80%	2	56,95%	2	85,42%	2	78,99%	1
0,3	91,73%	1	81,61%	2	45,34%	1	75,99%	1	78,99%	1
0,35	91,73%	1	70,52%	1	45,34%	1	75,99%	1	78,99%	1
0,4	91,73%	1	70,52%	1	45,34%	1	75,99%	1	78,99%	1
0,45	91,73%	1	70,52%	1	45,34%	1	75,99%	1	78,99%	1
0,5	91,73%	1	70,52%	1	45,34%	1	75,99%	1	78,99%	1
	SEA		WavGen		Covertype		Abalone		Connect	
Ψ	Accuracy	Trees count	Accuracy	Trees count	Accuracy	Trees count	Accuracy	Trees count	Accuracy	Trees count
0	**89,22%**	**5**	**87,26%**	**37**	**79,73%**	**54**	**20,90%**	**2**	**72,49%**	**20**
0,1	**88,23%**	**2**	**83,01%**	**5**	**77,87%**	**14**	**19,83%**	**1**	**71,17%**	**7**
0,15	88,23%	2	81,00%	4	75,03%	11	19,83%	1	71,10%	5
0,2	88,23%	2	78,48%	3	73,60%	7	19,83%	1	69,65%	4
0,25	78,20%	1	76,33%	2	73,60%	7	19,83%	1	70,11%	3
0,3	78,20%	1	76,33%	2	70,51%	7	19,83%	1	70,11%	3
0,35	78,20%	1	76,33%	2	68,45%	5	19,83%	1	67,35%	3
0,4	78,20%	1	73,22%	2	68,45%	5	19,83%	1	65,71%	2
0,45	78,20%	1	61,24%	1	67,31%	5	19,83%	1	65,71%	2
0,5	78,20%	1	61,24%	1	67,31%	4	19,83%	1	65,71%	2

[4] P. Berkhin, A survey of clustering data mining techniques, In Jacob C. N. Kogan, M. Teboulle, editors, Grouping Multidimensional Data, Springer Berlin Heidelberg, (2006) 25-71
[5] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, MOA: Massive Online Analysis; Journal of Machine Learning Research 11 (2010) 1601-1604
[6] A. Bifet, R. Kirkby, Data Stream Mining A Practical Approach, Center of the Open Source Innovation, 2009
[7] D. Brzezinski and J. Stefanowski, Reacting to different types of concept drift: The accuracy updated ensemble algorithm, IEEE Transactions on Neural Networks and Learning Systems, 25(1) (2014) 81-94
[8] D. Brzezinski, J. Stefanowski, Combining Block-based and Online Methods in Learning Ensembles from Concept Drifting Data Streams, Information Sciences, 265 (2014) 50-67
[9] A.K. Chandanan and M.K. Shukla, Removal of duplicate rules for association rule mining from multilevel dataset, Procedia Computer Science, International Conference on Advanced Computing Technologies and Applications (ICACTA), 45(0) (2015) 143-149
[10] V. Chandola, V. Kumar, Summarization - compressing data into an informative representation, Knowledge and Information Systems, 12(3) (2007) 355-378
[11] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv., 41(3) (2009) 15:1-15:58
[12] G. Ditzler, M. Roveri, C. Alippi, R. Polikar, *Learning in Nonstationary Environments: A Survey*, Computational Intelligence Magazine, IEEE, 10(4) (2015) 12-25
[13] P. Domingos, G. Hulten, Mining high-speed data streams, Proc. 6th ACM SIGKDD Internat. Conf. on Knowledge Discovery and Data Mining, (2000) 71-80
[14] R. Durrett, Probability: theory and examples (4th ed.), Cambridge University Press, (2004)
[15] W. Fan, F. Chu, H. Wang, P.S. Yu, Pruning and dynamic scheduling of cost-sensitive ensembles, In Eighteenth National Conference on Artificial Intelligence, Menlo Park, CA, USA, American Association for Artificial Intelligence, (2002) 146-151
[16] J. Gama, A survey on learning from data strems: current and future trends, Progress in Artificial Intelligence, 1(1) (2012) 45-55
[17] M. M. Gaber, S. Krishnaswamy, A. Zaslavsky, On-board Mining of Data Streams in Sensor Networks, Accepted as a chapter in the forthcoming book Advanced Methods of Knowledge Discovery from Complex Data, (Eds.) Sanghamitra Badhyopadhyay, Ujjwal Maulik, Lawrence Holder and Diane Cook, Springer Verlag, (2005)
[18] V. Grossi, F. Turini, Stream mining: a novel architecture for ensemble-based classification, Knowledge and Information Systems, 30(2) (2012) 247-281
[19] T.C. Havens, J.C. Bezdek, C. Leckie, L.O. Hall, M. Palaniswami, Fuzzy c-means algorithms for very large data. Fuzzy Systems, IEEE Transactions on, 20(6) (2012) 1130-1146
[20] D. Hernndez-Lobato, G. Martnez-Muoz, A. Surez, How large should ensembles of classifiers be? Pattern Recognition, 46(5) (2013) 1323-1336
[21] G. Hulten, L. Spencer, and P. Domingos, Mining time-changing data streams, in Proc. 7th ACM SIGKDD Internat. Conf. on Knowledge Discovery and Data Mining, (2001) 97106
[22] K. Jackowski, Fixed-size ensemble classifier system evolutionarily adapted to a recurring context with an unlimited pool of classifiers. Pattern Analysis and Applications, 17(4) (2014) 709-724
[23] G. Kesavaraj, S. Sukumaran, A study on classification techniques in data mining, In Computing, Communications and Networking Technologies (ICCCNT), 2013 Fourth International Conference on, (2013) 1-7
[24] A. Kale, M. D. Ingle, *SVM based Feature Extraction for Novel Class Detection from Streaming Data*, International Journal of Computer Applications, 110(9) (2015) 1-3
[25] L.I. Kuncheva, W.J. Faithfull, PCA feature extraction for change detection in multidimensional unlabelled data, IEEE Transactions on Neural Networks and Learning Systems, 25(1) (2014) 69-80
[26] Y.-N. Law, C. Zaniolo, An Adaptive Nearest Neighbor Classification Algorithm for Data Streams, Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD'05, Springer-Verlag, Berlin, Heidelberg, (2005) 108-120
[27] M. Last, Online Classification of Nonstationary Data Streams, Intelligent Data Analysis, 6(2) (2002) 129-147
[28] P. Li, X. Wu, X. Hu, H. Wang, Learning concept-drifting data streams with random ensemble decision trees, Neurocomputing, 166 (2015) 68-83
[29] X. Li, W. Yu, Data Stream Classification for Structural Health Monitoring via On-Line Support Vector Machines, Big Data Computing Service and Applications (BigDataService), 2015 IEEE First International Conference on, (2015) 400-405
[30] M. Lichman, UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, http://archive.ics.uci.edu/ml/, 2013 (accessed 14.10.2015)
[31] D. Mariammal, S. Jayanthi, and P. Patra, Classification methods in data mining: a detailed survey, IJRCCT, 3(4) 2014
[32] L. L. Minku, X. Yao, DDD: A New Ensemble Approach For Dealing With Concept Drift, IEEE Transactions on Knowledge and Data Engineering, 24(4) (2012) 619-633

[33] S. Muthukrishnan, Data Streams: Algorithms and Applications, 2005
[34] S. Rabbani, Proof that the Difference of Two Correlated Normal Random Variables is Normal, available at srabbani.com/bivariate.pdf
[35] M. Rathi, Regression modeling technique on data mining for prediction of crm. In VinuV Das and R. Vijaykumar, editors, Information and Communication Technologies, volume 101 of Communications in Computer and Information Science, Springer Berlin Heidelberg, (2010) 195200
[36] Y. Ren, L. Zhang and P. N. Suganthan, Ensemble Classification and Regression-Recent Developments, Applications and Future Directions [Review Article], in IEEE Computational Intelligence Magazine, 11(1) (2016) 41-53
[37] J. J. Rodriguez, L. I. Kuncheva and C. J. Alonso, Rotation Forest: A New Classifier Ensemble Method, IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(10) (2006) 1619-1630
[38] L. Rutkowski, M. Jaworski, L. Pietruczuk, P. Duda, A new method for data stream mining based on the misclassification error, IEEE Transaction on Neural Networks and Learning Systems, 26(5) (2015) 1048-1059
[39] L. Rutkowski, M. Jaworski, L. Pietruczuk, P. Duda, Decision Trees for Mining Data Streams Based on the Gaussian Approximation, IEEE Transactions on Knowledge and Data Engineering, 26(1) (2014) 108-119
[40] L. Rutkowski, M. Jaworski, L. Pietruczuk, P. Duda, The CART Decision Tree for Mining Data Streams, Information Sciences, 266 (2014) 1-15
[41] L. Rutkowski, L. Pietruczuk, P. Duda, M. Jaworski, Decision trees for mining data streams based on the McDiarmids bound, IEEE Transactions on Knowledge and Data Engineering, 25(6) (2013) 1272-1279
[42] N.V. Sawant, K. Shah, V.A. Bharadi, Survey on data mining classification techniques, In Proceedings of the International Conference 38; Workshop on Emerging Trends in Technology, ICWET 11, ACM, New York, NY, USA, (2011) 1380-1380
[43] W.N. Street, Y.S. 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, KDD 01, New York, NY, USA, (2001) 377382
[44] A. W. van der Vaart, Asymptotic statistics, New York: Cambridge University Press, 1998
[45] X.-Z. Wang, R. Aamir, A.-M. Fu, Fuzziness based sample categorization for classifier performance improvement, Journal of Intelligent & Fuzzy Systems, 29 (2015) 1185-1196
[46] X.-Z. Wang, H.-J. Xing, Y. Li, W. Pedrycz, A study on relationship between generalization abilities and fuzziness of base classifiers in ensemble learning, IEEE Transactions on Fuzzy Systems, 23(5) (2015) 1638-1654
[47] R. Wang, S. Kwon, X.-Z. Wang, Q.-S. Jiang, Segment based decision tree induction with continuous valued attributes, IEEE Transactions on Cybernetics, 45(7) (2015) 1262-1275
[48] K. Wankhade, S. Dongre, Data Streams Mining: Classification and Application, LAP Publication House, Germany, 2010
[49] J. Huang, J. Liu and X. Yao, A multi-agent evolutionary algorithm for software module clustering problems, Soft Computing, online on 4 February (2016)
[50] X.-C. Yin, K. Huang, H.-W. Hao, DE2: Dynamic ensemble of ensembles for learning nonstationary data, Neurocomputing, 165 (2015) 14-22
[51] I. Žliobaitė, A. Bifet, B. Pfahringer, G. Holmes, Active learning with drifting streaming data, IEEE Transactions on Neural Networks and Learning Systems, 25(1) (2014) 27-39