Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2016, Article ID 4218973, 9 pages
http://dx.doi.org/10.1155/2016/4218973

Research Article

Online Ensemble Using Adaptive Windowing for Data Streams with Concept Drift

Yange Sun,[1,2] Zhihai Wang,[1] Haiyang Liu,[1] Chao Du,[1] and Jidong Yuan[1]

[1] *School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China*
[2] *School of Computer and Information Technology, Xinyang Normal University, Xinyang 464000, China*

Correspondence should be addressed to Zhihai Wang; 13112074@bjtu.edu.cn

Received 8 November 2015; Revised 28 February 2016; Accepted 19 April 2016

Academic Editor: Antonino Staiano

Data streams, which can be considered as one of the primary sources of what is called big data, arrive continuously with high speed. The biggest challenge in data streams mining is to deal with concept drifts, during which ensemble methods are widely employed. The ensembles for handling concept drift can be categorized into two different approaches: online and block-based approaches. The primary disadvantage of the block-based ensembles lies in the difficulty of tuning the block size to provide a tradeoff between fast reactions to drifts. Motivated by this challenge, we put forward an online ensemble paradigm, which aims to combine the best elements of block-based weighting and online processing. The algorithm uses the adaptive windowing as a change detector. Once a change is detected, a new classifier is built replacing the worst one in the ensemble. By experimental evaluations on both synthetic and real-world datasets, our method performs significantly better than other ensemble approaches.

1. Introduction

In recent years, some promising computing paradigms have emerged to meet the needs of big data. The only thing that the parallel batch process model copes with is the stationary massive data. However, there are a lot of applications in practice, such as sensor networks [1], spam filtering [2], intrusion detection [3], and credit card fraud detection [4], which generate continuously arriving data, known as data streams [5]. Most big data can be regarded as data streams, in which data are produced continuously [6]. In fact, model in the data stream is coping with the problem of three features of big data: big volume, big velocity, and big variety.

In general, most of the existing solutions constructing stream data mining are under the hypothesis that data are stationary. However, in the real-world, the generation of data streams is usually in the nonstationary environment, which means that the underlying distribution of the data can change arbitrarily over time. This phenomenon is known as concept drift [7, 8], which exists commonly in the scenarios of big data mining. For example, weather prediction models change according to the seasons, and in recommend systems, user consumption patterns may change over time due to fashion, economy, and so forth. The occurrence of such change leads to a drastic drop in classification accuracy. Therefore, the learning models should be able to adapt to the changes quickly and accordingly.

According to their speed, concepts drifts have been divided into two types: sudden drifts and gradual drifts [7]. Sudden concept drift is characterized by large amounts of change between the underlying class distribution and the incoming instances in a relatively short amount of time, while gradual concept drift is featured by large amount of time to witness a significant change in differences between the underlying class distribution and the incoming instances. Most of the existing methods just deal with one of the two types. However, in the real-world, data stream probably contains more than one type of concept drift. Thus, being able to track and adapt to various kinds of concept drift instantly is highly expected from a better classifier.

Concept drift has become a popular research topic over the last decade and many algorithms have been developed [9, 10]. The methodologies proposed for tackling concept drifts can be organized into three main groups: window-based approaches, weight-based approaches, and ensemble classifiers [7]. Ensemble methods are widely used in concept drift learning. The techniques for using ensemble to handle concept drift fall into two categories: block-based ensembles and online ensembles [11].

For block-based ensembles [4, 11–14], the streams are segmented into a series of successive fixed-size blocks. Every time when a new block appears, a new classifier, which is learned from the block, will be added to the ensemble, and the weakest classifier will be eliminated in line with the result of the evaluation. Consequently, the component classifiers of ensemble will be evaluated and later updated. Such approach ensures accurate reactions to gradual concept drifts. The main drawback of block-based ensembles is their delay in reacting to the sudden concept drifts. Another disadvantage is the difficulty of defining an appropriate size of the block [4]. Online ensembles update component weights after each instance without the need for storage and reprocessing [15]. So this method can adapt to sudden changes as quickly as possible. However, some of these algorithms are usually characterized by higher computational costs compared with block-based methods.

In order to meet the above challenges, we have come up with a novel ensemble paradigm, called Adaptive Windowing based Online Ensemble (AWOE), which combines the best elements of block-based weighting and online processing. The main contributions can be summarized as follows.

(1) The proposed algorithm is designed to assign different size of block to each ensemble member using adaptive windowing as a change detector. Therefore, it can capture sudden drifts immediately.

(2) The proposed approach synthesizes the essential features of the two groups of ensembles to handle various types of concept drifts.

The performance of the proposed algorithms was evaluated on both synthetic and real-world datasets, and a comprehensive comparison study of online and block-based ensemble algorithms was presented. The results show that our method achieves better performance than previous methods, especially when concept drift occurs.

The remainder of this paper is organized as follows. Section 2 presents the related work. In Section 3, we describe the approach in detail. In Section 4, we evaluate the method on both artificial and real-world datasets. Finally, some conclusions are drawn and future researches are discussed in Section 5.

2. Related Work

In this section, some relevant concepts of this study are to be introduced first, and then some previous work will be summarized.

2.1. Basic Concepts and Notation

Definition 1. A data stream is an infinite sequence of training records:

$$S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_t, y_t), \ldots\}. \tag{1}$$

Each record is a pair (x_t, y_t), where x_t is a d-dimensional vector arriving at the time stamp t and y_t is the class label of x_t.

Definition 2. One considers that the term concept refers to the whole distribution of the problem in a certain point in time, being characterized by the joint probability $P(x_t, y_t)$.

Definition 3. Concept drift, that is, the underlying distribution of the data, is evolving over time [7]. It can be formally defined as any scenario where the posterior probability changes over time; that is, $P_t(x_i, y_i) \neq P_{t+1}(x_i, y_i)$ [10].

Definition 4. A change detector is an algorithm that takes a stream of instances as input and outputs an alarm if it detects a change in the distribution of the data.

2.2. Ensemble Classifiers for Data Streams with Concept Drift.

Block-based approaches have been designed to work in the environments where instances arrive in portions, called chunks or blocks. Most block-based ensembles periodically evaluate their components and substitute the weakest ensemble member with a new (candidate) classifier after each block of instances. Such an approach ensures accurate reactions to gradual concept drifts.

The first of such block-based ensembles was the Streaming Ensemble Algorithm (SEA) [12], which used a heuristic replacement strategy based on accuracy and diversity. Accuracy Weighted Ensemble (AWE) is a generic framework for dealing with concept drifts in data streams [4]. The idea is to train a group of classifiers from sequential blocks of the data streams. Each classifier is weighted and only the top K-classifiers are kept. And the final output is based on the decision made by the weighted votes of the classifiers. The Accuracy Updated Ensemble (AUE1) [13], which incrementally trains its component classifiers after every processed block of instances. Results obtained by AUE2 [11] suggested that by incremental learning of periodically weighted ensemble members one could preserve good reactions to gradual changes, while reducing the block size problem and, therefore, improving accuracy on suddenly changing streams.

It is significant to notice that the performance of the block-based ensembles primarily depends on the size of the blocks. A small block does not supply adequate data for building a new classifier, while a too large block may include data coming from various concepts, causing delay of the adaptation to new concepts.

Oza and Russell [16] developed online versions of bagging and boosting for data streams. They show how the process of sampling bootstrap replicates from training data can be simulated in a data stream context. They observe that the probability that any individual instance will be chosen for a replicate tends to a Poisson (1) distribution. Kolter and Maloof [17] proposed an algorithm called Dynamic Weighted Majority (DWM), which is one of the most cited online learning approaches to handle drifts. In DWM, weighted experts are dynamically created and removed according to their accuracy after each incoming instance. Bifet et al. [18] introduced an algorithm named Leveraging Bagging

```
Input: data stream $S$, confidence $\delta \in (0,1)$;
Output: ChangeAlarm;
(01) Initialize Window $W$;
(02)     for each $t > 0$ do
(03)         $W \leftarrow W \cup \{x_t\}$ (i.e., add $x_t$ to the head of $W$);
(04)         repeat
(05)             Drop elements from the tail of $W_L$;
(06)     until KL($W_L \parallel W_R$) < $\varepsilon$; (calculate $\varepsilon$ according to (4));
(07)     end for
(08)     Output ChangeAlarme;
(09) end
```

Algorithm 1: Pseudocode of adaptive windowing change detector.

(Lev), which intends to add more randomization to the base classifiers.

In terms of the sudden drifts, the online ensembles can respond faster with both of their components evolving over time. However, online ensembles do not take advantage of periodical component evaluations and do not weight or introduce new components periodically. As a result, on data streams with gradual changes, online ensembles are often less accurate than block-based approaches.

To address the above problems, a hybrid ensemble, which combines the strength of the above two, was proposed in this study.

3. Our Algorithm

In this section, an adaptive windowing change detector based on entropy will be introduced first, and then an online ensemble with internal change detector is demonstrated in detail. The complexity of the algorithm will be analyzed lastly.

3.1. Adaptive Windowing Change Detector Based on Entropy. This study proposed a two-window paradigm for change detection, which is inspired by Adaptive Window (ADWIN) [19]. The ADWIN algorithm increases the window size until two subwindows are found that are "distinct enough." Distinct enough means the average of the two subwindows is larger than a threshold defined by the Hoeffding bound [20]. The window will be dynamically magnified when no obvious change is detected and will be compressed when a change occurs.

Theorem 5 (Hoeffding bound). *The Hoeffding bound is stated as follows: with probability $1-\delta$, the estimated mean after n independent observations of range R will not differ from the true mean by more than ε, where*

$$\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}, \tag{2}$$

where $\delta \in (0,1)$ is a user-defined confidence parameter.

Theorem 6. *Let W_0 and W_1 denote the two subwindows (where W_1 contains the most recent instances). With probability $1-\delta$, one has $|\mu_{W_0} - \mu_{W_1}| \le 2\varepsilon$, where ε is the Hoeffding bound, μ_{W_0} and μ_{W_1} are the mean of two subwindows.*

Proof. Assume the true mean of W is μ. According to the Hoeffding bound, $|\mu_{W_0} - \mu| \le \varepsilon$ and $|\mu_{W_1} - \mu| \le \varepsilon$ could be obtained separately. Then, they can be transformed into $-\varepsilon \le |\mu_{W_0} - \mu| \le \varepsilon$ and $-\varepsilon \le |\mu - \mu_{W_1}| \le \varepsilon$. By summing these two inequalities,

$$\left|\mu_{W_0} - \mu_{W_1}\right| \le 2\varepsilon. \tag{3}$$

According to Theorem 5, (3) can be converted into

$$\left|\mu_{W_0} - \mu_{W_1}\right| \le \sqrt{\frac{2R^2 \ln(1/\delta)}{n}}. \tag{4}$$

Since the entropy can be viewed as an average value. This study adopts the relative entropy (Kullback-Leibler distance) [21] as a measure to compare the difference between two subwindows with the Hoeffding bound to determine if the target concept is drifted. Different from ADWIN, the change detector was used to obtain the entropy from window dynamically. The sliding window W was partitioned into two equal length subwindows: a left subwindow W_L and a right subwindow W_R. The Kullback-Leibler distance from W_L to W_R is defined as

$$\mathrm{KL}(W_L \parallel W_R) = \sum_{x \in X} p_{W_L}(x) \log \frac{p_{W_L}(x)}{p_{W_R}(x)}, \tag{5}$$

where the sum is taken (in the discrete setting) over the atoms of the space of events X. When the distance is greater than the threshold calculated according to (4), a change is detected. Then, the older portion of the window, W_L, is dropped. The full pseudocode of the entropy-based change detector is listed in Algorithm 1. □

3.2. Online Ensemble Using Adaptive Windowing. The primary disadvantage of the block-based ensembles lies in their delay in responding to the sudden concepts drifts, and this resulted from analyzing real labels only after every full block of instances. Another disadvantage is the difficulty of tuning the block size to offer a compromise between fast reactions to concept drifts and high accuracy in periods of concept stability.

In order to solve the above problems, an online ensemble with internal change detector was proposed, which retains

```
Input: S: data stream, $C_0$: online learner, D: adaptive windowing change detector,
        buffer of size d; k: number of ensemble members, B: long-term
Output: E: ensemble of k weighted classifiers;
(01) for all instances $x_t \in S$ do
(02)    incrementally train $C_0$ and D with $x_t$;
(03)    $B \leftarrow B \cup \{x_t\}$;
(04)    if $|B| = d$ or change detected then
(05)     build and weight new classifier $C'$ using B;
(06)         weight all classifiers $C_i$ in ensemble;
(07)         if $|E| < k$ then $E \leftarrow E \cup \{C'\}$;
(08)         else replace the weakest ensemble member with $C'$;
(09)         reinitialize $C_0$ with B;
(10)        reinitialize D;
(11)        $B \leftarrow \emptyset$;
(12)    end if
(13) end for
```

Algorithm 2: Pseudocode of AWOE algorithm.

a pool of weighted classifiers by obtaining the final output of components based on the weighted majority voting rule. The sliding window is chosen to monitor the classification error of the most recent data. Furthermore, a long-term buffer mechanism is selected to store the recent training instances, on which a new classifier is built when a change is detected. In this way, it can assign different size of block to each ensemble member.

Furthermore, the addition of an online learner and drift detector offers quicker reactions to sudden concept changes compared to most block-based ensembles. The online learner, which is incrementally trained with each incoming instance, is taken into account during component voting. Such strategy ensures that the most recent data is included in the final prediction. In the following experiments, we adopt an incremental algorithm for constructing decision trees, which is called Hoeffding Tree [20]. It builds a decision tree from data streams incrementally, without storing instances after they have been employed to renew the tree. The proposed Adaptive Window algorithm was selected as a change detector by monitoring the classification error. We consider a correct prediction to be 1 and an incorrect one to be 0. The full pseudocode of AWOE is listed in Algorithm 2.

Let S be a data stream; E represents the ensemble. When an instance arrives, online classifier is incrementally trained with internal change detector D. Instead of evaluating component classifiers after each block of instances, the ensemble members $C_i \in E$ are weighted after each incoming instance according to

$$w_{ij} = \frac{1}{\mathrm{MSE}_r + \mathrm{MSE}_i + \varepsilon},$$
$$\mathrm{MSE}_r = \sum_y p(y)\left(1 - p(y)\right)^2, \tag{6}$$
$$\mathrm{MSE}_{ij} = \frac{1}{|B|}\sum_{(x,y)\in B}\left(1 - f_y^i(x)\right)^2,$$

where MSE_{ij} represents the prediction error of C_i on long-term buffer B, while MSE_r represents the mean square error of a randomly predicting classifier and is used as a reference point to the current class distribution. Additionally, a very small positive value ε is added to avoid division by zero problems. Function $f_y^i(x)$ denotes the probability given by classifier C_i that x is an instance of class y. When a concept drift is detected (or the number of instances in the long-term buffer exceeds the maximum), a candidate classifier is built on the instances in B, weighted, and added to the ensemble. If the ensemble is full, the weakest classifier is replaced by new one based on the result of the evaluation.

3.3. Complexity. It is important to analyze the time and space complexity of the algorithms. At this point, now the AWOE algorithm has been described and a detailed analysis of this complexity is presented. It should be noted that the ensemble algorithm can be configured with different base classifiers, so the final details about complexity will depend on the final base classifier used. In our experiments, the Hoeffding Tree [20] was chosen as the base classifier, but one could use any online learning algorithm as a base learner.

Temporal Complexity. Therefore, the analysis can be done according to two situations: building new base classifiers or weighting them. As the Hoeffding Tree is learned in constant time per instance [20], the training of an ensemble of k Hoeffding Trees has a complexity of $O(k)$. Additionally, the weighting procedure requires a constant number of operations; thus, for weighting k components $O(1)$ time is needed. Therefore, in the worst-case scenario, the training and weighting of AWOE have a complexity of $O(2k)$ per instance since k is a user-defined constant.

Spatial Complexity. It is basically determined by the maximum number of base classifiers stored in the ensemble (max) and their maximum size. The memory requirements of an ensemble of Hoeffding Trees depend on the concept being learned and can be denoted as $O(kavcl)$, where a represents the number of attributes, v is the maximum number of values per attribute, c is the number of classes, and l is the number of leaves in the tree. We adopt a variation of exponential histograms [22] as the data structure, which maintains an approximation of the number of 1's in a window of length W with logarithmic memory and update time. In this way, the change detector consumes $O(\log W)$ memory. The total spatial complexity is $O(kavcl + \log(W))$.

4. Experimental Results

In this section, we demonstrate all the used datasets, describe experimental setup, and discuss experiment results.

4.1. Datasets. The experiments are implemented in Java with the help of Massive Online Analysis (MOA) [23]. MOA is a software environment for implementing algorithms and running experiments for online learning. In our experiments, we adopt four synthetic and three real-world datasets.

4.1.1. Synthetic Datasets. Synthetic datasets have several advantages: they are easier to reproduce and bear low cost of storage and transmission, and, most importantly, synthetic datasets provide an advantage of knowing the ground truth. For instance, we can know where exactly concept drift happens, what the type of drift is, and the best classification accuracies achievable on each concept. The synthetic datasets contain three types of concept drift: sudden, gradual, and mixture.

HyperPlane is a two-class dataset that models a rotating hyperplane in a d-dimensional space. It is represented by the set of points x that satisfy $\sum_{i=1}^{d} w_i x_i = w_0$, where x_i is the ith coordinate of x. Instances for which $\sum_{i=1}^{d} w_i x_i \geq w_0$ are labeled positive, and instances for which $\sum_{i=1}^{d} w_i x_i < w_0$ are labeled negative. This generator was used to create a dataset containing 1,000,000 instances with gradual drifts by the modification weight w_i changing by 0.001 with each instance and added 5% noise to streams.

The SEA dataset was first described in [12]. It consists of three attributes, where only two are relevant. All the attributes have values between 0 and 10. The points of the dataset are divided into four blocks with different concepts. In each block, the classification is done using $f_1 + f_2 \leq \theta$, where f_1 and f_2 represent the first two attributes and θ is a threshold value. The threshold values are 9, 8, 7, and 9.5. We generated a dataset containing 1,000,000 instances with sudden drifts occurring every 250,000 instances and having 10% of class noise.

The goal of LED dataset is to predict the digit displayed on a seven-segment LED display. The particular configuration of the generator used for the experiment produces 24 binary attributes, 17 of which are irrelevant. Concept drift is simulated by interchanging relevant attributes. We generated a stream of 1,000,000 instances with sudden and gradual concept drifts and 10% of noise.

Waveform is composed of a stream with three decision classes, in which the instances are depicted by 40 attributes. The aim of the task is to distinguish between three diverse classes of waveform, and each of them is produced by a synthesis of two or three base waves. We produce a stream consisting of 1,000,000 instances with no drift. It has been applied before, such as in [15].

4.1.2. Real-World Datasets. When working with real-world datasets, it is not possible to know exactly when a drift starts to occur, which type of drift is present, or even if there really is a drift. Therefore, it is not possible to perform a detailed analysis of the behavior of algorithms in the presence of concept drift using only pure real-world datasets. The real-world datasets employed in the experiments can be obtained at http://moa.cms.waikato.ac.nz/datasets/, and they can be simulated into data streams by the MOA generators.

The Covertype dataset comes from UCI archive [24] including the forest cover type for cells of 30 × 30 meters procured from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. It contains 581,012 instances, which are defined by 53 cartographic variables that depict one of seven possible forest cover types. The aim is to predict the forest cover type based on cartographic variables. It has been used in [16, 25].

The Poker Hand dataset represents the problem of identifying the hand in a Poker game. It consists of 1,000,000 instances representing all possible poker hands. Each instance represents a hand comprising five cards drawn from a standard deck of 52. Each card is described using two attributes (suit and rank), for a total of 10 predictive attributes. There is one class attribute that describes the "Poker Hand."

The Electricity dataset which consists of 45,312 instances, each described by 7 attributes, presents the problem of predicting whether the price in the Australian New South Wales Electricity Market will increase or decrease. The dataset is a collection of successive measurements at every 30 minutes, spanning the period from May 1996 to December 1998. The class label of each point is either UP or DOWN, referring to whether the electricity price at the specified time is higher or lower than the average price of the preceding 24 hours. It has been used in [17, 25, 26].

4.2. Experimental Setup. To evaluate the effectiveness of the methods, we use the prequential evaluation method [27]. This way the classifier is tested against all instances before seeing them. All the algorithms were implemented in Java as part of the MOA framework. The experiments were performed on 3.0 GHz Pentium PC machines with 8 GB of memory, running Microsoft Windows 7.

All the tested ensembles used $k = 10$ component classifiers. The Hoeffding Tree was selected as the base classifier. It set default parameters: grace period $n_{\min} = 100$, tie-threshold $\tau = 0.05$, and split confidence $\delta = 0.01$.

4.3. Results and Discussion

4.3.1. Drift Detection. The proposed entropy-based change detection was compared against the following change detections: Drift Detection Method (DDM) [25], Early Drift Detection Method (EDDM) [26], and Adaptive Window (ADWIN) [19] on the performance measures such as the false positive rate and false negative rate.

Table 1: The false positive rate of change detections.

	DDM	EDDM	ADWIN	Our method
HyperPlane	0.2572	**0.1027**	0.1134	0.2044
SEA	**0.0406**	0.1127	0.3940	0.1004
LED	0.4177	0.4133	0.3062	**0.2059**

Table 2: The false negative rate of change detections.

	DDM	EDDM	ADWIN	Our method
HyperPlane	0.0304	**0.0020**	0.1031	0.0973
SEA	0.0093	0.0176	0.0342	**0.0045**
LED	0.3211	0.2274	0.2154	**0.1178**

False Positive Rate. The false positive rate is the probability of falsely rejecting the null hypothesis for a given test.

False Negative Rate. The false negative rate is the probability of falsely accepting the null hypothesis when it is in fact true.

All of the change detections have freely available implementations in the MOA framework. The results are shown in Tables 1 and 2. The lower values indicated a better performance. It is clearly revealed that DDM is the method with the best performance on the dataset with sudden changes (SEA). However, its detection speed is very slow. EDDM is more suitable for detecting gradual changes, while most misdetection appeared under static environments because of their sensitivity to errors and noise. The relatively higher false positive rate for ADWIN is due to the use of compression to reduce storage size of its buffer. Our method achieves better false positive rates than ADWIN in the presence of the dataset with mixture concept drift (LED). The results showed that our method ensures certain superiority over others comparing change detections especially on datasets containing different types of drifts.

4.3.2. Comparative Performance Study. The AWOE was evaluated against the following methods: Accuracy Weighted Ensemble (AWE), Accuracy Updated Ensemble (AUE2), Dynamic Weighted Majority (DWM), Lev Bagging (Lev), and Online Accuracy Updated Ensemble (OAUE). They all have freely available implementations in the MOA framework, except for DWM, which was implemented and is available at (http://sites.google.com/site/moaextensions/) as MOA extensions.

The performance can be evaluated in terms of accuracy, time, and memory in Tables 3–5 (the best results for each dataset are indicated in bold).

Classification Accuracy. As Table 3 shows, in terms of accuracy, Lev and our method outperform all the other algorithms. On the dataset with no drift (Waveform), Lev, AWE, and DWM performed almost identically, with OAUE being slightly less accurate. For the dataset with gradual concept drift (HyperPlane), AWE is the best, followed by AUE. However, our method seems to be the most accurate in the case of sudden changes (SEA). This is partly because the addition of drift detector offers quicker reactions to sudden

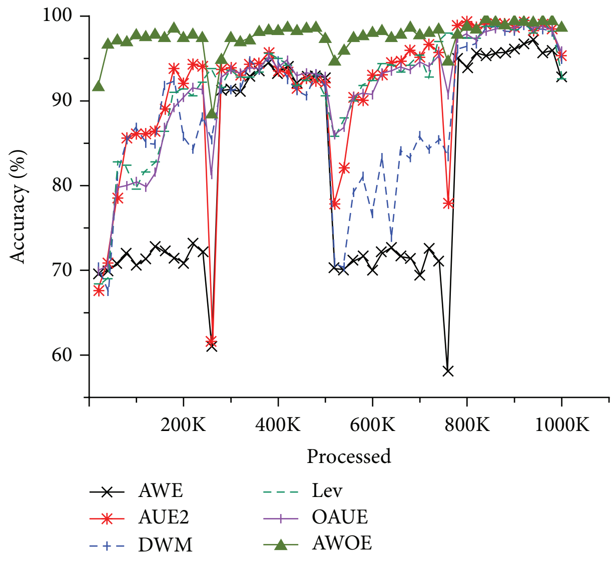

Figure 1: Accuracy on the SEA dataset.

concept changes compared to most block-based ensembles. For the dataset with mixed concept drift (LED), our proposed method largely outperformed other algorithms. On the real-world datasets, in terms of accuracy, there is no single best performing algorithm. On the Covertype, our method clearly outperformed all the other algorithms. On the Poker, OAUE is the most accurate followed by Lev, while on the Electricity all the algorithms perform almost identically.

Time Analysis. In terms of the running time, as shown in Table 4, through the comparative analysis, we found that DWM consumed the least, followed by our algorithm, and Lev is the longest time-consuming. Although Lev achieves the highest classification accuracy rate, it consumes more time. We have observed that the online ensemble is the best strategy in terms of accuracy, but it also had a poor performance in terms of the processing time.

Memory Usage. According to Table 5, in most cases, AUE2 achieved minimal memory consumption, followed by our algorithm, while the Lev consumed the most memory. It is clear that the memory usage of the AUE2 is lower than others because of the pruning strategy. This is partly because our algorithm not only uses an adaptive sliding window algorithm based on classification error rate to track changes in data streams but also just stores classification error rate instead of all the instances so that it consumes less memory compared with other algorithms.

In conclusion, the results proved that the proposed algorithm achieves better performance with regard to accuracy and costs less time and memory. The Lev enjoys the slight advantage over other algorithms in terms of accuracy. Unfortunately, it is also the costliest strategy in terms of processing time, as it requires estimating each component's predictive performance after each instance.

Figure 1 shows the classification accuracy on the SEA dataset, which was designed to evaluate the ability to handle

Table 3: Classification accuracies of different algorithms (%).

	AWE	AUE2	DWM	Lev	OAUE	AWOE
HyperPlane	**91.67 (1)**	89.12 (2)	81.36 (5)	80.21 (6)	84.25 (4)	86.04 (3)
SEA	79.59 (6)	80.81 (5)	87.10 (3)	88.93 (2)	86.85 (4)	**89.12 (1)**
LED	52.29 (6)	53.41 (4)	53.27 (5)	55.89 (2)	53.47 (3)	**57.78 (1)**
Waveform	83.32 (3)	82.46 (4)	82. 53 (2)	**83.64 (1)**	82.17 (6)	82.25 (5)
Covertype	78.74 (6)	88.14 (4)	85.52 (5)	90.37 (2)	89.54 (3)	**95.26 (1)**
Poker	62.22 (6)	71.06 (5)	75.51 (4)	80.20 (2)	**86.74 (1)**	80.45 (3)
Electricity	70.84 (6)	77.34 (5)	90.10 (2)	**91.02 (1)**	87.71 (4)	88.96 (3)
Average rank	4.86	4.14	3.71	2.29	3.57	2.43

Table 4: Times of different algorithms (seconds).

	AWE	AUE2	DWM	Lev	OAUE	AWOE
HyperPlane	33.34 (3)	48.11 (5)	25.33 (2)	765.03 (6)	45.54 (4)	**22.90 (1)**
SEA	**10.75 (1)**	11.64 (2)	52.23 (5)	82.29 (6)	14.60 (4)	12.62 (3)
LED	53.54 (6)	43.35 (5)	**12.11 (1)**	31.34 (2)	36.01 (4)	33.23 (3)
Waveform	6.79 (3)	7.45 (4)	**4.31 (1)**	14.56 (5)	15.47 (6)	5.84 (2)
Covertype	338.94 (5)	130.42 (2)	140.24 (3)	884.41 (6)	**106.89 (1)**	221.80 (4)
Poker	147.81 (4)	58.69 (2)	156.41 (5)	1247.79 (6)	**51.79 (1)**	68.92 (3)
Electricity	14.94 (4)	10.03 (3)	**7.14 (1)**	28.89 (6)	8.22 (2)	18.94 (5)
Average rank	3.71	3.29	2.57	5.29	3.14	3.00

Table 5: Memory usage of different algorithms (MB).

	AWE	AUE2	DWM	Lev	OAUE	AWOE
HyperPlane	1.22 (2)	1.30 (3)	**0.16 (1)**	5.91 (6)	2.87 (4)	3.03 (5)
SEA	0.71 (2)	1.76 (4)	**0.07 (1)**	67.30 (6)	6.83 (5)	1.14 (3)
LED	0.61 (4)	0.22 (2)	**0.04 (1)**	1.76 (6)	0.62 (5)	0.23 (3)
Waveform	**5.05 (1)**	6.49 (3)	6.78 (4)	480.29 (6)	50.71 (5)	6.37 (2)
Covertype	3.12 (3)	1.05 (2)	8.27 (6)	6.75 (5)	3.09 (4)	**0.60 (1)**
Poker	0.27 (3)	**0.20 (1)**	0.31 (4)	1.23 (5)	3.46 (6)	0.26 (2)
Electricity	0.91 (5)	**0.24 (1)**	0.30 (2)	0.46 (3)	1.54 (6)	0.63 (4)
Average rank	2.86	2.29	2.71	5.29	5.00	2.85

sudden concept drifts. Whenever a concept drift occurred, the accurate rates of all the algorithms will undergo instantaneous fluctuations except for our algorithm, which maintains a high, stable accuracy and suffered the smallest accuracy drops. This might be attributed to the addition of drift detector which could capture sudden concept drifts promptly, according to changes in the concept and in a timely manner to build a classifier to deal with this type of drift.

Figure 2 presents the classification accuracy on the LED dataset, which intended to verify the algorithms' response to mixed drifts. This dataset included a complex change by combining two gradually drifting streams. After 500K instances, the target concept was instantly switched from a concept to another. We observed that all the algorithms maintain a high and stable accuracy when the data was relatively stable. When the concept drift occurred at 500K, accuracy of all the algorithms declined sharply, except in our method. Since our method can track the various kinds of changes immediately, it reestablished a new classifier in real-time. Since the dataset contained 10% of the noise, it illustrated that the proposed method was more suitable for the noise environment.

Real-world stream environment conceptual changes have unpredictability and uncertainty which can better verify the performance of the algorithm. Figure 3 depicts the accuracy changes on the Covertype. We observed the accuracy curves of all algorithms with varying degrees of volatility, which indicates that concept drift may exist in the dataset. Our method is the most accurate one, followed by the OAUE. The accuracy curve of the proposed algorithm is relatively stable, as it is robust to concept drift, which also shows that our algorithm has better adaptability for real environment.

In conclusion, our approach has better performance than other ensembles in the following three aspects: (1) it better resolves the problem of setting an appropriate size of block; (2) it is more suitable for the scenarios with different types of drift; and (3) our algorithm is more efficient than other

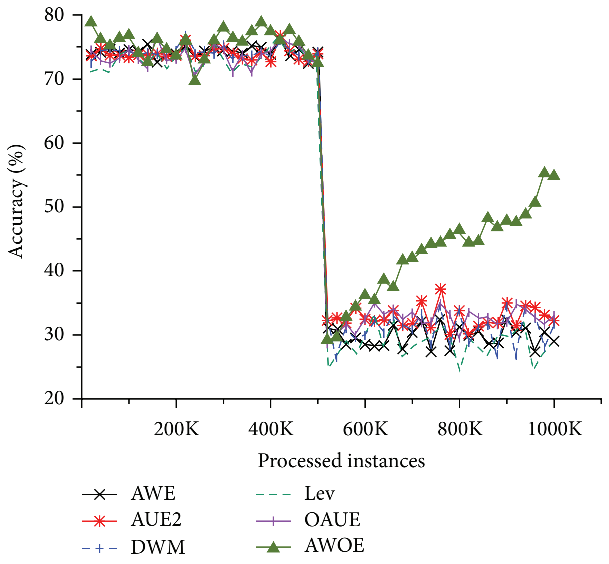

Figure 2: Accuracy on the LED dataset.

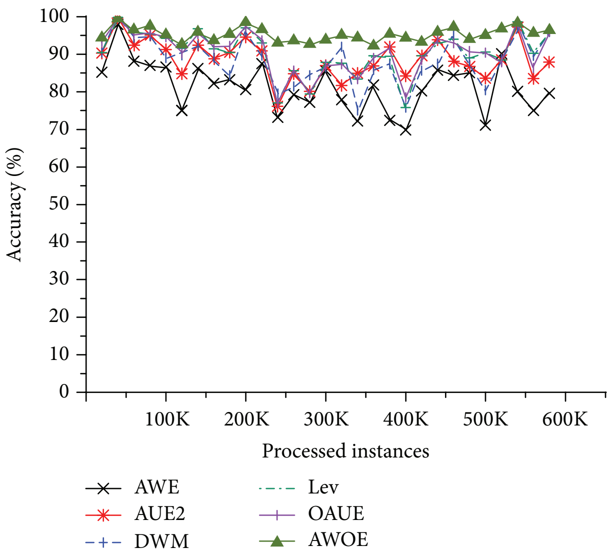

Figure 3: Accuracy on the Covertype dataset.

ensemble approaches in terms of accuracy and memory consumption.

5. Conclusion and Future Work

This study, through studying the influence of the size of data block on performance of the ensemble classifier, proposed an online ensemble with internal change detector to capture concept drifts in timely manner by determining block size dynamically. The experimental results prove that our approach performs better than other ensembles and gains the best tradeoff between accuracy and resources.

Most existing data stream algorithms assume that true labels are immediately and entirely available. Unfortunately, such assumption is often violated in real-world applications because it is expensive to obtain all true labels. As the future work, we intend to investigate the potentiality of adapting the proposed algorithm to the streams with unlabeled data.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (no. 61572417, no. 61563001, and no. 61572005), the Natural Science Foundation of Beijing (no. 4142042), and the Fundamental Research Funds for the Central Universities (no. 2015YJS049).

References

[1] L. Cohen, G. Avrahami-Bakish, M. Last, A. Kandel, and O. Kipersztok, "Real-time data mining of non-stationary data streams from sensor networks," *Information Fusion*, vol. 9, no. 3, pp. 344–353, 2008.

[2] S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle, "A case-based technique for tracking concept drift in spam filtering," *Knowledge-Based Systems*, vol. 18, no. 4-5, pp. 187–195, 2005.

[3] T. Lane and C. E. Brodley, "Approaches to online learning and concept drift for user identification in computer security," in *Proceedings of the 4th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '98)*, pp. 259–263, AAAI Press, Menlo Park, Calif, USA, 1998.

[4] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*, pp. 226–235, San Francisco, Calif, USA, August 2003.

[5] C. C. Aggarwal, *Data Streams: Models and Algorithms*, Springer, Berlin, Germany, 2007.

[6] A. Bifet, G. de Francisci Morales, J. Read, G. Holmes, and B. Pfahringer, "Efficient online evaluation of big data stream classifiers," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*, pp. 59–68, ACM, Sydney, Australia, August 2015.

[7] A. Tsymbal, "The problem of concept drift: definitions and related work," Tech. Rep., Department of Computer Science, Trinity College, Dublin, Ireland, 2004.

[8] G. Widmer, "Learning in the presence of concept drift and hidden contexts," *Machine Learning*, vol. 23, no. 1, pp. 69–101, 1996.

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

[10] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, pp. 231–238, 2014.

[11] 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*, vol. 25, no. 1, pp. 81–94, 2014.

[12] W. N. Street and Y. S. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proceedings of the 7th*

ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '01), pp. 377–382, ACM Press, San Francisco, Calif, USA, August 2001.

[13] D. Brzeziński and J. Stefanowski, "Accuracy updated ensemble for data streams with concept drift," in *Hybrid Artificial Intelligent System: 6th International Conference, HAIS 2011, Wroclaw, Poland, May 23–25, 2011, Proceedings, Part II*, E. Corchado, M. Kurzynski, and M. Wozniak, Eds., vol. 6679 of *Lecture Notes in Computer Science*, pp. 155–163, Springer, Berlin, Germany, 2011.

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

[15] D. Brzezinski and J. Stefanowski, "Combining block-based and online methods in learning ensembles from concept drifting data streams," *Information Sciences*, vol. 265, pp. 50–67, 2014.

[16] N. C. Oza and S. Russell, "Experimental comparisons of online and batch versions of bagging and boosting," in *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '01)*, pp. 359–364, ACM Press, New York, NY, USA, August 2001.

[17] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: an ensemble method for drifting concepts," *Journal of Machine Learning Research*, vol. 8, pp. 2755–2790, 2007.

[18] A. Bifet, G. Holmes, and B. Pfahringer, "Leveraging bagging for evolving data streams," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2010, Barcelona, Spain, September 20–24, 2010, Proceedings, Part I*, vol. 6321 of *Lecture Notes in Computer Science*, pp. 135–150, Springer, Berlin, Germany, 2010.

[19] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 7th SIAM International Conference on Data Mining (SDM '07)*, C. Apte, D. Skillicorn, B. Liu, and S. Parthasarathy, Eds., pp. 443–448, SIAM, 2007.

[20] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '00)*, pp. 71–80, ACM Press, Boston, Mass, USA, August 2000.

[21] S. Kullback and R. A. Leibler, "On information and sufficiency," *Annals of Mathematical Statistics*, vol. 22, pp. 79–86, 1951.

[22] M. Datar, A. Gionis, P. Indyk, and R. Motwani, "Maintaining stream statistics over sliding windows," *SIAM Journal on Computing*, vol. 31, no. 6, pp. 1794–1813, 2002.

[23] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: massive online analysis," *Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010.

[24] M. Lichman, *UCI Machine Learning Repository*, University of California, School of Information and Computer Science, Irvine, Calif, USA, 2013, http://archive.ics.uci.edu/ml.

[25] J. Gama, P. Medas, G. Castillo et al., "Learning with drift detection," in *Advances in Artificial Intelligence—SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29-Ocotber 1, 2004. Proceedings*, vol. 3171 of *Lecture Notes in Computer Science*, pp. 286–295, Springer, Berlin, Germany, 2004.

[26] M. Baena-García, D. J. Campo-Ávila, R. Fidalgo et al., "Early drift detection method," in *Proceedings of the 4th International Workshop on Knowledge Discovery from Data Streams (KDD '06)*, pp. 77–86, ACM Press, 2006.

[27] J. Gama, R. Sebastião, and P. P. Rodrigues, "Issues in evaluation of stream learning algorithms," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*, pp. 329–337, July 2009.