Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

# An ensemble method for data stream classification in the presence of concept drift

Omid ABBASZADEH, Ali AMIRI[‡], Ali Reza KHANTEYMOORI

(*Department of Computer Engineering, University of Zanjan, Zanjan 45371-38791, Iran*)

E-mail: o.abbaszadeh@znu.ac.ir; a_amiri@znu.ac.ir; khanteymoori@znu.ac.ir

Received Nov. 19, 2014; Revision accepted Apr. 15, 2015; Crosschecked Nov. 11, 2015

**Abstract:** One recent area of interest in computer science is data stream management and processing. By 'data stream', we refer to continuous and rapidly generated packages of data. Specific features of data streams are immense volume, high production rate, limited data processing time, and data concept drift; these features differentiate the data stream from standard types of data. An issue for the data stream is classification of input data. A novel ensemble classifier is proposed in this paper. The classifier uses base classifiers of two weighting functions under different data input conditions. In addition, a new method is used to determine drift, which emphasizes the precision of the algorithm. Another characteristic of the proposed method is removal of different numbers of the base classifiers based on their quality. Implementation of a weighting mechanism to the base classifiers at the decision-making stage is another advantage of the algorithm. This facilitates adaptability when drifts take place, which leads to classifiers with higher efficiency. Furthermore, the proposed method is tested on a set of standard data and the results confirm higher accuracy compared to available ensemble classifiers and single classifiers. In addition, in some cases the proposed classifier is faster and needs less storage space.

**Key words:** Data stream, Classificaion, Ensemble classifiers, Concept drift
**doi:**10.1631/FITEE.1400398   **Document code:** A   **CLC number:** TP391

## 1 Introduction

'Data stream' refers to a continuous and infinite series of data with a high generation rate. Recent advances in sensors technology, data processing, and communication have enabled automatic data storage. The need to process larger amounts of data has promoted the field of data mining and knowledge discovery (Xu *et al.*, 2011). These advances have also multiplied the applications that relate with data streams. Data stream mining is a research topic of growing interest. Unique features of data streams including immense volume, high generation rate, limited processing time, large dimensions, infinite data volume, and concept drift, distinguish data streams from standard types of data. This means that traditional data mining methods are not suitable for classifying data streams. The majority of classification methods assume that data distribution does not change over time, so they have a problem in recognizing concept drift when it comes to data streams (Jiang *et al.*, 2009; Gama, 2010).

A critical task of the data mining process is classification of data. Classification methods are composed of two phases: (1) A model is generated using training data with a specific class tag; (2) The generated model is employed to predict the class label of the test data (i.e., the data of which we know the class label but the model does not). In classification of data streams, labeled and unlabeled data are in the same set. Indeed, training and testing phases are performed in parallel, and a model is developed

‡ Corresponding author
ORCID: Omid ABBASZADEH, http://orcid.org/0000-0002-8923-940X

as soon as these two phases finished (Gama, 2010). Generally, classification algorithms are grouped in two groups of single and ensemble models. Single models learn incrementally and need new data for updating. The updating process in single models is complicated. Additionally, classification of the data by single models is not one hundred percent reliable. On the other hand, ensemble models are constituted of several single models. In data stream classification, ensemble learning methods enjoy several advantages over other models such as being easily scalable, having parallel function compatibility, and fast change adaptability through pruning of low-performance sections. Ensemble classifications also feature high accuracy (Kuncheva, 2004).

Under data stream classification, the unlabeled data is fed to the classification system and then assigned with the correct labels. The labeled data can be used to update the classifier model. Data stream classification is always performed at the training stage, as the feedback is the only way to determine concept drift, to adapt, and to update the classification model.

An ideal classifier for data streams needs to meet specific features, including: (1) high accuracy, (2) fast change adaptation, (3) low computation and storage load, (4) minimum number of parameters, (5) noise tolerance, and (6) compatibility with new concepts, recursiveness, and optimum use of the past data. Some of these features such as low storage load and recursiveness are controversial and not all these features can be collected in a single system. Due to the specific nature of the data stream, the following requirements must be considered for a data stream classifier (Street and Kim, 2001): (1) all samples are processed only once, (2) limited storage is needed, (3) there is limited data processing time, and (4) the model should provide the best prediction if it stops before the conclusion.

Integrating, composed of the voting method, classifier combination, and removal of low quality classifiers, is performed online in the proposed ensemble classifier. The present study is a development based on previous work. In addition to a data stream classifier, a novel method composed of concept drift detection, expert deletion, and dynamical weighting mechanism is introduced. Data processing is carried out on blocks of data and drifts are detected based on effectiveness of base classification between two consecutive blocks of data (Base classifiers used in this method are decision trees). The number of removed classes varies in each replication of classifier evaluation and in the case in which the number of concept drifts in the data exceeds a threshold, the number of removed classifications will be more than that in the previous stage. Dynamic weighting with a base classifier leads to higher diversity in the model. Along with higher accuracy, the Kappa statistics (an indicator of intelligence of the classifier) also increases. Eventually, with implementation of a weighting mechanism, base classification is achieved based on previous records and higher accuracy.

## 2 Related work

Before discussing similar studies in the field, three methods to detect concept drift are introduced. Gama *et al.* (2004) introduced the drift detection method (DDM). The main idea of DDM is that when data distribution is stable over time, the error level of the model decreases over time, and vice versa. Thus, the status of the model can take two modes: (1) warning level—when the input data is stored in temporary memory; (2) drift level—drift is noted when the system enters this level and the model generator algorithm starts relearning by using the data in the temporary memory. An extension to DDM is the early drift detection method (EDDM) (Baena-García *et al.*, 2006). The main idea is that the distance between two errors increases over time while the concepts adopt a more stable condition. It was assumed that when the distance between errors is less than a threshold level ($\alpha$), a warning mode will be triggered and recent data will be stored in the temporary memory. In addition, the method detects drift when the warning mode is triggered and distance between the errors is less than a threshold $\beta$ ($\alpha > \beta$). EDDM shows higher performance and needs less storage. Bifet (2009) proposed an adapting sliding window algorithm called ADWIN. That is, when the average value of two parts of sub-windows exceeds an acceptable level, the recent sub-window data will be dropped. One of the inputs of the algorithm is conceptual level ($\alpha$), which represents a confidence level to the test.

The concept adapting very fast decision tree (CVFDT) (Hulten *et al.*, 2001) is an extension to the very fast decision tree algorithm. The algorithm is known for high accuracy and a fast decision tree, showing the capability to detect and respond to change in the process of data sample generation. In fact, CVFDT is capable of detecting and dealing with concept drift. Dynamic weighted majority (DWM) (Kolter and Maloof, 2007) is an ensemble algorithm, which does not use any internal explicit detection method. Concept drift is detected by weighting on the performances of base classifiers. At first, a classifier is assigned a fixed weight; then, the weight of the classifier is increased/decreased based on its performance and parameter $\rho$ (a factor set by the operator for increasing/decreasing weights). When the classifier error exceeds a threshold level, one of the base classifiers is dropped and replaced by another classifier. Eventually, the majority voting is done by implementing a weighting function on the classifiers. OZAboost is a kind of online boosting algorithm (Oza, 2005). OZAboost updates the weight with a Poisson distribution and is a parallel boosting method that follows Adaboost. OZAboost-Adwin is an extension to OZAboost in which drifts are detected using the ADWIN method. Like the DWM method, an accuracy weighted ensemble (AWE) (Wang *et al.*, 2003) generates variation by weighting base classifiers and employs Hoeffding trees for classification. Instead of using a mechanism to detect drift, the method employs a function for weighting. Regardless of drift, the proposed classifier drops classifications with minimum efficiency and generates a new classification based on the data from the last training step.

Zhang *et al.* (2015) proposed an ensemble classifier based efficient indexing structure method, for creation and management of an ensemble. E-tree was introduced in the tree structure for the efficient insert, delete, and search operations, and then the tree as base classifiers uses four different ensemble methods to build ensemble classifiers. Among these four ensemble methods, the LE-tree method (where the upper bound for the number of classifications is defined and when this limit is met, the old class will be dropped from ensemble) takes the least time.

## 3 Concept drift

Concept drift refers to change in distribution of the data in the context of input data. The context is a stream of data with a stable distribution. Here we define the concept drift in detail: suppose $\boldsymbol{X}$ is a specimen of $p$ dimensions. $\boldsymbol{X} \in C$, and $C$ is a set of classes. The preferred classifier assigns data $\boldsymbol{X}$ to $c_i$ accurately, based on the probability of each class $p(c_i)$ and conditional probability $p(\boldsymbol{X}|c_i)$, to achieve the desired classification. A set $S$ is used to describe the behavior of a classifier:

$$\begin{aligned} S =&\{(p(c_1), p(\boldsymbol{X}|c_1)), (p(c_2), p(\boldsymbol{X}|c_2)), \ldots, \\ &(p(c_k), p(\boldsymbol{X}|c_k))\}, \end{aligned} \tag{1}$$

where the probability of each class and its conditional probability pertinent to the source are given. According to the Bayes theorem, the probability of the observed specimen $\boldsymbol{X}$ belonging to class $c_i$ is given by Eq. (2) and $\boldsymbol{X}$ belongs to the class with the maximum probability value:

$$p(c_i|\boldsymbol{X}) = \frac{p(c_i)p(\boldsymbol{X}|c_i)}{p(\boldsymbol{X})}, \tag{2}$$

where $p(\boldsymbol{X})$ is a piece of evidence of $\boldsymbol{X}$, which is constant for all the classes $c_i$. There is concept drift when:

1. Class priors $p(c)$ might change over time.

2. The distributions of one or several classes $p(\boldsymbol{X}|c)$ might change.

3. The posterior distributions of the class memberships $p(c|\boldsymbol{X})$ might change.

The probability of change in $p(c|\boldsymbol{X})$ is considered as concept drift and change in $p(\boldsymbol{X}|c)$ is called virtual or pseudo concept drift (Zliobaite, 2009).

Another definition of concept drift has been proposed by Zhang *et al.* (2008). In this definition, by using $p(x,c) = p(x|t) \cdot p(c|x)$, changes in the previous distribution $p(x)$ are made dependent on time $(t)$, and concept drift occurs when $p(x|t)$ and $p(c|x)$ change. So, concept drift is categorized as follows:

1. Loose concept drift (LCD): this kind of drift occurs if $p(x|t)$ is altered as systematically localized and $p(c|x)$ is altered gradually and evolves locally as time elapses. If $p(x|t)$ changes continuously but $p(c|x)$ remains unchanged, then LCD is exactly the sample selection bias problem in traditional machine learning.

2. Rigorous concept drift (RCD): this kind of drift occurs when $p(x|t)$ and $p(c|x)$ are altered randomly and suddenly.

By definition, 'systematically localized' is scarcity or absence of sample's overlapping in the current and previous chunks. This definition of concept drift is more accurate than other definitions, since, in addition to involving time, it includes other factors such as bias, which can lead to misperception of concept drift. Another acceptable classifier that classifies changes based on appearance is grouped in four groups: (1) sudden, (2) gradual, (3) recursive, and (4) predictable (Zliobaite, 2009). Based on how concept drift is handled, stream data classification methods can be grouped as (i) instance weighting (Ruping, 2001), (ii) window based (Oza, 2005), and (iii) ensemble methods (Street and Kim, 2001; Kolter and Maloof, 2007; Tsymbal *et al.*, 2008; Minku and Yao, 2012; Brzezinski and Stefanowski, 2014). In group (i), the input data is dropped from the training set based on the weight assigned to them. In group (ii), in addition to handling drift, classification is carried out by generating a sliding window with occasionally different sizes on the just added data, which also supplies training data. In group (iii), which is the most common method, drift is handled by assigning weights to base classifiers, dropping and replacing poor classifiers by new models.

## 4 Proposed method

Any ensemble classifier for processing data stream should be able to show: (1) how the class label of an input will be predicted, (2) how concept drift is detected, and (3) how an expert is dropped from the set of classifiers.

The proposed classifier employs the weighted majority voting to generate its final prediction. Drift is detected based on a new measure called 'Kappa statistics', which emphasizes intelligence of the algorithm and employs a new concept taking the quality of each classifier to drop base classifiers. The process is described in detail below.

### 4.1 Weighting mechanism

An active ensemble classifier was proposed by Zhu *et al.* (2010) to label an important part of the samples. By dividing the error into variance and bias, it is demonstrated that the bias error can be ignored when the same base classifiers are used. Furthermore, the model error is decreased by reducing the variance. Therefore, this weighting base classifier problem can be formulated as an optimization problem and the objective function is variance minimization. In this study, the bias error is avoided in weighting and, instead of applying the optimization function which is a computationally expensive task, to determine the weights, simple weighting functions are used in different conditions to take into account the optimization of weights.

The weighting process in the proposed method is carried out on the base classifiers; one linear and one nonlinear functions are used when input data are added under different conditions. As suggested by the tests, the nonlinear function is more effective when a concept is stationary, and the linear one is preferred when fluctuation of the input data is notable. On the other hand, the nonlinear function in absence of drift immunizes the classifier against noise and irrelevant data. According to statistics and pattern detection techniques, classification is a prediction process and authenticity of the predication can be ascertained in different ways such as by mean square error (MSE), mean absolute error (MAE), variance, and standard deviation. MSE and MAE are more common than the others. Wang *et al.* (2003) and Brzezinski and Stefanowski (2014) used MSE in the weighting algorithm, and we use MAE in this study. The reason is that the square error is mostly used to show the intensity of changes and thus more emphasis is put on large errors (e.g., noises) with a low probability. MAE is obtained as follows:

$$\mathrm{MAE}=\frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i-y_i|, \tag{3}$$

where $\hat{y}_i$ stands for the prediction results, $y_i$ the actual value of the class label, and $n$ the number of observations. The linear weighting function is

$$W_{\mathrm{Linear}}=\max\{\mathrm{MAE}_{\mathrm{r}}-\mathrm{MAE}_i,\epsilon\}, \tag{4}$$

and the nonlinear weighting function is

$$W_{\mathrm{Non\_linear}}=\frac{1}{\mathrm{MAE}_{\mathrm{r}}+\mathrm{MAE}_i+\epsilon}, \tag{5}$$

where $\epsilon$ denotes a small and almost zero constant and $\mathrm{MAE}_i$ represents the mean absolute error of the $i$th base classifier in the recent stream, which is given as

$$\mathrm{MAE}_i=\frac{1}{|S_n|}\sum_{(\boldsymbol{x},c)\in S_n}|1-p_c^i(\boldsymbol{x})|. \tag{6}$$

Here $S_n$ represents the input data, $|S_n|$ is the size of input data, $\boldsymbol{x}$ is an input instance of the classifier, and $c$ is the class label of the instance. $p_c^i(\boldsymbol{x})$ stands for the probability of input data belonging to class $c$ from the viewpoint of the $i$th classifier. Supposing that a classifier generates classification randomly based on the distribution of each class, with $p(c)$ being the probability of selecting class $c$, $\mathrm{MAE_r}$ is obtained as

$$\mathrm{MAE_r} = \sum_c p(c)|1 - p(c)|. \tag{7}$$

Now, let the input dataset be composed of two classes $\{0, 1\}$ with a uniform distribution. Then Eq. (7) yields 0.5. $\mathrm{MAE_r}$ in the proposed method acts as a measure of poorness of the classifier. The pseudocode for weighting the classifiers is given in Algorithm 1.

**Algorithm 1** Weighting

```
Require: $S_i$: data stream; $C_i$: example chunk of size $c$;
    $E$: ensemble
Ensure: $E_w$: weighted ensemble $E$
 1: if size($C_i$) = $c$ && drift is detected then
 2:     for each $e$ in $E$ do
 3:         $e_{\mathrm{weight}} = W_{\mathrm{Linear}}$
 4:         Add $e$ to $E_w$
 5:     end for
 6: end if
 7: if size($C_i$) = $c$ && drift is not detected then
 8:     for each $e$ in $E$ do
 9:         $e_{\mathrm{weight}} = W_{\mathrm{Non\_linear}}$
10:         Add $e$ to $E_w$
11:     end for
12: end if
13: return $E_w$
```

## 4.2 Drift detection

The main issue in working with a data stream is how to detect and handle concept drift. There are a variety of statistical tests to detect drift in data streams (Gama *et al.*, 2004; Baena-García *et al.*, 2006; Bifet, 2009). As noted earlier, these tests are characterized by heavy and slow computation. The proposed method here, however, is implicitly added to the classifier algorithm. It is also implicitly available in any ensemble algorithm so that the proposed method can be compatible with all systems that employ an ensemble of classifiers for classification. Given that measuring with considerable random error effects leads to different decision-making values, one may say that replication of the measurement does not lead to a fixed result and the results are not reliable. In such a case, the Kappa coefficient is used to measure the reliability. In practice, more than 75% harmony is reliable. The Kappa coefficient is also an index to assess harmony among decision makers (Sim and Wright, 2005). For example, in Table 1, harmonies are observed mainly on the points $a$ and $d$. Kappa measures the extent of these harmonies and it is obtained through

$$\kappa = \frac{\Pr(o) - \Pr(e)}{1 - \Pr(e)}, \tag{8}$$

where $\Pr(o)$ is the relative observed agreement among raters, and $\Pr(e)$ is the hypothetical probability of chance agreement. $\Pr(e)$ and $\Pr(o)$ are

$$\Pr(e) = \frac{n_1}{n}\frac{m_1}{n} + \frac{n_0}{n}\frac{m_0}{n}, \tag{9}$$

$$\Pr(o) = \frac{a + d}{n}. \tag{10}$$

We take Kappa coefficient above 80% as excellent harmony and above 65% as good harmony; a Kappa coefficient not larger than 65% indicates that the probability of harmony is by chance (≥50%) or non-chance (<50%) (In other words, agreement may also be a chance or not). A negative Kappa coefficient indicates that the harmony is by chance. A Kappa coefficient of 65% is considered as acceptable harmony. The proposed method calls a classification process 'random' when the Kappa coefficient for the last 100 specimens of each input package is less than 65%. When this happens, the weighting function is replaced and poor classifiers are dropped. In fact, the precision of running the algorithm is a measure to survey concept drift. The pseudocode is given in Algorithm 2.

**Table 1 Example of the harmony table**

| | | $B$ | | |
|---|---|---|---|---|
| | | Yes | No | Total |
| $A$ | Yes | $a$ | $b$ | $m_1 = a + b$ |
| | No | $c$ | $d$ | $m_0 = c + d$ |
| | Total | $n_1 = a + c$ | $n_0 = b + d$ | $n = n_0 + n_1$ |

## 4.3 Expert deletion

Generally, the number of base classifiers in ensemble classifiers, as a constant throughout the operation, is set by the operator. Recently proposed

**Algorithm 2** Drift detection

```
Require: $C_{100}$: last chunk of size 100; $E$: ensemble; $\alpha$: agreement factor
Ensure: True, False
1: // Default value ($\alpha = 65\%$)
2: if ($C_{100}$) && $\kappa(E) > \alpha$ then
3:     return False
4: end if
5: if ($C_{100}$) && $\kappa(E) \le \alpha$ then
6:     return True
7: end if
```

methods drop classifiers, if needed, within a specific time period, so that the number of classifiers to be removed can be constant (one in most cases). Under the proposed method, one of the poor classifiers is dropped at the end of each chunk of input data and more than one classifier is dropped when a concept drift is detected. In fact, the number of classifiers is set to maximum by the operator.

In the expert deletion algorithm (Algorithm 3), $\alpha_{\text{min}}$ is assigned with the minimum quality of classifiers and the classifier with the minimum quality is dropped; poorest$_{\text{classifiers}}$ stores the classifiers that must be dropped; $\alpha_{\text{avg}}$ represents the harmonic mean of the classifier's quality and it is obtained as follows:

$$\alpha_{\text{avg}} = \frac{|E|}{\sum_{i=1}^{n} 1/\alpha_i}. \tag{11}$$

Variable $\alpha_i$ carries the quality of each classifier and is stored along with each classifier. The classifiers with quality smaller than the mean point are dropped. To obtain $\alpha$, assume a set of input data as

$$D = \{(x_i, y_i) | i = 1, 2, \cdots, N\}, \tag{12}$$

where $x_i$ is data point, $y_i$ the correct label for $x_i$, and $N$ the number of base classifiers in the ensemble classifier. The error of the $i$th classifier on the input data is

$$\text{error}_i = \frac{1}{N} \sum_{j=1}^{N} I(c_i(x_j)) \neq y_j, \tag{13}$$

where $I(c_i(x_j))$ is classification on the specimen $\boldsymbol{x}$ carried on by classifier $C_i$. To add the effect of the weight of each classifier on its error, the inverse weight of each classifier is multiplied by its error:

$$\varepsilon_i = \frac{1}{w_i} \cdot \text{error}_i. \tag{14}$$

**Algorithm 3** Expert(s) deletion

```
Require: $S_i$: data stream; $C_i$: example chunk of size $c$; $E$: ensemble
Ensure: poorest$_{\text{classifiers}}$
1: $\alpha_{\text{min}} = 1.0$
2: poorest$_{\text{classifiers}}$=null
3: if size($C_i$) = $c$ && drift is not detected then
4:     if size($E$) = default ensemble size then
5:         for each $e$ in $E$ do
6:             if $e_\alpha \le \alpha_{\text{min}}$ then
7:                 $\alpha_{\text{min}} = e_\alpha$
8:                 poor$_{\text{classifier}}$ = $e$
9:             end if
10:        end for
11:        poorest$_{\text{classifiers}}$.add(poor$_{\text{classifier}}$)
12:    end if
13: end if
14: if size($C_i$) = $c$ && drift is detected then
15:     for each $e$ in $E$ do
16:         $\alpha_{\text{total}} += 1/(e_\alpha + \epsilon)$
17:     end for
18:     $\alpha_{\text{avg}} = |E|/\alpha_{\text{total}}$
19:     for each $e$ in $E$ do
20:         if $e_\alpha < \alpha_{\text{avg}}$ then
21:             poorest$_{\text{classifiers}}$.add($e$)
22:         end if
23:     end for
24: end if
25: return poorest$_{\text{classifiers}}$
```

Eventually, the quality of each classifier is obtained by normalization:

$$e_\alpha = \frac{1}{2} \ln \frac{1 - \varepsilon_i}{\varepsilon_i}. \tag{15}$$

Under the proposed method, examinations to drop classifiers are carried out when a new block of data is added and classifiers with minimum values are removed. As illustrated in the dropping algorithm (Algorithm 3) and based on different mechanisms, the number of classifiers varies between the cases in which a drift is detected or no drift is detected. No classifier is dropped when there is no drift and the number of base classifier does not exceed the upper limit, and one classifier is dropped otherwise. On the other hand, the classifiers with minimum mean harmonic values are dropped when a drift is detected.

## 5 Experimental results

As discussed in the previous section, the proposed method handles a variety of ensemble classifications by using weighting functions and dropping poor classifiers. When a drift is detected, the number of dropped classifiers might be more than one. The Kappa coefficient was introduced to detect drift, which also shows the precision and reliability of the proposed method. To compare the results with those of other methods, measures such as accuracy, classification time, precision of prediction, and required storage were employed under slow and sudden drift conditions with different cluster sizes. Experiments were implemented on a Linux machine with 2.20 GHz CPU and 4 GB memory using a data stream management system MOA (Bifet *et al.*, 2010).

### 5.1 Dataset and assessment

Standard datasets used in other similar research were used here. The data is available in MOA, which enables us to exactly set the point and place of drift. Therefore, the accuracy and error level of the model can be measured when drift is induced. Different types of data were tested (Table 2).

In the following subsections we compare the proposed method with CVFDT, DWM, OZA, and AWE models regarding accuracy, average required memory, prediction precision, and classification time. The models compared were set in default mode.

**Table 2 Characteristics of different datasets**

| Dataset | Number of drifts | Number of labels | Number of attributes |
|---|---|---|---|
| $\mathrm{SEA_S}$ | 3 | 4 | 3 |
| $\mathrm{SEA_G}$ | 9 | 4 | 3 |
| $\mathrm{Hyper_S}$ | 4 | 2 | 10 |
| $\mathrm{Hyper_M}$ | 8 | 2 | 10 |
| $\mathrm{LED_M}$ | 4 | 4 | 5 |
| Wave | 0 | 3 | 40 |
| $\mathrm{Wave_M}$ | 9 | 3 | 40 |

S: sudden drift; G: gradual drift; M: combination of drifts including sudden and gradual drifts. For all the datasets, the number of instances is $1\times10^5$

### 5.2 Accuracy

To compare the accuracy of the proposed method with those of other methods, a window size of 500 was used for ensemble classifiers. In addition, the effect of window size was checked using different window sizes. Accuracy comparison between the proposed method and other methods is given in Table 3.

**Table 3 Average classification accuracies (%)**

| Dataset | Proposed | DWM | OZA | CVFDT | AWE |
|---|---|---|---|---|---|
| $\mathrm{SEA_S}$ | 84.84 | 84.82 | 82.64 | 87.71 | 87.19 |
| $\mathrm{SEA_G}$ | 87.37 | 84.91 | 83.37 | 85.00 | 85.10 |
| $\mathrm{Hyper_S}$ | 85.70 | 88.70 | 71.65 | 82.40 | 87.30 |
| $\mathrm{Hyper_M}$ | 79.90 | 76.84 | 71.79 | 71.36 | 72.17 |
| $\mathrm{LED_M}$ | 74.10 | 73.95 | 71.63 | 68.11 | 73.58 |
| Wave | 85.71 | 83.82 | 83.37 | 83.90 | 81.57 |
| $\mathrm{Wave_M}$ | 84.15 | 83.75 | 83.22 | 82.71 | 81.31 |

S: sudden drift; G: gradual drift; M: combination of drifts including sudden and gradual drifts

Evidently, the accuracy of the proposed method is acceptable with datasets LED, Wave, Hyper, and SEA under gradual changes. Furthermore, the proposed method outperformed other methods regarding accuracy under combined drifts mode (sudden and gradual drifts). Classifiers AWE and DWM had higher performance under sudden drift.

The proposed classifier outperformed the ensemble classifiers regarding accuracy under sudden drift; however, two classifiers were better than the proposed method in this regard. This is illustrated in Fig. 1. The cause of sudden changes in the classifier accuracy of the proposed method is the use of small decision tree. Although the decision trees used in the base classifiers are small, accurate weighting leads to acceptable harmony of the classifier. When working with artificial data, the operator may accurately manipulate the place of drift and length of drift by setting the parameters of the data generation function.

As indicated earlier, the proposed classifier has higher accuracy when the drift is gradual and combined. This is due to two weighting functions for drift and no drift conditions. The proposed method also has acceptable tolerance for noise to guarantee that when the concepts are received with fixed distribution, the proposed method shows less fluctuation compared with other models. Table 4 lists the effect of window size on the set of data used in this work. Clearly, increase in window size leads to a decrease in the time needed to accomplish data classification, while the classification accuracy decreases at the same time. The extent of the accuracy variation between maximum and minimum window sizes notably reaches 2% in some cases. The window size also influences the accuracy and time classification, which is more notable when fewer characteristics are under consideration (e.g., SEA). Furthermore, larger

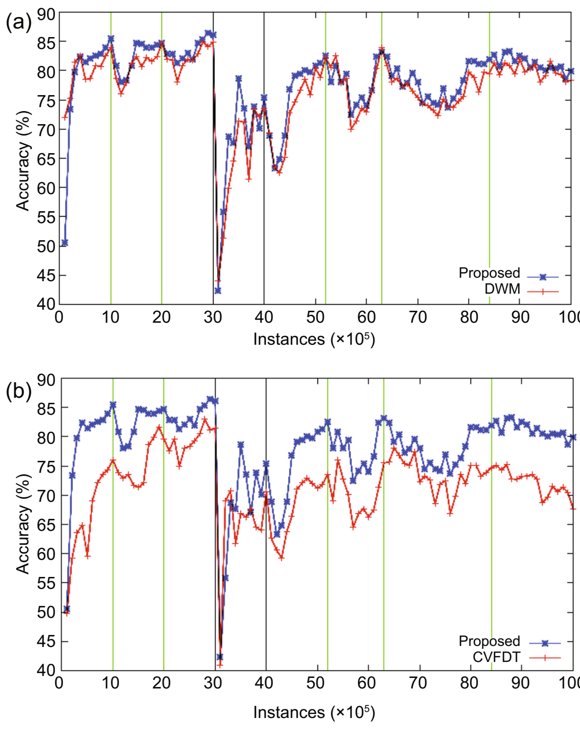

**Fig. 1 Comparison of classification accuracy on the Hyper$_M$ dataset: (a) the proposed method vs. DWM; (b) the proposed method vs. CVFDT. The black and green horizontal lines represent sudden and gradual drifts, respectively. The first drift occurs at time 0. References to color refer to the online version of this figure**

changes of window size on the dataset with more characteristics (i.e., Wave) lead to different behavior so that increase of window size leads to the increase of classification accuracy.

## 5.3 Memory

The number of base classifiers in the proposed method is set to maximum by the operator; this number increases or decreases under different conditions (drift and no drift). This is a great saving of memory demand compared with other ensemble methods. Table 5 lists the memory usage. With only one decision tree, the classifier CVFDT needs smaller memory, which is not listed in the table.

**Table 5 Average memory usage**

| Dataset | Memory (MB) | | | |
|---|---|---|---|---|
| | Proposed | DWM | OZA | AWE |
| SEA$_S$ | 1.56 | 1.32 | 6.23 | 2.66 |
| SEA$_G$ | 1.22 | 1.73 | 4.82 | 1.93 |
| Hyper$_S$ | 3.38 | 4.24 | 11.31 | 3.41 |
| Hyper$_M$ | 3.22 | 4.37 | 10.19 | 3.71 |
| LED$_M$ | 0.48 | 0.61 | 2.56 | 0.32 |
| Wave | 56.30 | 6.18 | 69.73 | 50.63 |
| Wave$_M$ | 15.81 | 6.42 | 26.16 | 12.29 |

S: sudden drift; G: gradual drift; M: combination of drifts including sudden and gradual drifts

## 5.4 Precision

There are different definitions for reliability of an algorithm; here we used precision of classifier, which is composed of smaller elements. In most cases, accuracy and precision are used interchangeably, but in fact, they represent two independent concepts. There are different ways to test precision. As mentioned, the Kappa coefficient was used here. Table 6 lists the Kappa coefficient value for different sets of data and classifiers. The proposed method is similar to the Boosting method (strong classifier) in the way of selecting base classifiers. We expected higher reliability of the harmony used for classification. Table 6 confirms high precision of the proposed method in all cases except for SEA and Hyper when sudden drifts take place. In addition, the relationship between accuracy and precision is highlighted by the table. For instance, like accuracy, classifier DWM resulted in higher precision on Hyper.

**Table 4 Average classification accuracy and time using different window sizes**

| Window size | Time (s) | | | | | Accuracy (%) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | SEA$_S$ | SEA$_G$ | Hyper$_S$ | Hyper$_M$ | Wave | SEA$_S$ | SEA$_G$ | Hyper$_S$ | Hyper$_M$ | Wave |
| 500 | 40.19 | 38.72 | 41.95 | 106 | 2120.96 | 84.84 | 87.37 | 85.70 | 79.90 | 85.71 |
| 750 | 39.83 | 36.70 | 41.25 | 104 | 2110.91 | 84.57 | 87.14 | 85.50 | 78.84 | 85.94 |
| 1000 | 37.92 | 36.45 | 40.06 | 109 | 2111.87 | 84.21 | 86.79 | 85.29 | 78.80 | 86.29 |
| 1250 | 37.45 | 35.70 | 39.84 | 103 | 2106.12 | 83.94 | 86.52 | 85.47 | 78.51 | 86.52 |
| 1500 | 36.30 | 35.12 | 38.88 | 99 | 2097.11 | 83.92 | 86.25 | 85.23 | 79.33 | 86.50 |
| 1750 | 35.59 | 34.41 | 38.02 | 98 | 2081.21 | 83.68 | 85.95 | 85.62 | 78.91 | 86.41 |
| 2000 | 34.88 | 33.58 | 37.05 | 96 | 2064.91 | 83.30 | 85.64 | 85.12 | 78.11 | 86.83 |

S: sudden drift; G: gradual drift; M: combination of drifts including sudden and gradual drifts

**Table 6 Average classification precision**

| Dataset | Classification precision (%) | | | |
|---|---|---|---|---|
| | Proposed | DWM | OZA | AWE |
| SEA$_\mathrm{S}$ | 66.56 | 65.59 | 60.92 | 69.96 |
| SEA$_\mathrm{G}$ | 71.62 | 66.15 | 62.70 | 68.72 |
| Hyper$_\mathrm{S}$ | 69.50 | 76.82 | 41.58 | 74.59 |
| Hyper$_\mathrm{M}$ | 55.41 | 53.11 | 40.81 | 43.84 |
| LED$_\mathrm{M}$ | 71.20 | 70.33 | 67.78 | 69.92 |
| Wave | 79.10 | 75.63 | 74.95 | 72.26 |
| Wave$_\mathrm{M}$ | 80.19 | 73.86 | 74.57 | 71.70 |

S: sudden drift; G: gradual drift; M: combination of drifts including sudden and gradual drifts

## 5.5 Time

The average run time of the algorithm for 1000 test samples was obtained (Table 7). The proposed method shows high processing speed compared with DWM and AWE. This is because the proposed method checks the quality of each base classifier at the end of each stage, while this is done at the weighting stage in AWE and DWM methods. Due to preprocessing, OZA has a high volume of computation and memory space. In Table 8, we compare the four ensemble methods in terms of execution time. As listed in the table, the proposed method was better on the first three datasets.

**Table 7 Average of time consumption for 1000 test examples**

| Dataset | Time (s) | | | |
|---|---|---|---|---|
| | Proposed | DWM | OZA | AWE |
| SEA$_\mathrm{S}$ | 0.19 | 0.07 | 7.94 | 0.14 |
| SEA$_\mathrm{G}$ | 0.09 | 0.06 | 5.97 | 0.09 |
| Hyper$_\mathrm{S}$ | 0.31 | 0.20 | 9.16 | 0.20 |
| Hyper$_\mathrm{M}$ | 0.24 | 0.21 | 3.90 | 0.22 |
| LED$_\mathrm{M}$ | 0.54 | 0.15 | 0.75 | 0.15 |
| Wave | 3.67 | 0.48 | 33.65 | 2.97 |
| Wave$_\mathrm{M}$ | 3.87 | 0.46 | 33.46 | 1.05 |

S: sudden drift; G: gradual drift; M: combination of drifts including sudden and gradual drifts

## 6 Conclusions

The proposed method here is based on observation of error, intelligence, and weighting the votes of base classifiers. In practice, each classification operation on the input data creates a classifier on the recent data, which is added to the set of classifiers when a new set of data is added. This ensures real-time classification and updating.

**Table 8 Time consumption for training and testing in datastream**

| Dataset | Time (s) | | | |
|---|---|---|---|---|
| | Proposed | DWM | OZA | AWE |
| SEA$_\mathrm{S}$ | 40.19 | 41.22 | 221.09 | 43.22 |
| SEA$_\mathrm{G}$ | 34.71 | 35.93 | 206.11 | 38.66 |
| Hyper$_\mathrm{S}$ | 41.95 | 47.08 | 327.40 | 48.73 |
| Hyper$_\mathrm{M}$ | 80.82 | 74.09 | 260.55 | 79.09 |
| LED$_\mathrm{M}$ | 430.09 | 384.77 | 597.00 | 369.56 |
| Wave | 2230.44 | 2111.20 | 18932.00 | 2120.96 |

S: sudden drift; G: gradual drift; M: combination of drifts including sudden and gradual drifts

As the comparisons based on standard datasets show, the proposed method has acceptable accuracy under gradual and combined drifts, while under sudden drift, DWM and AWE have higher accuracy. Compared with other methods, the proposed method approaches concept drift differently and uses a precision measure to detect drift. Good harmony is another advantage of the algorithm, which is better with gradual and mixed drift than with sudden drift.

## References

Baena-García, M., del Campo-Ávila, J., Fidalgo, R., *et al.*, 2006. Early drift detection method. ECML PKDD.

Bifet, A., 2009. Adaptive learning and mining for data streams and frequent patterns. *ACM SIGKDD Explor. Newsl.*, **11**(1):55-56. [doi:10.1145/1656274.1656287]

Bifet, A., Holmes, G., Kirkby, R., *et al.*, 2010. MOA: massive online analysis. *J. Mach. Learn. Res.*, **11**:1601-1604.

Brzezinski, D., Stefanowski, J., 2014. Reacting to different types of concept drift: the accuracy updated ensemble algorithm. *IEEE Trans. Neur. Netw. Learn. Syst.*, **25**(1):81-94. [doi:10.1109/TNNLS.2013.2251352]

Gama, J., 2010. Knowledge Discovery from Data Streams. Chapman & Hall/CRC, London.

Gama, J., Medas, P., Castillo, G., *et al.*, 2004. Learning with drift detection. Brazilian Symp. on Artificial Intelligence, p.286-295. [doi:10.1007/978-3-540-28645-5_29]

Hulten, G., Spencer, L., Domingos, P., 2001. Mining time-changing data streams. Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery Data Mining, p.97-106. [doi:10.1145/502512.502529]

Jiang, T., Feng, Y.C., Zhang, B., *et al.*, 2009. Monitoring correlative financial data streams by local pattern similarity. *J. Zhejiang Univ.-Sci. A*, **10**(7):937-951. [doi:10.1631/jzus.A0820445]

Kolter, J.Z., Maloof, M.A., 2007. Dynamic weighted majority: an ensemble method for drifting concepts. *J. Mach. Learn. Res.*, **8**:2755-2790.

Kuncheva, L.I., 2004. Combining Pattern Classifiers: Methods and Algorithms. John Wiley & Sons, Hoboken.

Minku, L.L., Yao, X., 2012. DDD: a new ensemble approach for dealing with concept drift. *IEEE Trans. Knowl. Data Eng.*, **24**(4):619-633. [doi:10.1109/TKDE.2011.58]

Oza, N.C., 2005. Online bagging and boosting. IEEE Int. Conf. on System and Man Cybernetics, p.2340-2345. [doi:10.1109/ICSMC.2005.1571498]

Ruping, S., 2001. Incremental learning with support vector machines. IEEE 13th Int. Conf. on Data Mining, p.641-642. [doi:10.1109/ICDM.2001.989589]

Sim, J., Wright, C.C., 2005. The kappa statistic in reliability studies: use, interpretation, and sample size requirements. *Phys. Ther.*, **85**(3):257-268.

Street, W.N., Kim, Y.S., 2001. A streaming ensemble algorithm (SEA) for large-scale classification. Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, p.377-382. [doi:10.1145/502512.502568]

Tsymbal, A., Pechenizkiy, M., Cunningham, P., *et al.*, 2008. Dynamic integration of classifiers for handling concept drift. *Inform. Fus.*, **9**(1):56-68. [doi:10.1016/j.inffus.2006.11.002]

Wang, H., Fan, W., Yu, P.S., *et al.*, 2003. Mining concept-drifting data streams using ensemble classifiers. Proc. 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, p.226-235. [doi:10.1145/956750.956778]

Xu, W.H., Qin, Z., Chang, Y., 2011. Clustering feature decision trees for semi-supervised classification from high-speed data streams. *J. Zhejiang Univ.-Sci. C (Comput. & Electron.)*, **12**(8):615-628. [doi:10.1631/jzus.C1000330]

Zhang, P., Zhu, X., Shi, Y., 2008. Categorizing and mining concept drifting data streams. Proc. 14th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, p.812-820. [doi:10.1145/1401890.1401987]

Zhang, P., Zhou, C., Wang, P., *et al.*, 2015. E-tree: an efficient indexing structure for ensemble models on data streams. *IEEE Trans. Knowl. Data Eng.*, **27**(2):461-474. [doi:10.1109/TKDE.2014.2298018]

Zhu, X., Zhang, P., Lin, X., *et al.*, 2010. Active learning from stream data using optimal weight classifier ensemble. *IEEE Trans. Syst. Man Cybern. B*, **40**(6):1607-1621. [doi:10.1109/TSMCB.2010.2042445]

Žliobaite, I., 2009. Learning under Concept Drift: an Overview. Technical Report. Vilnius University, Lithuania.