The Computer Journal Advance Access published June 20, 2014

doi:10.1093/comjnl/bxu050

A Selective Detector Ensemble for Concept Drift Detection

LEI DU*, QINBAO SONG, LEI ZHU AND XIAOYAN ZHU

Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China
*Corresponding author: qbsong@mail.xjtu.edu.cn

Concept drifts usually originate from many causes instead of only one, which result in two types of concept drifts: abrupt drifts and gradual drifts. From the point of view of speed, concept drifts pose strong challenges for data stream mining. In this paper, we propose a selective detector ensemble to detect both abrupt and gradual drifts. We first present our detector ensemble construction method, and then introduce how to use this ensemble to detect concept drifts with the proposed *early-find-early-report* rule. To evaluate the performance of our method, we compare it with four drift detection methods on eight publicly available data sets containing various concept drifts. The experimental results show that compared with those benchmarks, our ensemble method can effectively improve the recall and false negative rate without significantly increasing the false positive rate, and has stronger generalization ability than those single-change-indicator-based methods.

Keywords: data stream; concept drifts; detector ensemble; selective ensemble method

Received 23 October 2013; revised 19 May 2014
Handling editor: Ethem Alpaydin

1. INTRODUCTION

A data stream is an ordered sequence of examples that is generated continuously on the fly [1]. Examples of data streams include industrial processes records, financial time series, computer network traffic data, retail chain transactions, phone conversations records and various sensor data. Thereinto, classification data streams are one of the most widely used stream models, each of which is made up of an attribute vector and a class label [2]. For brevity, classification data streams are abbreviated to data streams hereafter in this paper.

Data streams are very different from traditional stationary databases—they can be read only once, too large to fit in main memory, and different concepts probably appear from time to time [3]. Of these, concept drifts, which result from the time-changing property of the underlying distribution of data streams [4, 5], are the most compelling issue. They can lead to expiration of models or patterns learned from past examples [4, 5]. Therefore, detecting concept drifts is full of importance and challenge.

Concept drifts are diverse and abundant [6, 7], because it is usually incurred by many causes other than only one, which means concept drifts in different periods probably have different causes and present differently [8]. From the point of view of speed, there are two different types of concept drifts: the abrupt/sudden drifts, when a complete drift takes only one or very few timesteps, and gradual/slow drifts, otherwise [7, 8].

Unfortunately, current detection methods cannot well address both concept drifts consistently under different conditions [9, 10]. Instead, they either perform well with abrupt drifts, or do well on data streams containing gradual drifts. The reason is that these methods utilize only one statistic as the change indicator and cannot deal with concept drifts presenting different speeds [9–13]. Recently, Minku *et al.* [7, 8] studied the speed of concept drifts and analyzed their impact on the online classifier; they however do not detect concept drifts.

The ensemble method integrates several base methods and combines advantages of them, so it can obtain better predictive performance than any of the constituent methods can [14–17], and is easier to scale [18]. Recently, a selective ensemble paradigm is proposed and has gained great success [19], and has been used to mine data streams [7, 8, 18, 20–25]. However, these proposed methods integrate different *classifiers* aiming to accurately predict the class labels of data stream examples. That means they do not explicitly detect concept drifts [7, 8, 18, 20–22, 25].

In this paper, we propose a novel concept drift detection method based on the selective *e*nsemble of *Detector*s (*e*-Detector) rather than the ensemble of *classifiers*, with the purpose of simultaneously identifying both abrupt and gradual drifts. Before detecting, *e*-Detector chooses diverse detection methods with powerful detection ability to construct the ensemble based on reference data streams. During detecting, *e*-Detector combines base detection methods using the *early-find-early-report* fusion rule to discover concept drifts. The experimental results on eight publicly available data sets show that, compared with four well-known benchmark methods, i.e. DDM [9], EDDM [10], ADWIN [11] and STEPD [12], *e*-Detector is able to statistically improve the *Recall* (reduce the false negative rate) significantly, while its false alarm rate is not significantly increased. The results also reveal that *e*-Detector has strong generalization ability, and performs better than or is comparable to those benchmarks in terms of prequential error rate.

The remainder of this paper is organized as follows. In Section 2, we briefly review the related work. In Section 3, we systemically introduce the *e*-Detector and how it is constructed. In Section 4, we present how to use *e*-Detector to detect both abrupt and gradual drifts. In Section 5, we conduct experiments to evaluate *e*-Detector on both artificial and real-world data sets, and compare it with four benchmark methods. In Section 6, we discuss the results and conclude this paper.

2. RELATED WORK

Generally, there are two types of work regarding the concept drift detection.

The first one is based on interleaved test-then-train mode (see Section 4.2 for details). Gama *et al.* [9] detected concept drifts using online classifier's error rate as the change indicator. When monitoring, once the error rate exceeds the threshold, a concept drift is reported. As the error rate demands enough wrong predictions to find concept drifts, it takes a long time to react to gradual drifts, suggesting it does not perform well on gradual drifts. Baena-García *et al.* [10] employed the interval between incorrect classifications to find concept drifts. The interval is sensitive to concept changes thus this method performs well on gradual drifts. But, it does not do well on abrupt drifts. Nishida and Yamauchi [12] used the difference between the recent accuracy and the overall accuracy to signal concept drifts. However, the method produces too many false alarms if the sliding window, in which the recent accuracy is calculated, is small. Moreover, if the sliding window is quite large, it does not perform well on graduate drifts. The last but not the least problem is that different sliding windows should be used due to various concept drifts. Bifet and Gavaldà [11] located concept drifts based on a sliding window with different size at different times. To detect concept drifts, the sliding window is split into two unequally sized sub-windows and the mean difference between their examples' test results is used as the change indicator. This method is time expensive because its window gets larger and larger until a concept drift is found. The authors also decreased the time consumption by reducing the split times, but this decreased the accuracy. Du *et al.* [13] utilized information entropy over an adaptive sliding window to discover concept drifts, but the method cannot well address concept drifts with different speeds.

Another kind of detection method is based on the batch mode (see Section 4.2 for details). Klinkenberg and Joachims [26] employed support vector machines to detect concept drifts. This method artificially divides a data stream into batches/blocks, and uses the batch error rate to signal concept drifts. However, it also cannot detect concept drifts with different speeds, either.

A common issue of these above-mentioned detection methods, called the single-change-indicator-based methods here, is that they use only one statistic as the change indicator, and expect it to work once and for all. As a result, they cannot address concept drifts with different speeds consistently. Our method, which takes advantage of the ensemble of detection methods, rather than the ensemble of classifiers, aims to detect both abrupt and gradual drifts simultaneously. Therefore, *e*-Detector is quite different from these single-change-indicator-based methods or those ensembles of classifiers.

3. CONSTRUCTING ENSEMBLE OF DETECTORS

In this section, we first introduce the selective ensemble paradigm, then describe how to use it to build *e*-Detector, i.e. how to set the number of base detectors and which detector will be selected.

3.1. Selective ensemble paradigm

Although an ensemble of as many detectors as possible can help locate more concept drifts, indiscriminate usage of detectors is not the best choice for time-changing data streams. For example, suppose detector dm_1 outperforms dm_2 on abrupt drifts, while dm_2 is better than dm_1 on gradual drifts; then the ensemble of them will be more powerful than each of them since the ensemble performs well for detecting both abrupt and gradual drifts. However, if dm_1 is superior to dm_2 on both abrupt and gradual drifts, the ensemble of them will be less effective than dm_1.

A solution to this problem is the selective ensemble paradigm [19], which only chooses the most effective detectors. We use this strategy to build the *e*-Detector because: (1) there are so many types of data streams, but usually a single detection method only works well for one type of them; (2) for distinct streams and different periods, the speed of concept drifts are quite different but, unfortunately, a method can only handle concept drifts with a particular speed and has limited generalization capability; and (3) several detectors may use equivalent change indicators[1] and their combination may lead to performance decrease.

[1]Equivalent change indicators mean that two detectors uses similar change indicators. For more details we refer the reader to Section 3.2.1.

Based on the selective ensemble paradigm, before building an effective detection method, the following two problems should be addressed:

1. How many base detectors should be selected for the ensemble?
2. When choosing the base detectors, which selection rules will be used? In other words, which evaluation metric is more useful and powerful?

Both questions play key roles in building e-Detector, and we will answer them in the next two subsections, respectively.

3.2. Determining the number of base detectors

In ensemble learning, the higher the diversity among the base learners is, the better the ensemble is [27]. This is true for the ensemble of concept drift detectors as well, because base detectors with more diversity are more powerful in finding different concept drifts. So the issue of how to measure the diversity between two detectors is raised.

3.2.1. Diversity

To measure the diversity between two detection methods, we define two types of change indicators, i.e. the homogeneous change indicators and the heterogeneous change indicators.

DEFINITION 3.1 (Homogeneous/Heterogeneous change indicators). *Suppose there are m concept drift detection methods CdM_k $(k = 1, 2, \ldots, m)$, and s_i and s_j are the statistics of methods CdM_i $(i \in \{1, 2, \ldots, m\})$ and CdM_j $(j \in \{1, 2, \ldots, m\} \wedge j \neq i)$, separately. If s_i and s_j are equivalent in finding concept drifts, we say s_i and s_j are homogeneous change indicators; otherwise, they are heterogeneous change indicators.*

The statistic refers to the change indicator, which is a variable that a detection method uses to signal concept drifts. It is the only difference between two detectors because a detector here is wrapped with a classifier. For example, the error rate of DDM [9] and the accuracy of STEPD [12].

DEFINITION 3.2. *The diversity* $\mathrm{div}(CdM_i, CdM_j)$ *between detectors CdM_i and CdM_j is defined as follows:*

$$\mathrm{div}(CdM_i, CdM_j) = \begin{cases} 0 \textit{ if Condition } 1 \textit{ applies}; & (1) \\ 1 \textit{ if Condition } 2 \textit{ applies}. & (2) \end{cases}$$

Condition 1 : CdM_i and CdM_j have homogeneous change indicators; Condition 2 : CdM_i and CdM_j have heterogeneous change indicators.

This definition clearly states how to calculate the diversity between two detectors. For example, DDM uses the error rate as its change indicator, STEPD's change indicator is the accuracy, while the change indicator of EDDM is the interval between incorrect classifications. Since these statistics are all from the results of a classifier, the error rate is the ratio of the incorrect classifications to all examples, while the accuracy is the ratio of the correct classifications to all examples. It is easily known that error rate and accuracy perform equally in recognizing concept drifts. Based on the definition, we say that the error rate and accuracy are homogeneous change indicators, thus $\mathrm{div}(DDM, STEPD) = 0$. We further recognize that the interval of wrong classifications is different from the error rate or accuracy. Therefore, $\mathrm{div}(DDM, EDDM) = 1$ and $\mathrm{div}(EDDM, STEPD) = 1$.

DEFINITION 3.3 (Diversity vector of a detector). *For a detector CdM_i, the diversity vector consists of the diversity between itself and all the detectors, i.e.,*

$$\vec{V}_{CdM_i} = (\mathrm{div}(CdM_i, CdM_1), \ldots, \mathrm{div}(CdM_i, CdM_i), \ldots, \mathrm{div}(CdM_i, CdM_m)).$$

The *div* score is either 0 when two detectors are homogeneous, or 1 when they are heterogeneous. The diversity vector reveals the relationship between a detector and the rest of candidate detectors. With diversity vectors of all detectors, it is easy to decide which pair of candidates is homogeneous and which pair is heterogeneous.

3.2.2. Determination of the number of base detectors

With the help of the diversity vectors, we can cluster all candidate detectors into groups. Within each group, all detectors have homogeneous change indicators, and detectors from different groups are heterogeneous. Thus, we think detectors within one group are equivalent, while those in different groups are distinct, and consider the number of groups as the number of detectors. This means we only choose one detector from each group, then build e-Detector. This is valid because e-Detector integrates different detectors, which ensures that it can perform consistently.

Based on the above analysis, the number of base detectors can be determined by the following two-step procedure:

1. Generation of diversity vectors. For each candidate detector $CdM_i (i \in \{1, 2, \ldots, m\})$, we first calculate the diversity between itself and other detectors CdM_j $(j \in \{1, 2, \ldots, m\} \wedge j \neq i)$, then its diversity vector $\vec{V}_{CdM_i}$ is generated. Repeat this process until every candidate detector is covered, and then the diversity vectors of all the candidate detectors are obtained.
2. Determination of the number of base detectors. For each candidate detector CdM_i $(i \in \{1, 2, \ldots, m\})$, we compare its diversity vector $\vec{V}_{CdM_i}$ with those of CdM_j $(j \in \{1, 2, \ldots, m\} \wedge j \neq i)$, and group it and CdM_j into one cluster if $\mathrm{div}(CdM_i, CdM_j) = 0$; then for

the remaining detectors that are not clustered, we repeat this process until all candidate detectors are grouped. Finally, several clusters of heterogeneous detectors are obtained and the number of the clusters is the number of the base detectors.

After clustering, we clearly know how many types of distinct detectors there are; then we can easily get the number of base detectors. This process works cooperatively with the base detectors selection algorithm, thus its algorithmic description is integrated into Algorithm 1 (Part 1).

Algorithm 1 Base Detectors Selection (BDS)

```
Input:
    D - the set of reference data sets;
    CdM - the set of all candidate detectors.
Output:
    e-Detector.
    /* Part 1: Association rule mining */
1:  $s = 0$, e-Detector $\leftarrow \emptyset$; //$s$ is # of selected detectors
2:  for each $CdM_i \in CdM$ which has not been clustered do
3:      $s++$, $Clu_s \leftarrow \emptyset$;
4:      calculate $div(CdM_i, CdM_j)$ for each $CdM_j (j \neq i)$;
5:      $Clu_s = Clu_s \cup CdM_i$;
6:      if $div(CdM_i, CdM_j) == 0$ then
7:          $Clu_s = Clu_s \cup CdM_j$;
8:      end if
9:  end for
    /* Part 2: Selecting base detectors */
10: $CoF(min) \leftarrow \infty$; //$CoF(min)$ is the reference value
11: for each $Clu \in Clu_s$ do
12:     for each $CdM_i \in Clu$ do
13:         for each $D_j \in D$ do
14:             compute $Perr(CdM_i)_j$ and $Rdr(CdM_i)_j$;
                // Eqs. 4 and 6
15:         end for
16:         compute $CoF(CdM_i)$; // Eq. 7
17:         if $CoF(CdM_i) < CoF(min)$ then
18:             $CoF(min) = CoF(CdM_i)$;
19:             $temp = CdM_i$;
20:         end if
21:     end for
22:     e-Detector = e-Detector $\cup$ temp;
23: end for
24: return e-Detector;
```

3.3. Selecting base detectors

Based on the above process, it is easy to choose a base method if there is only one detector in a cluster. But if a cluster has more than one detector, how to choose the corresponding base detector? Next we concentrate on answering this question.

3.3.1. *The evaluation metric*

The prequential error rate ($Perr$) is usually used to evaluate a detector's performance [28], and the prequantial error rate at timestamp t was defined as follows:

$$Perr(t) = \begin{cases} p_t, & \text{if} \quad t = 1; \qquad (3) \\ Perr(t-1) + \dfrac{p_t - Perr(t-1)}{t} & \text{otherwise.} \qquad (4) \end{cases}$$

where, p_t is 0 if the current example is correctly predicted; otherwise it is 1. Obviously, a higher $Perr$ suggests a worse performance.

However, only using *Perr* cannot directly reveal a detector's detection ability, and so Baena-García *et al.* [10] used the number of declared drifts for evaluation. But this metric does not take into consideration the actual number of drifts in a data stream. It is natural to employ the effective detection rate (*Edr*) to make complement to that. Suppose a data stream contains T concept drifts, and a detector *CdM* obtains T' effective detections. Then the *Edr* of *CdM* is defined as follows:

$$Edr(CdM) = \frac{T'}{T}. \qquad (5)$$

The higher the Edr value, the better is the detector *CdM*.

Unfortunately, Edr is too difficult to obtain due to the following two problems. The first one is the detection delay. For a data set containing concept drifts, both empirical and experimental results have shown that no method can declare a concept drift immediately when it occurs, which means every method has a detection delay. In addition, no method guarantees to locate a drift on time [29]. As expected, as long as the acceptable delay is determined, the detection rate is easy to calculate. However, it is hard and impracticable to set a specific value to the acceptable delay, which is usually different under different conditions. As a result, we cannot get the number of drifts correctly detected by a method, i.e. the numerator of Equation (5).

The second difficulty is that it is hard to define gradual concept drifts for a data set, because there is a long transition between an old concept and a new one. We can consider that there are several abrupt drifts between intermediate concepts or only one gradual drift [7]. Thus we cannot obtain the actual number of drifts in a gradual drifting data stream, i.e. the denominator of Equation (5).

For abrupt drifts, we can address the above problems by introducing the *relative detection rate (Rdr)* to replace Edr. For a data stream containing T concept drifts, *CdM* declares T^* drifts; then the Rdr of *CdM* is defined as

$$Rdr(CdM) = \frac{|T^* - T|}{T}. \qquad (6)$$

Rdr denotes the detection ability of a detector, and it is easier to be calculated than Edr. A smaller Rdr value stands for a better detection ability.

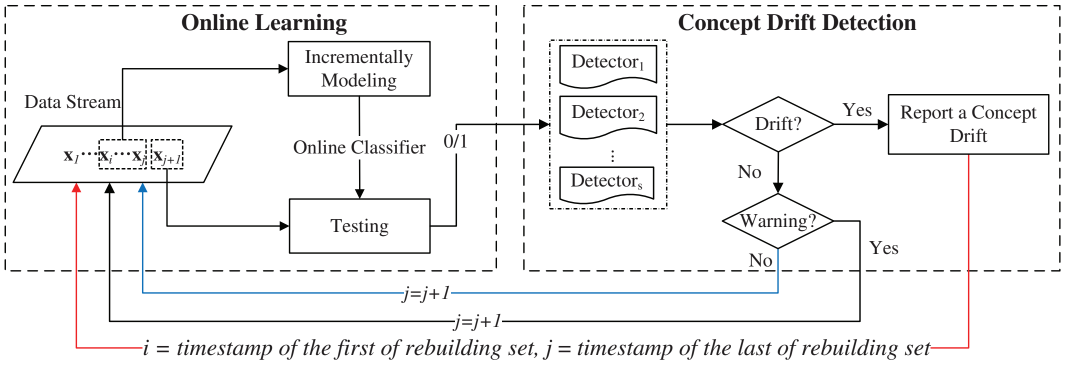

FIGURE 1. Framework of ensemble drift detection.

For gradual drifts, we cannot use Rdr instead of Edr because the denominator of Equation (5) is still unknown. By now, there are no other criteria that can be used. Therefore, we choose $Perr$ to evaluate a detector's capability to detect gradual drifts.

However, we still do not know which detector is the best because different methods' performances vary greatly if different evaluation criteria are adopted. Thus it is reasonable to integrate distinct criteria together. The *Coefficient of Failure (CoF)* of a detector is an integrated measure, and it is defined as follows:

$$CoF = \frac{\sum_{i=1}^{n_a} Perr_i \times Rdr_i}{n_a} + \frac{\sum_{j=1}^{n_g} Rerr_j}{n_g}. \qquad (7)$$

where Rdr_i and $Perr_i$ stand for a detector's relative detection rate and the prequential error rate on the i^{th} reference data set, respectively; n_a and n_g denote the numbers of reference data which contain abrupt drifts and gradual drifts, respectively. According to $Perr$ and Rdr, a smaller CoF value means better performance.

With CoF we can obtain the performance of a detector within each cluster we have generated in Section 3.2.2. Further, all detectors in one cluster are sorted according to their values of CoF on all reference data sets.

3.3.2. *Base detector selection algorithm*

When choosing base detectors, we first compute the CoF value of each candidate, and then cluster these candidates into groups according to their CoF values; finally, the representative of each group is selected and all the representatives constitute the base detectors' set. Let D be the set of reference data sets in which n_a are the reference data sets containing abrupt drifts and n_g are the reference data sets containing gradual drifts, then the specific selection procedure is described as follows:

1. Clustering candidate detectors. For each candidate detector $CdM_i(i \in \{1, 2, \ldots, m\})$, we first calculate $\text{div}(CdM_i, CdM_j)(j \in \{1, 2, \ldots, m\} \wedge j \neq i)$, and then CdM_i and CdM_j will be grouped into one cluster if $\text{div}(CdM_i, CdM_j) = 0$. After all detectors are covered, we have s clusters $Clu_i(i = 1, 2, \ldots, s \leq m)$ (see Section 3.2.2 for details).
2. Selecting base detectors. For each detector $CdM_i \in Clu_k(k \in \{1, 2, \ldots, s\})$ and each reference data set $D_j \in D(j \in \{1, 2, \ldots, n_a + n_g\})$, we compute $Perr(CdM_i)_j$ and $Rdr(CdM_i)_j$ if D_j contains abrupt drifts, or calculate $Perr(CdM_i)_j$ if D_j contains gradual drifts; once every D_j is processed, we calculate the $CoF(CdM_i)$ for CdM_i. After each detector $CdM_i \in Clu_k$ is covered, we select the detector with the smallest CoF as a base detector. Repeating this process for every cluster, we obtain all the base detectors for *e*-Detector.

This procedure selects detectors through both static (diversity measurement) and dynamic (CoF over reference data sets) analysis, which guarantees *e*-Detector's performance. The algorithmic presentation is contained in Algorithm 1. Note that the construction procedure of *e*-Detector goes before a detection task.

4. DETECTING CONCEPT DRIFTS BY *E*-DETECTOR

In this section, we first present the general framework of concept drift detection, and then introduce how to use *e*-Detector to discover concept drifts.

4.1. Overview

According to the methods introduced in Section 3, *e*-Detector is developed to locate both abrupt and gradual concept drifts. Figure 1 portrays the whole detection framework, which consists of two phases: the online learning and the concept drift detection.

At the online learning phase, stream examples[2] are continuously provided to build the online classifier. Old examples

[2]Here a data stream is denoted as an infinite set $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_i, y_i), \ldots\}$, where $(\mathbf{x}_i, y_i)$ means an example; $\mathbf{x}_i$ is an attribute vector and

$\{\mathbf{x}_i, \ldots, \mathbf{x}_j\}$ are employed to train the classifier, which is then tested on (used to predict the label of) the latest example $\mathbf{x}_{j+1}$. On the other hand, the classifier is updated according to the feedback from the concept drift detection phase. Concept drift detection is conducted via the ensemble method with the test results (information) provided by the online classifier. At the same time, the concept drift detection phase also returns relevant information to the online learning phase for updating the classifier.

4.2. Online learning

There are two major strategies for online learning from data streams: (1) the prequential mode (or the interleaved test-then-train mode), and (2) the batch/block mode (or the holdout mode). In the former one, examples are processed separately and successively [8, 9]; while in the latter mode, a data stream is intentionally split into continuous blocks and each block is treated integrally as a stationary set [30].

The prequential mode has three advantages: (1) no extra storage is needed for the examples; (2) it is capable of making sufficient use of the available examples and (3) it can be easily converted to batch mode [30]. So the prequential mode is adopted by most detection methods, i.e. DDM [9] and ADWIN [11]. In this paper, we also use this type of online learning.

Let $\{\mathbf{x}_i, \ldots, \mathbf{x}_j\}$ be the up-to-now example set of a data stream; the prequential online learning can be described as follows:

(i) At the beginning $i = j = 1$, and a classifier is built on the examples set $\{\mathbf{x}_i, \ldots, \mathbf{x}_j\}$. Then the next example $\mathbf{x}_{j+1}$ is used to test the classifier. Zero is generated if the label of $\mathbf{x}_{j+1}$ is correctly predicted[3], and 1 otherwise. After that, $\mathbf{x}_{j+1}$ becomes an old example and the online classifier will be updated according to the feedback from the concept drift detection phase.

(ii) If a warning feedback is received, $\mathbf{x}_{j+1}$ is added to the up-to-now example set and the online classifier is incrementally trained with this example set. After that, the latest example $\mathbf{x}_{j+2}$ is used as a tester. At the same time, $\mathbf{x}_{j+1}$ is stored to construct the rebuilding set, which consists of $\{\mathbf{x}_{j+1}\}$ by now. If the online learning continuously receives warning feedback, the rebuilding set will be extended. This extension will not stop unless a non-warning feedback is obtained.

(iii) If the feedback shows that there is no drift, $\mathbf{x}_{j+1}$ is also added to the up-to-now set which is further used to incrementally train the online classifier. Then $\mathbf{x}_{j+2}$ is used to test the classifier. Besides, the rebuilding set will be cleared if it is not empty.

(iv) If the feedback signals a concept drift, the online classifier is first discarded and then rebuilt according to the rebuilding set. The up-to-now example set is also first cleared and then renewed using the rebuilding set. That means $\mathbf{x}_i$ is set to the first example of the rebuilding set, and $\mathbf{x}_j$ is set to the last one. Likewise, $\mathbf{x}_{j+1}$ is used to test the classifier.

During the above process, the test results, a time series that is made up of 0 and 1, are produced. This time series is the input of the concept drift detection phase, which is also provided on the fly.

y_i is the class label. The $\mathbf{x}_i$ is the first example of each new concept, and it is changed whenever a concept drift is found.

[3]A correct prediction means an example's predicted label $\hat{y}_i$ equals its actual class label y_i; similarly, an incorrect prediction indicates that $\hat{y}_i \neq y_i$.

4.3. Concept drift detection

In this subsection, we first describe the detection paradigm, and then introduce the fusion rules. Finally, we present the details of employing these rules to detect concept drifts.

4.3.1. *Detection paradigm*

As explained in Section 3.2.1, every detector uses a statistic or variable as a change indicator to signal concept drifts. The change indicator is compared with a predefined threshold to determine the current state of the data stream. The predefined thresholds have two types, which are used for reporting warning and drift, respectively.

To describe the detection process, we take detector DDM as an example. At the stage of initialization, DDM calculates the value of change indicator (error rate) using the first 30 examples[4] of the time series. Then from the 31st example on, the error rate is incrementally updated as follows. Suppose the latest example is $\mathbf{x}_{j+1}$, and the error rate has been calculated from the test results of $\{\mathbf{x}_i, \ldots, \mathbf{x}_j\}$. The predicted value 0/1 of $\mathbf{x}_{j+1}$ is used to update the error rate, which means by now the error rate has been calculated according to the test results of $\{\mathbf{x}_i, \ldots, \mathbf{x}_{j+1}\}$. Depending on the relationship between the error rate and the predefined thresholds, there are three different cases: (1) if the error rate exceeds the warning threshold, DDM declares a warning at timestamp $j + 1$ and returns a warning signal to the online learning; (2) if the error rate exceeds the drift threshold, a concept drift is reported at current timestamp $j + 1$, and DDM informs the online learning of a concept drift feedback; otherwise, (3) DDM gives a feedback signaling there is no drift.

e-Detector is made up of several base detectors, each of which has its own warning and drift thresholds.

4.3.2. *Drift detection method*

e-Detector is a committee of base detectors; thus a fusion rule is required to consolidate detection results from different base detectors.

There are many fusion rules which can be used for an ensemble, such as the majority voting, the weighted voting,

[4]DDM assumes that there is no concept drift within the first 30 examples of a data stream [9].

and so on. However, both the majority voting and weighted voting are inappropriate for a concept drifting environment. The reason lies in the fact that different base detectors usually find a concept drift at different timesteps, and so it is hard to combine these different results. For example, if detector CdM_i locates a concept drift at time step t while CdM_j finds it at $t+3$, both the majority voting and weighted voting are unavailable because they cannot combine the concept drifts detected at different timestamps, t and $t+3$.

As we know, every detector has a detection delay when finding concept drifts. So, a good strategy is that if any base detector finds a concept drift, e-Detector declares it. This guarantees that e-Detector locates a concept drift as early as possible. We call this the *early-find-early-report* rule. One may argue that this fusion rule can incur more false alarms, but fortunately, our selective paradigm (see Section 3.1 for details) makes this increase of false alarms negligible. The reason lies in that we only select one representative detector from those with homogeneous change indicators, which avoids repetitive reports. Besides, we use CoF to evaluate all candidate detectors and those with poor identification ability will not participate in the detecting task.

Suppose the change indicator of every base detector has been updated by the predicted value 0/1 of the latest example $\mathbf{x}_{j+1}$; then the *early-find-early-report* rule can be presented as follows:

1. If one base detector finds a concept drift, e-Detector reports it at current time step $j+1$. Then e-Detector returns a concept drift feedback to the online learning, and all base detectors are reset. This makes e-Detector react as quickly as possible.
2. If one base detector reports a warning, e-Detector also reports it at time step $j+1$. At the same time, a warning feedback is sent to the online learning. This makes e-Detector able to store adequate examples (from time step $j+1$ to the drift time) for rebuilding the online classifier.
3. If all base detectors recognize that the data stream is stable, e-Detector is consistent with them and provides the online learning a non-drift signal.

Based on the fusion rule, the concept drift detection (CDD) algorithm is proposed here, and its algorithmic description is included in Algorithm 2.

Algorithm 2 Concept Drift Detection (CDD)

```
Input:
    $ds$ - to be detected data stream; $e$-Detector - selective ensemble of detectors.
1:  $RT \leftarrow \emptyset$;// $RT$ is the rebuilding set
2:  for the latest example $\mathbf{x}_{j+1}$ in $ds$ do
3:      $p_t$ = OnlineLearning($\mathbf{x}_{j+1}$); // predicting $\mathbf{x}_{j+1}$ by the online classifier
4:      for each base $CdM_i \in e$-Detector do
5:          update $CdM_i$ by $p_t$;
6:          if $CdM_i$'s change indicator exceeds the drift level then
7:              declare a concept drift at timestamp $j+1$;
8:              reset all base detectors;
9:              rebuild the online classifier from $RT$;
10:         else
11:             if $CdM_i$'s change indicator exceeds the warning level then
12:                 declare a warning at timestamp $j+1$;
13:                 $RT = RT \cup \mathbf{x}_{j+1}$;
14:             end if
15:         else
16:             if $RT \neq \emptyset$ then
17:                 $RT \leftarrow \emptyset$;
18:             end if
19:         end if
20:     end for
21:     signal the online classifier to update itself by $\mathbf{x}_{j+1}$.
22: end for
```

5. EXPERIMENTAL STUDY

In this section, we first present the data sets and the benchmark methods, then test all methods and analyze their performances.

5.1. Benchmark data sets

We use both artificial and real data sets, which contain different types of concept drifts such as abrupt drifts and gradual ones, to investigate e-Detector's performances under different conditions. Eight data sets are utilized, and we only present four of them here due to space limitations. The full data sets can be found in the supplement.

5.1.1. *Artificial Data Sets*

(i) Stagger [9]. Stagger contains abrupt drifts. Examples have three categorical attributes, and they are from *size={small, medium, large}*, *color={red, green, blue}* or *shape={square, circular, triangular}*. Concept drifts occur between the following two concepts: (1) *{size=small* $\wedge$ *color=red}* or *{size=medium* $\vee$ *large}*; (2) *{color=green* $\wedge$ *shape=circular}*.

(ii) Sine1g [10]. This is a very slow gradual concept drifting data set, and it is noise free. It has two relevant attributes, and each of them is uniformly distributed from range [0,1]. At the beginning, examples located below the curve $y = \sin(x)$ are set to positive, and this rule is reversed when a concept drift happens. The important point is that it has a long transition period between two concepts.

TABLE 1. Details of experimental data sets.

Name	Type	Speed of drifts	# attributes	# examples	Concept length	# drifts	Noise
Stagger	Artificial	Abrupt	3	3000	1000	2	No
Sine1g	Artificial	Gradual (very slow)	2	32 000	–	–	No
Elist	Real	Abrupt	913	1500	300	4	No
Elec2	Real	Gradual	6	45 312	–	–	No

5.1.2. Real Data Sets

1. Elist (Emailing list data) [25]. The Elist collects email messages from usenet posts existing in the twenty Newsgroup[5], and messages are about different topics that are continuously provided to users, who then assign them to spam or not according to their personal interests. There are 1500 examples and 913 attributes. Elist can be split into 5 periods, and each period can be viewed as a concept. The transformation between two periods suggests a concept drift. Table 2 presents the details, where ($\surd$) denotes an interesting email and ($\times$) is a spam.
2. Elec2 (Electricity Market data) [31]. This data set comes from Australian New South Wales Electricity Market. It is about the electricity price being adjusted according to the demand and supply in the market. The price is set every 5 min. This data set has 45312 examples dated from May 1996 to December 1998, and each example has six attributes: {day of week, the timestamp, NSW electricity demand, Vic electricity demand, scheduled electricity transfer between states, class label}, which spans a period of 30 min. Thus there are 48 examples in a day. The class label stands for the change of the price of a moving average over the past day.

Finally, the characters of each data set are contained in Table 1. It is clear that not only the number of examples and attributes but also the types of concept drifts are different. Thus these data sets are full of diversity.

5.2. Experimental method

5.2.1. Experimental setup

e-Detector is compared with the following five benchmark methods: DDM [9], EDDM [10], ADWIN [11], STEPD [12] and EnDDM [13]. DDM uses the online classifier's error rate as the change indicator. The error rate is updated as the data stream goes. A concept drift is reported once it exceeds a predefined threshold. EDDM's change indicator is the interval between incorrect predictions. A concept drift is found when it becomes in excess of the threshold. STEPD identifies concept drifts when the difference between the global accuracy and the local accuracy is significant. ADWIN recognizes concept drifts using two unequally sized sliding windows. If the *mean* between them exceeds the threshold, a concept drift is reported. EnDDM uses the entropy of a sliding window as the change indicator. A concept drift is found whenever the entropy reaches its maximum. These methods are well designed and have proved to be effective in identifying concept drifts. More importantly, they explicitly find concept drifts in data streams. We use the recommended parameters of each method in experiments.

[5]UCI machine learning repository.

TABLE 2. Emailing list data (Elist).

	0–300	300–600	600–900	900–1200	1200–1500
Medicine	$\surd$	$\times$	$\surd$	$\times$	$\surd$
Space	$\times$	$\surd$	$\times$	$\surd$	$\times$
Baseball	$\times$	$\surd$	$\times$	$\surd$	$\times$

e-Detector and those methods work with an online classifier; thus their performances are related to the classifier more or less. So four different classifiers are employed. We present the results of NNge (nearest-neighborhood with generalization) [32] here, and the results of three other classifiers, IB1 (nearest-neighborhood) [33], C4.5 (decision-tree based) [34] and SVM (kernel-based) [35], are contained in the supplement due to space limitations.

5.2.2. e-Detector Setup

The benchmark methods DDM, EDDM, ADWIN, STEPD and EnDDM are also used as candidates of the selective ensemble. That is, the candidate detector set $CdM = \{DDM, EDDM, ADWIN, STEPD, EnDDM\}$.

When building *e*-Detector, the *leave-one-out* strategy is employed to handle those data sets.

As DDM uses the error rate while STEPD uses the difference between two *accuracies* as change indicators, and the accuracy and the error rate have the same effect for evaluating classification, STEPD's change indicator is similar to that of DDM, which means they have homogeneous change indicators, i.e. $\mathrm{div}(DDM, STEPD) = 0$. The change indicator of ADWIN is the *mean* difference between two subwindows. As is known, the accuracy and the error rate are types of *mean*, so the change indicators of DDM, STEPD and ADWIN are all similar and homogeneous. EnDDM's change indicator is the entropy of a sliding window. Although the entropy can be viewed as a type of *mean*, it uses both accuracy and error rate, which means it makes use of information provided by both correct and incorrect classifications. Hence, it is not homogeneous to DDM, STEPD or ADWIN. For EDDM, the change indicator is the interval between incorrect classifications. Thus, it is heterogeneous from DDM, STEPD, ADWIN and EnDDM.

According to Definition 3.3, we obtain the diversity vector of every detector and include them in Table 3. After that, by using these diversity vectors, the Part 1 of algorithm BDS clusters these detectors into three groups, which are $Clu_1 = \{DDM, ADWIN, STEPD\}$, $Clu_2 = \{EDDM\}$ and $Clu_3 = \{EnDDM\}$. Obviously, the number of the base detectors is 3.

Next, we evaluate the detectors in each cluster based on *CoF*. Take the data set Stagger as an example; then the rest of the data sets are reference data sets based on which *CoF* is calculated. The results are contained in Table 4. Then, according to Part 2 of algorithm BDS, the detector with the smallest *CoF* in each cluster is chosen. As a result, e-Detector $= \{DDM, EDDM, EnDDM\}$ is formed.

Likewise, we can obtain e-Detector for different data sets. Table 5 contains the selection results for each data set. For all data sets, the BDS results are the same. We have explained in Section 2 that DDM uses the error rate that requires enough

TABLE 4. *CoF* of candidate detectors in each cluster.

	Clu_1			Clu_2	Clu_3
detector	DDM	ADWIN	STEPD	EDDM	EnDDM
$CoF(\cdot)$	0.80	2.86	6.48	1.07	0.16

TABLE 3. Diversity vector of every candidate detector.

	DDM	EDDM	ADWIN	STEPD	EnDDM
DDM	0	1	0	0	1
EDDM	1	0	1	1	1
ADWIN	0	1	0	0	1
STEPD	0	1	0	0	1
EnDDM	1	1	1	1	0

TABLE 5. BDS results for different data sets.

Data set	Base detectors
Stagger	DDM, EDDM, EnDDM
Sine1g	DDM, EDDM, EnDDM
Elist	DDM, EDDM, EnDDM
Elec2	DDM, EDDM, EnDDM

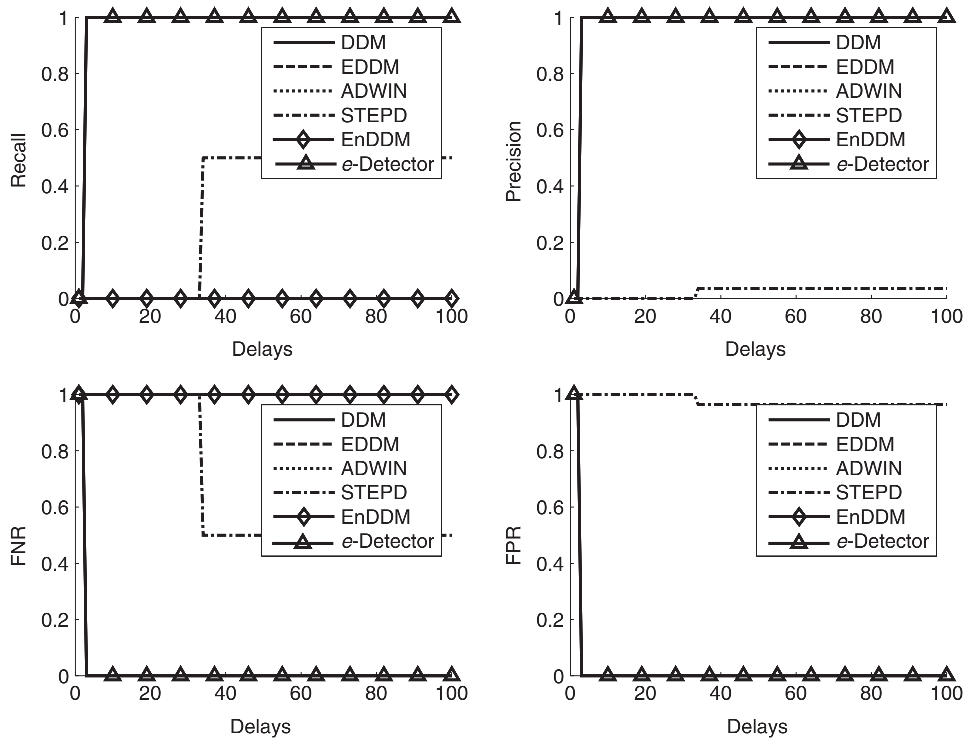

FIGURE 2. *Recall*, *Precision*, *FNR* and *FPR* comparisons on Stagger.

TABLE 6. Final *Perr* comparison on artificial data sets.

		Detection method					
Data set	Classifier	DDM	EDDM	ADWIN	STEPD	EnDDM	*e*-Detector
Stagger	NNge	**0.0027**	0.0050	0.0050	0.0173	0.0050	**0.0027**
Sine1g	NNge	0.2258	0.2010	0.1935	0.2079	0.1874	**0.1432**
AvgPERR		0.1143	0.1030	0.0996	0.1126	0.0962	**0.0730**

incorrectly predicted values to signal concept drifts, while abrupt drifts make the classifier generate a large number of incorrect predictions within a short period of time. Thus DDM is good at abrupt drift detection. EDDM's change indicator depends on the interval, not the number, of incorrect predictions, so it is good at gradual drift detection. EnDDM finds concept drifts using both correct and incorrect predictions, hence it performs generally well in detecting both abrupt and gradual drifts. Therefore, the results of BDS are consistent with our preliminary analysis.

5.2.3. *Evaluation criteria*

It is easy to define and detect an abrupt concept drift since its occurring time is easily known. It is true that an abrupt drift usually refers to a significant change in data steams. On the contrary, a gradual/slow concept drift is hard to be defined or detected because there is no substantial difference during the drifting period. Therefore, it is reasonable to employ different evaluation criteria for different types of concept drifts.

For an abrupt drift, it is easy to judge if a declared concept drift is correct. So it is easy to apply the traditional evaluation criteria. Firstly, we entail the following three definitions [36]:

(i) True positive (*TP*): a concept drift declared by a method is a true concept drift that has happened in a data stream. Also called *hit*.
(ii) False positive (*FP*): a concept drift is declared, but no concept drift has occurred in a data stream. Also known as *false alarm*.
(iii) False negative (*FN*): a concept drift in fact occurs in a data stream, but the method cannot detect it. Also called *miss* or *failure*.

According to the above definitions, we define the *Precision* and *Recall* as follows:

$$Precison = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN}. \tag{8}$$

The higher the values of *Precision* and *Recall*, the better the methods are.

The false positive rate (*FPR*) and the false negative rate (*FNR*) also can be defined as follows:

$$FPR = \frac{FP}{TP + FP}, \quad FNR = \frac{FN}{TP + FN}. \tag{9}$$

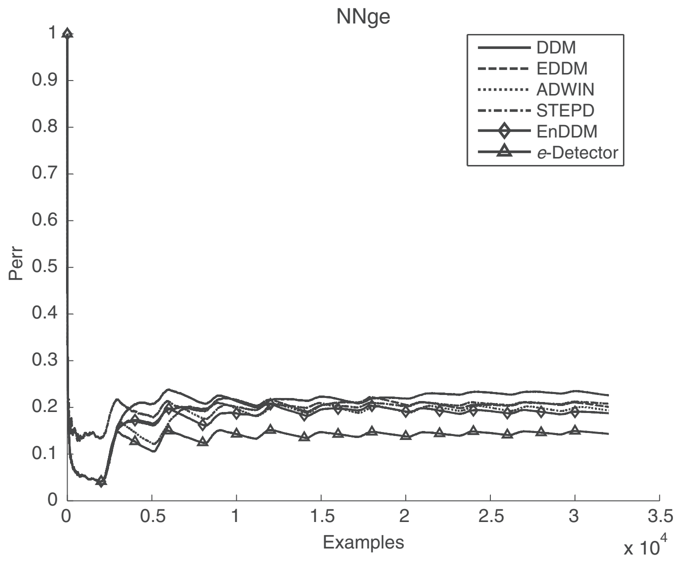

FIGURE 3. *Perr* comparison on Sine1g.

where *FPR* is also called the false alarm rate, and *FNR* denotes the miss detection rate or failure detection rate. For both *FPR* and *FNR*, a smaller score stands for a better performance.

As explained earlier, a detection method always has detection delay. Only when the delay threshold is determined, can the *Precision*, *Recall*, *FPR* and *FNR* be calculated. However, it is not easy to set a specific value for the threshold since different applications have different requirements in delay. Hence, in this paper, we intend to set a range to the delay threshold. We set the maximum threshold to 100, which means it is acceptable if a real concept drift is detected after at most 100 examples arrive. For every specific threshold within this range, we compute the *Precision*, *Recall*, *FPR* and *FNR*, respectively. By tuning the delay threshold, we portray the *Precision-delay*, *Recall-delay*, *FPR-delay* and *FNR-delay* curves, in which the horizontal axis denotes the value of delay and the vertical axis denotes the corresponding rate. From the definitions, we know that $Recall + FNR = 1$ and $Precision + FPR = 1$. Thus, the *FNR-delay* curve is the mirror figure of the *Recall-delay* curve across the vertical direction, and the *FPR-delay* curve is the mirror figure of the *Precision-delay* curve. On

TABLE 7. *Runtime* comparison on artificial data sets.

		Detection method					
Data set	Classifier	DDM	EDDM	ADWIN	STEPD	EnDDM	*e*-Detector
Stagger	NNge	1	1	1	1	1	1
Sine1g	NNge	15	15	11	9	14	**4**
AvgTime		8	8	6	5	7.5	**2.5**

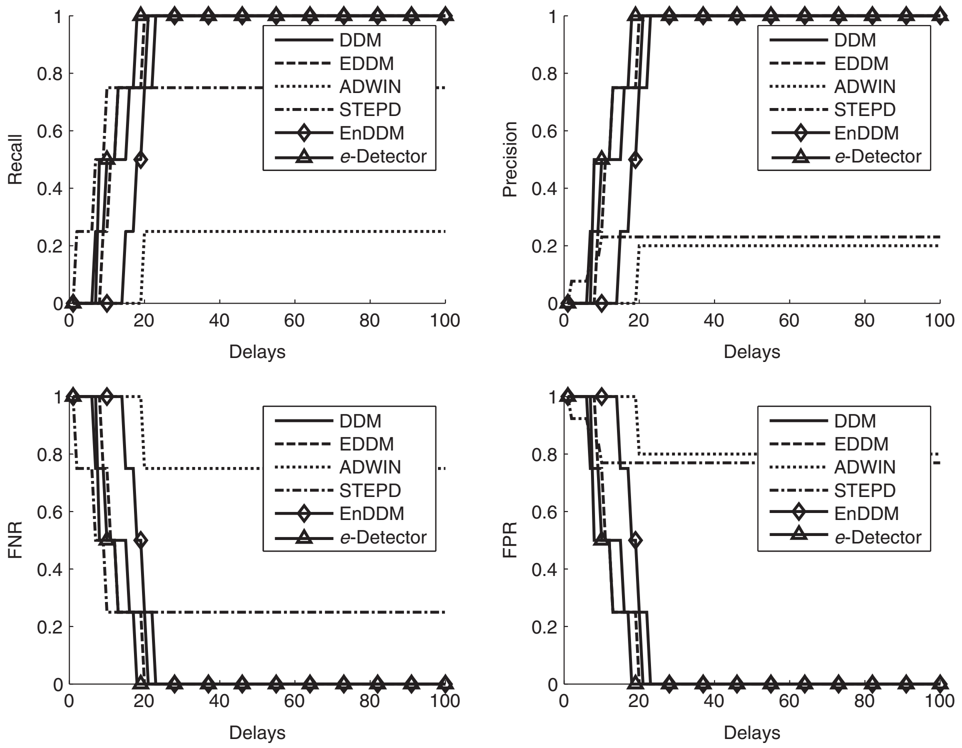

FIGURE 4. *Recall, precision, FNR* and *FPR* comparisons on Elist.

account of this, we concentrate on the results of *Recall* and *Precision*.

Perr and the *Runtime* are used as the evaluation criteria for both abrupt and gradual concept drifts, and a smaller score suggests a better performance. We are unable to calculate the *Precision*, *Recall*, *FPR* and *FNR* for a gradual concept drift, so these criteria are not used.

5.3. Results and analysis

In this subsection, we present the detection results of *e*-Detector and those benchmarks in detail. We analyze the performances of every method on different data sets, respectively. Finally, we also statistically test their performances.

5.3.1. *Performance on artificial data*

As stated in Section 5.1, we present the performance on data set Stagger and Sine1g using classifier NNge. The complete results on all data sets and classifiers are contained in supplement.

Figure 2 shows the detection results about Stagger. As expected, both the *Recall* and the *Precision* increase from left to right for all curves, and the *FNR* and the *FPR* decrease from left to right. The reason is that the number of *TP* increases with the increase in delay. From the figure we also observe that: (1) for *Recall*, *e*-Detector and DDM get the best value with very few delays, while the rest of benchmarks cannot. This means that *e*-Detector and DDM successfully find two concept drifts within Stagger, but, the other detectors fail to do so; (2) for *Precision*, *e*-Detector and DDM also obtain the best score among all methods. These positive results suggest that *e*-Detector is as good as DDM and outperforms others in terms of both *Recall* and *Precision*.

Table 6 presents the comparison results of the final *Perr* of Stagger and Sine1g, where the best value is in boldface. We observe that: (1) *e*-Detector performs better than those

TABLE 8. Final *Perr* comparison on real data sets.

		Detection method					
Data set	Classifier	DDM	EDDM	ADWIN	STEPD	EnDDM	*e*-Detector
Elist	NNge	0.1413	0.1473	0.3907	0.2407	0.1460	**0.1387**
Elec2	NNge	0.1298	0.1011	0.1798	**0.0961**	0.1490	0.0962
AvgPERR		0.1356	0.1242	0.2853	0.1684	0.1475	**0.1175**

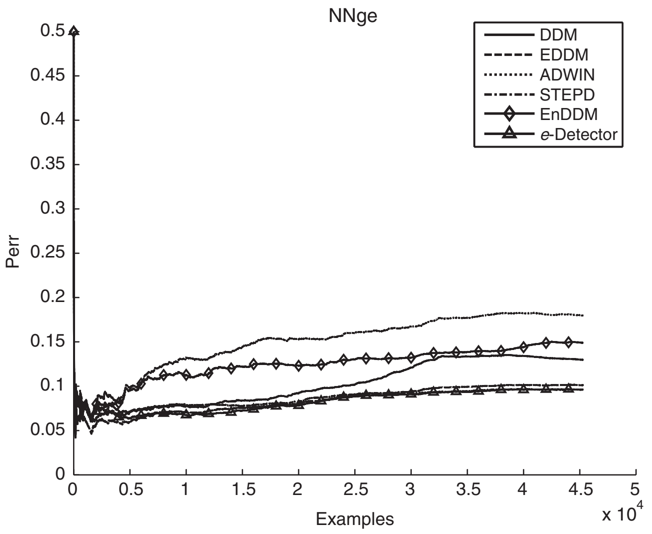

FIGURE 5. *Perr* comparison on Elec2.

benchmarks on both abrupt drifts and gradual drifts. The superiority is quite obvious for gradual drift; (2) *e*-Detector has the smallest average *Perr* meaning so that it outperforms those methods in terms of generalization ability. Being unable to use the *Recall* and *Precision* on gradual drifts, we portray the curve of *Perr* vs. stream examples of Sine1g in Fig. 3 here to present different methods' performance. It is clear that *e*-Detector's *Perr* curve is below those benchmarks, which further demonstrates the positive performance of *e*-Detector for gradual concept drift detection.

The *Runtime* comparison of the six detection methods are shown in Table 7, where the least time is reported in boldface. We observe that (1) all methods perform similarly on Stagger, which is because this data set is short; (2) *e*-Detector takes less time than all single-change-indicator-based methods on Sine1g. So its average time consumption is also the smallest. This reveals that *e*-Detector outperforms those benchmarks in terms of *Runtime* on artificial data sets.

5.3.2. *Performance on real data*

In this experiment, we use two real-world data sets Elist and Elec2, one containing abrupt concept drifts and the other showing gradual changing, to evaluate *e*-Detector.

Figure 4 shows the results of the six detection methods on Elist. We observe that: (1) for *Recall*, *e*-Detector is the first one to achieve the best score, which means it performs better than those benchmarks; (2) for *Precision*, *e*-Detector also wins out. This suggests that *e*-Detector can find all concept drifts with the least delay, which reveals that it is the best detector.

Table 8 gives the final *Perr* on Elist and Elec2, and the best value is in boldface. We observe that: (1) *e*-Detector performs better than other methods on Elist; (2) *e*-Detector outperforms those methods except STEPD on Elec2. But the difference between it ($Perr = 0.0962$) and STEPD (the best performer, $Perr = 0.0961$) is very insignificant. This can also be observed from Fig. 5 in which *e*-Detector's curves are as low as that of STEPD. This reveals that, in terms of *Perr*, *e*-Detector's performance is better than those benchmarks on both real data sets.

The *Runtime* of every method on two real data sets are contained in Table 9. We observe that: (1) on Elist, *e*-Detector's time consumption is smaller than that of DDM, EnDDM and ADWIN, and is inferior to that of STEPD and EDDM. But STEPD has more false alarms. (2) on Elec2, *e*-Detector uses less time than STEPD and EDDM. At the same time, *e*-Detector has better *Perr* than them. The results suggest that *e*-Detector obtains the best detection results at the cost of time. Fortunately, the increase of time consumption is insignificant.

5.3.3. *Analysis and discussion*

All the above results show that *e*-Detector is more effective than those single-change-indicator-based benchmarks. In this subsection, we statistically verify this conclusion via the Friedman test followed by the Nemenyi's *post hoc* test if applicable [37].

Table 10 contains the results of Friedman test. The null hypotheses are that every criterion is equivalent for each of the six drift detection methods. All the *p* values are smaller than 0.05, which is an evidence to reject the null hypotheses, suggesting that all methods are significantly different. Thus we conduct the Nemenyi test to figure out which one is the best.

Firstly, we conduct the Nemenyi test on the *Recall* and *FNR* – they are consistent with each other in evaluating methods – of the different detection methods. Figure 6 and 7 show the results. In the figures, we connect *e*-Detector and a detection method if there is no significant difference between them. In

TABLE 9. *Runtime* comparison on real data sets.

Data set	Classifier	DDM	EDDM	ADWIN	STEPD	EnDDM	*e*-Detector
		Detection method					
Elist	NNge	61	55	493	**27**	64	68
Elec2	NNge	28	11	30	11	20	**10**
AvgTime		44.5	33	261.5	**19**	42	39

TABLE 10. *p* value of Friedman test.

Classifier	*Recall*	*FNR*	*Precision*	*FPR*	*Perr*
NNge	0.00	0.00	0.00	0.00	0.0081

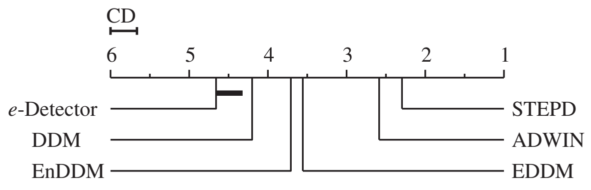

FIGURE 6. *Recall* comparison of the six detection algorithms against each other with the Nemenyi test.

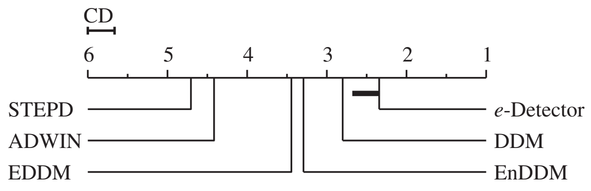

FIGURE 7. *FNR* comparison of the six detection algorithms against each other with the Nemenyi test.

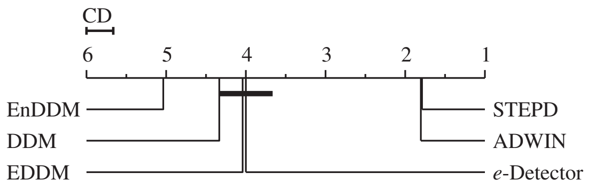

FIGURE 8. *Precision* comparison of the six detection algorithms against each other with the Nemenyi test.

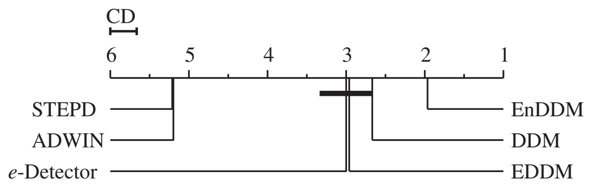

FIGURE 9. *FPR* comparison of the six detection algorithms against each other with the Nemenyi test.

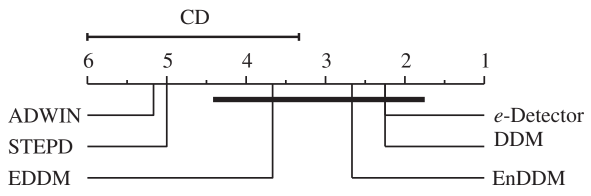

FIGURE 10. *Perr* comparison of the six detection algorithms against each other with the Nemenyi test.

both figures, we observe that the *Recall* and *FNR* of *e*-Detector are all statistically better than those of other detection methods.

From Fig. 8 and 9, we observe that both the *Precision* and the *FPR* of NNge with *e*-Detector are statistically better than those with ADWIN or STEPD. But there is no consistent evidence to indicate statistical differences between *e*-Detector, DDM and EDDM, respectively.

From Fig. 10, we observe that the *Perr* of NNge with *e*-Detector is statistically better than those of NNge with ADWIN or STEPD. But there is no consistent evidence to indicate statistical *Perr* differences between *e*-Detector, DDM, EDDM and EnDDM, respectively.

From the above results, we know that *e*-Detector performs quite well for different data sets on different evaluation criteria. For the purpose of exploring the whole performance of every method, i.e., which algorithms are more suitable for which types of concept drifts, we rank the six detection methods according to their performances on data streams with different types of concept drifts. Table 11 shows the results. From this table, we observe that

1. for data sets with abrupt drifts, *e*-Detector gets the first rank and should be the undisputed first choice. Besides, DDM can be a good alternative;
2. for data sets with gradual drifts, *e*-Detector gets the first rank again, and is the undisputed first choice. Besides, STEPD can be a good alternative.

The above analysis verifies that *e*-Detector performs very well on both abrupt drifts and gradual drifts. There are two reasons. The first one is that it combines the merits of its base detectors, which makes it find concept drifts more accurately;

TABLE 11. Ranks of detection methods on all data sets ('–' means this criterion does not apply).

	Data sets with abrupt drifts						Data sets with gradual drifts					
	DDM	EDDM	ADWIN	STEPD	EnDDM	*e*-Detector	DDM	EDDM	ADWIN	STEPD	EnDDM	*e*-Detector
Recall	3	2	6	5	4	1	–	–	–	–	–	–
Precision	2	3	6	4	4	1	–	–	–	–	–	–
Perr	2	4	5	5	3	1	5	3	4	2	3	1
Runtime	3	2	6	1	4	5	6	3	4	2	4	1
Sum	10	11	23	15	15	8	11	6	8	4	7	2
Rank	2	3	5	4	4	1	6	3	5	2	4	1

the second is its *early-find-early-report* rule, which allows it to discover concept drifts more timely.

6. CONCLUSION

In this paper, we have proposed a novel approach, e-Detector, using the selective ensemble strategy to explicitly detect concept drifts.

The proposed method is different from those previously published in the following aspects: (1) e-Detector integrates different detection methods (we call them detectors) to find concept drifts with different speeds, hence it is able to detect both abrupt and gradual drifts simultaneously; (2) e-Detector uses selective ensemble strategy, hence it can find concept drifts that cannot be detected by single-change-indicator-based methods; (3) e-Detector employs an *early-find-early-report* rule, and so it reacts faster to concept drifts than benchmarks. In addition to these differences, we also introduce the selection criterion CoF (Coefficient of Failure), which helps construct a well-designed and effective e-Detector.

We have compared e-Detector with four well-known drift detection methods, i.e. DDM, EDDM , ADWIN and STEPD, and a newly published one EnDDM, on eight publicly available data sets which contain not only artificial and real data sets but also abrupt concept drifts and gradual ones. The experimental results show that: (1) e-Detector performs consistently better than those benchmarks for both abrupt concept drifts and gradual ones, no matter whether they are artificial data or real data; (2) for abrupt concept drifts, e-Detector significantly improves the *Recall*, i.e. it reduces *FNR*, of those single-change-indicator-based methods without significantly decreasing (increasing) the *Precision (FPR)*. Actually, its *Precision* gets better for Stagger and Sine1 with IB1; (3) e-Detector also performs better than other detection methods in terms of *Perr* though the improvement is not statistically significant; 4) moreover, e-Detector has strong generalization and self-optimization capability. This is quite promising and encouraging, indicating that our selective ensemble detector has considerable potential in concept drift detection.

ACKNOWLEDGEMENTS

The authors would like to thank all the editors and reviewers for their insightful comments and kindly encouragement. The authors would also like to thank Yuanjiang Du for his help regarding the language and grammar.

FUNDING

This work was supported by the National Natural Science Foundation of China (No. 61373046).

REFERENCES

[1] Aggarwal, C. (2007) *An Introduction to Data Streams*. Springer, Berlin.

[2] Aggarwal, C. (2007) *A Survey of Classification Methods in Data Streams*. Springer, Berlin.

[3] Babcock, B., Babu, S., Datar, M., Motwani, R. and Widom, J. (2002) Models and Issues in Data Stream Systems. *Proceedings of PODS 02*, Madison, WI, June 3–5, pp. 1–16. ACM, New York.

[4] Widmer, G. and Kubat, M. (1996) Learning in the presence of concept drift and hidden contexts. *Mach. Learn.*, **23**, 69–101.

[5] TCD-CS-2004-15 (2004) The problem of concept drift: definitions and related work. Trinity College Dublin, Dublin, Ireland.

[6] Gama, J. and Castillo, G. (2006) Learning with Local Drift Detection. *Proceedings of ADMA 06*, Xi'an, China, August 14–16, pp. 42–55. Springer, Berlin.

[7] Minku, L., White, A. and Yao, X. (2010) The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Trans. Knowl. Data Eng.*, **22**, 730–742.

[8] Minku, L.L. and Yao, X. (2012) Ddd: A new ensemble approach for dealing with concept drift. *IEEE Trans. Knowl. Data Eng.*, **24**, 619–633.

[9] Gama, J., Medas, P., Castillo, G. and Rodrigues, P. (2004) Learning with Drift Detection. *Proceedings of SBIA 04*, Sao Luis, Maranhao, September 29–October, pp. 286–295. Springer, Berlin.

[10] Baena-García, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavalda, R. and Morales-Bueno, R. (2006) Early Drift Detection Method. *Proceedings of IWKDD 06*, Philadelphia, PA, August 20–23, pp. 77–86. ACM, New York.

[11] Bifet, A. and Gavaldà, R. (2007) Learning from Time-Changing Data with Adaptive Windowing. *Proceedings of SDM 07*, Minneapolis, MN, April 26–28, pp. 443–448. SIAM, PA.

[12] Nishida, K. and Yamauchi, K. (2007) Detecting Concept Drift Using Statistical Testing. *Proceedings of DS 07*, Omaha, Nebraska, October 28–31, pp. 264–269. Springer, Berlin.

[13] Du, L., Song, Q. and Jia, X. (2014) Detecting concept drift: an information entropy based method using an adaptive sliding window. *Intell. Data Anal.*, **18**, 337–364.

[14] Opitz, D. and Maclin, R. (1999) Popular ensemble methods: an empirical study. *J. Artif. Intell. Res.*, **11**, 169–198.

[15] Polikar, R. (2006) Ensemble based systems in decision making. *IEEE Circuits Syst. Mag.*, **6**, 21–45.

[16] Rokach, L. (2010) Ensemble-based classifiers. *Artif. Intell. Rev.*, **33**, 1–39.

[17] Xiao, J., He, C., Jiang, X. and Liu, D. (2010) A dynamic classifier ensemble selection approach for noise data. *Inf. Sci.*, **180**, 3402–3421.

[18] Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R. and Gavaldà, R. (2009) New Ensemble Methods for Evolving Data Streams. *Proceedings of KDD 09*, Paris, France, June 28–July 1, pp. 139–148. ACM, New York.

[19] Zhou, Z., Wu, J. and Tang, W. (2002) Ensembling neural networks: many could be better than all. *Artif. Intell.*, **137**, 239–263.

[20] Street, W. and Kim, Y. (2001) A Streaming Ensemble Algorithm (SEA) for Large-Scale Classification. *Proceedings of KDD 01*, San Francisco, CA, August 26–29, pp. 377–382. ACM, New York.

[21] Fern, A. and Givan, R. (2003) Online ensemble learning: An empirical study. *Mach. Learn.*, **53**, 71–109.

[22] Wang, H., Fan, W., Yu, P. and Han, J. (2003) Mining Concept-Drifting Data Streams Using Ensemble Classifiers. *Proceedings of KDD 03*, Washington, DC, August 24–27, pp. 226–235. ACM, New York.

[23] Kolter, J. and Maloof, M. (2003) Dynamic Weighted Majority: A New Ensemble Method for Tracking Concept Drift. *Proceedings of ICDM 03*, Melbourne, Florida, December 19–22, pp. 123–130. IEEE Computer Society, Washington D.C.

[24] Kolter, J. and Maloof, M. (2007) Dynamic weighted majority: an ensemble method for drifting concepts. *J. Mach. Learn. Res.*, **8**, 2755–2790.

[25] Katakis, I., Tsoumakas, G. and Vlahavas, I. (2010) Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowl. Inf. Syst.*, **22**, 371–391.

[26] Klinkenberg, R. and Joachims, T. (2000) Detecting Concept Drift with Support Vector Machines. *Proceedings of ICML 00*, Stanford, CA, USA, June 29–July 2, pp. 487–494. Morgan Kaufmann, San Francisco.

[27] Tang, E., Suganthan, P. and Yao, X. (2006) An analysis of diversity measures. *Mach. Learn.*, **65**, 247–271.

[28] Philip Dawid, A. and Vovk, V. (1999) Prequential probability: principles and properties. *Bernoulli*, **5**, 125–162.

[29] Kifer, D., Ben-David, S. and Gehrke, J. (2004) Detecting Change in Data Streams. *Proceedings of VLDB 04*, Toronto, Ontario, August 31–September 3, pp. 180–191. Morgan Kaufmann, San Francisco.

[30] Bifet, A. and Kirkby, R. (2009) *Data Stream Mining: A Practical Approach*. University of Waikato, Hamilton, New Zealand.

[31] UNSW-CSE-TR-9905 (1999) Splice-2 comparative evaluation: electricity pricing. The University of New South Wales, Sydney, Australia.

[32] Martin, B. (1995) *Instance-Based Learning: Nearest Neighbor with Generalization*. University of Waikato, Hamilton, New Zealand.

[33] Aha, D., Kibler, D. and Albert, M. (1991) Instance-based learning algorithms. *Mach. Learn.*, **6**, 37–66.

[34] Quinlan, J. (1993) *C4. 5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco.

[35] Joachims, T. (1998) Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Proceedings of ECML 98*, Chemnitz, Germany, April 21–23, pp. 137–142. Springer, Berlin.

[36] Nikovski, D. and Jain, A. (2010) Fast adaptive algorithms for abrupt change detection. *Mach. Learn.*, **79**, 283–306.

[37] Demsar, J. (2006) Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, **7**, 1–30.