Learning with Concept Drift Detection based on Sub-concepts from k Time Sub Windows

Li Liu[1], Nathalie Japkowicz[2], Dan Tao[3], Zhen Liu[4]

[1] School of Information Science and Technology, Huizhou University, China
[2] Department of Computer Science, American University, Washington DC, USA
[3] School of Electronic and Information Engineering, Beijing Jiaotong University, China
[4] College of Medical Information Engineering, Guangdong Pharmaceutical University, China
liuli@hzu.edu.cn, japkowic@american.edu, dtao@bjtu.edu.cn, liu.zhen@gdpu.edu.cn

Abstract

Concept drift detection has attracted much interest recently, due to its pervasive nature in the massive amount of streaming data available for analysis. Traditional concept drift detection methods, based on the monitoring performance of a base learner on a whole time window of data stream, are not sensitive enough to sub-concept drifts and discover them late or not at all. This is because, when aggregated together, the sub-concepts that form a concept are not precisely described. To solve this problem, we propose the *k*TSW (*k* Time Sub-concepts Window) based framework that divides instances from a whole time window into *k* sub-concept windows, and then builds a drift monitor for each sub-concept window. Once a sub-concept window's instances have experienced a concept drift, we update the learned model. We propose three schemes with different base learner numbers for our framework. Each of the schemes takes advantage of a different degree of sub-concept knowledge. Two real data sets are used to verify the validity of our method in data stream classification. Experimental results show that our method is able to obtain higher accuracy and recall than methods based on a whole time window.

Keywords: Concept drift detection, Sub concept, Data stream

1 Introduction

In recent years, due to the development of intelligent computer networks, massive amounts of streaming data have been continuously generated [1-2] e.g., from smart mobile phone, social networks, sensor networks etc. Due to these innovations, data stream mining has received a lot of attention. An important aspect of data stream mining is that the underlying data distribution changes over time. This is known as concept drift. The performance of learning models will decrease when a concept drift occurs, so we need methods that adapt to these changes. Many systems were designed to adapt to concept drift [3-4]. A common method consists of performing concept drift detection and updating the model or retraining it with new data when a concept drift has been detected.

In supervised online learning, most concept drift detection algorithms [5-8] are based on the evaluation of the learner's performance. These algorithms monitor the learner's performance, such as accuracy, and when that performance decreases below a given threshold, it is assumed that a drift may have occurred. Given that data streams are unlimited and that these algorithms only calculate statistics on the performance of a subset of the data stream, gathering data batches from data streams as time progresses is critical [8]. At present, using a time window is the most common method, and algorithms based on this method use all the data from the time window to calculate their statistics. We call these algorithms ***W****hole* ***T****ime* ***W****indow* (WTW) based methods in this paper. The drawback of WTW based methods is that they are not always sensitive to certain more subtle concept drifts that may occur only on the sub-concept of the overall data stream. We may be able to improve concept drift detection accuracy if we find a way to fully use sub-concept knowledge.

For example, assuming that there exists three sub concepts from a data stream, and the instances sequences are shown in Figure 1. We highlight them as yellow, green and blue blocks respectively to indicate different sub concepts, i.e., the first instance and the fifth instance are represented with same yellow color, which means that they belong to same sub concept. We also assume that a drift detector monitors the error rate of the classification model, and the number on each block is the number of instances that the drift detector has observed. Let's assume that at time window *t*, sub concept 3 changed suddenly, from blue to red, and the classification model cannot identify the drifted sub concept (red color). Assuming that the error rate up to the 9th instance at time window *t*-1 in the data stream is

*Corresponding Author: Li Liu; E-mail:liuli@hzu.edu.cn
DOI: 10.3966/160792642020032102024

0 and that the prediction of red at time t will be false, then the next error rates will be 0.1, 0.1, 0.1, 0.1, 0.14, 0.14, 0.17..., i.e. the error rate obtained at t is 0+(1-0)/10=0.1, and, more generally, the error rates can be computed using formula (1).

$$p\_error = p\_error + (pre - p\_error) / num \quad \textbf{(1)}$$

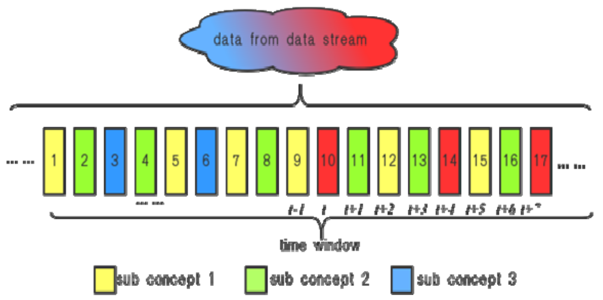


Figure 1. Instances from data stream's time window

The value of pre is 1 or 0, where 1 indicates that the prediction of the classification model is false. The value of num is the number of instances that the drift detector monitor has observed to date. We can see that the change in error rate is slow due to the fact that it combines changes in all three sub concepts. In certain cases, this change may even be ignored by the concept drift detector since, relatively speaking, it is quite small.

To solve this problem, we propose a method to monitor the classification model's performance based on k time sub-concept windows (kTSW). If we have knowledge of sub-concepts, we can use three drift detectors to monitor the classification model's performance on instances from each time sub-concept window. Each concept drift detector monitors a different time sub-concept window. As shown in Figure 2, in our illustration, there are three time sub-concept windows corresponding to the three sub-concepts. One concept drift detector will be built from each of these windows. In our illustration, the 3rd drift monitor only receives data prediction results (true or false) from sub concept 3, so the error rate, computed with formula (1), on sub concept 3 data is 0.3, 0.5, 0.6..., which is a more significant difference than the one observed on the entire time window (the WTW based method). Therefore the sub-concept based drift detector can detect concept drifts earlier, if we can find a method that divides one concept (one time window) into k sub concept (sub time window) effectively.

Rather than using the simple error rate method described previously to illustrate our idea, the kTSW based method was combined with more powerful concept drift algorithms, such as DDM [5], EDDM [6], HDDM [7]etc. We tested kTSW on real world data sets, and compared its performance to that of the WTW based method. Our experimental results show that the kTSW based method obtains higher performance than the WTW based methods. The main contributions of this paper are:

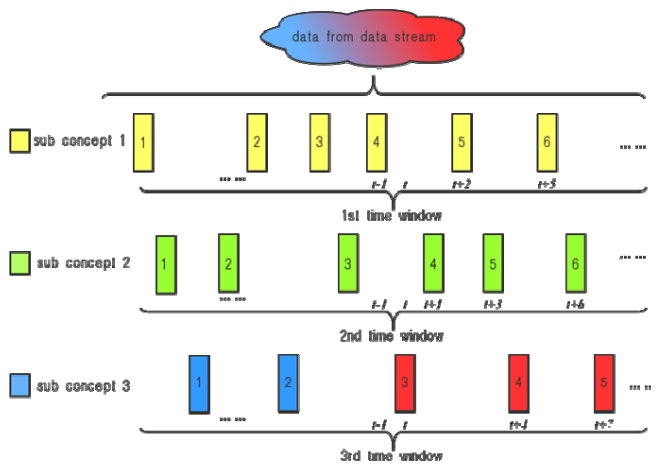


Figure 2. Instances from data stream's k time sub-concept window

We propose a concept drift detection method based on kTSW which can be used as a front-end to error rate based concept drift detection algorithms, such as DDM, EDDM etc.

Based on [9]and [10], we propose three schemes to implement our kTSW based concept drift detection framework, and each scheme makes use of a different degree of knowledge about the sub-concept structure of the data.

We test our method on two real word data sets : ELEC2 data set [11] and mobile traffic data set [12]. The experimental results show that our three schemes for kTSW based concept drift detection outperform WTW based concept drift detection.

2 Error Based Drift Detection

Many concept drift detection methods have been developed. Lu et al. [13] have classified these algorithms into three categories, including error rate-based drift detection, data distribution-based drift detection and multiple hypothesis test drift detection. Our work is related to the first category of algorithms that monitor the online error rate of a base decision model. If the change of error rate is statistically significant and reaches the threshold of drift, then a learning process for the new decision model will be triggered.

Since data streams are unlimited, it is impossible to compute test statistics or train a model over the entire data stream. Most strategies use time windows to compute the test statistics over a subset from a specific time interval of the data stream and select the sub data set for training. Taking the Drift Detection Method (DDM) [5] as an example, we explain how it uses time windows to detect a concept drift. For each point i in the data stream, DDM tracks the online error rate p_i (the probability of observing a False decision) of the decision model and standard deviation s_i which is given by $s_i = \sqrt{p_i(1-p_i)/i}$. DDM manages two registers (p_{min} and s_{min}) during the time window period. These values are updated when $p_i + s_i < p_{min} + s_{min}$. It defines the following levels and the conditions associated with that level are triggered:

- Warning level: $p_i + s_i \geq p_{min} + 2*s_{min}$, the time point is denoted as t_w.
- Drift level: $p_i + s_i \geq p_{min} + 3*s_{min}$, the time point is denoted as t_d.

When the sum of error rates and standard deviations reach the drift level, a new decision model is induced using only the subset of the data stream collected from t_w to t_d. The values for p_{min} and s_{min} are reset and a new time window begins. DDM calculates the test statistics with all the examples contained in a time window of the data stream. It does not calculate the test statistics separately for each sub-concepts. Error rate based drift detection algorithms that use the same windowing method as DDM are the Early Drift Detection Method (EDDM), Hoeffding's inequality based Drift Detection Method (HDDM), EWMA for Concept Drift Detection (ECDD), ADaptive WINdowing (ADWIN) and Fuzzy Windowing Drift Detection Method (FW-DDM).

EDDM [6] uses the distance-error-rate of the base learner to identify whether a drift has occurred. EDDM improved upon DDM by introducing slow gradual changes. HDDM [7] uses Hoeffding's inequality to identify the region of a concept drift. ECDD [14] uses an Exponentially Weighted Moving Average (EWMA) chart to monitor the error rate of the base learner. ADWIN [15] can recompute the size of the time window online according to the variation of error rate observed from the data in the window itself. FW-DD [8] assigns different grades to instances from a time window, and calculates the test statistics with the instances' grades. In contrast to all these algorithms, our *k*TSW based method divides the whole time window into *k* sub-concept windows based on the sub-concept knowledge. The instances from each time sub-concept window belong to the same sub concept. Our method then calculates test statistics over instances from each time sub-concept window separately.

3 Problem Definitions and Description

We define a data stream as a sequence of data objects with time stamps $DS = \{x_{t_1}, x_{t_2}, \ldots, x_{t_i}, \ldots\}$, where $t_i < t_{i+1}$ for all i. x_{t_i} is the data object at time point t_i, and $x_{t_i} \cdot \mathbf{X} = \{x_1, x_2, \ldots, x_c\}$, where **X** is the set of features or covariates. For classification learning, each data object x_i has a class label $y_i \cdot \mathbf{Y} = \{y_1, y_2, \ldots, y_c\}$. In this case, data streams are defined over $\mathbf{X} \cup \mathbf{Y}$, which represents the joint distribution **X** and **Y**, and are denoted as a sequence of $<x_t, y_t>$.

Definition 1: concept= P(X, Y)

A concept is defined as the joint probability distribution P(**X**, **Y**) over objects and class labels [16-17].

Definition 2 (concept drift): $\boldsymbol{P}_{t_i}(\mathbf{X},\mathbf{Y}) \neq \boldsymbol{P}_{t_j}(\mathbf{X},\mathbf{Y})$.

Data streams are usually non-stationary, which implies that their underlying distribution can change dynamically over time. This is known as concept drift [16-17]. Formally the joint probability distribution $P_{t_i}(\mathbf{X},\mathbf{Y})$ is not equal to $P_{t_j}(\mathbf{X},\mathbf{Y})$, where $t_i \neq t_j$ when concept drift happens. This is a kind of WTW based concept drift.

Definition 3: $\mathbf{O} = \bigcup_{i=1}^{k} O_i$.

We assume that there are *k* sub concepts. And the instances from the *i*th sub-concept are denoted by $\mathbf{O}_i$. And all $\mathbf{O}_i$ compose the all instances in a time window.

Definition 4: sub-concept=*P*(O*i*)

The concept over a time window is denoted as P(**O**)=P(**X**, **Y**) and the concept from a time sub-concept window as P($\mathbf{O}_i$). In a time window t_i, each upcoming instance would be assigned to the nearest sub-concept according to the distance from the instance to the centers of existing sub-concepts. When a sub-concept drift is detected, the P($\mathbf{O}_i$) (i=1,…k) would be updated based on newly arrived instances as shown in Algorithms 2 and 4.

Definition 5 (sub-concept drift): $\boldsymbol{P}_{t_j}(\boldsymbol{O}_i) \neq \boldsymbol{P}_{t_k}(\boldsymbol{O}_i)$.

Sub-concept drifts are detected by computing test statistics within each time sub-concept windows. This way, the test statistics of a sub-concept is more sensitive than that of a concept on the whole domain. When $P_{t_j}(O_i)$ is not equal to $P_{t_k}(O_i)$, a sub-concept drift between time point t_j and time point t_k occurs. This is a kind of *k*TSW based concept drift.

Using these definitions, we construct a method able to detect sub-concept drifts based on sub-concept knowledge. This way we are able to find concept drifts early and find some drifts that may be ignored by WTW based methods.

4 Concept Drift Detection Based on *k* Time Sub-Concept Windows

4.1 The Framework of *k*TSW Based Concept Drift Detection

Concept drift detection based on *k*TSW includes the following three components: a *Divide Module*, a *Base Learner* and a *Drift Monitor*. The framework's structure is shown in Figure 3. The *divide module* can divide instances from a whole time window into *k* time sub-concept windows. This means that the *divide module* can identify the sub-concepts. The *base learner* predicts the value of each unknown instance. The *drift monitors* (actually, sub drift monitors) observe the performance evaluation over instances from the sub-concept windows. The number of drift monitors needed is equal to the number of time sub-concept windows which is, itself, determined by domain knowledge or by inspection. When an instance arrives, the system first determines which sub drift monitor should be used, then submits the predicted result to that sub drift monitor. It outputs a drift level (Normal/ Warning/Drift) based on the sub drift monitors' feedback. If one of the Sub Drift Monitors' feedback is "Drift", then our method outputs "Drift" and resets the whole system; Else if one of Sub Drift Monitors' feedback is "Warning", then it outputs "Warning"; Otherwise it outputs "Normal".

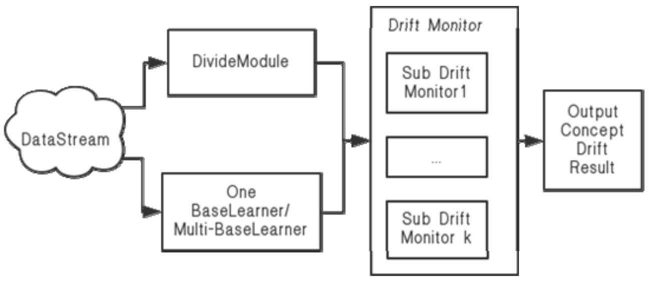


Figure 3. the framework structure chart of *k*TSW based concept drift detection

The divide module. The first step of Concept drift detection with the *k* time sub-concept window method is to divide instances from the whole time window into *k* time sub-concept windows. Based on Definitions 3 to 5, we can transform this problem into dividing concepts into sub-concepts. Sharma et al. [9] explored sub-concepts structure and applied it to one class classification. Moulton et al. [10] adapted that idea to data streams. They identified three levels of knowledge of the sub-concepts structure, including complete knowledge, fuzzy knowledge and no knowledge. When we have knowledge of the sub-concepts structure, we can use that knowledge to identify sub-concepts. This knowledge can be used in both the initial and the online phases. If we have fuzzy knowledge of the sub-concept structure, we can use it to identity sub-concepts in the initial phase. This information, however, does not carry to the online phase. Finally, if we have no knowledge of the sub-concepts but there exists an underlying sub-concepts structure, we can use a clustering algorithm during both the initial and online phase to identify the sub-concepts [18].

The base learner and drift monitor. Base learners are the inductive learning models that learn from the data stream. The drift monitor is the module which tracks the performance of the base learner, and then checks whether a concept drift has occurred. In our method, the concept from the data stream is divided into sub-concepts, which means that the instances from the data stream are divided into sub sets. As a result, we can use one or multiple base learners in our method.

The drift monitor consists of the *k* Sub *Drift Monitors* whose number *k* is equal to the k number of time sub-concept windows. In our method, we can use any concept drift algorithm as Sub Drift Monitor. These algorithms could be DDM, EDDM etc. which are error based algorithm. They output the drift level (Normal/ Warning/Drift) of the stream (or sub stream) they are monitoring.

In order to apply sub-concepts structure to improve concept drift detection and the performance of classification under the proposed method, this paper utilizes three schemes based on different numbers of base learners, and presents strategies for each scheme which take advantage of the kind of sub-concepts structure knowledge that Sharma et al. [9] defined. In scheme I, there are multiple drift monitors, but only one whole base learner. In scheme II, there are multiple drift monitors and multiple sub base learners. Scheme III is the combination of scheme I and scheme II. There are multiple drift monitors with one whole base learner and multiple sub base learers. When predictions of whole base lerner and sub base learner are all true, then submit true to sub drift monitor, otherwise submit false to sub drift monitor. We will describe the algorithms of scheme I and II in the next section.

4.2 Scheme I: Multiple Drift Monitors with One Base Learner

In this scheme, there is only one base learner for all instances or subsets. This means that all initial instances are used to build one base learner in the initial phase. We track this base learner's error-rate over each subset in the online concept drift monitoring phase.

There are generally three levels of knowledge of the sub-concepts structures [10]. In the complete knowledge scenario, the number of Sub Drift Monitors is equal to the number of classes or categories, and each Sub Drift Monitor tracks the performance of the base learner over one class or category. The sub-concept's index can be calculated by the sub-concepts' knowledge. In the fuzzy knowledge scenario, we first obtain the number of classes or categories which we can then use as the number of sub-concepts. In the no knowledge scenario, we cannot get any knowledge about the sub-concepts structure, so we use a clustering model to identify the underlying sub-concepts structure.

The initial phase and online phase of the algorithm with one base leaner is shown as Algorithm 1 and Algorithm 2 respectively, where *N* is the number of training instances to be used during initialization; *k* is the number of time sub-concept windows determined by domain knowledge or by inspection; DriftM is the concept drift monitor chosen to detect sub-concepts; CLASS is the base classifier; CLUS is the cluster model used to divide the concept into sub-concepts.

During the initiation phase, the first N instances are buffered, and used to build a base classifier whose performance will be tracked over future data stream instances. In order to divide instances from the whole time window into sub-concept windows, a clustering model needs to be built with parameter *k* in the case of fuzzy knowledge or no knowledge. In the case of fuzzy knowledge, the value of *k* will depend on prior knowledge. In algorithm 1, Lines 1 to 4 cache the initiation instances. Line 5 builds the base classifier. Lines 6 to 11 build the clustering (CLUS) when there is fuzzy or no knowledge.[1] Lines 12 to 16 return the clustering models built in each of the knowledge levels.

Algorithm 1. One base learner-initiation phase

```
Input: Datastream, N, k
OutPut: CLASS, CLUS
1. While (DS has more instances && numInst < N) do
2.     Add next instance to Buffer
3.     numIns++
4. End While
5. Build classifier model (CLASS) in Buffer
6. If fuzzy knowledge then
7.     k= the number of instances' class or category
8. End if
9. If no Knowledge or fuzzy knowledge then
10.    Build Cluster model (CLUS) in Buffers with k
11. End if
12. If Complete Knowledge then
13.    Return CLASS
14. Else
15.    Return CLASS, CLUS
16. End if
```

During the online phase, our method uses existing concept drift detection methods, such as DDM and EDDM, to create *k* sub-concept drift monitors. The performance of CLASS over each sub-concept instance set will be tracked by drift monitors. It checks whether a drift occurs by considering the information output by the drift monitors. In algorithm 2, Lines 1 to 3 reset the value of *k* in the case of fuzzy knowledge. Line 4 creates *k* sub-concept drift monitors (DriftM). Lines 6 to 10 compute the instance's sub concepts index (subWindowIndex). Lines 11 to 12 get the predictive result. Lines 13 to 19 check the drift level based on the drift results of the drift monitors (DriftM). Line 20 to 22 update the clustering model and the classifier if a drift occurs.

Algorithm 2. One base learner-online phase

```
Input: Datastream, k, DriftMonitor, CLASS, CLUS
OutPut: Level (Normal/ Warning/Drift)
1. If complete knowledge or fuzzy knowledge then
2.     k= the number of instances' class
3. End if
4. Use DriftMonitor to create k SubConcept Drift Monitors DriftM [k];
5. While (DS has more instances) do
6.     If complete knowledge then
7.         subWindowIndex= instance's class or category
8.     Else
9.         subWindowIndex=CLUS (instance)
10.    End if
11.    predictValue=CLASS (instance)
12.    submit predictResult (true/false) to DriftM [subWindowIndex]
13.    If DriftM [subWindowIndex] drift then
14.        Level="Drift"
15.    ElseIf DriftM [subWindowIndex] warning then
16.        Level="Warning"
17.    Else
18.        Level="Normal"
19.    End If
20.    If Level=="Drift" then
21.        Use Instances from Warning to Drift to update CLUS and CLASS
22.    End If
23.    OutPut Level
24. End While
```

4.3 Scheme II: Multiple Drift Monitors with Multiple Base Learners

In this scheme, there are multiple base learners. That is, a base learner and a drift monitor are built for instances from each time sub-concept window that describe different sub-concepts. The number of base learners is equal to the number of time sub-concept windows and the number of drift Monitors. In the initial phase, the initial instances are divided into subsets, we build a base learner for each subset. In the online phase, we monitor the error-rate of each base learner over the instances from the corresponding time sub-concept window.

In the complete knowledge scenario, we build a base learner for each sub-concept data set, and track the base learner's error-rate over the sub-concepts sets. In the fuzzy knowledge scenario, we cannot get the sub-concept information of novel instances from the data stream. We identify the underlying sub-concepts using a clustering algorithm, and the number of clusters *k* can be obtained from domain knowledge. In the no knowledge scenario, we cannot get the sub-concept knowledge of all the instances from the data stream. We use a clustering model to identify the underlying sub-concepts structure as well as the number of clusters.

The initial and online phases of the algorithm with multiple base leaners are as shown in Algorithm 3 and

[1] In the case of complete knowledge, no clustering algorithm needs to be applied as we already know how to divide the data into separate classes.

Algorithm 4, respectively, where the meaning of the parameters are same as in Algorithm 2 and Algorithm 3. During the initialization phase, we first divide the first N data stream instances into sub-concept sets, and build classifiers over each subset. If we have fuzzy or no knowledge, we should build a clustering model to learn the sub-concept instances sets.

Algorithm 3. Multi base learner-initiation phase

```
Input: Datastream, N, k
OutPut: subCLASS, CLUS
1.  While (DS has more instances && numInst < N)
      do
2.    Add next instance to Buffer
3.    numIns++
4.  End While
5.  If complete knowledge or fuzzy knowledge then
6.     k= the number of instances' class or category
7.  End if
8.  If fuzzy knowledge or no knowledge then
9.    Build Cluster model (CLUS) with k
10.   IF no knowledge then
11.       Run CLUS to divide Buffer into k subBuffers
      [k]
12.    End if
13. End if
14. If complete knowledge or fuzzy knowledge then
15.   divide Buffer into k subBuffers [k] with the sub-
      concept knowledge of instances
16. End if
17. Build classifier models (subClass [k]) in each
      subBuffer
18. If Complete Knowledge then Return subCLASS
19. Else  Return subCLASS, CLUS
```

In algorithm 3, lines 1 to 4 cache the initiation instances. Lines 5 to 16 build the clustering (CLUS) in the case of fuzzy or no knowledge, and divide Buffer into subBuffers with CLUS when there is no knowledge. Line 17 builds k base classifiers on each subBuffer. Lines 18 to 19 return the model for the different knowledge levels.

During the online phase, our method creates k drift monitors which will track the performance of the k base learners to detect concept drift. In algorithm 4, line 4 creates k sub-concept drift monitors. Lines 5 to 24 detect concept drifts based on data from the sub time windows. If a drift is detected, then the updating of the cluster and sub classifier will be triggered.

Algorithm 4. Multi base learner-online phase

```
Input: Datastream, N, k, DriftMonitor, subCLASS,
       CLUS
OutPut: Level: Concept drift level (Normal/
        Warning/ Drift)
1.  If complete or fuzzy knowledge then
2.    k= the number of instances' class
3.  End if
4.  UseDriftMonitor to create k SubConcept Drift
      Monitors DriftM [k];
5.  While (DS has more instances) do
6.    If  complete knowledge then
7.      subWindowIndex= instance's class or category
8.    Else
9.      subWindowIndex=CLUS (instance)
10.   End if
11.   predictValue=subCLASS [subWindowIndex]
      (instance)
12.   submit predictResult (true/false) to DriftM
      [subWindowIndex]
13.   If DriftM [subWindowIndex] drift then
14.       Level="Drift"
15.   ElseIf DriftM [subWindowIndex] warning then
16.       Level="Warning"
17.   Else  Level="Normal"
18.   End If
19.   If Level=="Drift" then
20.       Use Instances in Buffer from Warning to Drift
      to update   CLUS
21.       Use Instances in subBuffer
      [subWindowIndex]  from
    Warning to Drift to update   subCLASS
      [subWindowIndex]
22.   End If
23.   OutPut Level
24. End While
```

5 Experiments

In our experiments, we implemented the algorithms described in the previous section using WEKA (available at https://www.cs.waikato.ac.nz/ml/weka/) and MOA [19] (available athttps://moa.cms.waikato.ac.nz/). To evaluate our method, we compared our proposed algorithm, ***k*TSW,** outfitted with DDM [5] and EDDM [6] to the standard whole window algorithm, **WTW**, also outfitted with DDM and EDDM on classification tasks. These experiments were carried out on two real world data sets. All experiments were conducted on a computer with 1.8GHz CPU and 16GB RAM. In addition, we used k-means as the base clustering system and HoeffdingTree as the base learner (classifier). In our experiments, k was set in the range of 2 to10. The parameters of the compared algorithms were set to the default values suggested by their authors. The meaning of the acronyms used to describe the compared methods is shown in Table 1.

Table 1. Meaning of methods' acronyms

Method Name	Means
WTW-***	Concept drift detection algorithm (***) based on **W**hole instances from **T**ime **W**indow
*k*TSW-***-C	Concept drift detection algorithm (***) based on instances from ***k*** **S**ub-concept **T**ime **W**indows in **C**omplete knowledge scenario
*k*TSW-***-F	Concept drift detection algorithm (***) based on instances from ***k*** **S**ub-concept **T**ime **W**indows in **F**uzzy knowledge scenario
*k*TSW-***-N	Concept drift detection algorithm (***) based on instances from ***k*** **S**ub-concept **T**ime **W**indows in **N**o knowledge scenario
*k*TSW-***-One	Concept drift detection algorithm (***) based on instances from ***k*** **S**ub-concept **T**ime **W**indows with **One** base learner (Scheme I)
*k*TSW-***-Multi	Concept drift detection algorithm (***) based on instances from ***k*** **S**ub-concept **T**ime **W**indows with **Multi**ple base learner (Scheme II)
*k*TSW-***-Com	Concept drift detection algorithm (***) based on instances from ***k*** **S**ub-concept **T**ime **W**indows with Scheme III (**Com**bining Scheme I and II)

5.1 Experiment1: ELEC2

The ELEC2 dataset [11] contains 45,312 instances, 7 attributes and 2 classes. On this data set, 5% of the instances are used as initial instances. The classification results are shown in Table 2 to Table 4. Table 2 shows the results using scheme I. Table 3 shows the results using scheme II. Table 4 shows the results using scheme III. Because the ELEC2 data set only has two classes, Schemes II and III are not suitable to this data set in the case of complete knowledge. Indeed, with knowledge of the instances' labels, the classification accuracy of the sub base learner is very high and we will fail to detect concept drift by tracking its error rate.

Table 2. Classification results with scheme I on ELEC2

Methods name	Accuracy	Recall	Time(s)	Methods name	Accuracy	Recall	Time(s)
WTW-DDM	0.798801	0.784419	0.11558±0.03283	WTW-EDDM	0.768788	0.758974	0.07662±0.02801
*k*TSW-DDM-C	0.802611	0.797760	0.13811±0.03926	*k*TSW-EDDM-C	0.776128	0.772554	0.08224±0.03127
*k*TSW-DDM-F	0.807606	0.802561	0.19338±0.09949	*k*TSW-EDDM-F	0.767022	0.765618	0.11925±0.07231
*k*TSW-DDM-N	**0.814482**	**0.810034**	0.38722±0.12828	*k*TSW-EDDM-N	**0.796223**	**0.794449**	0.31095±0.11218

Table 3. Classification results with scheme II on ELEC2

Methods name	Accuracy	Recall	Time(s)	Methods name	Accuracy	Recall	Time(s)
WTW-DDM	0.798801	0.784419	0.11558+0.03283	WTW-EDDM	0.768788	0.758974	0.07662+0.02801
*k*TSW-DDM-F	0.795503	0.781201	0.18578+0.07076	*k*TSW-EDDM-F	0.739215	0.733695	0.18248+0.07346
*k*TSW-DDM-N	**0.812600**	**0.805781**	0.35757+0.10123	*k*TSW-EDDM-N	**0.794945**	**0.793371**	0.35205+0.17954

Table 4. Classification results with scheme III on ELEC2

Methods name	Accuracy	Recall	Time(s)	Methods name	Accuracy	Recall	Time(s)
WTW-DDM	0.798801	0.784419	0.11558+0.03283	WTW-EDDM	0.768788	0.758974	0.07662±0.02801
*k*TSW-DDM-F	**0.821892**	**0.8184415**	0.32651+0.18189	*k*TSW-EDDM-F	0.796989	**0.793024**	0.27710±0.09118
*k*TSW-DDM-N	0.819755	0.8124215	0.54086+0.18096	*k*TSW-EDDM-N	**0.798825**	0.781092	0.48046±0.13164

From tables 2 to 4, we can see that our methods (*k*TSW outfitted with DDM and EDDM) outperform traditional windowing methods (WTW outfitted with DDM and EDDM). The results show that the knowledge of sub-concepts can improve the performance of DDM and EDDM, and yield higher accuracy and recall. But the time consumption is greater. The time consumption mainly includes the time used in building the base learner model (A), in building the clustering model (B), in monitoring the drift (C), in updating the base learner (D) and in updating the cluster model (E). The time consumption details are shown in Table 5. When a concept drift is detected, the base learner and the clustering need to be updated, so the greater the number of concept drifts detected, the greater the time cost. From this, we can conclude that the reason our methods are improving the classification accuracy is that they can detect more concept drifts. The underlying reason is that our methods monitor the instances from the *k* sub concept time window, so they are more sensitive to sub concept drifts than traditional methods.

Table 5. The time cost details of difference methods

Method Name	A	B	C	D	E
WTW-***	YES	NO	YES	YES	NO
*k*TSW-***-C	YES	NO	YES	YES	NO
*k*TSW-***-F	YES	YES	YES	YES	YES
*k*TSW-***-N	YES	YES	YES	YES	YES

It costs time to update the base learner and the clustering algorithm when a concept drift occurs is different from one scheme to the other. That cost is the lowest in scheme II, because it only updates one base learner and the other base learners do not need to be updated. However, the time cost of forming the initial clusters of instances for scheme II is higher than in scheme I because the instances are divided into *k* subsets, and each subset must satisfy the minimum number of training instances required.

5.2 Experiment2: Mobile Traffic Data Set

The Mobile traffic data set [12] includes 11,8020 instances, 36 attributes and 12 classes. The distribution of the mobile traffic data set is shown in Table 6. This data set is an imbalanced one, so we also use the G-mean to evaluate our methods. We use 1%, 5% and 20% of the data set as initial instances. The knowledge of the sub-concepts is the category of instances. Category I is QQ, WeChat, Facebook, Weibo, Youku, Category II is Youku, TencentVideo, MgTV, Browser, Category III is JdShop, VipShop, and Category IV is QQMail and YahooMail. So the value of *k* is 4 for complete or fuzzy knowledge. The classification results are shown in Table 7 to Table 9.

Table 6. The detail of instance number in each class

Class Name	QQ	WeChat	Facebook	Weibo	Youku	Tencent Video	MgTV	Browser	JdShop	VipShop	QQMail	Yahoo Mail
Instances Number	17104	13631	1401	25407	5825	1593	14046	25512	4008	4577	1432	3484

Table 7. Classification results with scheme I on MobileDS

Methods name	Accuracy	Recall	G-means	Methods name	Accuracy	Recall	G-means
WTW-DDM	0.638737	0.584183	0.574169	WTW-EDDM	0.625873	0.576908	0.563337
*k*TSW-DDM-C	0.649170	0.601099	0.591877	*k*TSW-EDDM-C	0.630820	0.585996	0.575952
*k*TSW-DDM-F	0.663266	0.615754	0.606624	*k*TSW-EDDM-F	0.640979	0.572255	0.553894
*k*TSW-DDM-N	**0.672193**	**0.621638**	**0.611802**	*k*TSW-EDDM-N	**0.651883**	**0.588673**	**0.577138**

Table 8. Classification results with scheme II on MobileDS

Methods name	Accuracy	Recall	G-means	Methods name	Accuracy	Recall	G-means
WTW-DDM	0.658773	0.607162	0.595651	WTW-EDDM	0.667924	0.611037	0.600456
*k*TSW-DDM-C	**0.781224**	**0.7508643**	**0.741349**	*k*TSW-EDDM-C	**0.767354**	**0.74213991**	**0.732557**
*k*TSW-DDM-F	0.665454	0.613998	0.602731	*k*TSW-EDDM-F	0.667862	0.615320	0.603631
*k*TSW-DDM-N	0.668031	0.613603	0.602203	*k*TSW-EDDM-N	0.666997	0.616512	0.607364

Table 9. Classification results with scheme III on MobileDS

Methods name	Accuracy	Recall	G-means	Methods name	Accuracy	Recall	G-means
WTW-DDM	0.653586	0.601903	0.590203	WTW-EDDM	0.646977	0.597031	0.586200
*k*TSW-DDM-C	**0.796804**	**0.7726095**	**0.763369**	*k*TSW-EDDM-C	**0.793467**	**0.778905**	**0.770425**
*k*TSW-DDM-F	0.670098	0.6246085	0.614381	*k*TSW-EDDM-F	0.659814	0.600604	0.589136
*k*TSW-DDM-N	0.672291	0.617889	0.606754	*k*TSW-EDDM-N	0.670374	0.607260	0.617848

From Table 7 to Table 9, we can see that our method can improve the performance of concept drift monitors when compared to WTW. Especially, when we have complete knowledge we can obtain higher accuracy, recall and G-mean. This demonstrates that the knowledge of sub-concepts is useful to sub-concepts drift detection. But in scheme I, the advantage of knowledge is not much obvious. So when we have enough initial instances and multiple classes, we can choose scheme II or III for higher classification accuracy. The mobile traffic data set has 12 classes. To explore the performance of our methods in different classes, we compare the accuracy of the different methods on each class. The experimental results are shown as Figure 4 to Figure 9. We can see that our method can improve the performance of most classes using either DDM or EDDM, especially, when using scheme II and III with complete knowledge. For example, in scheme III, when we have complete knowledge and use *k*TSW outfitted with DDM, the accuracy improves on all classes; when we have fuzzy knowledge and we use *k*TSW outfitted with DDM, the accuracy improves on 8 out of 12 classes; when we have no knowledge and we use *k*TSW outfitted with DDM the accuracy improves in 6 out of 12 classes.

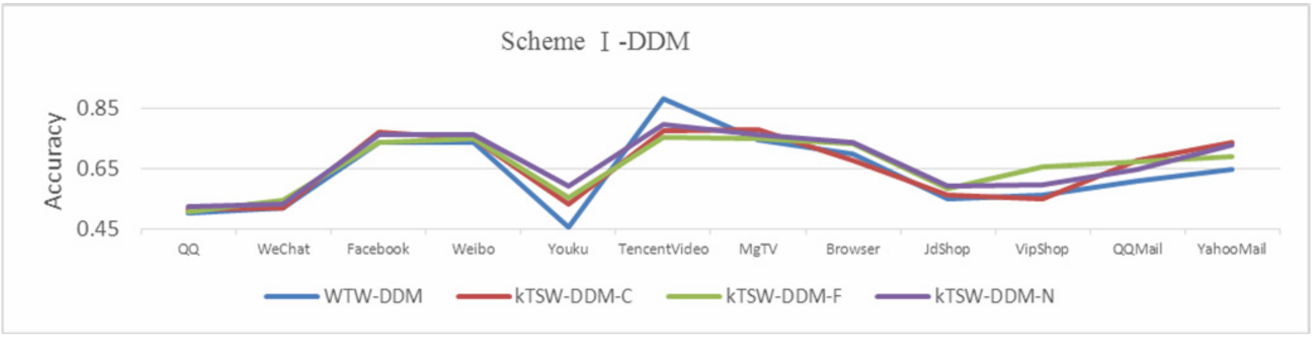


Figure 4. The accuracy of each class with scheme I and DDM

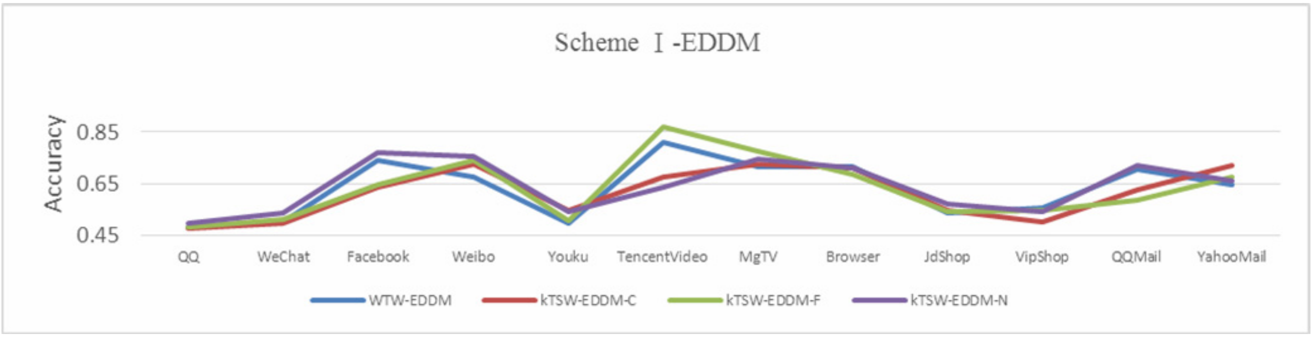


Figure 5. The accuracy of each class with scheme I and EDDM

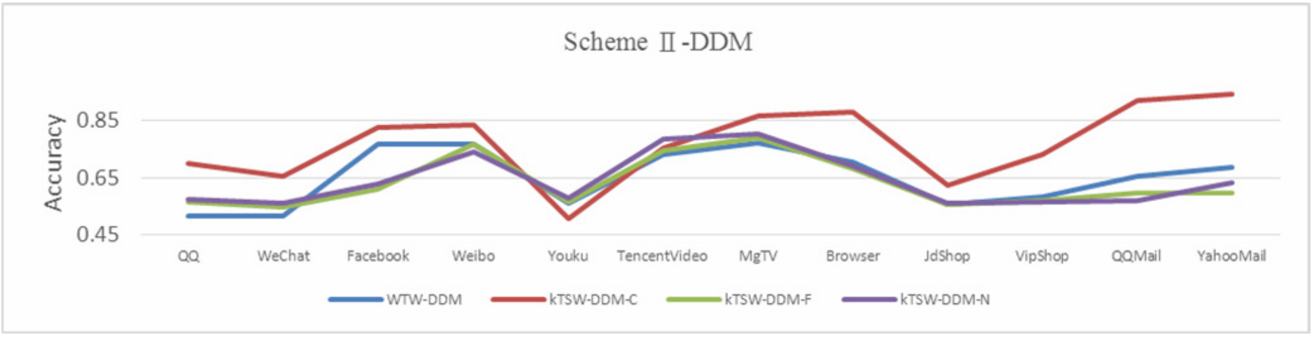


Figure 6. The accuracy of each class with scheme II and DDM

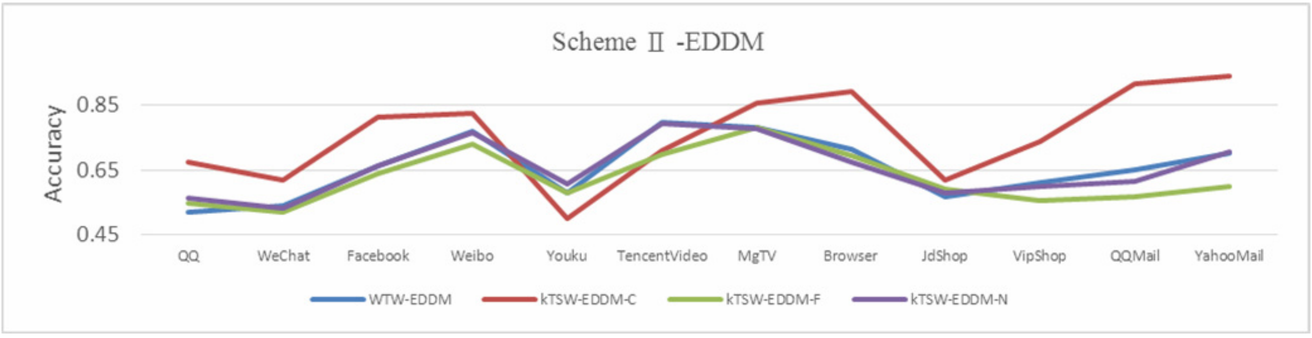


Figure 7. The accuracy of each class with scheme II and EDDM

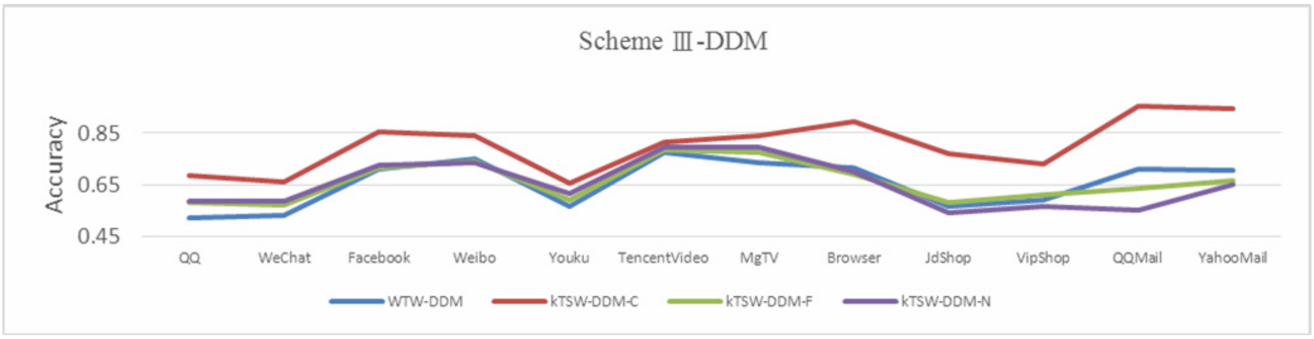


Figure 8. The accuracy of each class with scheme III and DDM

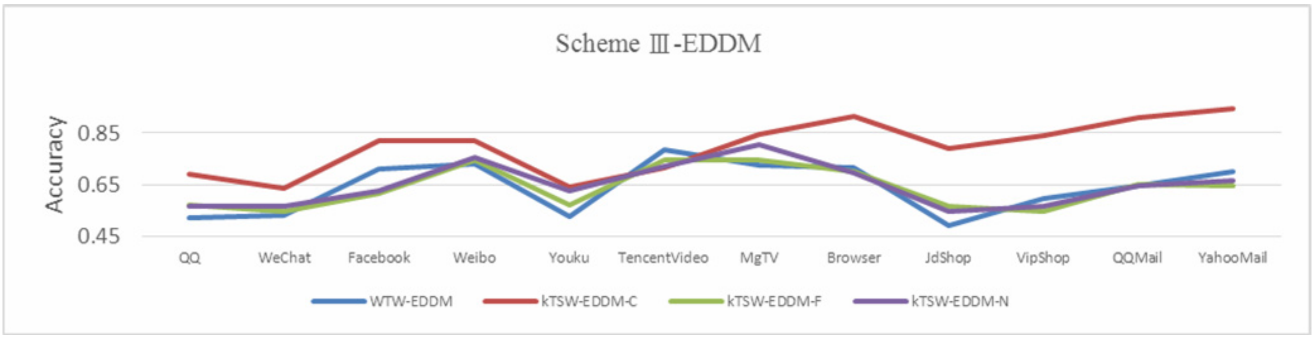


Figure 9. The accuracy of each class with scheme III and EDDM

5.3 Parameter Discussion

5.3.1 The Value of k

The value of *k* represents the number of time sub-concept windows. When we have knowledge of the number of sub-concepts, we know the value of *k*. To explore the performance of our methods with different values of *k*, we compare the accuracy of our methods with value of *k* varying from 1 to 12. The experimental results are shown in Figure 10 to Figure 13. When the value of *k* is 1, our method is the same as the WTW based method, for example, Figure 10 is the results of *k*WST-DDM, when the value of *k* is 1, the *k*WST-DDM equal to WTW-DDM. In Figures10 to Figure 13, we can see that most of values of *k* can improve accuracy, especially on the mobile traffic data set. Since the underlying sub-concepts for mobile traffic are more complex, the division offered by the sub-concepts methods performs well on the problem of detecting concept drift in mobile traffic data.

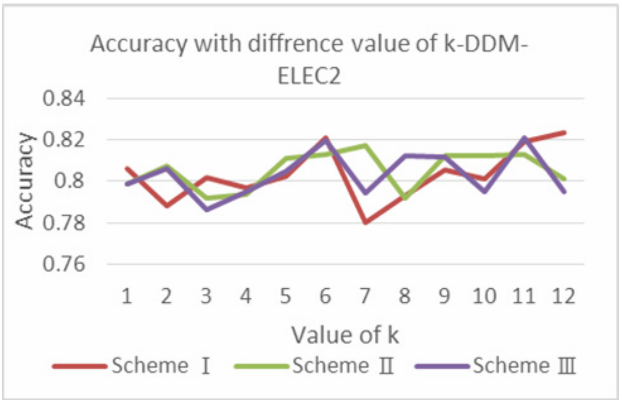


Figure 10. The accuracy of *k*WST-DDM with difference value of *k* on ELEC2

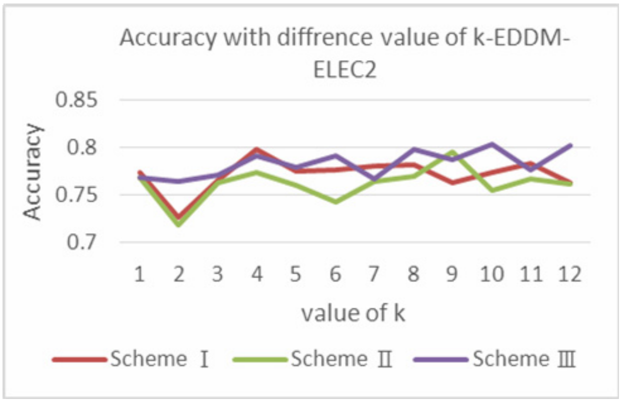


Figure 11. The accuracy of *k*WST-EDDM with difference value of *k* on ELEC2

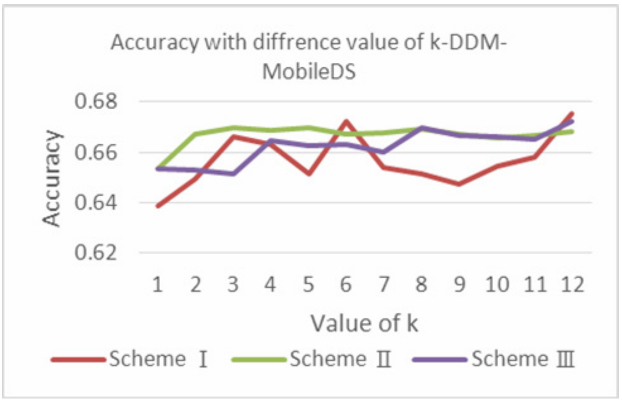


Figure 12. The accuracy of *k*WST-DDM with difference value of *k* on MobileDS

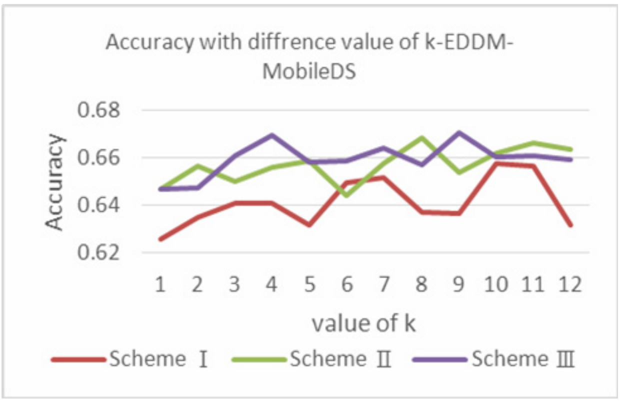


Figure 13. The accuracy of kWST-EDDM with difference value of k on MobileDS

5.3.2 The Number of Initial Data from Data Stream

Because error-based concept drift detection methods detect concept drift by tracking the error rate of the base learner, we set a fixed number of instances for building the base learner. This section compares the classification accuracy under different numbers of initial training instances to study the performance of our method. The experimental results under the situation with no sub-concepts knowledge are shown in Figure 14 and Figure 15. The x-axis represents the ratio of the number of initial training instances to the whole experimental data set, and the y-axis is the classification accuracy. The results in Figure 14 represent the comparison of WTW-DDM and *k*TSW-DDM when the value of *k* is 6. We can see that our method can obtain higher accuracy than WTW-DDM in most cases. Scheme II's accuracy is lower when the initial ratio is lower than 0.03, because scheme II needs to build multiple base learners, and therefore, it needs more initial instances than scheme I. We can also see that the effect of the number of initial instances on scheme I is not significant.

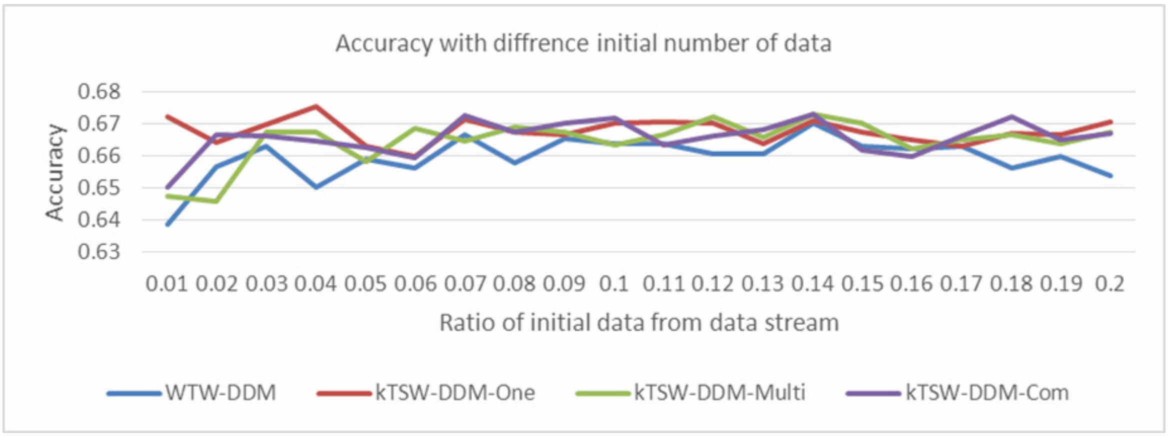


Figure 14. The methods with DDM accuracy comparison under difference ratio initial data

The results in Figure 15 represent the comparison between WTW-EDDM and *k*TSW-EDDM. The accuracy of *k*TSW-EDDM is higher with *k* from 2 to 20 in Figure 15. We can see that *k*TSW-EDDM can obtain higher accuracy than WTW-DDM in most cases as well. But when the ratio of initial to whole training data equals 0.06, 0.07, 0.09 and 0.1, some of our schemes perform worse than WTW-DDM. This is because the minimum number of errors of EDDM is set to 30. If the instances are divided into multiple time sub-concept windows and the change is slow, the detection of a concept drift will be delayed and that will negatively influence the classification accuracy. So how to divide instances from the data stream is crucial to our method when we have no knowledge of sub-concepts

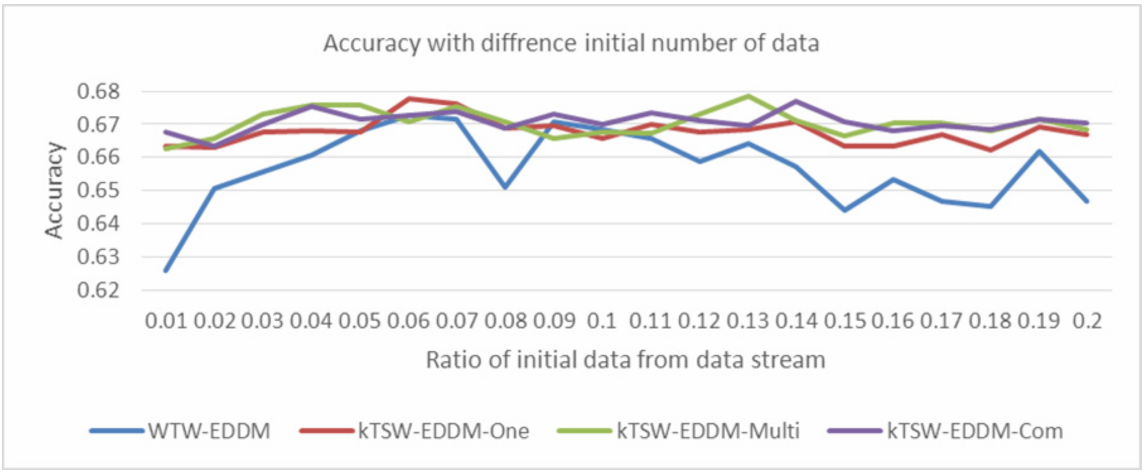


Figure 15. The methods with EDDM accuracy comparison under difference ratio initial data

6 Conclusions

This paper explores the idea of a concept-drift detection method based on sub concepts. Our method divides the data from a whole time window into *k* time sub-concept windows. Three schemes are proposed to improve the performance of error rate-based concept drift detection algorithms. The advantage of our method is that it can take full advantage of any knowledge it has of the sub-concepts structure to detect sub-concepts drift that maybe be ignored by the whole time window methods.

We tested our method on two real world data sets. From our experimental results, we can see that the accuracies and recalls of *k*TSW-DDM and *k*TSW-EDDM are higher than those of WTW-DDM and WTW-EDDM respectively. The characteristics of our schemes and the results we obtained are summarized below.

(1) Scheme I uses multiple drift monitors with one base learner, it obtains higher classification accuracy than scheme II when we have no knowledge of the sub-concepts structure, and it needs a smaller number of initial instances than scheme II. On the other hand, its time consumption is greater than that of scheme II.

(2) Scheme II uses multiple drift monitors with multiple base learners. It obtains higher classification accuracy than scheme I when we have complete knowledge of the sub-concept structure, and its time consumption is lower. However, it needs more initial instances than scheme I.

(3) Scheme III is the combination of Scheme I and II. It obtains higher classification accuracy than scheme I and II, but its time consumption is more than that of scheme I and II.

(4) As a whole, our experimental results show that the proposed *k*TSW based concept drift detection method can achieve higher classification accuracy and recall—no matter which scheme it uses—than WTW based methods. It performs especially well when we have complete knowledge of the sub-concepts structure.

In this paper we only use *k*-means to divide the original window. We will investigate other clustering algorithms in future.

Acknowledgments

This work was partly supported by the National Natural Science Foundation of China under Grant No. 61872027 and No. 61501128.

References

[1] D. Tao, T. Y. Wu, J. G. Jiang, P. P. Liu, uIP Stack based Embedded Network Control System for Bridge Safety Detection, *Journal of Internet Technology*, Vol. 17, No. 6, pp. 1109-1116, November, 2016.

[2] R. Z. Wang, D. Tao, Context-Aware Implicit Authentication of Smartphone Users Based on Multi-Sensor Behavior, *IEEE ACCESS*, Vol. 7, No. 1, pp. 119654-119667, August, 2019.

[3] A. S. Iwashita, J. P. Papa, An Overview on Concept Drift Learning, *IEEE Access*, Vol. 7, pp. 1532-1547, December, 2018.

[4] Y. Sun, K. Tang, Z. Zhu, X. Yao, Concept Drift Adaptation by Exploiting Historical Knowledge, *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 29, No. 10, pp. 4822-4832, October, 2018.

[5] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with Drift Detection, *17th Brazilian Symposium on Artificial Intelligence*, São Luis, Maranhão, Brazil, 2004, pp. 286-295.

[6] M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, R. Morales-Bueno, Early Drift Detection Method, *Fourth International Workshop on Knowledge Discovery from Data Streams*, 2006, pp. 77-86.

[7] I. Frias-Blanco, J. d. Campo-Avila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-Diaz, Y. Caballero-Mota, Online and Non-parametric Drift Detection Methods Based on Hoeffding's Bounds, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 27, No. 3, pp. 810-823, March, 2015.

[8] A. Liu, G. Zhang, J. Lu, Fuzzy Time Windowing for Gradual Concept Drift Adaptation, *26th IEEE International Conference on Fuzzy Systems*, Naples, Italy, 2017, pp. 1-6.

[9] S. Sharma, A. Somayaji, N. Japkowicz, Learning over Subconcepts: Strategies for 1-class Classification, *Computational Intelligence*, Vol. 34, No. 2, pp. 440-467, May, 2018.

[10] R. H. Moulton, H. L. Viktor, N. Japkowicz, J. Gama, *Contextual One-Class Classification in Data Streams*, CoRR abs/1907.04233, July, 2019.

[11] M. Harries, *Splice-2 Comparative Evaluation: Electricity Pricing*, Report No. UNSW-CSE-TR-9905, University of New South Wales, July, 1999.

[12] R. Y. Wang, Z. Liu, Y. M. Cai, D. Tang, J. Yang, Z. Yang, Benchmark Data for Mobile App Traffic Research, *Mobile and Ubiquitous Systems Conferenc*, New York, USA, 2018, pp. 402-411.

[13] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, G. Zhang, Learning under Concept Drift: A Review, *IEEE Transactions on Knowledge and Data Engineering* (Early Access), October, 2018, pp. 1-18.

[14] G. J. Ross, N. M. Adams, D. K. Tasoulis, D. J. Hand, Exponentially Weighted Moving Average Charts for Detecting Concept Drift, *Pattern Recognition Letters*, Vol. 33, No. 2, pp. 191-198, January, 2012.

[15] A. Bifet, R. Gavalda, Learning from Time-changing Data with Adaptive Windowing, *2007 SIAM International Conference on Data Mining*, Minneapolis, Minnesota, USA, 2007, pp. 443-448.

[16] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, A. Bouchachia, A Survey on Concept Drift Adaptation, *ACM Computing Surveys*, Vol. 46, No. 4, pp. 44/1-37, April, 2014.

[17] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, F. Petitjean, Characterizing Concept Drift, *Data Mining and Knowledge Discovery*, Vol. 30, No. 4, pp. 964-994, July, 2016.

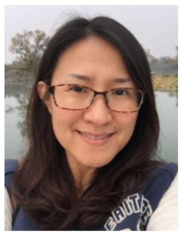
[18] R. H. Moulton, H. L. Viktor, N. Japkowicz, J. Gama, Clustering in the Presence of Concept Drift, *European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, Dublin, Ireland, 2018, pp. 339-355.

[19] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, MOA: Massive Online Analysis, *The Journal of Machine Learning Research*, Vol. 11, pp.1601-1604, May, 2010.

Biographies

Li Liu currently works as an associate professor in the School of Information Science and Technology, Huizhou University, Huizhou, China. She received the M.S. degree in School of Computer Science, Guangxi, Nanning, China in 2004. Her research interests are in the areas of machine learning, image retrieval and mobile traffic classification.

Nathalie Japkowicz is a Professor and Chair in the Computer Science at American University. She was previously with the School of Electrical Engineering and Computer Science at the University of Ottawa where she led the Laboratory for Research on Machine Learning for Defense and Security. Over the years, she has published over 100 articles, papers and books.



Dan Tao currently works as a professor in the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China. She received the Ph.D. degree in School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China in 2007. Her research interests include wireless network, IoT, mobile computing and big data security.

Zhen Liu received the Ph.D. degree from the School of Computer Science and Technology of South China University of Technology, China, in 2013. She is now a Lecturer in the School of Medical Information Engineering, Guangdong Pharmaceutical University, Guangzhou, China. Her research interests are in the areas of machine learning, mobile traffic classification