CADM: Confusion Model-Based Detection Method for Real-Drift in Chunk Data Stream

Songqiao Hu[1], Zeyi Liu[2], and Xiao He[2(✉)]

[1] School of Automation, Beijing Institute of Technology, Beijing 100081, China
1120193091@bit.edu.cn

[2] Department of Automation, Tsinghua University, Beijing 100084, China
liuzy21@mails.tsinghua.edu.cn, hexiao@tsinghua.edu.cn

Abstract. Concept drift detection has attracted considerable attention due to its importance in many real-world applications such as health monitoring and fault diagnosis. Conventionally, most advanced approaches will be of poor performance when the evaluation criteria of the environment has changed (i.e. concept drift), either can only detect and adapt to virtual drift. In this paper, we propose a new approach to detect real-drift in the chunk data stream with limited annotations based on concept confusion. When a new data chunk arrives, we use both real labels and pseudo labels to update the model after prediction and drift detection. In this context, the model will be confused and yields prediction difference once drift occurs. We then adopt cosine similarity to measure the difference. And an adaptive threshold method is proposed to find the abnormal value. Experiments show that our method has a low false alarm rate and false negative rate with the utilization of different classifiers.

Keywords: Concept drift · Confusion model · Chunk data stream · Similarity

1 Introduction

Concept drift is a nonnegligible factor in data analysis of dynamic systems. For example, much data is collected by sensors, but the sensors' output is vulnerable to its structure and the surrounding environment, of which temperature is the most influential. Under different temperatures, the distribution range of the collected data and even their categories will change. If the monitor system fails to detect the change, it will be difficult to provide correct decision-making suggestions when risks happen [1].

In the literature, several advanced studies have been proposed for solving this problem. Gama *et* al. [2] proposed Drift Detection Method (DDM) to detect whether the overall online error rate increased greatly, which was used to judge whether the warning level or the drift level was reached. [3] proposed

H. R. Karimi and N. Wang (Eds.): S-Cube 2022, LNICST 487, pp. 191–201, 2023.
https://doi.org/10.1007/978-3-031-34899-0_13

a two-time window-based drift detection algorithm named ADaptive WINdowing (ADWIN). ADWIN indicates that concept drift occurs if the means of data in two windows differ significantly. In [4], conformal prediction was introduced to detect concept drift. Concept drift was confirmed if the conformal prediction result in the two data chunks showed a great difference. Otherwise, pseudo labels were used to update the model. Lu *et* al. [5] adopted ensemble learning and adjusted each base classifier's weight according to their performance. Once a base classifier was dropped out and a new classifier needed to generate, the size of the train data chunk increased continuously until the variance stopped rising. Using the selected size data chunk to train a new base classifier could adapt to concept drift. DSPOT proposed by Siffer *et* al. [7] believed that anomalies are usually extreme values and utilized extreme theory to fit the extreme value distributions for calculating the dynamic concept drift threshold. Sethi *et* al. [8] proposed the GC3 framework, which uses grid density clustering and a unified grid density sampling mechanism to achieve better performance with a lower label rate. The concept drift was then detected, which is the idea of feedback.

Generally, there are two main types of concept drift: real concept drift changes in $P(y|x)$, and virtual concept drift changes in $P(x)$. Nevertheless, many of the above methods can only detect virtual drift but do nothing for real drift. [3] only considers the mean of the data but is unconcerned about the labels of the data. So it can just handle virtual drift in principle. [4] seems to be put forward for real concept drift. However, judging whether two data chunks have significant distribution differences only uses the pseudo labels predicted by the same classifier. Therefore, it cannot find the change in the labels. In [7], only the case of one-dimensional time series is taken into account, and whether it is an outlier is given according to the value of the feature, and the samples do not have labels at all. So the method is similar to judging concept drift from $P(x)$. Another typical problem in data mining and concept drift is label cost. In real situations, obtaining actual data labels usually requires a lot of time and energy, especially in the case of a high-speed data stream. Therefore, it is unrealistic to regard the concept drift as a supervised learning problem [2] [5] [6]. [8] is one of the few methods that consider both actual drift and labeling cost. The main framework is based on the idea of feedback. Namely, some samples are selected for annotations after the prediction. Then, the classifier is adjusted according to the difference between the prediction and labeling results.

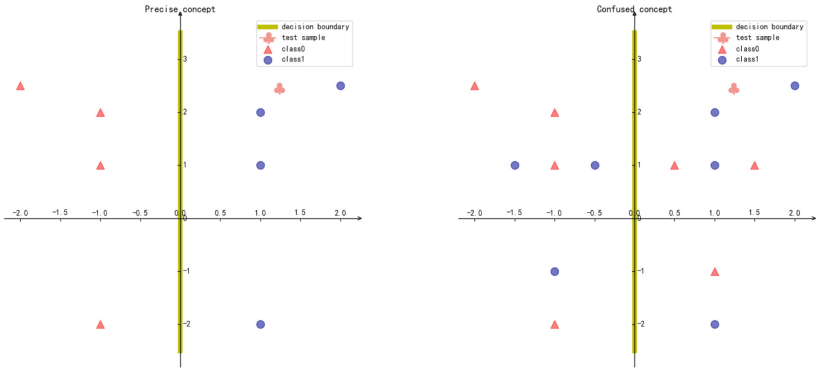
In this paper, we propose a novel chunk-based confusion model method called **C**onfusion **A**nd **D**etection **M**ethod (CADM) to deal with real concept drift with limited annotations. The main contribution of this paper can be summarized as follows: (1) A novel method is provided to detect real concept drift with limited labels in chunk data streams; (2) To the best of our knowledge, it is the first method that requires labels after concept drift detection. The drift detection results can then be used to guide the annotation costs; (3) The effectiveness is verified based on numerous experiments.

The rest of the paper is organized as follows. Section 2 introduces the method and datasets adopted in this study. Section 3 describes and discusses some experimental results and Sect. 4 presents the conclusions.

2 Proposed Method

2.1 Motivation

The main idea is generated from the confusion model, which refers to a model that has a low degree of certainty in the classification judgment of samples. Generally, it can be caused by different concepts contained in training data. An example can be shown in Fig. 1.



(a) PC = [4.55×10^{-6}, 0.999995] (b) PC = [0.3954, 0.6046]

Fig. 1. The prediction confidence (PC) for different training data by Naive Bayes model. (a) The training data is from the same concept. PC is then completely certain. (b) Add another batch of data with a different concept on the basis of (a). PC becomes uncertain. Such a model is called the confusion model.

In general, incremental learning models [9] have greater advantages to serve as the confusion model compared to common models due to the ability to update the model without storing historical data. In the case of the chunk data stream, the model can predict and select some samples to update when each data chunk arrives. To detect drift and improve the prediction performance of the model, we provide some samples with hard pseudo labels according to the prediction and annotate some samples manually. The purpose of pseudo labels is to increase training samples, improve prediction performance and adapt to virtual drift. While the purpose of manual annotations is to confuse the model when the real drift occurs, which makes the difference in prediction.

2.2 Difference Measurement

Difference measurement aims to measure the prediction difference in the new data chunk after the incremental update. Let U_0={x_1, x_2, ..., x_n} be the set

of n unlabeled samples in new chunk, h_{t-1} denote the model before update, h_t represent the model after update, $h(\cdot)$=$[p_1, ..., p_m]^T$ be the confidence probability vector for m classes. In this case, we define the matrix H_{t-1}=$[h_{t-1}(x_1), ..., h_{t-1}(x_n)]$ and H_t=$[h_t(x_1), ..., h_t(x_n)]$ as shown in Eqs. (1) and (2).

$$H_{t-1} = [h_{t-1}(x_1), ..., h_{t-1}(x_n)] = [\alpha_{t-1,1}^T, ..., \alpha_{t-1,m}^T]^T \tag{1}$$

$$H_t = [h_t(x_1), ..., h_t(x_n)] = [\alpha_{t,1}^T, ..., \alpha_{t,m}^T]^T \tag{2}$$

In this context, the function $Sim(\cdot)$ can be defined with the utilization of model similarity as:

$$Sim(H_{t-1}, H_t) = \frac{1}{m} \sum_{i=1}^{m} \frac{\alpha_{t-1,i}^T \cdot \alpha_{t,i}}{||\alpha_{t-1,i}|| \cdot ||\alpha_{t,i}||}. \tag{3}$$

In Eq. (3), the cosine similarity between $\alpha_{t-1,i}$ and $\alpha_{t,i}$ is calculated for each class. The mean of cosine similarity of m classes is used to denote the cosine similarity of two models. Clearly, $Sim(h_{t-1}, h_t) \in [-1, 1]$ is always valid. The smaller the value is from 1, the more different the predictions of the two models are, which indicates that more likely concept drift occurs.

2.3 Adaptive Threshold

For drifting samples of the same number, the difference degree of the models after updating varies with the number of samples that have been trained. The more samples have been trained, the smaller the impact of new samples on the model. Therefore, it is necessary to adopt the adaptive threshold.

Given that the model changes greatly when updates at the beginning, the oscillation of cosine similarity will be large at first and then gradually decrease. To this end, we propose *Deviation-Adaptive Threshold* (DAT) which considers the statistical features such as mean and variance. A fixed-size window is adopted to store latest cosine similarity values. At each time stamp, the current average level of cosine similarity is judged according to the mean of the values in the window. The current oscillation level can then be obtained according to the standard deviation. Then the abnormal cosine similarity value can be obtained with such an average level and oscillation level. To understand the whole procedure, the pseudo-code is shown in Algorithm 1.

2.4 Drift Detection

When the new data chunk arrives, it is needed to calculate the cosine similarity between the model updated by the last data chunk and the non-updated model. The window and threshold can then be updated. If the cosine similarity is lower than the threshold, the drift can then be judged to occur. The details of the algorithm can then be summarized in Algorithm 2. And the overall flow chart is summarized as shown in Fig. 2.

Algorithm 1. DAT (Deviation-Adaptive Threshold)

Input: New data $\mathbf{C}_{new}$, previous window $\mathbf{W}_{pre}$, the size of window $\boldsymbol{l}$, deviation coefficient $\boldsymbol{k}$.

Output: Threshold $\mathbf{t}$, new window $\mathbf{W}_{new}$.

if length($\mathbf{W}_{pre}$)<$\boldsymbol{l}$ **then**
    Add $\mathbf{C}_{new}$→$\mathbf{W}_{pre}$
else
    Pop($\mathbf{W}_{pre}$,0) //delete the first element
    Add $\mathbf{C}_{new}$→$\mathbf{W}_{pre}$
$\mathbf{W}_{pre}$→$\mathbf{W}_{new}$
mean($\mathbf{W}_{new}$)→$\overline{x}$
std($\mathbf{W}_{new}$)→σ
$\overline{x}$ - $\boldsymbol{k}\sigma$→$\mathbf{t}$
return $\mathbf{t}$, $\mathbf{W}_{new}$

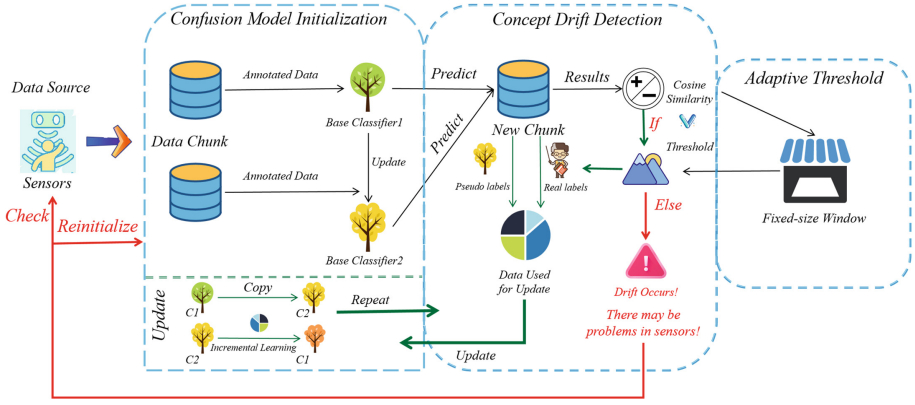


Fig. 2. The overall flow chart.

3 Experimental Analysis

In this section, we empirically compare the proposed method with other state-of-the-art (SOTA) methods for chunk data streams with various concept drifts. The effectiveness of the adaptive threshold setting method in CADM and how it changes with cosine similarity are also illustrated.

3.1 Experimental Settings

Four simulation datasets with decision boundaries of different shapes are selected, which generally contain two variables .The extreme concept drift, i.e. labels of all samples are reversed, is simply introduced (see Fig. 3, 4). Several advanced methods (DWM [10], ARF [11], HT [12], NB [13], CPSSDS [4], OSELM

Algorithm 2. CADM (Confusion And Detection Method)

Input: Data chunk ***stream***={D_i, i=1,...,N,D_i = {$\boldsymbol{x}_j \in \mathcal{X}$, j=1,...,$|D_i|$}}, classfier type $\mathcal{C}$, label ratio λ, the size of window $\boldsymbol{l}$, deviation coefficient $\boldsymbol{k}$.
Output: Drift points list $\boldsymbol{L}_{drift}$.

```
//Confusion model initialization.
C1, C2 = C()
D = stream.next_chunk()
x_labeled = Random(D, λ·|D|)
y_labeled = Give_label(x_labeled)
C1.fit(x_labeled, y_labeled)
C2.fit(x_labeled, y_labeled)
W = [ ]
L_drift = [ ]
while (D = stream.next_chunk())≠∅ do
    H_{t-1} = C1.predict_prob(D)
    H_t = C2.predict_prob(D)
    cos = Sim(H_{t-1},H_t)
    t, W = DAT(cos, W, l, k)    //calculate the threshold and update the window
    if cos < t then
        Add chunk_index → L_drift
        W = [ ]
        C1, C2 = C()
        x_labeled = Random(D, λ·|D|)
        y_labeled = Give_label(x_labeled)
        C1.fit(x_labeled, y_labeled)
        C2.fit(x_labeled, y_labeled)          //drift occurs and reinitialize the models
    else
        x_labeled = Random(D, λ·|D|)
        y_labeled = Give_label(x_labeled)
        x_pseudo = Random(D−{x_labeled}, λ·|D|)
        y_pseudo = Hard_pseudo_label(x_pseudo, C2)
        C1 = C2
        C2.partial_fit([x_labeled, x_pseudo], [y_labeled,y_pseudo])
return L_drift
```

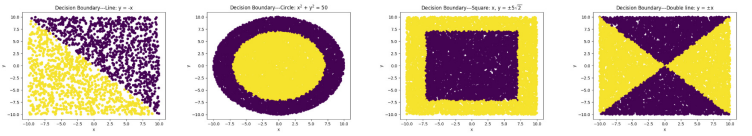
[14], BLS [15]) are selected to be compared, which are implemented by *skmultiflow*[1].

For the parameters of CADM, the size of the window is set at 10, the label ratio is set at 0.2, and the size of the chunk is 200. We set k as 2 by analogy to the probability criterion of Gaussian distribution[2]

3.2 Experimental Results

Drift Detection and Threshold Change. In four simulation datasets, we set an extreme real drift at every 25 chunks, respectively. Naive Bayes [11] is used

[1] https://scikit-multiflow.github.io/.
[2] The code is available at https://github.com/songqiaohu/CADM-Confusion-Model.



(a) Line (b) Circle (c) Square (d) Double lines

Fig. 3. Simulation datasets with different shape-decision boundaries.

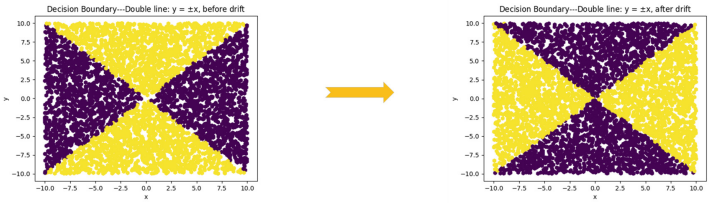


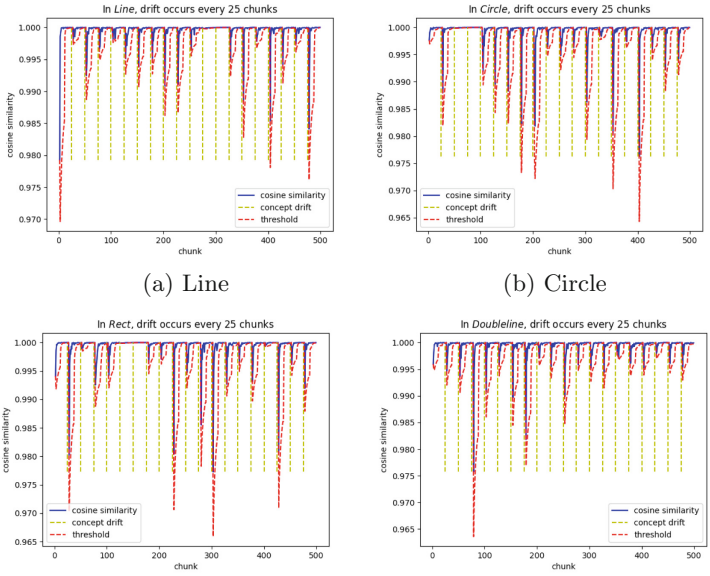
Fig. 4. Distribution of samples and labels before and after drifts.

in CADM for this experiment, where the default parameters provided by *scikit-multiflow* library in Python are considered. The change in cosine similarity and threshold are shown in Fig. 5. The effect of drift detection is reported in Table 1. Clearly, *drift chunk* refers to the chunk where drift begins, *FA* refers to *False Alarm*, and the table content is the position where drift is detected.

Table 1. Drift detection of CADM in different datasets.

Dataset	Drift chunk																			
	26	51	76	101	126	151	176	201	226	251	276	301	326	351	376	401	426	451	476	FA
Line	27	52	77	102	127	152	178	202	227	251	×	×	327	352	377	403	427	452	477	–
Circle	27	×	×	103	127	152	177	202	227	252	277	302	327	352	377	402	427	452	477	–
Square	27	52	77	102	×	×	179	202	227	252	279	302	327	352	377	402	427	452	477	–
Doubleline	27	52	78	102	127	153	177	203	226	252	277	302	327	352	377	402	427	452	477	–

From this experiment, it can be verified that the proposed method has great advantages against false alarms. However, each drift detection has a certain delay, which is caused by two reasons: one is the principle of the method itself. The model should need to be confused before drift can be detected; the other is the randomness of samples. More samples are needed if a small number of samples cannot sufficiently confuse the model.



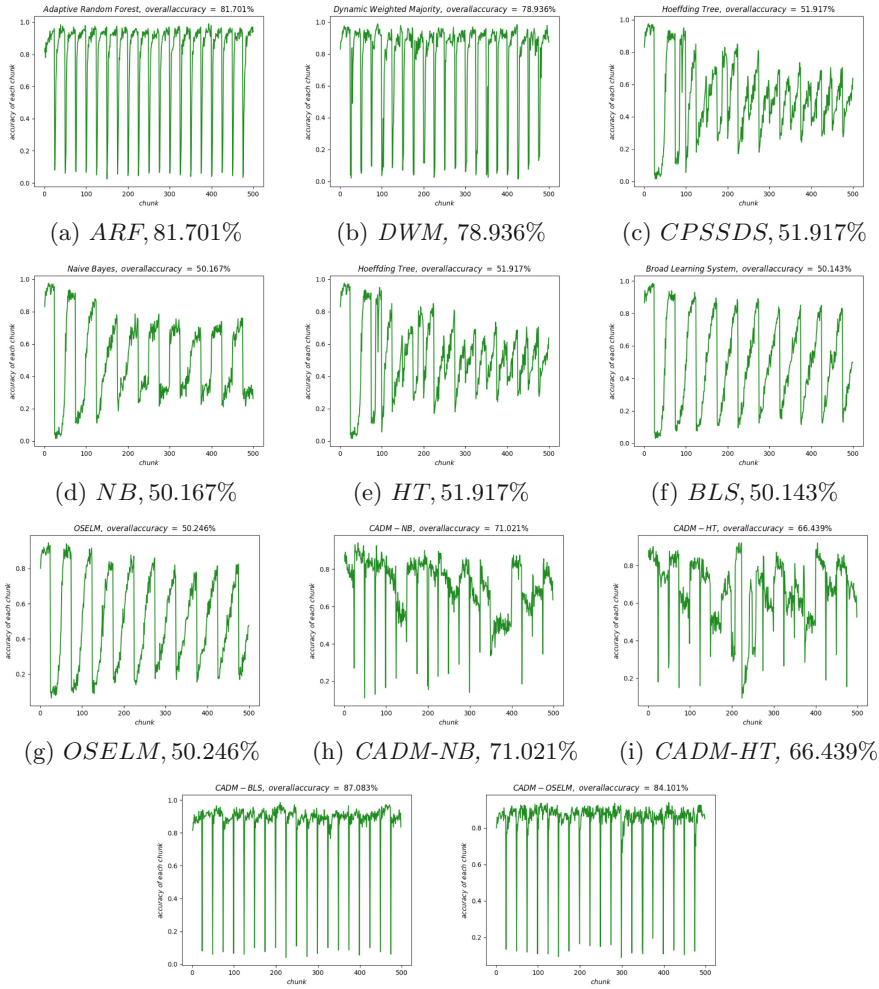
(a) Line (b) Circle

(c) Square (d) Double lines

Fig. 5. Cosine similarity and threshold varies with chunk.

Prediction Accuracy. In this experiment, we compare the performance of the previously mentioned algorithms and their combination with CADM in prediction accuracy. The adjustable parameters of all methods are identical, including training data, label ratio, etc. The dataset used is *Doubleline*, where the decision boundary is nonlinear.

From Fig. 6 and Table 2, the following conclusions can be drawn: **i)**Drifts in simulation datasets frequently occur. Without drift detection, the updating and adaptation speed of *NB, HT, BLS* and *OSELM* is slower than that of drift change, especially in the environment with few labels. Therefore, the accuracy curve almost changes following the drifts as shown in Fig. 6 (a)-(d); **ii)**After combining with *CADM*, all methods can quickly detect and adapt to the drifts. Hence, the accuracy curve can rapidly rise to a high position after a sharp fall. The overall accuracy improves significantly, as shown in Fig. 6 (e)-(h), which also shows the scalability and applicability of *CADM*; **iii)**Among all methods,



(a) *ARF*, 81.701% (b) *DWM*, 78.936% (c) *CPSSDS*, 51.917%

(d) *NB*, 50.167% (e) *HT*, 51.917% (f) *BLS*, 50.143%

(g) *OSELM*, 50.246% (h) *CADM-NB*, 71.021% (i) *CADM-HT*, 66.439%

(j) *CADM-BLS*, 87.083% (k) *CADM-OSELM*, 84.101%

Fig. 6. Prediction accuracy of different methods.

CADM-BLS achieves the highest accuracy, which is higher than *ARF* and *DWM*. *CADM-OSELM* is also higher than *DWM*, which shows that the combinations of *CADM* and some base classifiers outperform the current mainstream algorithms with drift detection in *scikit-multiflow*. **iv)** For *CPSSDS* shown in Fig. 6 (k), its performance is similar to the methods without drift detection. As mentioned in the first section, it can only detect virtual drift rather than real drift.

Table 2. Overall accuracy for all methods in *Doubleline* (over ten runs).

Methods		Accuracy ± std(%)
Methods with drift detection	ARF	**81.164 ± 0.215**
	DWM	**78.973 ± 0.872**
	$CPSSDS$	51.917 ± 0.000
Methods without drift detection	NB	49.846 ± 0.297
	HT	51.969 ± 0.590
	BLS	49.780 ± 0.134
	$OSELM$	49.801 ± 0.193
Methods with $CADM$	$CADM - NB$	65.773 ± 4.338
	$CADM - HT$	64.893 ± 3.881
	$CADM - BLS$	**86.851 ± 1.156**
	$CADM - OSELM$	**79.293 ± 5.780**

4 Conclusion

In this paper, we have proposed the *CADM* to deal with real concept drift with limited annotations. *CADM* updates the model with manually annotated samples and pseudo-label-samples predicted by the model simultaneously, which can improve the performance of the model when there is no drift and cause confusion when there is drift. Then, cosine similarity and adaptive threshold have also been proposed to judge whether drift occurs. We have verified its effectiveness and superiority in extreme-drifting simulation datasets by combining CADM with different classifiers.

Acknowledgment. This work was supported by National Natural Science Foundation of China under Grant 61733009, National Key Research and Development Program of China under Grant 2017YFA0700300, and Huaneng Group science and technology research project.

References

1. Liu, Z., Deng, Y., Zhang, Y., Ding, Z., He, X.: Safety assessment of dynamic systems: an evidential group interaction-based fusion design. IEEE Trans. Instrum Measur. **70**, 1–14 (2021)
2. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. In: Proceedings of the 17th Brazilian Symposium On Artificial Intelligence, pp. 286–295 (2004)
3. Bifet, A., Gavalda, R.: Learning from time-changing data with adaptive windowing. In: Proceedings of the 2007 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, pp. 443–448 (2007)

4. Tanha, J., Samadi, N., Abdi, Y., et al.: CPSSDS: conformal prediction for semi-supervised classification on data streams. Inf. Sci. **584**, 212–234 (2022)
5. Lu, Y., Cheung, Y.M., Tang, Y.Y.: Adaptive chunk-based dynamic weighted majority for imbalanced data streams with concept drift. IEEE Trans. Neural Netw. Learn. Syst. **31**(8), 2764–2778 (2019)
6. Liu, Z., Zhang, Y., Ding, Z., He, X.: An online active broad learning approach for real-time safety assessment of dynamic systems in nonstationary environments. IEEE Trans. Neural Netw. Learn. Syst., (2022)
7. Siffer, A., Fouque, P.A., Termier, A., et al.: Anomaly detection in streams with extreme value theory. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1067–1075 (2017)
8. Sethi, T.S., Kantardzic, M., Hu, H.: A grid density based framework for classifying streaming data in the presence of concept drift. J. Intell. Inf. Syst. **46**(1), 179–211 (2016)
9. Losing, V., Hammer, B., Wersing, H.: Incremental on-line learning: a review and comparison of state of the art algorithms. Neurocomputing **275**, 1261–1274 (2018)
10. Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: an ensemble method for drifting concepts. J. Mach. Learn. Res. **8**, 2755–2790, December (2007). ISSN 1532–4435
11. Gomes, H.M., et al.: Adaptive random forests for evolving data stream classification. Mach. Learn. **106**(9), 1469–1495 (2017). https://doi.org/10.1007/s10994-017-5642-8
12. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: KDD'01, pp. 97–106. ACM Press, San Francisco, CA (2001)
13. Murphy, K.P.: Naive Bayes classifiers. Univ. British Columbia **18**(60), 1–8 (2006)
14. Ding, S., Zhao, H., Zhang, Y., et al.: Extreme learning machine: algorithm, theory and applications. Artif. Intell. Rev. **44**(1), 103–115 (2015)
15. Chen, C.L.P., Liu, Z.: Broad learning system: an effective and efficient incremental learning system without the need for deep architecture. IEEE Trans. Neural Netw. Learn. Syst. **29**(1), 10–24 (2017)