Semi-supervised Classification on Data Streams with Recurring Concept Drift Based on Conformal Prediction

ShiLun Ma[1], Wei Kang[1], Yun Xue[2], and YiMin Wen[1(✉)]

[1] Guangxi Key Laboratory of Image and Graphic Intelligent Processing, Guilin University of Electronic Technology, Guilin 541004, China
ymwen@guet.edu.cn

[2] School of Municipal and Surveying Engineering, Hunan City University, Yiyang 413000, China

Abstract. In this article, we consider the problem of semi-supervised data stream classification. The main difficulties of data stream semi-supervised classification include how to jointly utilize labeled and unlabeled samples to adress concept drift detection and how to use unlabeled to update trained classifier. Existing algorithms like the CPSSDS method constantly retrain a new classifier when concept drift is detected, it is very consuming and wasteful. In this paper, the algorithm of data stream semi-supervised classification with recurring concept drift named as CPSSDS-R is proposed. First, the labeled samples in the first data block are used to initialize a classifier, which is added into a pool and actived for classification. While a new data block arrives, concept drift is detected by computing conformal prediction results. If no concept drift is detected, the pseudo-labeled samples in the previous data block are added with the labeled samples in the current data block to incrementally train the active classifier. If a new concept is detected, a new classifier is trained on the labeled samples of the current data block and added into the pool and actived for classification, else if a recurring concept is detected, the pseudo-labeled samples and labeled samples in the current data block are used to incrementally update the classifier corresponding to the recurring concept in the pool and actived for classification. The proposed algorithm is tested on multiple synthetic and real datasets, and its cumulative accuracy and block accuracy at different labeling ratios demonstrate the effectiveness of the proposed algorithm. The code for the proposed algorithm is available on https://gitee.com/ymw12345/cpssds-r.

Keywords: Semi-supervised classification · Ensemble learning · Model reuse · Concept drift

Supported by the National Natural Science Foundation of China (62366011), the Key R&D Program of Guangxi under Grant (AB21220023), and Guangxi Key Laboratory of Image and Graphic Intelligent Processing (GII P2306).

1 Introduction

With the development of technology, a large amount of data are generated in form of data stream from real computing devices [1]. For example, applications like social networks, network intrusion detection, and weather forecasts [2]. More seriously, the number of labeled data is limited in many cases, the use of machine learning methods to effectively handle semi-supervised data stream classification is very challenging.

In CPSSDS [3], conformance prediction is used to calculate the confidence of unlabeled samples in adjacent data blocks and used for concept drift detection. If concept drift occurs, the existing model is eliminated, and a new model is trained on the current data block. One of the main advantages of this paper is that it uses conformal prediction to both concept drift detection and self-labeled data selection for updating the trained modal. However, when a concept reappears, a new model needs to be retrained, and no historical model cannot be reused [4], which affects the classification accuracy of CPSSDS.

In response to the above problem, this paper proposes a semi-supervised classification algorithm called CPSSDS-R based on CPSSDS [3]. The proposed algorithm maintains a classifier pool, which stores classifiers trained using different concept data. It detects whether a concept drift has occurred between the current data block and the previous data block. If so, the conformance prediction output of the current data block is compared with the conformance prediction outputs of the component classifiers in the classifier pool to detect reoccurring concept. If detected, the component classifiers corresponding to the reoccurring concept is updated and actived for classification, otherwise, initialize a new classifier on the labeled data of the current data block for classification and add it into the pool. The main innovations of this paper are as follows:

1) Combining the idea of ensemble learning, maintaining a classifier pool that retains classifiers trained with different concept data, thus solving the problem of CPSSDS's inability to improve classification accuracy using historical model.
2) Proposing a method for detecting reoccurring concept based on conformance prediction. By comparing the conformal Prediction prediction outputs of the component classifiers in the classifier pool with the conformal Prediction prediction output of the current data block, recurring concept drift can be effectively detected.

2 Related Work

In this section, we first introduce algorithms related to concept drift detection and then discuss related work on semi-supervised data stream classification.

Concept drift detection algorithms identify change points or change time intervals by quantifying statistical measures of change [5]. Based on the test statistics they use, they can be roughly divided into two categories: methods based on classifier accuracy and methods based on data distribution. In accuracy-based drift detection methods [6], changes in performance are detected through statistical tests. The second category of detection algorithms quantifies the dissimilarity between previous and current data distributions [7]. The disadvantage of accuracy-based drift detection methods is that concept drift can only be detected when the classifier's performance declines. This article proposes a distribution-based concept drift detection method that utilizes conformed prediction output of unlabeled data. Recurring concepts can be detected.

Wen et al. first proposed a review on semi-supervised data stream classification algorithms [8]. Next, we will introduce some of the more classical semi-supervised data stream classification algorithms. In semi-supervised classification algorithms for data streams, SmSCluster [9] and ReaSC [10] adopt a semi-supervised approach to construct a collection of classifiers for classification, discarding the worst-performing models from both the old and newly trained models. Going further, SPASC [11] uses the EM algorithm to update classifiers based on labeled data in data blocks and performs sequential classification on instances in the block through a dynamic weighting adjustment mechanism. Addressing the limitations of SPASC, Liu et al. [12] proposed SCBELS, which uses BIRCH ensembles and local structure mapping. [13] maintains a set of clusters as learning models to capture the distribution of data streams and uses KNN for classification of data streams. Each cluster is assigned a weight that dynamically changes based on real-time classification accuracy. Another algorithm similar to SSE-PBS [14] is a high-confidence-based unlabeled sample selection strategy, which can effectively mine and utilize information from unlabeled samples and implicitly adapt to concept drift. Zheng et al. [15] proposed the ESCR algorithm, which calculates the Jensen-Shanon divergence between two distributions, detects significant changes in classifier confidence, and detects repeated concept drift. Khezri et al. [16] proposed the STDS algorithm based on a self-training algorithm, which uses the Kullback-Leibler divergence to measure the distribution difference between consecutive blocks and detect concept drift. Tanha et al. [3] proposed CPSSDS, which uses incremental classifiers as base learners and a self-training framework to augment labeled samples, thereby addressing the scarcity of labeled samples.

3 Proposed Algorithm

This section introduces the algorithm framework of CPSSDS-R, and then provides a more detailed description of the proposed method.

3.1 CPSSDS-R

Like CPSSDS, the CPSSDS-R algorithm is also a new semi-supervised self-training [17] data stream classification framework. In a semi-supervised data stream environment, only a small portion of samples in the data chunk may be labeled, and a data stream is represented as D_1, D_2, D_3, ..., D_∞. Each data

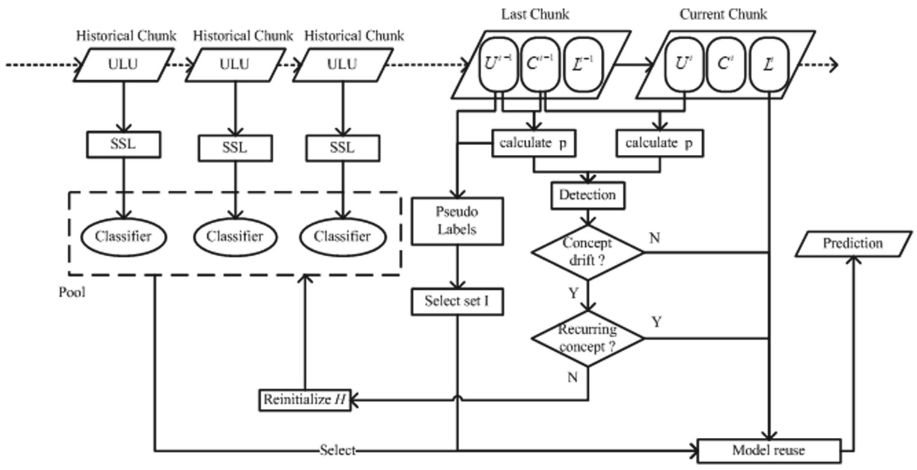

Fig. 1. The framework of CPSSDS-R algorithm

block consists of labeled samples and unlabeled samples, and the data block is divided into a training set (80%) and a test set (20%), where the labeled samples are further divided into a labeled sample set L (70%) and a validation set C (30%). The CPSSDS-R algorithm framework is shown in Fig. 1, and its technical details will be provided in the following sections. Table 1 provides commonly used symbols.

The pseudocodes for the CPSSDS-R algorithm are shown in Algorithm 1. The algorithm uses an incremental Hoeffding tree as base classifier. The steps of the algorithm are as follows: lines 1–2 initialize the model and classifier pool. Lines 6–7 perform inductive conformance prediction on the unlabeled samples of adjacent data blocks, with the method shown in the Algorithm 2. Line 8 indicates that concept drift detection is performed as shown in Algorithm 3. Line 9–10 indicates that if concept drift has occurred, Then Algorithm 4 is used for recurring concept drift detection. Lines 12–16 indicate that if a reoccurring concept is detected, the component classifier with highest similarity to the current data block concept is selected and incrementally updated. Lines 18–22 indicate that if no recurring concept is detected, a new classifier is created using the labeled samples from the current data block and added into the classifier pool. Lines 24–25 indicate that if the test value is greater than or equal to 0.05, the Algorithm 5 is used to select a pseudo-labeled sample set with high information gain and add it to the labeled sample set of the current data block. Line 26 indicates that the model is being updated incrementally.

3.2 Inductive Conformal Prediction

Inductive Conformal Prediction (ICP) proposed in CPSSDS [3] is often used as a framework to determine the set of prediction labels for the relevant confidence

Table 1. Symbols Table

Variables	Descriptions
D_t	The t^{th} data chunk
L^t	Labeled set of the t^{th} chunk
C^t	Calibration set of the t^{th} chunk
U^t	Unlabeled set of the t^{th} chunk
A	Non-conformity measure
H^t	Trained model on the t^{th} chunk
M	Classifier pool
Y	class set
ϵ	Significance level
p_t	p-values of U^t
p_i	the p-value of the i^{th} classifier in the M
p_t^y	The p-value of the sample in the t^{th} data block with predicted category y
p_{t,x_i}^y	p-values of pair (x_i, y) in the t^{th} chunk
S_u	Confidence level for predicted label of sample u
$\alpha_{x_i}^y$	Non-conformity value of pair (x_i, y)
I	Informative samples set

values of test samples. The core components of ICP are the inconsistency function and p-values. The inconsistency function measures the degree of consistency of each (x, y) pair relative to other samples. The latter value measures the proportion of training samples that are different from the class y for a new sample. Next, we will introduce the relevant steps of ICP. The Algorithm 2 presents its pseudocode.

1) Using the labeled sample set L in the data chunk to train an incremental Hoeffding Tree classifier;
2) Calculate the inconsistency value $\alpha_{x_c}^y$ of the calibration set C. The calculation is shown in the Eq. (1),

$$\alpha_{x_i}^y = 1 - \frac{1 + g(y)}{K + \sum_{j=1}^{K} g(y_j)} \tag{1}$$

Where $g(y)$ is the number of samples belonging to the y class, which are in the same cotyledon as x_i, and K is the number of samples for class y.;
3) Calculate the inconsistency value $\alpha_{x_i}^y$ of each sample in the unlabeled sample set one by one, and then compare it with the inconsistency value of the calibration set. Calculate p-values using the Eq. (2),

$$p_{x_i}^y = \frac{\left|\left\{c = 1, \ldots, n \mid \alpha_{x_c}^y \geq \alpha_{x_i}^y\right\}\right|}{n} \tag{2}$$

Where n is the number of in the calibration set, and the superscript y represents the p-value of sample x_i as class y.
4) Select the unlabeled samples with p-value greater than or equal to the significance level parameter ϵ and label them with pseudo labels.

Algorithm 1: CPSSDS-R

```
Input: $D = \{D_1, D_2, \cdots, D_t, \cdots\}$, significance level $\epsilon$,$Y$
1  Initialize the classifier $H^1$ based in $L^1$, $M = \emptyset$
2  $M \leftarrow M \cup H^1$
3  $t \leftarrow 2$
4  while data chunk $D_t$ is available do
5      $p_{t-1} \leftarrow ICP(H^{t-1}, C^{t-1}, U^{t-1})$
6      $p_t \leftarrow ICP(H^{t-1}, C^{t-1}, U^t)$
7      $drift \longleftarrow Drift\_Detection(p_{t-1}, p_t, Y)$
8      if $drift$ then
9          $recurring\_concept = Recurring\_Detection(M, U^t, Y)$
10         if $recurring\_concept$ then
11             $I = select\_information\_sample(U^t)$
12             $L^t \leftarrow L^t \cup I$
13             $H^{'} \leftarrow \text{argmax}_{H^i \in M}(p_i, p_t)$
14             $H^{'} \leftarrow incremental\_train(H^{'}, L^t)$
15             $H^{'}$ actived for classification
16         else
17             Initialize $H^t$ based on $L^t$
18             if $len(M) == max\_pool\_size$ then
19                 remove the earliest classifier in $M$
20                 $M \leftarrow M \cup H^t$
21                 $H^t$ actived for classification
22     else
23         $I = select\_information\_sample(U^{t-1})$
24         $L^t \leftarrow L^t \cup I$
25         $H^t \leftarrow incremental\_train(H^{t-1}, L^t)$
26     $t \leftarrow t + 1$
```

3.3 Concept Drift Detection

Concept drift detection consists of two steps. The first step is to detect whether there is a concept drift between the current data block and the previous concept drift. The second step is to detect whether the current data block is a recurring concept.

The Algorithm 3 presents the process of concept drift detection. Line 4 represents using the KS (Kolmogorov Smirnov) test for each class y. Lines 5–6 represents calculating the p-value distribution of the adjacent unlabeled sample sets of two data blocks, then accumulating them. Lines 7–8 indicate that concept drift is detected if $R/|Y|$ is less than 0.05.

The pseudocode for recurring concept drift detection is presented in the Algorithm 4. Line 3 computes the p-value of each unlabeled sample in U^i to each class label for the classifier H^i and the calibration set C^i, while Line 4 computes the p-value of each unlabeled sample in U^t to each class label for the classifier

Algorithm 2: *ICP*

```
   Input: $H$, $C$, $U$
   Output: p-values
1  $p\_values = \emptyset$
2  $C = \{Z_1, Z_2 \cdots Z_n\} = \{(x_1, y_1), (x_2, y_2), \cdots (x_n, y_n)\}$
3  foreach $x_i \in U$ do
4      foreach $class \quad y \in Y$ do
5          $N_i = (x_i, y)$
6          $\alpha_{x_i}^y = 1 - \frac{1+g(y)}{K+\sum_{j=1}^{K} g(y_j)}$
7          foreach $x_c \in C$ do
8              $\alpha_{x_c}^y = 1 - \frac{1+g(y)}{K+\sum_{j=1}^{K} g(y_j)}$
9      $p_{x_i}^y = \frac{\left|\left\{c=1,\ldots,n \mid \alpha_{x_c}^y \geq \alpha_{x_i}^y\right\}\right|}{n}$
10 return $p\_values$
```

Algorithm 3: *Drift_Detection*

```
   Input: $p_{t-1}, p_t, Y$
   Output: $ks$
1  $boolean \quad drift = false$
2  $R = 0$
3  foreach $y \in Y$ do
4      $pVal \leftarrow KStest(p_{t-1}^y, p_t^y)$
5      $R \leftarrow R + pVal$
6  if $(R/|Y| < 0.05)$ then
7      $drift = true$
8  return $drift$
```

H^i and the calibration set C^i. Lines 5–7 represent that after all component classifiers are computed, the *index* and *mas_ks* value of the component classifier that is most similar to the concept of the current data block are obtained. Lines 8–9 represent that if the *mas_ks* value is greater than 0.1, the *index* value is returned, indicating the detection of a recurring concept.

3.4 Information Sample Selection Strategy

The key step in the self-training framework is the information sample selection step, where the most reliable unlabeled samples are chosen to improve the generalization ability of the classifier. The pseudocode for selecting samples is shown in Algorithm 5. For each sample in the unlabeled sample set, line 2 represents calculating the largest p-value for all class labels. Line 3 represents calculating the confidence of the sample belonging to class y. Lines 4–5 represent that if the confidence is greater than the significance level, the sample is added to the

Algorithm 4: *Recurring_Detection*

```
   Input: $M$, $U^t$, $Y$
   Output: classifier index
1  $index = 0, max\_ks = 0$
2  foreach $H^i \in M$ do
3      $p_i \leftarrow ICP(H^i, C^i, U^i)$
4      $p_t \leftarrow ICP(H^i, C^i, U^t)$
5      $ks \leftarrow ks\_test(p_i, p_t, Y)$
6      if $max\_ks < ks$ then
7          $index = i$
8          $max\_ks = ks$
9  if $mas\_ks > 0.1$ then
10     return $index$
```

Algorithm 5: *select_information_sample*

```
   Input: $U$
   Output: $I$
1  foreach $x_u \in U$ do
2      $y_u = argmax_y \left\{p^y_{t,x_u}\right\}$
3      $S_u = p^{y_u}_{t,x_u}$
4      if $S_u \geq \epsilon$ then
5          $I = I \cup x_u$
6  return $I$
```

informative sample set. Finally, after all unlabeled samples are calculated, the informative sample set is return.

4 Experiments

4.1 Datasets

To verify the performance of the algorithm, we used 3 real and 6 synthetic datasets [18]. All synthetic datasets are class balanced. The detailed information of these 9 datasets is shown in Table 2.

4.2 Experimental Setup

We use a block-based model to evaluate the proposed framework, where a fixed portion of instances is selected as labeled at the beginning of each block's training set, set to 10% in the experiments. The experiments are conducted under the following parameter settings: for CPSSDS-R, the significance level for adaptive predictions is tested and adjusted for each dataset. The parameters for comparing algorithms are fixed and use the default values from the original paper. All experiments in the paper are the average results of 10 runs.

Table 2. Properties of the Datasets

datasets	Attributes	Instances	Classes	Chunk Size
Agrawal	9	130000	2	5000
RandomRBF	10	300000	4	5000
Stagger	3	250000	2	5000
FG2C2D	2	200000	2	5000
GEAR2C2D	2	200000	2	5000
MG2C2D	2	200000	2	5000
Electricity	8	45312	2	5000
SLDD	48	58509	11	5000
Shuttle	9	845228	10	4000

4.3 Algorithm Accumulative Accuracy and Result Analysis

In this section, the accumulative accuracy of CPSSDS-R algorithm compared to CPSSDS algorithm on 9 datasets with different labeled ratio was analyzed. The average results of ten runs are shown in Table 3, and paired t-tests with a 95% confidence level are performed on the results. In order to observe the performance of the proposed algorithm more intuitively, the Run chart of the accumulative accuracy of the algorithm on each dataset versus the number of data chunks is shown in Fig. 2.

Table 3. Accuracy comparison for CPSSDS and CPSSDS_R on the experimental datasets with a 10% labeling ratio.

datasets	CPSSDS	CPSSDS-R
Agrawal	53.01	**55.24**
RandomRBF	57.79	**58.98**
Stagger	91.75	**97.28**
FG2C2D	88.94	**91.14**
GEAR2C2D	96.07	**96.79**
MG2C2D	91.37	**92.17**
Electricity	65.16	**65.41**
SLDD	46.45	**49.51**
Shuttle	90.51	**92.49**

The cumulative accuracy comparison of datasets with a labeling rate of 10% is shown in Table 3. It can be seen that, on all datasets, the cumulative accuracy of the CPSSDS-R algorithm is significantly better than that of the CPSSDS algorithm. Figure 2 shows the cumulative accuracy comparison of algorithms for

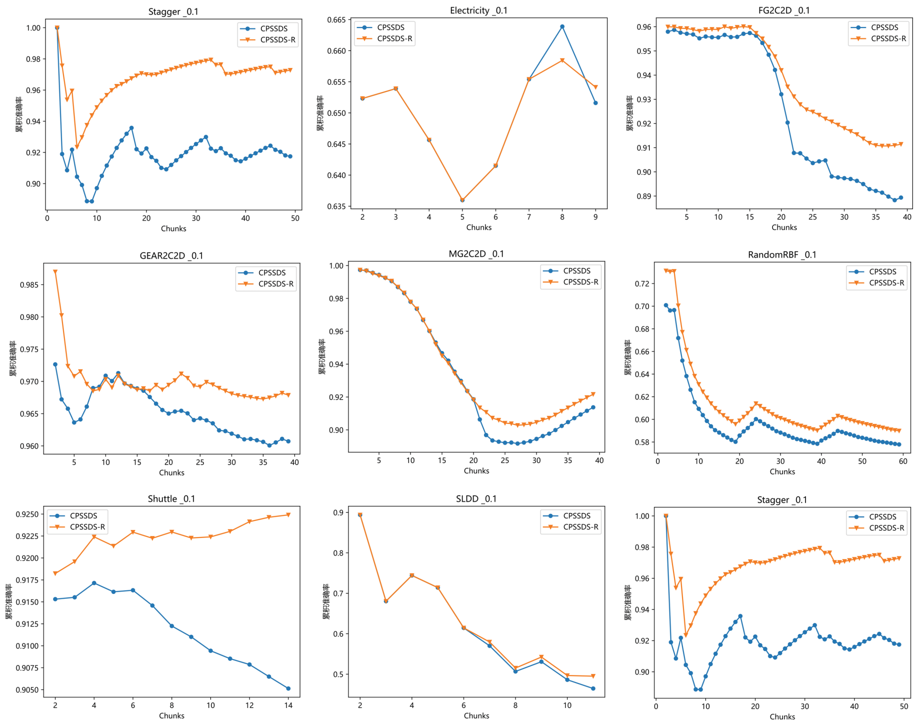

Fig. 2. Accuracy comparison on the experimental datasets with 10% labeled each

each dataset with a labeling rate of 10%. From the figure, it can be observed that, on the synthetic datasets, the cumulative accuracy of the proposed algorithm is consistently higher than that of the CPSSDS algorithm. On the real-world datasets, however, due to the unknown location and type of concept drift, the proposed algorithm has slightly lower cumulative accuracy than the CPSSDS algorithm in the early stages. However, as the number of data blocks increases, the cumulative accuracy of CPSSDS-R gradually surpasses that of the CPSSDS algorithm, which is more pronounced in the Shuttle dataset.

In Tables 4, the CPSSDS-R algorithm outperforms other algorithms in classification performance on most datasets, indicating that detecting recurring concept drift through conformal prediction can more effectively address the problem of continuous model initialization, reduce the cold start problem when recurring concepts, and achieve better classification performance on different datasets. But on the GEAR2C2D dataset, the accuracy of the SCBELS algorithm is higher because the component classifier of the SCBELS algorithm is an integrated classifier, which performs better on this dataset. In contrast, the CPSSDS-R algorithm has a slightly lower accuracy than the SCBELS algorithm due to its ability to detect recurrence concepts in a timely manner.

Table 4. Accuracy comparison with other semi-supervised classification algorithms on the experimental datasets with a 10% labeled each

datasets	SPASC	SCBELS	CPSSDS-R
Agrawal	49.89	51.81	**55.24**
RandomRBF	55.11	55.39	**58.98**
Stagger	80.89	78.38	**97.28**
FG2C2D	79.98	88.11	**91.14**
GEAR2C2D	94.45	**97.35**	96.79
MG2C2D	55.27	84.86	**92.17**
Electricity	53.56	58.96	**65.41**
SLDD	18.16	32.95	**49.51**
Shuttle	91.16	88.56	**92.49**

5 Conclusions

In this paper, we propose the semi-supervised data stream classification algorithm CPSSDS-R. Firstly, the Hoeffding tree model is trained using labeled sample sets in data chunk, and on the basis of the self training framework, conformal prediction is used to label unlabeled samples. Samples with high information content are selected to expand the labeled sample set. Secondly, by maintaining a classifier pool and calculating the KS statistical values of the current data chunk and component classifiers in the classifier pool, the recurrent concept can be detected, and the occurrence of recurrent drift detected before the classifier can be updated. The appropriate component classifier can be selected in a timely manner to adapt to the recurrence concept drift. Finally, the algorithm is validated on multiple synthetic and real datasets for its effectiveness.

Future work include: on the one hand, the classifier trained by incremental Hoeffding tree is always updated incrementally, which may lead to overfitting. It should consider designing a reasonable pruning strategy based on the characteristics of semi-supervised data stream to reuse trained model more effectively. On the other hand, based on conformal prediction, the algorithm selects pseudo labeled samples with high confidence to expand the labeled sample set. Generally, samples far away from the Decision boundary are used to update the classifier. Failure to take into account the global distribution may lead to a decline in the performance of classifier.

References

1. Kuo, Y.-H., Kusiak, A.: From data to big data in production research: the past and future trends. Int. J. Prod. Res. **57**(15–16), 4828–4853 (2019)
2. Tan, C.H., Lee, V.C., Salehi, M.: Information resources estimation for accurate distribution-based concept drift detection. Inf. Process. Manage. **59**(3), 102911 (2022)

3. Tanha, J., Samadi, N., Abdi, Y., Razzaghi-Asl, N.: CPSSDS: conformal prediction for semi-supervised classification on data streams. Inf. Sci. **584**, 212–234 (2022)
4. Zhao, P., Zhou, Z.: Learning from distribution-changing data streams via decision tree model reuse. Scientia Sinica Informationis **51**(1), 1–12 (2021)
5. Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G.: Learning under concept drift: a review. IEEE Trans. Knowl. Data Eng. **31**(12), 2346–2363 (2018)
6. Lu, N., Lu, J., Zhang, G., De Mantaras, R.L.: A concept drift-tolerant case-base editing technique. Artif. Intell. **230**, 108–133 (2016)
7. Gu, F., Zhang, G., Lu, J., Lin, C.-T.: Concept drift detection based on equal density estimation. In: 2016 International Joint Conference on Neural Networks (IJCNN), pp. 24–30. IEEE (2016)
8. Wen, Y., Liu, S., Miao, Y., Yi, X., Liu, C.: Survey on semi-supervised classification of data streams with concepts. J. Softw. **33**(4), 1287–1314 (2022)
9. Masud, M.M., Gao, J., Khan, L., Han, J., Thuraisingham, B.: A practical approach to classify evolving data streams: training with limited amount of labeled data. In: 2008 Eighth IEEE International Conference on Data Mining, pp. 929–934. IEEE (2008)
10. Masud, M.M., et al.: Facing the reality of data stream classification: coping with scarcity of labeled data. Knowl. Inf. Syst. **33**, 213–244 (2012)
11. Hosseini, M.J., Gholipour, A., Beigy, H.: An ensemble of cluster-based classifiers for semi-supervised classification of non-stationary data streams. Knowl. Inf. Syst. **46**, 567–597 (2016)
12. Wen, Y.-M., Liu, S.: Semi-supervised classification of data streams by birch ensemble and local structure mapping. J. Comput. Sci. Technol. **35**, 295–304 (2020)
13. Din, S.U., Shao, J., Kumar, J., Ali, W., Liu, J., Ye, Y.: Online reliable semi-supervised learning on evolving data streams. Inf. Sci. **525**, 153–171 (2020)
14. Khezri, S., Tanha, J., Ahmadi, A., Sharifi, A.: A novel semi-supervised ensemble algorithm using a performance-based selection metric to non-stationary data streams. Neurocomputing **442**, 125–145 (2021)
15. Zheng, X., Li, P., Hu, X., Yu, K.: Semi-supervised classification on data streams with recurring concept drift and concept evolution. Knowl.-Based Syst. **215**, 106749 (2021)
16. Khezri, S., Tanha, J., Ahmadi, A., Sharifi, A.: STDS: self-training data streams for mining limited labeled data in non-stationary environment. Appl. Intell. **50**, 1448–1467 (2020)
17. Zhu, X., Goldberg, A.B., Brachman, R., Dietterich, T.: Synthesis lectures on artificial intelligence and machine learning. In: Introduction to Semi-Supervised Learning, vol. 3, no. 1, pp. 1–130. Morgan & Claypool (2009)
18. Montiel, J., Read, J., Bifet, A., Abdessalem, T.: Scikit-multiflow: a multi-output streaming framework. J. Mach. Learn. Res. **19**(1), 2914–2915 (2018)