Semi-supervised Classification of Data Streams Based on Adaptive Density Peak Clustering

Changjie Liu[1], Yimin Wen[1(✉)], and Yun Xue[2]

[1] Guangxi Key Laboratory of Image and Graphic Intelligent Processing, Guilin University of Electronic Technology, Guilin, China
techat17@163.com, ymwen2004@aliyun.com

[2] School of Municipal and Surveying Engineering, Hunan City University, Yiyang, China
yunxue1209@163.com

Abstract. In the real-world scenario of data stream classification, label scarcity is very common. More challenges are data streams always include concept drifts. To handle these challenges, an algorithm of semi-supervised classification of data streams based on adaptive density peak clustering (SSCADP) is proposed. In SSCADP, to generate concept clusters at leaves in a Hoeffding tree, a density peak clustering method and a change detection technique are combined to adaptively locate the clustering centers. Concerning concept drift detection, we argue that the change of cluster with higher density more likely reflect the change of data distribution. Hence, to detect concept drifts, an adaptive weighting method for density change detection is proposed to calculate the deviations between the history concept clusters and new ones. Experiments on synthetic and real datasets confirm the advantages of SSCADP.

Keywords: Semi-supervised classification · Data stream · Decision tree · Clustering · Concept drift

1 Introduction

The classification of data steams with concept drift is one of the main challenges of data mining [1–3]. In various real-applications, including network intrusion detection, spam filtering and credit card fraud detection[4] etc., due to labeling cost and time consuming, it is unrealistic that all instances are labeled by expert. Therefore, a semi-supervised classification algorithm which can handle concept drifts plays a critical role in addressing the issue of data stream mining.

To overcome these challenges, many researches have been reported in recent years. However, there are still some limitations. Firstly, many existing methods [5,6] commonly take clustering methods to label the unlabeled instances which should assign the number of clusters in advance and keep it constant during the

H. Yang et al. (Eds.): ICONIP 2020, LNCS 12533, pp. 639–650, 2020.
https://doi.org/10.1007/978-3-030-63833-7_54

processing of data streams. Secondly, many methods ignore the impact of high density clusters on concept drift detection [5,7].

Specifically in SUN [5], firstly, K-modes is used to form clusters and label the unlabeled data. However, using K-modes requires setting the number of clusters in advance and keeping it unchanged during the processing of data streams, which is unreasonable in many real-applications. Since the dynamic feature of data streams, new concept clusters may appear while old may disappear. Secondly, the average deviation between the history and new concept clusters is adopted to detect concept drifts, which ignores changes in high-density clusters which are more likely result in concept drifts.

In light of these limitations, an approach of **s**emi-**s**upervised **c**lassification of data streams based on **a**daptive **d**ensity **p**eak clustering (SSCADP) is proposed. The framework of SSCADP is the same as SUN, but there are two main differences.

Firstly, to generate concept clusters at the leaves in a Hoeffding tree, a density peak clustering method [8] and a change detection technique [9] are combined to adaptively locate the cluster centers, instead of using K-modes. Secondly, we consider that the change of clusters with higher density is more likely to reflect the change of data distribution. And hence an improved detection method based on SUN is proposed that an adaptive weighted average strategy is adopted to assign higher weight value to the higher density clusters.

The rest of this paper is organized as follows. Section 2 presents some related work. Section 3 describes the proposed algorithm in detail including adaptively locating the cluster centers, labeling the unlabeled samples, and concept drift detection method. Experiments and results are shown in Sect. 4. Finally, Sect. 5 summarizes this paper and future work.

2 Related Work

The proposed approaches for semi-supervised classification of data stream with concept drifts can be broadly divided into decision tree-based and non-decision tree-based methods. The decision tree-based methods like SUN [5] and REDLLA [7] adopt a Hoeffding tree as base classifier. During the construction of the base classifier, unlabeled data are labeled by clustering in leaves, and then added into the detection of concept drifts and the updating of the base classifier. Concept drifts are detected based on the deviation between history concept clusters and the new ones. Others like Sco-forest [10] extends Co-forest algorithm to handle evolving data streams. The concomitant ensemble is used to select samples with high classification confidence and label these samples to update the corresponding base classifier. If a concept drift is detected by Adwin2 [11], the base classifier with the worst accuracy will be discarded.

The non-decision tree-based methods usually take cluster model for data streams classification. Reasc [12] maintains an ensemble of cluster-based classifiers. When updating the ensemble, the cluster-based classifier with the worst accuracy will be removed out. SPASC [6] maintains a classifier pool which is

composed of weighted clusters-based model. The weight value will be adjusted adaptively according to the correctness of classification. SCBELS [13] maintains an ensemble of cluster-based classifiers which are constructed by BIRCH [14] and incrementally updated during the classification of data stream. Local structural information of data is taken into account to deal with concept drifts. [15] proposed to dynamically maintain a set of micro-clusters, each instance is used to update the model, outdated or micro-clusters with low reliability are removed to adapt to the evolving concepts of data streams.

Other models like ECU [16], it constructs an ensemble model which combines both classifiers and clusters for classification. [17] proposed a neural network framework for streaming data classification that each layer consists of a generative network, a discriminant structure and a bridge.

3 Proposed Algorithm

3.1 The Framework of the Proposed Algorithm

In this paper, a data stream is represented as $\boldsymbol{D} = \{\boldsymbol{D}^0, \boldsymbol{D}^1, \boldsymbol{D}^2 \ldots, \boldsymbol{D}^t, \ldots\}$, in which $\boldsymbol{D}^t = \{\boldsymbol{x}_1^t, \boldsymbol{x}_2^t, \ldots, \boldsymbol{x}_m^t\}$ indicates the data batch collected at the time t. SSCADP[1] is described in the algorithm 1. It employs a Hoeffding tree as its base classifier. After $\boldsymbol{D}^t$ is classified by the Hoeffding tree, each instance in it is sorted into a leaf, the corresponding statistics of the leaf are updated. If the number of instances arrived at the tree meets dp, all the labeled instances in a leaf l are utilized to label the unlabeled instances in it, and then concept drift detection is installed to detect drift at the leaf. Then, a pruning strategy is adopted on the Hoeffding tree, and if the number of instances in a new leaf meet n_{min}, the leaf attempts to split. After splitting, the updated Hoeffding tree is ready for new concept.

3.2 Adaptively Locate Cluster Centers and Label Unlabeled Instances

If a detection period is reached, a clustering method named Clustering by fast search and find of density peaks (CFSDP) [8] and a change detection technique [9] are combined to adaptively locate the cluster centers. After concept clusters at each leaf are created, graph-based label propagation [18] is installed to label the unlabeled data in each cluster. If a cluster without any labeled instance, all unlabeled instances in it will be assigned the majority label of the closest cluster.

The basic idea of CFSDP is that the clustering centers should be in the region with high data density and far away from each other. CFSDP is based on two quantities: (1) ρ_i, the local density of the i-th instance; (2) δ_i, the minimum distance between the i-th instance and the instances which have higher density than the i-th instance. ρ_i and δ_i are defined as

$$\rho_i = \sum_{j \in I_s / \{i\}} \exp\{-(d_{ij}/d_c)^2\}, \ \delta_i = \min_{j:\rho_j > \rho_i} (d_{ij}). \tag{1}$$

[1] Source code: https://gitee.com/ymw12345/sscadpsrc.git.

Algorithm 1: SSCADP

Input: A data stream in the form of chunk:$\boldsymbol{D} = \{\boldsymbol{D}^0, \boldsymbol{D}^1, \boldsymbol{D}^2..., \boldsymbol{D}^t, ...\}$ and parameters of n_{min}, dp, α

Output: The predicted labels of $\boldsymbol{D}^t$

```
Initialize a leaf for tree T, t = 0;
while data chunk D^t is available do
    if t > 0 then
        T.classify(D^t);
    for each x_i ∈ D^t do
        sort x_i into a leaf l ;
        update the statistics of the leaf l according to it is labeled or unlabeled;
        if the number of arrived instances at T meets dp then
            for each leaf l from bottom to top do
                D_l = get_data(l);
                labeling_unlabeled_data(D_l, α);
                concept_drift_detection(l);
            Installing pruning;
            if the number of arrived instances at a leaf l meets n_min then
                Installing split-test and growing child leaves ;
    t++;
```

where d_{ij} refers to the Euclidean distance between i-th and j-th instance. d_c represents the cutoff distance, which is set the same as CFSDP according to experience. I_s is the set of all instances indexes. Hence, a cluster center has such characteristic that ρ_i and δ_i are as large as possible. The $\rho_i - \delta_i$ plot (decision graph) can provide a visual way to determine the number of clusters.

In the function 1, $\gamma_i = \rho_i \delta_i$ is computed for each instance and then all γ_i are sorted in ascending order. CFSDP assumed that the sorted $\gamma_i(\gamma)$ follows the power-law distribution, and hence jump point of the sorted γ_i can be found.

After the cluster centers are determined, the clustering is installed, and then label propagation is conducted.

In order to find the jump point of the sorted γ_i, we refer to the idea of change detection in SAND [9], in which a change detection method is proposed to detect the location of the most significant changes in a series of values along a direction. Then, we further assume that the sorted γ_i follows a Pareto distribution. After the jump point is found, the number of clusters can be determined. As shown in the Fig. 1, the point in the red circle are jump point, the points above it are the center points.

The probability density function of Pareto distribution can be expressed as $f(x, a, k) = ak^a x^{-(a+1)}$ where a is the shape parameter and k is the proportion parameter. The corresponding logarithmic probability density function can be expressed as $\log f(x, a, k) = a \log(k) + \log(a) - (a + 1) \log(x)$. Define a random variable $\gamma \sim \text{Pareto}(a, k)$, and $\{\gamma_i\}$ is the observed value of γ. The maximum

Function 1: *labeling_unlabeled_data*

Input: Data $\boldsymbol{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$; Confidence parameter α
Output: The expanded labeled data $\boldsymbol{D}'$; Concept clusters set C

$\boldsymbol{D}'$=null, labeled data are added to $\boldsymbol{D}'$;
Construct distance matrix on $\boldsymbol{D}$ based on d_{ij} and record it as $\boldsymbol{M}$;
Calculate ρ_i and δ_i for each data $\boldsymbol{x}_i$ based on $\boldsymbol{M}$;
Normalize ρ_i and δ_i, $\gamma_i = \rho_i \delta_i$ and sort the elements in γ in ascending order;
$JumpPoint = jump\_point\_detection(\gamma, \alpha)$, $CN = n - JumpPoint$;
The instances corresponding to the first CN values in γ are selected as the cluster centers. $\{C_i\}_{i=1}^{CN}$ denotes the corresponding clusters;
for *each* $\boldsymbol{x}_i \in \boldsymbol{D}'$ **do**
    if $\boldsymbol{x}_i$ *is not a cluster center* **then**
        $\boldsymbol{x}_i$ is assigned to the cluster the same as its nearest higher density point belongs to;
for *each* $C_i \in C$ **do**
    Label unlabeled data at C_i by graph-based label propagation;
    Labeled data are added to $\boldsymbol{D}'$;
return $\boldsymbol{D}'$, C;

likelihood estimation of the parameters are calculated as follow, where N is the total number of observed values.

$$\hat{k}_{MLE} = \min_{1 \le i \le N} \{\gamma_i\},\ \hat{a}_{MLE} = N / \sum_{i=1}^{N} (\ln \gamma_i - \ln \hat{k}_{MLE}) \tag{2}$$

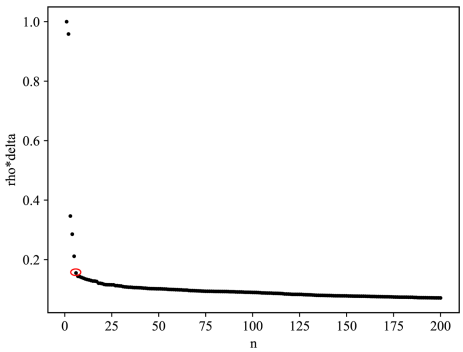

Fig. 1. Jump point detection.

To detect the jump point, in the function 2, γ is divided into two sub-windows by k from $N/2$ to $N-3$. The k value corresponding to the maximum statistical difference between the two sub-windows is the index of the jump point.

Function 2: *jump_point_detection*

Input: Array γ with elements in ascending order; Confidence parameter α
Output: The corresponding index of the jump point e

```
N = size(γ);
e = −1, w_n = 0;
for k = N/2 : N − 3 do
    m_a = mean(γ[1 : k]), m_b = mean(γ[k + 1 : N]);
    if m_a < αm_b then
        Pareto[scale_a, shape_a] < −estimateParam(γ[1 : k]);
        Pareto[scale_b, shape_b] < −estimateParam(γ[k + 1 : N]);
        sk = 0;
        for i = k + 1 : N do
            ta = logf(γ[i], scale_a, shape_a), tb = logf(γ[i], scale_b, shape_b);
            sk+ = log(tb/ta);
        if sk > w_n then
            w_n = sk, e = k;
return e;
```

3.3 Concept Drift Detection

A concept drift detection method is installed at each leaf. Before introducing the detail of concept drift detection, many variables should be defined. Respectively, r_{hist} and r_{new} denote the radius of the set of history concept clusters C_{hist} and the new ones C_{new}, n_{hist} and n_{new} represents the number of clusters in C_{hist} and C_{new}. r_k denotes the radius of a cluster and is computed as the average Euclidean distance from all instances in the cluster to its center: $r_k = \sum_{i=1}^{|C_k|} \sqrt{\sum_{j=1}^{D} (c_{kj} - x_{ij})^2} / |C_k|$, where $x_i = \{x_{i1}, x_{i2}, \ldots, x_{iD}\} \in C_k$ is the i-th instance in cluster C_k. D represents the attribute dimension. c_k refers to the cluster center of C_k and $|C_k|$ is the total number of instance in C_k. $r_{hist} = \sum_{i=1}^{n_{hist}} r_i / n_{hist}$. Similarly, r_{new} is calculated by this way. *dist* is used to measure the average distance between these two concept cluster sets. $dist = (\sum_{i=1}^{n_{new}} \min[\sqrt{\sum_{j=1}^{D} (c_{ij} - c_{kj})^2}, c_k \in C_{hist}, 1 \leq k \leq n_{hist}]) / n_{new}$, where c_k and c_i denote cluster centers in C_{hist} and C_{new}, respectively. In SUN, the value of *dist* greater than $\max(r_{hist}, r_{new})$ means concept drift.

However, in our algorithm, it is assumed that the change of points with higher density is more likely to reflect the change of data distribution. Therefore, *dist* is redefined as $dist = \sum_{i=1}^{n_{new}} w_i dist_i$ to capture the distribution change of data more accurately. $dist_i$ and w_i are calculated by (3) and (4) respectively. $dist_i$ refers to the distance of the cluster center c_i to C_{hist}. ρ_{c_i} refers to the average density of the clusters C_i and C_k which is the closest cluster to C_i in C_{hist} , and $\rho_{c_{ij}}$ refers to the density of j-th instance belonging to C_i, n_i and n_k mean the total number of data in C_i and C_k respectively. And hence larger w_i and $dist_i$ mean concept drift. Like SUN, if the value of *dist* is more than $\max(r_{hist}, r_{new})$,

a real concept drift is considered. The process of drift detection is described in the function 3.

$$dist_i = \min[\sqrt{\sum_{j=1}^{D} (c_{ij} - c_{kj})^2}, c_k \in C_{hist}, 1 \leq k \leq n_{hist}] \tag{3}$$

$$w_i = \rho_{c_i} \Big/ \sum_{i=1}^{n_{new}} \rho_{c_i}, \rho_{c_i} = (\sum_{j=1}^{n_i} \rho_{c_{ij}} + \sum_{j=1}^{n_k} \rho_{c_{kj}})/(n_i + n_k). \tag{4}$$

Function 3: *concept_drift_detection*

Input: Concept clusters set C_{new} and C_{hist} saved in leaf l
Output: *Flag*

```
Flag = False;
if C_hist = ∅ then
    C_hist = C_new
else
    calculate r_hist, r_new and dist using C_hist and C_new;
    if dist > max(r_hist, r_new) then
        Flag = True
return Flag
```

dist can be utilized to detect concept drifts caused by the change of P(x). In addition, considering concept drift can be caused by the distribution change of class labels, the class labels of history concept clusters and new ones are compared when concept drift is not detected. If the class labels of C_{hist} and C_{new} are completely opposite, it is also defined as a real concept drift.

After the bottom-up search is implemented to find all drift leaves, a pruning method is installed for adjusting the tree to cope with concept drifts. Each level of the tree will be traversed once to check concept drift of each leaf from bottom to top until the root is reached. If all child nodes of a node are detected concept drift, these child nodes are pruned and the new leaf node maybe split again.

4 Experiments

In this paper, all synthetic datasets are generated by MOA [19]. xxx-abr, xxx-gra and xxx-inc represent the concept drift types of abrupt, gradual and incremental, respectively. In the dataset with gradual drifts, it takes 5000 instances to change from one concept to another. In addition, to verify the performance of SSCADP on the datasets where the number of clusters varies apparently, we generate a Gaussian dataset with clusters change dynamically and concept changes, which are shown in Fig. 2(a), (b) and (c) represent three different distributions which

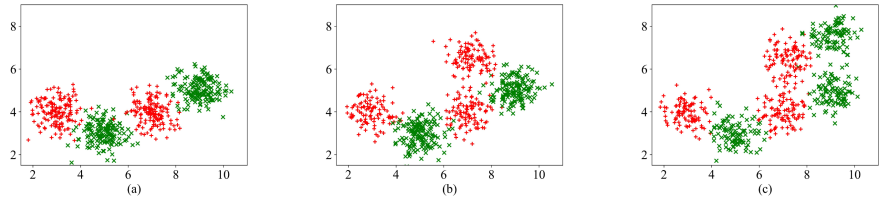

Fig. 2. Changes in clusters.

evolve along the time sequence, with the positive instances in red '+' and the negative instances in green '×', each figure contains 600 instances.

Table 1 shows the properties of all datasets. For Sea, four concepts are generated by setting $\theta = 8$, 9, 7, and 9.5. For Sine, the definition of Sine is if $a * sin(b * x_1 + \theta) + c > x_2$, the label is 0, otherwise is 1, four concepts are generated by setting $a = b = 1$ and $c = \theta = 0$, $a = b = 1$ and $c = \theta = 0$ with class labels are changed oppositely, $a = 0.3$, $b = 3\pi$, $c = 0.5$, $\theta = 0$ and $a = 0.3$, $b = 3\pi$, $c = 0.5$, $\theta = 0$ with class labels are changed oppositely. For HyperPlane-abr and HyperPlane-gra, four concepts are generated by setting $w^1 = (0, 0.5, 0.5), w^2 = (0, 1, 0), w^3 = (1, 0, 0)$ and $w^4 = (0.5, 0, 0.5)$, while for HyperPlane-inc, $d = 10$. For Agrawal, function 1, 2, 5, 6 are selected as the concepts of 1, 2, 3, 4. The Weather and Electricity dataset are used in this paper. For each synthetic data, 10 copies are randomly generated while for each real dataset labeled instances are randomly selected 10 runs.

In this paper, n_{min} refers to the minimum number of instances when a leaf attempts to do split-test and it is set to 200. dp means the detection period and $dp = 200$ empirically. $\alpha = 0.95$ is the confidence used in the clustering algorithm.

4.1 Experimental Results

SSCADP is compared with three baseline methods including SUN, SPASC, and Reasc. Three groups of experiments are conducted to evaluate the accuracy, the impact of label ratio, and concept drift tracking. In the first and last groups of the experiments, in order to simulate the situation of limited labeled data in real applications, the label ratio of all datasets are set to 0.1.

Accuracy. The accumulative accuracy is utilized to evaluate the performance of baseline methods and SSCADP. Table 2 shows the detailed results. For all datasets, each result is obtained by averaging the results of 10 runs.

It can be observed that SSCADP performs better than other baseline algorithms on almost of all datasets. The Friedman test is conducted on the results in Table 2, and the average rank is shown in Table 2, SSCADP achieves the best. Test statistic $F_F = 10.654$, the critical value for $\alpha = 0.05$ is 2.892, hence we can reject the null-hypothesis that there is no difference among the performance of

Table 1. Properties of the datasets

Datasets	Attributes	Instances	Classes	Chunk size	Concept change
Sea-abr	3	80000	2	1000	1-2-3-4-1-2-3-4
Sea-gra	3	115000	2	1000	1-2-3-4-1-2-3-4
Sine-abr	4	80000	2	1000	1-2-3-4-1-2-3-4
Sine-gra	4	115000	2	1000	1-2-3-4-1-2-3-4
HyperPlane-abr	3	80000	2	1000	1-2-3-4-1-2-3-4
HyperPlane-gra	3	115000	2	1000	1-2-3-4-1-2-3-4
HyperPlane-inc	10	80000	2	1000	Unknown
Agrawal-abr	9	80000	2	1000	1-2-3-4-1-2-3-4
Agrawal-gra	9	11500	2	1000	1-2-3-4-1-2-3-4
Gaussian	2	3600	2	600	1-2-3-1-2-3
Weather	8	18159	2	360	Unknown
Electricity	8	45312	2	1000	Unknown

all algorithms. Furthermore, in Nemenyi test $CD = 1.35$ which means SSCADP performs significantly better than SUN and Reasc. There is no significant difference in the performance between SSCADP and SPASC.

Table 2. Accumulative accuracy (%) on all datasets.

Datasets	SUN	SPASC	Reasc	SSCADP
Sea-abr	78.58 ± 2.64	83.36 ± 1.41	83.17 ± 1.58	$\mathbf{83.48 \pm 1.02}$
Sea-gra	81.16 ± 2.50	82.83 ± 1.10	82.45 ± 0.86	$\mathbf{84.45 \pm 0.51}$
Sine-abr	51.68 ± 2.37	$\mathbf{57.15 \pm 2.91}$	51.98 ± 0.31	51.92 ± 2.20
Sine-gra	53.21 ± 1.82	$\mathbf{55.35 \pm 1.76}$	51.03 ± 0.37	51.06 ± 1.40
HyperPlane-abr	64.47 ± 1.86	66.57 ± 2.29	51.38 ± 1.47	$\mathbf{67.14 \pm 0.82}$
HyperPlane-gra	65.54 ± 1.10	66.92 ± 1.99	51.17 ± 1.06	$\mathbf{68.31 \pm 1.73}$
HyperPlane-inc	74.15 ± 4.66	62.02 ± 3.38	50.06 ± 0.34	$\mathbf{74.34 \pm 7.53}$
Agrawal-abr	52.37 ± 1.03	57.78 ± 0.45	43.73 ± 0.32	$\mathbf{58.88 \pm 0.62}$
Agrawal-gra	53.36 ± 0.98	57.09 ± 0.53	44.34 ± 0.28	$\mathbf{57.86 \pm 0.46}$
Gaussian	52.64 ± 3.05	58.93 ± 5.86	50.38 ± 6.31	$\mathbf{62.17 \pm 6.37}$
Weather	67.89 ± 0.76	66.77 ± 1.21	67.95 ± 0.00	$\mathbf{68.53 \pm 0.67}$
Electricity	57.73 ± 5.04	54.75 ± 1.15	57.12 ± 1.49	$\mathbf{70.52 \pm 3.03}$
Average rank	3.00	2.25	3.41	1.33

Impact of Label Ratio. Considering the influence of labeling ratio on classification accuracy, we simulate the real scene to compare the algorithms by setting the label ratio to 0.05 and 0.2. Detailed results are shown in Table 3. In the case of 0.05, Friedman test is conducted and $F_F = 3.175$, critical value for $\alpha = 0.05$ is 2.892. This results indicate that the performance of all the algorithms is significantly different. Furthermore, in Nemenyi test $CD = 1.35$, and hence it can be concluded that SSCADP performs significantly better than SUN. These results indicate that even there are very limited labels available, SSCADP can achieve better performance, and it is suitable for semi-supervised classification of data stream.

In the case of 0.2, Friedman test is conducted and $F_F = 6.567$, critical value for $\alpha = 0.05$ is 2.892. This results indicate that the performance of all the algorithms is significantly different. Furthermore, in Nemenyi test $CD = 1.35$, and hence it can be concluded that SSCADP performs significantly better than SUN and Reasc. There is no significant difference in the performance between SSCADP and SPASC in the cases of 0.05 and 0.2.

Table 3. Impact of the label ratio on accumulative accuracy (%).

	SUN		SPASC		Reasc		SSCADP	
	0.05	0.2	0.05	0.2	0.05	0.2	0.05	0.2
Sea-abr	69.87	81.28	80.81	84.04	82.09	84.23	79.56	84.51
Sea-gra	74.07	82.53	80.89	84.78	83.13	85.18	77.33	85.22
Sine-abr	51.11	51.95	56.67	61.66	52.12	51.86	50.40	51.67
Sine-gra	52.19	52.28	57.81	54.36	53.01	53.05	52.22	50.16
H-abr	63.29	65.83	65.83	66.79	50.66	50.69	66.00	66.94
H-gra	64.22	66.97	66.04	68.32	51.39	50.65	67.28	68.86
H-inc	72.15	77.61	60.34	65.90	50.03	49.95	73.31	78.99
Agr-abr	51.69	52.98	56.87	58.82	43.69	43.80	57.63	59.30
Agr-gra	52.48	53.66	57.19	58.64	44.41	44.38	57.60	58.75
Gaussian	52.84	58.40	56.60	59.88	49.96	52.79	57.16	61.45
Weather	67.59	67.58	65.25	68.05	67.85	67.93	68.23	68.65
Electricity	59.36	70.56	53.67	56.15	59.69	57.38	68.09	70.82
Rank	3.16	3.00	2.25	2.25	2.83	3.25	1.75	1.50

Concept Drift Tracking. The drift tracking performance of all algorithms on all datasets with abrupt drift type are shown in Fig. 3. The number at bottom represent the kind of concept, and the vertical line indicate the location of concept drift. In the dataset of Sea, when new concepts arrive, the accuracy of SSCADP declined less in most cases, especially in concept 3 and 4 which are quite different. In the dataset of HyperPlane, SSCADP performed well in most cases except concept 2. In the dataset of Agrawal, the accuracy of SSCADP also

declined less in most cases. In the dataset of Sine, SSCADP performed not well since it is a single model and the large differences exist between each concept. SPASC adopts ensemble model as well as can deal with recurring drifts which can achieve better performance.

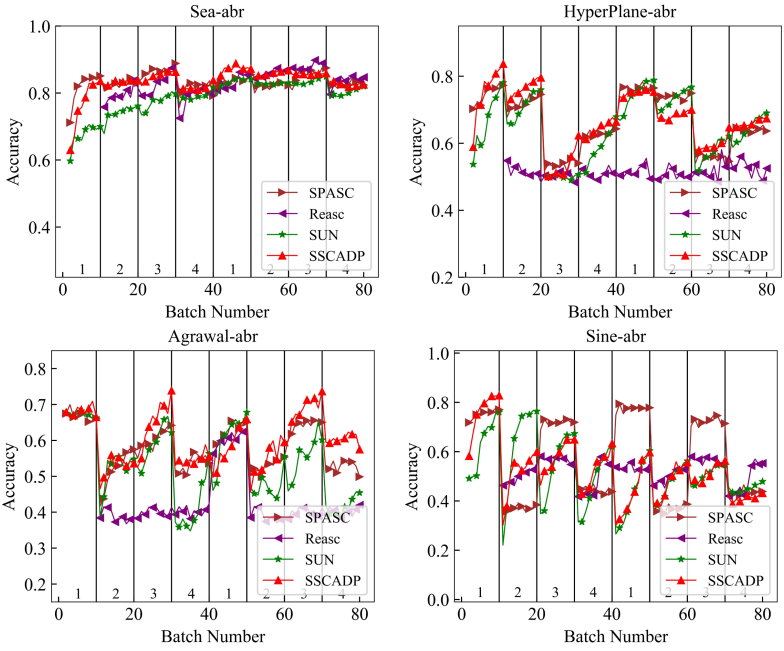

Fig. 3. Drift tracking graph.

5 Conclusions

In this paper, we propose a method of SSCADP to handle the semi-supervised classification of data streams. SSCADP can adaptively locate the cluster centers by considering both the local density and the distance between two points, and detect concept drifts by combining both the density of clusters and the distance between the historical clusters and the new ones. Experimental results illustrated that SSCADP can achieve better performance than the baseline algorithms in most datasets. In future, we will focus on how to effectively combine labeled data with unlabeled data to detect concept drift. We will also explore how to deal with recurring concept drift more effectively.

Acknowledgments. This work was partially supported by the Natural Science Foundation of Guangxi District (2018GXNSFDA138006), National Natural Science Foundation of China (61866007, 61662014), Collaborative Innovation Center of Cloud Computing and Big Data (YD16E12) and Image Intelligent Processing Project of Key Laboratory Fund (GIIP201505).

References

1. Ditzler, G., Roveri, M., Alippi, C., et al.: Learning in nonstationary environments: a survey. IEEE Comput. Intell. Mag. **10**(4), 12–25 (2015)
2. Gama, J., Žliobaitė, I., Bifet, A., et al.: A survey on concept drift adaptation. ACM Comput. Surv. **46**(4), 44–80 (2014)
3. Lu, J., Liu, A., Dong, F., et al.: Learning under concept drift: a review. IEEE Trans. Knowl. Data Eng. **31**(12), 2346–2363 (2019)
4. Sedhai, S., Sun, A.: Semi-supervised spam detection in Twitter stream. IEEE Trans. Comput. Soc. Syst. **5**(1), 169–175 (2018)
5. Wu, X., Li, P., Hu, X., et al.: Learning from concept drifting data streams with unlabeled data. Neurocomputing **92**, 145–155 (2012)
6. Hosseini, M.J., Gholipour, A., Beigy, H., et al.: An ensemble of cluster-based classifiers for semi-supervised classification of non-stationary data streams. Knowl. Inf. Syst. **46**(3), 567–597 (2016)
7. Li, P.P., Wu, X., Hu, X.: Mining recurring concept drifts with limited labeled streaming data. ACM Trans. Intell. Syst. Technol. **13**(2), 241–252 (2012)
8. Rodriguez, A., Laio, A.: Clustering by fast search and find of density peaks. Science **344**(6191), 1492–1496 (2014)
9. Haque, A., Khan, L., Baron, M., et al.: SAND: semi-supervised adaptive novel class detection and classification over data stream. In: Proceedings of the 13th AAAI Conference on Artificial Intelligence, pp. 1652–1658. AAAI, Menlo Park, CA (2016)
10. Wang, Y., Li, T.: Improving semi-supervised co-forest algorithm in evolving data streams. Appl. Intell. **48**(10), 3248–3262 (2018)
11. Bifet, A., Gavalda, R.: Learning from time-changing data with adaptive windowing. In: Proceedings of the 7th SIAM International Conference on Data Mining, pp. 443–448. SIAM, Philadelphia, PA (2007)
12. Masud, M.M., Woolam, C., Gao, J., et al.: Facing the reality of data stream classification: coping with scarcity of labeled data. Knowl. Inf. Syst. **33**(1), 213–244 (2012)
13. Wen, Y.M., Liu, S.: Semi-supervised classification of data streams by BIRCH ensemble and local structure mapping. J. Comput. Sci. Technol. **35**(2), 295–304 (2020)
14. Zhang, T., Ramakrishnan, R., Livny, M., et al.: BIRCH: an efficient data clustering method for very large databases. In: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, pp. 103–114. ACM, New York (1996)
15. Din, S.U., Shao, J., Kumar, J., et al.: Online reliable semi-supervised learning on evolving data streams. Inf. Sci. **525**, 153–171 (2020)
16. Zhang, P., Zhu, X., Tan, J., et al.: Classifier and cluster ensembles for mining concept drifting data streams. In: Proceedings of the 10th IEEE International Conference on Data Mining, pp. 1175–1180. IEEE, Los Alamitos, CA (2010)
17. Li, Y., Wang, Y., Liu, Q., et al.: Incremental semi-supervised learning on streaming data. Pattern Recogn. **88**, 383–396 (2019)
18. Zhu X, Ghahramani Z.: Learning from labeled and unlabeled data with label propagation. Technical report CMU-CALD-02-107, Carnegie Mellon University (2002)
19. Bifet, A., Holmes, G., Kirkby, R., et al.: MOA: massive online analysis. J. Mach. Learn. Res. **11**, 1601–1604 (2010)