Information Sciences 516 (2020) 20–32

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Selective prototype-based learning on concept-drifting data streams

Dongzi Chen, Qinli Yang*, Jiaming Liu, Zhu Zeng

School of Resources and Environment, University of Electronic Science and Technology of China, Chengdu 611731, China

ARTICLE INFO

Article history:
Received 15 June 2019
Revised 27 November 2019
Accepted 22 December 2019
Available online 23 December 2019

MSC:
00-01
99-00

Keywords:
Data stream
Concept drift
Classification
Prototype

ABSTRACT

Data stream mining has gained increasing attention in recent years due to its wide range of applications. In this paper, we propose a new selective prototype-based learning (SPL) method on evolving data streams, which dynamically maintains representative instances to capture the time-changing concepts, and make predictions in a local fashion. As an instance-based learning model, SPL only maintains some important prototypes (i.e., *ISet*) via error-driven representativeness learning. The fast condensed nearest neighbor *(FCNN)* rule, is further introduced to compress these prototypes, making the algorithm also applicable under memory constraints. To better distinguish noises from the instances associated with the new emerging concept, a potential concept instance set (i.e., *PSet*) is used to store all misclassified instances. Relying on the potential concept instance set, a local-aware distribution-based concept drift detection approach is proposed. SPL has several attractive benefits: (a) it can fit the evolving data streams very well by maintaining a small size of instance set; (b) it is capable of capturing both gradual and sudden concept drifts effectively; (c) it has great capabilities to distinguish noise/outliers from drifting instances. Experimental results show that the SPL has better classification performance than many other state-of-the-art algorithms.

© 2019 Elsevier Inc. All rights reserved.

1. Introduction

Data streams learning is a challenging task due to their unique characteristics: potential infinite and time evolution. Therefore, the learning algorithms must work efficiently with limited time and restricted memory resources. Meanwhile, the learning models should update quickly to capture the gradual or sudden concept drifts in data streams [1–6].

In recent years, two main strategies for data stream classification have been proposed: single model learning and ensemble learning. Single model learning algorithms attempt to capture the current concept with a fix or adaptive window [7,8]. However, usually it is very challenging to define a suitable window size in real-world applications. Different from the single model learning, ensemble learning algorithms divide a given data stream into many small data chunks, and then train several base learners for each data chunk. The final predictions are derived from aggregating all the results from these base classifiers with different strategies (e.g., major vote, bagging). The concept drift is implicitly learned by using these base learners. One potential problem of the ensemble learning is that each base learner is trained blindly from a single data chunk in a black box way and thus make it difficult to determine a suitable window size and useful instances. Besides, the

* Corresponding author.
E-mail address: qinli.yang@uestc.edu.cn (Q. Yang).

https://doi.org/10.1016/j.ins.2019.12.046
0020-0255/© 2019 Elsevier Inc. All rights reserved.

accuracy of such data stream classification depends on the performance of each base learner. One efficient solution to address these shortcomings is the use of instance-based learning. In instance-based learning, instead of taking time to build a global model, the target function is approximated locally by means of selected instances. Because of the inherent incremental nature of instance-based learning algorithms and their simple and flexible adaptation mechanism, this type of algorithm is quite suitable for learning in complexly dynamic environments. IBLStreams is one of the representative instance-based algorithms on data streams. It maintains an instance set as the training set of lazy learning. An example will be retained if it improves the predictive performance (classification accuracy) of the classifier; otherwise, it will be removed. In addition, IBLStreams only keeps the track of recent instances [9]. However, instance-based learning algorithms are sensitive to noise, and typically it is very time consuming for prediction with nearest neighbor searching.

In this paper, we propose a novel and robust instance-based data stream classification algorithm (SPL), to capture the current concept by dynamically selecting the most important instances. To this end, we maintain two dynamic sets over time: representative instance set (*ISet*) and potential concept-drifting instance set (*PSet*). Based upon the error-driven representativeness learning, the *ISet* stores the most representative instances to capture the current concept and these representative instances are used to predict the new incoming examples by nearest neighbor classifier. Considering the memory constraints in the context of data stream mining, a fast condensed nearest neighbor rule is introduced to compress the *ISet*. To identify the abrupt concept drift and filter noisy instances, a potential concept instance set (*PSet*) is proposed to store the misclassified examples. These misclassified instances are further examined to detect the abrupt concept drifts. Like other algorithms in this field, we assume that the class label of an instance to be classified becomes available immediately after its classification, and the data in streams are distributed not identically, but still independently.

1.1. Contributions

By introducing our instance-based learning algorithm, SPL has several attractive benefits for data stream classification, most importantly:

- **Concept modeling**. By dynamically maintaining the important instances and updating the *ISet*, SPL allows the learning of the gradual concept drift effectively without any needs of window-size data selection. By detecting the abrupt concept drift with the misclassified examples *PSet*, the instances in *ISet* can be updated quickly to adapt to the new emerging concept.
- **Robust to noise/outliers**. Since SPL examines all misclassified examples, the instances representing new concepts can be identified effectively, and the noise can be further removed through the error-driven representativeness learning.
- **High performance**. SPL only needs to maintain a small set of instances to capture the current concept, thus it allows higher classification accuracy than traditional single model algorithm. Furthermore, it shows its advantages over other instance-based algorithm(e.g., IBLStreams) since it has a low space complexity.

The remainder of this paper is organized as follows: In the following section, we briefly survey related work; Section 3 presents our algorithm in details; Section 4 contains the extensive experimental evaluations, and we finally conclude this paper in Section 5.

2. Related work

In the past several decades, researchers proposed many approaches for data streams learning, such as [2,3,7,9–13]. Due to space limitation, we only report the closest approaches from the literature.

Concept drift adaption. Data stream algorithms can be roughly divided into active and passive approaches in dealing with concept drift. Active approaches explicitly detect the change and make the adaptations [14,15]. Usually, statistics extracted from different time periods are analyzed to get the significant deviations. Classification error is the most common statistic [10,16] for concept drift detection. Kullback Leibler divergence is also used to measure the difference between models' probability distributions [17,18]. However, with the increment of the data dimensions, the run-time of the algorithms will grow exponentially and/or their performance will drop significantly [17,19]. To deal with the evolving data streams, ensembles are often coupled with drift detectors. For instance, Leveraging Bagging [20] and ADWIN Bagging [21] use the ADaptive WINdowing (ADWIN) algorithm [10], while DDD [22] uses Early Drift Detection Method (EDDM) [23] to detect concept drifts. Active methods are able to detect abrupt concept drift quickly. However, they have limited performance in dealing with gradual concept drift, which may not be significant enough and remains undetected. Passive approaches continuously adapt their models without explicit awareness of occurring drift. These methods can prevent pitfalls such as missed or false detected drifts, however, the adaptation speed is more or less constant, which may lead to costly delay in the case of abrupt drift [24]. Ensemble learning algorithms can deal with concept drifts with passive approaches, where they constantly reset the low performance classifiers to update the model [25–27]. To handle the gradual concept drift, Lughofer et al. [28,29] propose an incremental rule splitting concept for generalized fuzzy rules in order to autonomously compensate these negative effects of gradual drifts (i.e., oversized rules and accumulated errors). These passive approaches are useful in terms of adapting gradual drifts, while the active methods are more appropriate for rapidly recovering from abrupt drifts. In addition, some deep learning methods feature an open structure where their network structure, depth and width can be automatically evolved to adapt to the concept drift [30,31].

Data stream classification. Data stream classification has attracted growing attention in recent years. Single model learning algorithms on evolving data were firstly proposed in early years [7]. But very soon, the ensemble learners received great interest. To obtain more accurate predictions, the ensemble learners were selected to replace the single models in dealing with evolving stream classification [32]. The principle of ensemble classifiers is also based on the distinctive trait that ensembles allow for selectively reseting/removing/adding/updating the base models in response to drifts. However, ensemble models also have some disadvantages. For example, the model performance is greatly related to the base classifiers, and it is neither easy to divide the data chunks nor to define the ensemble size [33]. Instance-based learning is an efficient solution to these issues, and has been considered in some recent works [9,34]. Instead of learning the model, the instance-based learning attempts to maintain an instance set to capture the concept, and then perform classification by lazy learning. IBLStreams is one of the representative instance-based learning algorithms on data stream. It naturally supports the incremental update and thus fits the context of data streams well [9]. However, the instance-based learning also has weaknesses. For instance, it needs large memories to store the instances in order to keep the model's accuracy. Besides, noisy data have great effects on instance-based learning. To overcome such weaknesses, some algorithms handling the data streams with prototype have been proposed such as SyncStream [35,36]. SyncStream takes the idea based on synchronization to compress the representative instances as prototypes. One main difference between SyncStream and IBLStreams is that IBLStreams keeps track of recent instances while SyncStream dynamically learns a set of prototypes and supports efficient data maintenance [36].

In this work, we propose an algorithm based on selective prototypes. An instance set is proposed to store the representative examples. Additionally a potential concept drift data set is proposed to store the misclassified examples, which is used to detect the abrupt concept drift, making the algorithm robust to noise. Therefore, with implicit and explicit operations on concept drift, our algorithm can handle the concept drift (either gradual or sudden) well and is robust to noise.

3. Proposed method

In this section, we present our selective prototype-based learning algorithm. Before that, we first provide an overview of the algorithm and the intuition behind it.

3.1. Overview

As aforementioned, the basic idea of our algorithm is to dynamically maintain a set of important instances (i.e., *ISet*) to capture the time-changing concepts, and then use these data to predict the labels of new incoming instances with nearest neighbor classifier. Meanwhile, during the classification process, an error-driven learning approach is employed to determine the representativeness of all instances in the *ISet*. If the nearest prototype in *ISet* correctly predicts the new example, it indicates that the prototype is in line with the trend of the current concept. Therefore, the representativeness of the prototype in *ISet* increases, and vice versa. However, in the context of data stream classification, we cannot store all historical data due to the restricted memory. Therefore, these instances (prototypes) will be summarized as a smaller set based on the fast condensed nearest neighbor rule.

For concept-drifting data streams, how to maintain and update the instances to capture the current concept is the key point for data stream classification. To tackle this problem, we propose a heuristic way to detect abrupt concept drifts. The basic idea is to monitor the sharply degradation of prediction performance in a short time. If there are many wrongly classified examples with the same label in a short period, it is reasonable to speculate that an abrupt concept may emerge since the current model cannot predict the examples very well. To this end, we use another set named Potential Concept Instance Set (*PSet*), to store the instances which are wrongly classified. These instances largely represent the instances of new emerging concept in a data stream, and the noisy instances or outliers. Therefore, for a wrongly classified example, its closest examples (based on the Euclidean distance) with the same true label in *PSet* will be searched as the potential concept data. If the number of these potential concept instances is large enough, it means there is a local concept drift occurring. These potential concept drifting instances will be added into *ISet*, and meanwhile, these instances with low representativeness will be removed dynamically.

For illustration, Fig. 1 shows the general framework of our proposed algorithm. As the figure shows, when the new data comes, *ISet* is used to predict its label with nearest neighbor classifier firstly. Then the representativeness of the instance in *ISet* is updated with representativeness learning based on the prediction results. If the prediction is wrong, SPL will detect the concept drift with local-aware distribution-based concept drift detection approach based on *PSet* and update the *ISet*. Whats more, once the volume of ISet is too large and exceeds the threshold, *FCNN* is used to compress the ISet to summarize the instances and ensure the efficiency of the algorithm. In the following, we will elaborate it in detail.

3.2. Online data maintenance

In this section, we firstly introduce how to learn the representativeness of instances. Then, the instance set (*ISet*) is used to dynamically maintain the most importance instances to capture the time-changing concept, and the potential concept set (*PSet*) is used to store the misclassified instances to detect and get rid of the abrupt concept drifts and the noisy data.

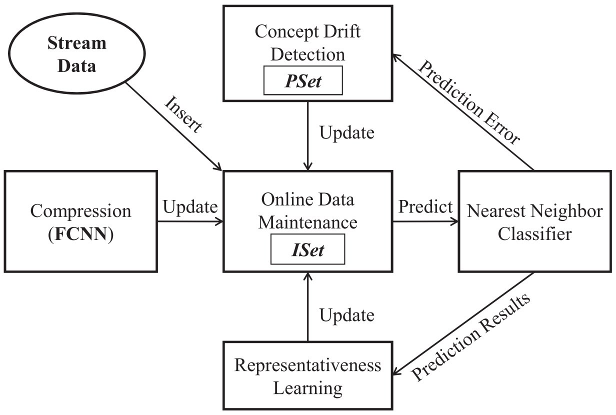


Fig. 1. The framework of the proposed algorithm.

3.2.1. *Representativeness learning*

Under the environment of evolving data streams, traditional data mining algorithms always focus on the short-term data since short-term data have higher probability to represent the current concept. However, some long-term historical data may also contain important information for data stream classification. Moreover, traditional data stream mining algorithms always choose a sliding window or a set of data chunks to capture the current concept. Unfortunately, the selection of the size of the sliding window or the horizon of the training data is a non-trivial task. If the window size is too small, the model may miss some important information. Conversely, if the window size is set too large, the training set may contain some outdated data. Therefore, to capture the gradual-changing concepts, here we introduce a representativeness learning approach to dynamically learn the representativeness of instances.

The basic idea of the representativeness learning is to infer the importance of a given instance by monitor its prediction ability. This idea is quite intuitive: if an instance contributes to the correctly labeling of a new incoming example, its importance should increase since it characterizes the current concept. Formally, let $x \in \mathcal{R}^d$ be an incoming example from a data stream $\mathcal{D}$. *ISet* is the current maintained instances, and $y \in$ *ISet* is the nearest instance to x. x^l and x^{pl} are the true label and the predicted label of x by p, respectively. The importance of the nearest instance p is updated as follows:

$$Im(p) = Im(p) + Sign(x^{pl}, x^l) \tag{1}$$

where $Sign(x^{pl}, x^l)$ is the sign function, which equals 1 if x^{pl} and x^l are equal, and equals -1 otherwise.

However, in the context of evolving data streams, the representativeness of instances should further consider their recency (i.e., the time effect is also an important factor that should be considered). Therefore, the representativeness for each instance p in *ISet* is updated as follows:

$$Im(p) = Im(p) \times 2^{-\lambda \cdot T_e} \tag{2}$$

3.2.2. *Fast condensed nearest neighbor rule*

Relying on the prediction ability and recency, the representativeness of each instance in *ISet* can be updated. More and more instances will be included in *ISet*, and this will definitely increase the spent of memory. Thereby, to fit the memory constraint the instances with low representativeness should be removed or compressed in time.

In this study, to maintain the effectiveness and efficiency of the algorithm, we introduce a technique to summarize the historical data via the fast condensed nearest neighbor rule. *Fast Condensed Nearest Neighbor Rule* (*FCNN*) aims to select a subset of the training set which classifies the remaining data correctly through the *NN* rule [37]. It uses a consistent subset of training set instead of the entire training set, to implement the NN rule. Therefore, it has the additional advantage that it would guarantee improved classification accuracy. The basic idea of selecting the consistent subset is that once the example cannot be classified correctly by its nearest neighbor, it will be kept in the subset and otherwise it will be discarded. The reason is that if the example can be correctly classified by its nearest neighbor, it may be redundant and should be discarded.

Typically, the fast condensed nearest neighbor algorithm needs a definition during the process.

Definition 1 (Voronoi cell)**.** Given a set T, let S be a subset of T, and let $p \in S$, the Voronoi cell of point p, denoted by $Vor(p, S, T)$ is defined as:

$$Vor(p, S, T) = \{q \in T | \forall p' \in S, dist(p, q) \leq dist(p', q)\} \tag{3}$$

FCNN works as follows. First, the consistent subset S is initialized to the centroids of the classes contained in the training set T. Then, during each iteration of the algorithm, for each point p in S, if there exists a point q($q \in T$ and $q \in Vor(p, S, T)$)

which has a different class label, it will be selected and added to *S*. The algorithm stops when no further point can be added to *S* (*i.e.* when *T* is correctly classified using *S*).

Under the compression of the training set with *FCNN, 1NN* classifier with compressed set can get the similar accuracy as *1NN* with the whole training set.

Finally, depending on performance of the prediction, the importance of the instances in *ISet* is updated by representativeness learning. Once the volume of *ISet* is large enough and exceeds the threshold, the instances in *ISet* with negative representativeness will be removed and the instances in *ISet* with unchanged representativeness will be compressed with *FCNN*. The algorithm of online *ISet* maintenance is summarized in Algorithm 1.

Algorithm 1 Online *ISet* maintenance.

```
Input: $x$, $ISet$, $maxI$.
1: if $ISet$.size $==$ $maxI$ then
2:     $S_n$ =$ISet$.getNegativePrototypes();
3:     $ISet$.remove($S_n$);
4:     $S_u$ =$ISet$.getUnchangedPrototypes();
5:     $P = FCNN(S_u)$;
6:     $ISet$.remove($S_u$);
7:     $ISet$.insert($P$);
8: end if
9: $ISet$.insert($x$)
Output: $ISet$.
```

3.2.3. Potential concept set update

For concept-drift data streams, we need to monitor whether and when a new abrupt concept occurs. To this end, we further maintain a potential concept set, called *PSet*. Once a new incoming example is wrongly classified, it is difficult to distinguish whether it is a new concept instance or a noisy example. However, if a relatively large number of instances are wrongly classified in a short period, and meanwhile, located in a local dense region, it is reasonable to assume that a new concept emerges. Therefore, we use *PSet* to store these wrongly classified instances, and then examine them for abrupt drift detection. Obviously, *PSet* faces the same situation as *ISet*, and only a limited size of the potential concept instances can be stored. Once the size of *PSet* is larger than its maximum allowed, we will remove the oldest potential concept instances with the low representativeness. The *PSet*, together with *ISet*, will be used for abrupt drift detection, which is illustrated in the next section.

3.3. Abrupt concept drift detection

When a new abrupt concept drift occurs, the incoming instances will have different labels with their local neighbors and thus tend to be wrongly classified. One principle to distinguish them from noisy instances is that, all these drifting instances in *PSet* will tend to appear in a local dense region (i.e., their neighbors tend to have the same label). Based on the observation, a local-aware distribution-based concept drift detection approach is proposed. As shown in Fig. 2, the shapes (square and circle) represent the true labels of instances, and the green symbols are the misclassified examples. For each misclassified instance, its associated instances from *PSet* are searched. If a sufficient number of neighbors in *PSet* are found to have the same labels as the classified instance, we will infer that an abrupt concept drift happens. After the concept drift is confirmed, *ISet* will be updated according to these new concept data.

During the process of concept drift detection, two things should be considered: a) How to determine a local dense region of the new wrongly classified instance in *PSet*? b) How to determine the threshold to judge whether there are enough neighbors of potential drifting instances with the same label?

For the first problem, given a new wrongly classified instance x, we cannot choose its k nearest neighbors in *PSet* directly since the instances in *PSet* are usually sparse, and the k nearest neighbors of x may be far away from x, which cannot ensure

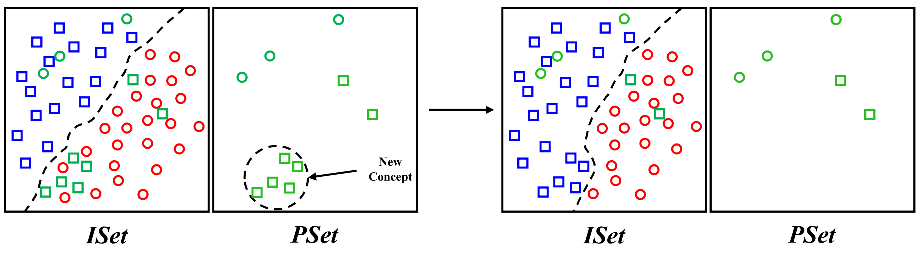


Fig. 2. Illustration of concept drift detection and model update. In this figure, the blue and red symbols are correctly classified instances, the green symbols are misclassified instances and the shapes (square and circle) represent the true labels of instances. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Algorithm 2 Abrupt concept drift detection.

```
Input: $x, k, ISet, PSet, maxP$.
 1: $C_x = PSet$.getCorrPotentialConcept($x$);
 2: if $C_x$.size$> k/N_c$ then
 3:     $ISetUpdate(x)$
 4:     $PSet$.remove($C_x$);
 5: else
 6:     $PSet$.add($x$);
 7:     //maxP: the maximum size of $PSet$
 8:     if $PSet$.size $==$ $maxP$ then
 9:         $PSet$.removeOldest();
10:     end if
11: end if
Output: $ISet, PSet$.
```

that all of them are located in a local dense region. Therefore, we first search the k nearest neighbors of x in $ISet$ as $N_{k,ISet}(x)$ instead, and let r be the maximum distance between x and its k nearest neighbors. Next, we search all nearest neighbors of x in $PSet$ with the distance no bigger that r (i.e., $NN(x) = \{p \in PSet|dist(p, x) \leq r\}$. For the second problem, we solve it with the help of k-nearest neighbor algorithm. Let N_c be the total number of the classes. For a given instance, if one specific label has more than k/N_c votes, the instance tends to be labeled by this specific label. Similarly, once the number of nearest neighbors of x with the same label exceeds k/N_c, the prediction will be most likely correct if we put these potential drifting instances into the $ISet$. Thereby, we set the threshold as k/N_c to judge whether there are enough potential drifting instances with the same label.

Once the abrupt concept drift is confirmed, these drifting instances in $PSet$ are removed. Accordingly, the $ISet$ should also be updated. On the one hand, their nearest neighbors in $ISet$ with the same label allow the characterization of the current concept, and accordingly we increase their representativeness. On the other hand, their nearest neighbors in $ISet$ with the different label characterize the old concepts, and thus these outdated instances are removed correspondingly. Algorithm 2 illustrates the pseudocode of our abrupt concept drift detection method.

Finally, based upon the online data maintenance and abrupt concept detection module, we can perform data stream classification in combination with the 1-nearest neighbor classifier. The pseudocode of SPL is finally summarized in Algorithm 3.

Algorithm 3 Algorithm of *SPL*.

```
Input: $\mathcal{D}, \mathcal{D}_{init}, k, \lambda, \delta, maxI, maxP$.
 1: //$ISet$ Initialization
 2: $ISet$.insert ($\mathcal{D}_{init}$);
 3: repeat
 4:     for Each new example $x \in \mathcal{D}$ do
 5:         //Search the nearest instance of $x$
 6:         $p$=nearestNeighbor($ISet, x$);
 7:         [$Acc, Pre, Rec, F_1$]=computeAccuracy($x^l$, $p^l$);
 8:         //Online $ISet$ Maintenance
 9:         $ISet$ = ISetUpdate($x, ISet, maxI$);
10:         //Representativeness learning
11:         if $p^l == x^l$ then
12:             $Im(p) = Im(p) \times 2^{-\lambda T_e} + 1$;
13:         else
14:             $Im(p) = Im(p) \times 2^{-\lambda T_e} - 1$;
15:             if noiseFlag == TRUE then
16:                 Flag = CheckNoise($ISet, x, \delta$);//$\delta = -3$
17:                 if Flag == TRUE then
18:                     $ISet$.remove($x$);
19:                 end if
20:             end if
21:             //Abrupt concept drift detection with Algorithm 2
22:             $ISet, PSet$ = ConceptDriftDetection($x, k, ISet, PSet, maxP$)
23:         end if
24:     end for
25: until Error
Output: $ISet, Acc, Pre, Rec, F_1$.
```

4. Experimental evaluation

In this section, we evaluate our proposed algorithm on both synthetic and real-world data streams.

4.1. Experimental setup

Data sets. We evaluate the proposed method on synthetic data and five different types of real-world data: *Shuttle, Covtype, Occupancy, Spam* and *Sensor*. Table 1 lists the statistics of the five real-world data streams.

Synthetic Data was generated based on a moving hyperplane [38]. The hyperplane in 2-dimensional space was used to simulate the time-changing concepts according to the change of its orientation and position. In this study, two synthetic data consisting of one million examples each were created to simulate the abrupt and gradual concept drift, respectively.

Shuttle is consisted of 58,000 instances and 9 numerical attributes. Approximately 80% of the data belongs to class 1.

Forest Covtype contains the forest cover type for 30 x 30 m cells obtained from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. It contains 581,012 instances and 54 attributes, and it has been used in several published studies on data stream classification.

Occupancy Detection Data contains 20,560 instances with 7 attributes. Experimental data used for binary classification (room occupancy) from temperature, humidity, light and CO_2. The data were obtained from time stamped pictures that were taken every minute.

Spam Filtering Data is a collection of 9324 email messages derived from the Spam Assassin collection1. Each email is represented by 500 attributes using the Boolean bag-of-words approach.

Sensor Data is collected from 54 sensors with 4 attributes (temperature, humidity, light and sensor voltage) deployed in the Intel Berkeley Research lab between February 28th and April 5th, 2004. It contains 2,219,803 instances.

Selection of comparison methods. To better evaluate our algorithm, we compare the performance of our proposed algorithm with that of other representative data stream classification algorithms.

Adaptive Hoeffding Tree [39] is a classification algorithm for evolving data streams. It uses ADWIN to monitor the performance of branches on the tree and to replace them with new branches when their accuracy decreases.

IBLStreams [9] is an instance-based learning algorithm for classification and regression problems on data streams. It makes prediction based on the neighbors of the new example in the training data.

OzaBagAdwin [40] is an online bagging method of Oza and Rusell with the addition of the ADWIN algorithm as a change detector and as an estimator for the weights of the boosting method. When a change is detected, the worst classier of the ensemble of classiers is removed and a new classier is added to the ensemble.

Table 1
The performances of different data stream classification algorithms on the real-world data sets.

Data	Methods	Acc.	Prec.	Rec.	F_1	Time (ms)
Shuttle	SPL	**0.9947**	**0.7996**	0.5157	**0.6270**	6060
	HoeffdingAdaTree	0.9704	0.5000	**0.5531**	0.5252	**564**
	OzaBagAdwin	0.9756	0.5265	0.5233	0.5249	3436
	WeightedEnsemble	0.9703	0.6049	0.4346	0.5058	8138
	AUE2	0.9815	0.586	0.5500	0.5674	3631
	IBLStreams	0.9907	0.7192	0.5246	0.6067	16,203
Covtype	SPL	**0.9495**	**0.9090**	**0.9101**	**0.9096**	592,632
	HoeffdingAdaTree	0.8189	0.7028	0.7894	0.7436	**24721**
	OzaBagAdwin	0.8474	0.7716	0.8163	0.7933	150,641
	WeightedEnsemble	0.8023	0.7463	0.6429	0.6907	440,758
	AUE2	0.8712	0.7618	0.7906	0.7759	164,565
	IBLStreams	0.9226	0.8709	0.8511	0.8608	5,207,061
Occupancy	SPL	**0.9934**	**0.9910**	**0.9904**	**0.9907**	3261
	HoeffdingAdaTree	0.9847	0.974	0.9837	0.9788	**247**
	OzaBagAdwin	0.9892	0.9798	**0.9904**	0.9851	687
	WeightedEnsemble	0.8949	0.9224	0.7808	0.8457	876
	AUE2	0.8978	0.923	0.7878	0.85	683
	IBLStreams	0.9768	0.974	0.9602	0.967	1492
Spam	SPL	**0.9688**	**0.9597**	**0.9583**	**0.9590**	61,618
	HoeffdingAdaTree	0.9071	0.8717	0.8935	0.8825	**1539**
	OzaBagAdwin	0.9107	0.8764	0.8972	0.8867	7825
	WeightedEnsemble	0.7415	0.7043	0.7595	0.7309	9548
	AUE2	0.7358	0.7113	0.774	0.7413	9731
	IBLStreams	0.9368	0.906	0.9389	0.9221	471,651
Sensor	SPL	**0.8902**	**0.8640**	**0.8627**	**0.8633**	339,426
	HoeffdingAdaTree	0.6504	0.6439	0.6412	0.6426	**159543**
	OzaBagAdwin	0.8728	0.8502	0.8479	0.8491	1,413,459
	WeightedEnsemble	0.6753	0.764	0.6567	0.7063	3,291,935
	AUE2	0.825	0.8059	0.8022	0.804	1,988,571
	IBLStreams	0.1168	0.1642	0.136	0.1488	258,198

AUE2 [41] is a block-based stream ensemble classifier which is able to react to different types of concept drifts.

Weighted Ensemble [11] is a general framework for mining concept-drifting data streams using weighted ensemble classifiers.

We have implemented SPL in Java. The IBLStreams algorithm is available at: https://cs.uni-paderborn.de/fileadmin/informatik/fg/is/Software/Learning.zip. Adaptive Hoeffding tree, Weighted Ensemble and OzaBagAdwin have been implemented in MOA framework available at https://moa.cms.waikato.ac.nz. All experiments have been performed on a computer with 3.6 GHz CPU and 8 GB RAM.

4.2. Proof of concept

We first use the two-dimensional synthetic data streams to evaluate the performance of SPL, and demonstrate its desirable properties.

Concept modeling. We first evaluated whether SPL can capture the current concept or not. Figs. 3 (a)–(d) and 4 (a)–(d) show the real distributions of the synthetic data stream with gradual concept drift and abrupt concept drift corresponding

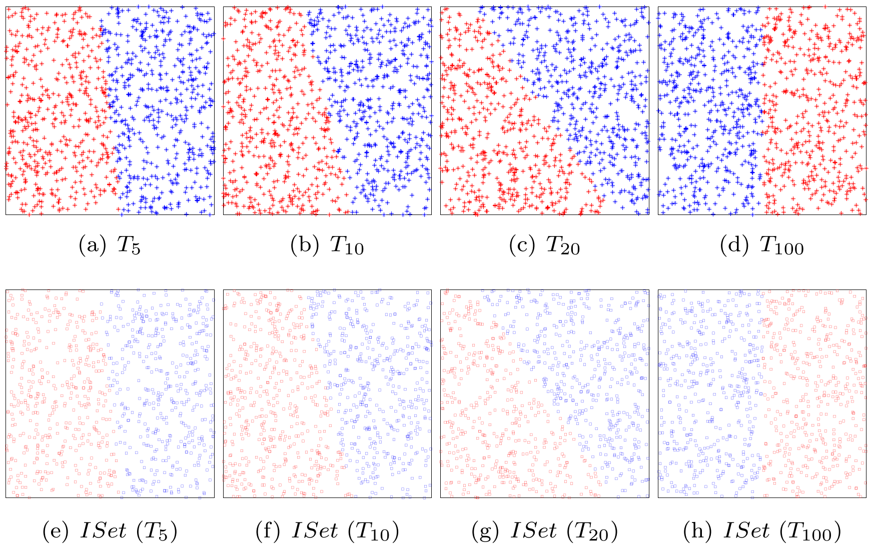


Fig. 3. Performance of SPL algorithm on the synthetic data stream with gradual concept drift. (a) - (d) show the real distributions of the synthetic data stream with gradual concept drift corresponding to the time units, respectively. (e) - (f) show the distributions of the data from *ISet* corresponding to the time units.

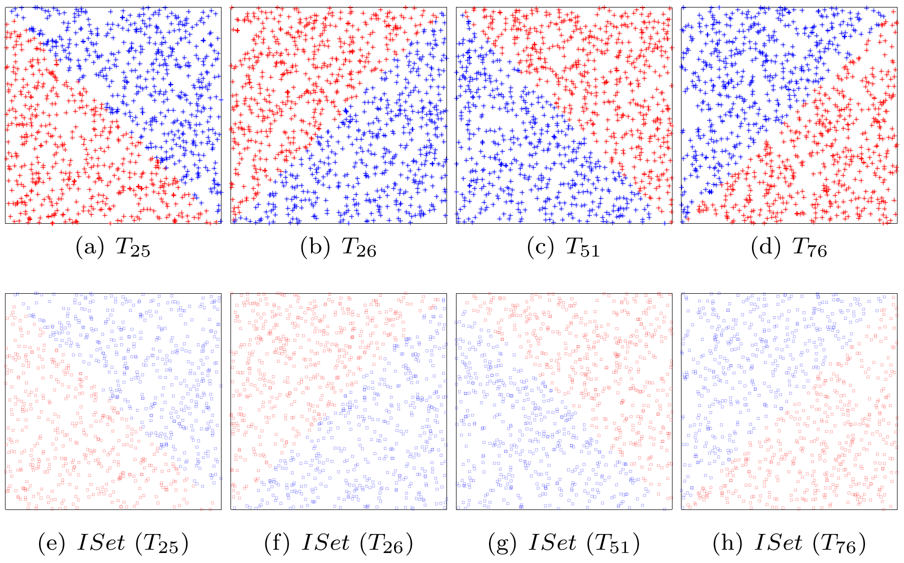
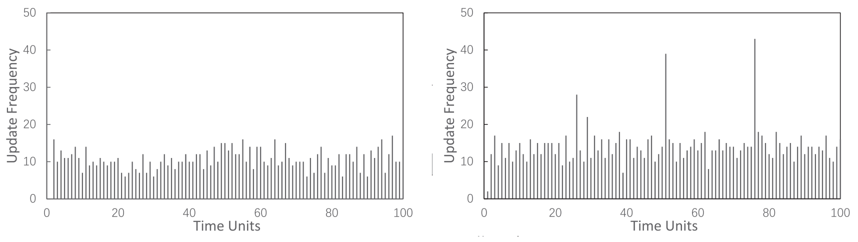
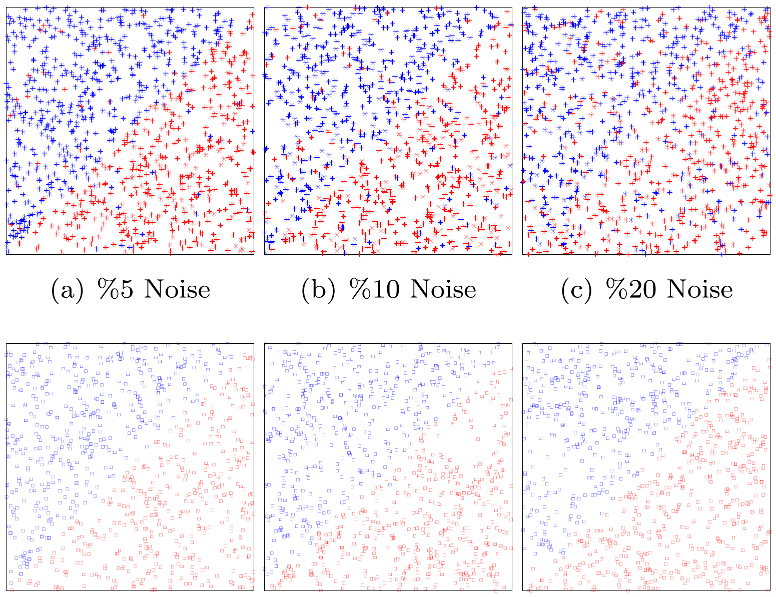


Fig. 4. Performance of SPL algorithm on the synthetic data stream with abrupt concept drift. (a) - (d) show the real distributions of the synthetic data stream with abrupt concept drift corresponding to the time units, respectively. (e) - (f) show the distribution of the data from *ISet* corresponding to the time units.



(a) Gradual concept drift data stream (b) Abrupt concept drift data stream

Fig. 5. The update frequency of *ISet* in SPL on the synthetic data stream with gradual concept drifts and abrupt concept drifts corresponding to the time units.



(a) %5 Noise (b) %10 Noise (c) %20 Noise

(d) $ISet$ (%5) (e) $ISet$ (%10) (f) $ISet$ (%20)

Fig. 6. Performances of SPL algorithm on the synthetic data stream with different percentages of noisy examples. (a) - (c) show the real distributions of the synthetic data stream with different percentages of noisy examples at a certain moment, respectively, (d) - (f) show the distributions of the data from *ISet* corresponding to different noise ratios at the same moment.

to the time units, respectively. Each data stream has one million examples, and they were divided into one hundred time units and each time unit had 1000 examples. In each subfigures, Figs. 3 (e)–(h) and 4 (e)–(h) show the distributions of the data from *ISet* in terms of time units. From the figures, we can see that our SPL algorithm can dynamically update the *ISet* to capture the current concept in time, and thus can keep almost the same distribution with the evolving data stream.

Concept drift detection. Here we further evaluated the performance of SPL on concept drift detection. For SPL, the *ISet* should be additionally updated when an abrupt concept drift is detected via the removal of all the outdated instances.The updating frequency is shown in Fig. 5. The *ISet* updating frequency on gradual concept drift data stream is stable. But the update frequency on abrupt concept drift data stream has some extreme values on 26, 51 and 76 time unit. As shown in Fig. 4, the abrupt concept drifts occurred at the beginning of the 26, 51, 76 time units. From the figure, we can conclude that SPL is able to detect the abrupt concept drifts and update the *ISet* to capture the current concept drift quickly.

Noise handling. We further evaluate the robustness of SPL on data stream with noisy instances. Fig. 6 shows the distributions of the synthetic data stream with concept drifts at a certain moment and the distribution of the data from *ISet* on data stream corresponding to the noise ratio at the same moment. Obviously, the SPL can filter these noisy instances well, and it can maintain the most important instances in *ISet*.

4.3. Prediction performance analysis

In this section, we compared the performances of different stream classification algorithms based on the five real data streams mentioned above. For SPL, unless otherwise specified, the parameters were set to $maxP = 3000$, $maxI = 1000$. For all the other algorithms, the default parameters suggested by the authors were used correspondingly.

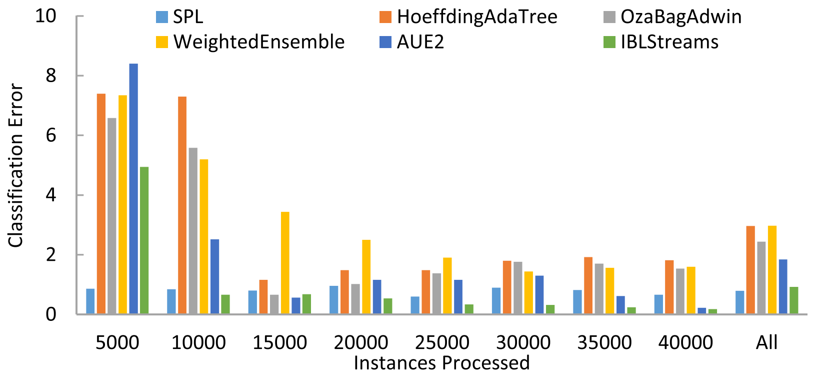


Fig. 7. Performances of different data stream classification algorithms on shuttle data.

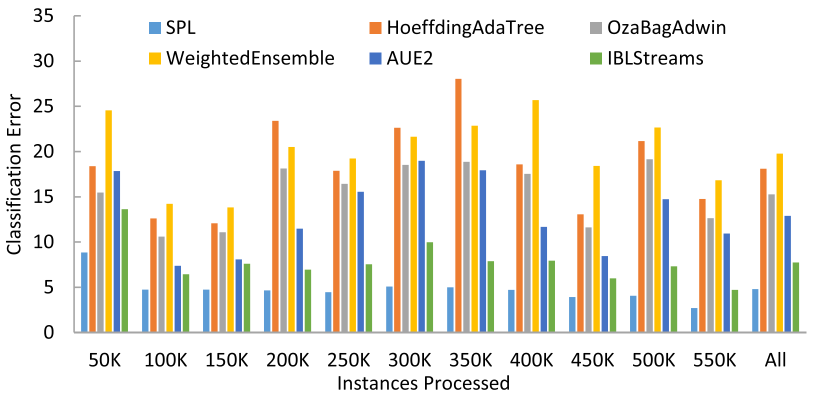


Fig. 8. Performances of different data stream classification algorithms on covtype data.

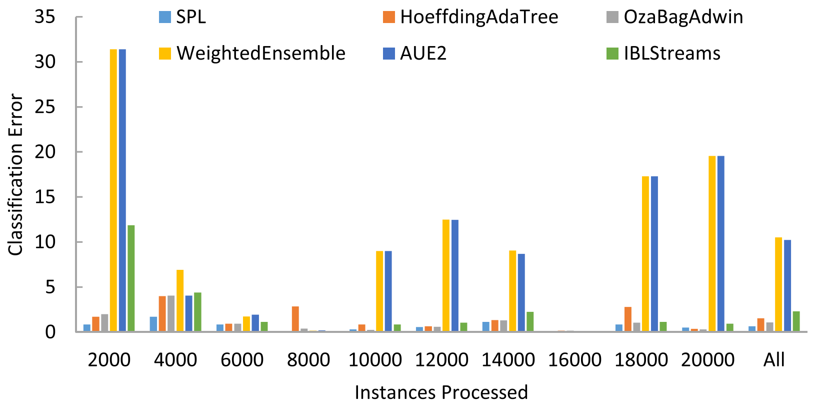


Fig. 9. Performances of different data stream classification algorithms on occupancy detection data.

Figs. 7–11 plot the prediction accuracies of the algorithms against the number of processed instances. As we can see, the classification error of SPL algorithm is low and stable with the increasing number of processed instances. HoeffdingAdaTree is a typical single model learner. It uses ADWIN to monitor performance of branches on the tree and update the model, and this method is outdated and cannot yield good performance. It almost has the worst performance over all five data sets. OzaBagAdwin, WeightedEnsemble and AUE2 are three ensemble learning algorithms, but their performances are not stable. They have the worst performance in some cases. Similar to SPL, IBLStreams is also an instance-based learning algorithm. However, it faild to yield good performance in some cases.

Table 1 shows the details about the performances of different classification algorithms in terms of different evaluation measures. According to the Table 1, HoeffdingAdaTree has the minimal time complexity, but worst accuracy. IBLStreams has better performance but it is the most time-consuming algorithm. From the experiment, we can see that SPL has good performance on accuracy, precision, recall and F_1 score, and it even significantly faster than IBLStreams.

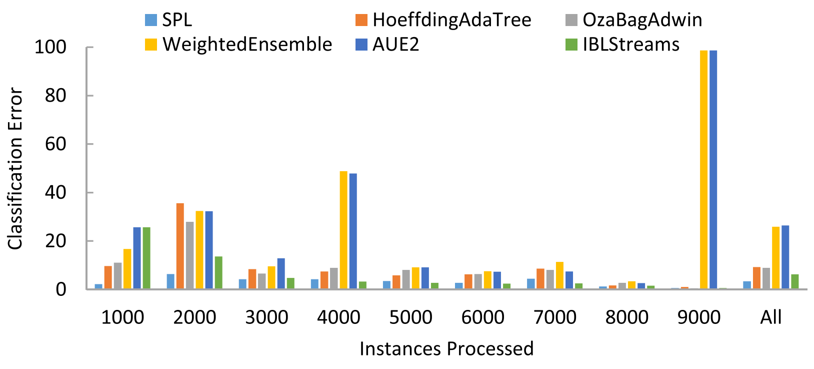


Fig. 10. Performances of different data stream classification algorithms on spam data.

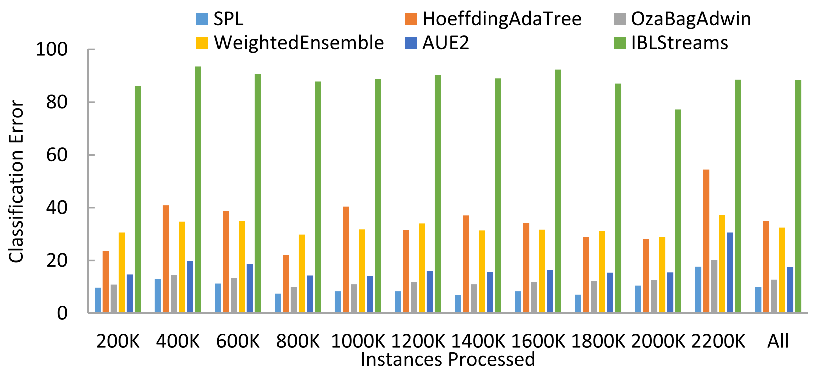


Fig. 11. Performances of different data stream classification algorithms on sensor data.

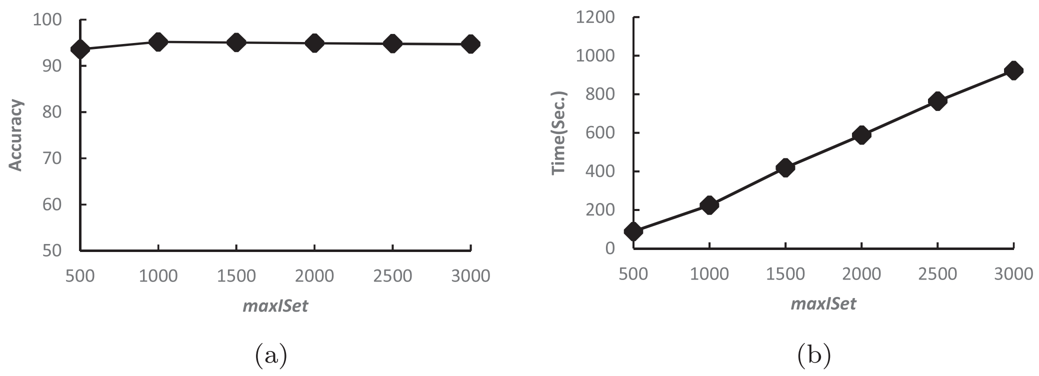


Fig. 12. The sensitivity analysis with *maxI*. (a) shows the classification accuracy with different values of *maxI*, (b) shows the computation time with different values of *maxI*.

4.4. Sensitivity analysis

In this section, we conducted a sensitivity analysis of SPL on the covertype data.

Maximum ISet size: As demonstrated in Section 3.2.2, the maximum boundary of the *ISet* should be specified. In this experiment, we varied *maxI* from 500 to 3000. Fig. 12 shows the classification accuracies in terms of the *maxI* and the corresponding computation time. It suggests that the classification performance is stable with the increase of the *maxI*, but the computation time is almost proportional to the maximum size of the *ISet*.

Maximum *PSet* size: Fig. 13 shows the classification performance with different size of the potential concept instance set ranging from 500 to 3000. The *maxP* determines the maximum number of the potential concept instances which can be stored to detect the concept drift. From Fig. 13, we can conclude that the classification accuracy is stable by varying the values of *maxP*, but the computation time is greatly increased with a larger *maxP*.

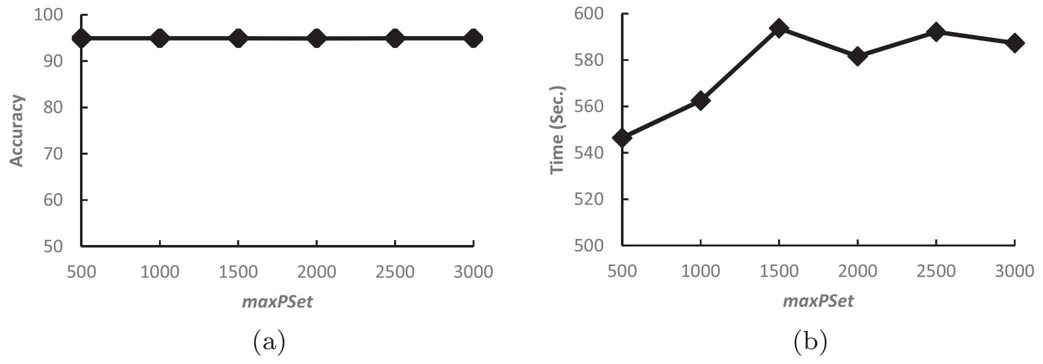


Fig. 13. The sensitivity analysis with *maxP*. (a) shows the classification accuracy with different values of *maxP*, (b) shows the computation time with different values of *maxP*.

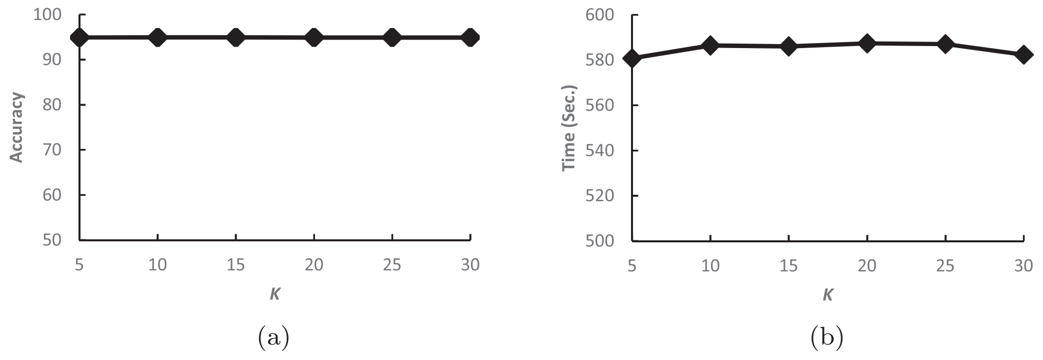


Fig. 14. The sensitivity analysis with K. (a) shows the classification accuracy with different values of K, (b) shows the computation time with different values of K.

K: As aforementioned, the K value determines the size of the area where the instances in this area can affect the target data sample. Fig. 14 shows that both classification accuracies and the computation time are stable with K value ranging from 5 to 30.

5. Conclusion

In this paper, we introduce a new instance-based data stream classification algorithm, called SPL. Instead of using a window to focus on the recent data, SPL dynamically maintains an instance set to keep important instances to model the current concept with error-driven representativeness learning. To adapt to abrupt concept drift, SPL further maintains a potential concept instance set to store the misclassified examples. Once enough neighbors of the wrongly classified example with the same label in the potential concept instance set are found, an abrupt concept drift is defined. Accordingly, the instance set will be updated to capture the concept. In addition, as to the limit of the memory size on data stream learning, the fast condensed nearest neighbor rule is used to compress the instance set to reduce the time complexity of the proposed algorithm. In contrast to the traditional instance-based learning such as IBLStreams, SPL allows maintaining both short-term and long-term representative instances. Extensive experiments have demonstrated that SPL has better performance than many state-of-art data stream classification methods. Although SPL allows a higher classification accuracy, it is slower than traditional single model algorithms since they use adaptive windows to focus on recent data. However, SPL shows its advantages over other instance-based algorithms (e.g., IBLStreams) since it only needs to dynamically maintain a small set of instances.

Declaration of Competing Interest

The authors declare no conflict of interest.

CRediT authorship contribution statement

Dongzi Chen: Formal analysis, Methodology, Visualization, Writing - original draft, Writing - review & editing. **Qinli Yang:** Supervision, Methodology, Writing - original draft, Writing - review & editing, Funding acquisition. **Jiaming Liu:** Validation, Visualization, Writing - review & editing. **Zhu Zeng:** Validation, Visualization, Writing - review & editing.

Acknowledgments

This work is supported by National Key Research and Development Program (2016YFA0601501, 2016YFB0502300), the National Natural Science Foundation of China (61976044, 41601025, 41830863), Fundamental Research Funds for the Central Universities (ZYGX2019Z014), Fok Ying-Tong Education Foundation for Young Teachers in the Higher Education Institutions of China (161062), the Belt and Road Fund on Water and Sustainability of the State Key Laboratory of Hydrology-Water Resources and Hydraulic Engineering (2019), State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin (IWHR-SKL-201911).

References

[1] J. Gama, J. Aguilar-Ruiz, R. Klinkenberg, Knowledge discovery from data streams, Intell. Data Anal. 12 (3) (2008) 251–252.
[2] M. Sayed-Mouchaweh, E. Lughofer, Learning in Non-stationary Environments: Methods and Applications, Springer Science & Business Media, 2012.
[3] I. Khamassi, M. Sayed-Mouchaweh, M. Hammami, K. Ghdira, Discussion and review on evolving data streams and concept drift adapting, Evol. Syst. 9 (1) (2018) 1–23.
[4] Q. Yang, H. Zhang, G. Wang, S. Luo, D. Chen, W. Peng, J. Shao, Dynamic runoff simulation in a changing environment: a data stream approach, Environ. Model. Softw. 112 (2019) 157–165.
[5] H. Zhang, Q. Yang, J. Shao, G. Wang, Dynamic streamflow simulation via online gradient-boosted regression tree, J. Hydrol. Eng. 24 (10) (2019) 4019041.
[6] S.U. Din, J. Shao, Exploiting evolving micro-clusters for data stream classification with emerging class detection, Inf. Sci. 507 (2020) 404–420.
[7] G. Hulten, L. Spencer, P. Domingos, Mining time-changing data streams, in: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2001, pp. 97–106.
[8] S. Papadimitriou, A. Brockwell, C. Faloutsos, Adaptive, hands-off stream mining, in: Proceedings of the 29th international conference on Very large data bases, 29, VLDB Endowment, 2003, pp. 560–571.
[9] A. Shaker, E. Hüllermeier, Iblstreams: a system for instance-based classification and regression on data streams, Evol. Syst. 3 (4) (2012) 235–249.
[10] A. Bifet, R. Gavalda, Learning from time–changing data with adaptive windowing, in: Proceedings of the 2007 SIAM international conference on data mining, SIAM, 2007, pp. 443–448.
[11] H. Wang, W. Fan, P.S. Yu, J. Han, Mining concept-drifting data streams using ensemble classifiers, in: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2003, pp. 226–235.
[12] A. Sancho-Asensio, A. Orriols-Puig, J. Casillas, Evolving association streams, Inf. Sci. 334 (2016) 250–272.
[13] J. Shao, Y. Tan, L. Gao, Q. Yang, C. Plant, I. Assent, Synchronization-based clustering on evolving data stream, Inf. Sci. 501 (2019) 573–587.
[14] S.G. Santos, R.S. Barros, P.M. Gonçalves Jr, A differential evolution based method for tuning concept drift detectors in data streams, Inf. Sci. 485 (2019) 376–393.
[15] D.R. de Lima Cabral, R.S.M. de Barros, Concept drift detection based on fishers exact test, Inf. Sci. 442 (2018) 220–234.
[16] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Brazilian symposium on artificial intelligence, Springer, 2004, pp. 286–295.
[17] T. Dasu, S. Krishnan, S. Venkatasubramanian, K. Yi, An information-theoretic approach to detecting changes in multi-dimensional data streams, in: In Proc. Symp. on the Interface of Statistics, Computing Science, and Applications, Citeseer, 2006.
[18] A. Shaker, E. Lughofer, Self-adaptive and local strategies for a smooth treatment of drifts in data streams, Evol. Syst. 5 (4) (2014) 239–257.
[19] D. Kifer, S. Ben-David, J. Gehrke, Detecting change in data streams, in: Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, VLDB Endowment, 2004, pp. 180–191.
[20] A. Bifet, G. Holmes, B. Pfahringer, Leveraging bagging for evolving data streams, in: Joint European conference on machine learning and knowledge discovery in databases, Springer, 2010, pp. 135–150.
[21] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, R. Gavaldà, New ensemble methods for evolving data streams, in: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2009, pp. 139–148.
[22] L.L. Minku, X. Yao, Ddd: a new ensemble approach for dealing with concept drift, IEEE Trans. Knowl. Data Eng. 24 (4) (2012) 619–633.
[23] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, R. Morales-Bueno, Early drift detection method, in: Fourth international workshop on knowledge discovery from data streams, 6, 2006, pp. 77–86.
[24] V. Losing, B. Hammer, H. Wersing, Tackling heterogeneous concept drift with the self-adjusting memory (sam), Knowl. Inf. Syst. 54 (1) (2018) 171–201.
[25] J.Z. Kolter, M.A. Maloof, Dynamic weighted majority: a new ensemble method for tracking concept drift, in: Third IEEE International Conference on Data Mining, IEEE, 2003, pp. 123–130.
[26] D. Brzezinski, J. Stefanowski, Combining block-based and online methods in learning ensembles from concept drifting data streams, Inf. Sci. 265 (2014) 50–67.
[27] H.M. Gomes, F. Enembreck, Sae2: advances on the social adaptive ensemble classifier for data streams, in: Proceedings of the 29th Annual ACM Symposium on Applied Computing, ACM, 2014, pp. 798–804.
[28] E. Lughofer, M. Sayed-Mouchaweh, Autonomous data stream clustering implementing split-and-merge concepts - towards a plug-and-play approach, 304, Elsevier, 2015.
[29] E. Lughofer, M. Pratama, I. Skrjanc, Incremental rule splitting in generalized evolving fuzzy systems for autonomous drift compensation, IEEE Trans. Fuzzy Syst. 26 (4) (2018) 1854–1865.
[30] A. Ashfahani, M. Pratama, Autonomous deep learning: continual learning approach for dynamic environments, 2019, pp. 666–674.
[31] M. Pratama, C. Za'in, A. Ashfahani, Y.S. Ong, W. Ding, Automatic construction of multi-layer perceptron network from streaming examples, 2019.
[32] M. Pratama, W. Pedrycz, E. Lughofer, Evolving ensemble fuzzy classifier, IEEE Trans. Fuzzy Syst. 26 (5) (2018) 2552–2567.
[33] L. Pietruczuk, L. Rutkowski, M. Jaworski, P. Duda, How to adjust an ensemble size in stream data mining? Inf. Sci. 381 (2017) 46–54.
[34] V. Losing, B. Hammer, H. Wersing, Knn classifier with self adjusting memory for heterogeneous concept drift, in: 2016 IEEE 16th International Conference on Data Mining (ICDM), IEEE, 2016, pp. 291–300.
[35] J. Shao, Z. Ahmadi, S. Kramer, Prototype-based learning on concept-drifting data streams, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2014, pp. 412–421.
[36] J. Shao, F. Huang, Q. Yang, G. Luo, Robust prototype-based learning on data streams, IEEE Trans. Knowl. Data Eng. 30 (5) (2017) 978–991.
[37] F. Angiulli, Fast condensed nearest neighbor rule, in: Proceedings of the 22nd international conference on Machine learning, ACM, 2005, pp. 25–32.
[38] H. Wang, W. Fan, P.S. Yu, J. Han, Mining concept-drifting data streams using ensemble classifiers, in: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, AcM, 2003, pp. 226–235.
[39] A. Bifet, R. Gavaldà, Adaptive learning from evolving data streams, in: International Symposium on Intelligent Data Analysis, Springer, 2009, pp. 249–260.
[40] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, R. Gavaldà, New ensemble methods for evolving data streams, in: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2009, pp. 139–148.
[41] D. Brzezinski, J. Stefanowski, Reacting to different types of concept drift: the accuracy updated ensemble algorithm, IEEE Trans. Neural Netw. Learn. Syst. 25 (1) (2013) 81–94.