Received December 31, 2019, accepted January 7, 2020, date of publication January 10, 2020, date of current version January 23, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2965766

CODES: Efficient Incremental Semi-Supervised Classification Over Drifting and Evolving Social Streams

XIN BI[1], CHAO ZHANG[2], XIANGGUO ZHAO[2], DONGHANG LI[3], YONGJIAO SUN[2], AND YULIANG MA[4]

[1]Key Laboratory of Ministry of Education on Safe Mining of Deep Metal Mines, Northeastern University, Shenyang 110819, China
[2]School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China
[3]College of Medicine and Biological Information Engineering, Northeastern University, Shenyang 110819, China
[4]School of Business Administration, Northeastern University, Shenyang 110819, China

Corresponding author: Xin Bi (bixin@mail.neu.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFC1401902, in part by the National Natural Science Foundation of China under Grant 61702086, Grant 61672145, Grant 61572121, and Grant 61972077, in part by the Fundamental Research Funds for the Central Universities of China under Grant N171904007 and Grant N171604007, and in part by the China Postdoctoral Science Foundation under Grant 2018M631806.

ABSTRACT Classification over data streams is a crucial task of explosive social stream mining and computing. Efficient learning techniques provide high-quality services in the aspect of content distribution and event browsing. Due to the concept drift and concept evolution in data streams, the classification performance degrades drastically over time. Many existing methods utilize supervised and unsupervised learning strategies. However, supervised strategies require labeled emerging records to update the classifiers, which is unfeasible to work in the practical social stream applications. Although unsupervised strategies are popularly applied to detect concept evolution, it takes tremendous run-time computation cost to run online clustering. To this end, in this paper, we address these major challenges of social stream classification by proposing an efficient incremental semi-supervised classification method named CODES (Classification Over Drifting and Evolving Stream). The proposed CODES method consists of an efficient incremental semi-supervised learning module and a dynamic novelty threshold update module. Thus, in the drifting and evolving social streams, CODES is able to provide: 1) semi-supervised learning ability to eliminate dependency on the labels of emerging records; 2) fast incremental learning with real-time update ability to tackle concept drift; 3) efficient novel class detection ability to tackle concept evolution. Extensive experiments are conducted on several real-world datasets. The results indicate a higher performance than several state-of-the-art methods. CODES achieves efficient learning performance over drifting and evolving social streams, which improves practical significance in the real-world social stream applications.

INDEX TERMS Social stream, incremental learning, semi-supervised learning, extreme learning machine.

I. INTRODUCTION

In this era of explosive information distribution, social media platforms release feeds of up-to-date information and user-generated content to the timeline of users. The key to enhance the stickiness of content subscribers is to improve the quality of content distribution and event browsing quality, in which social stream classification plays a critical role to identify the topic of each event or record.

A major issue of the social stream classification problem is *concept drift and evolution* [1]. Concept drift occurs when the emerging social records have drifted content centroids. Concept evolution indicates class label space changes when records with new topics emerge in the social stream. Training classifiers within a fixed and known label space leads to dramatic performance degradation over time. Thus, social stream classification urges incremental update mechanism and novelty detection method to provide the abilities of both dynamic adaption to content changes and novel class detection.

The associate editor coordinating the review of this manuscript and approving it for publication was Shirui Pan.

Many researchers have studied the problems of concept drift and evolution. However, most of these existing methods are studied under the assumption that the emerging records are labeled [2]–[4], so that they can participate the iterative or incremental training procedure. In the practical applications, the emerging records are *lack of label information*. Furthermore, as the volume of feeds data and user-generated social data keeps proliferating, it is unfeasible to label the records manually.

Then unsupervised learning methods are applied to avoid dependency on record labels [5]–[7]. However, in the real-world social stream applications, the instance-based unsupervised learning methods invoke tremendous online computation to realize clustering the records without labels. The *extremely high computation cost* cannot meet the requirement of real-time social stream analysis.

Therefore, designing an efficient social stream classification faces the following thriving challenges: 1) emerging records have no training labels, which requires a *semi-supervised* learning method to eliminate dependency on social record labels; 2) the centroids of classes keep drifting over time, which requires an *incremental* learning method with real-time *update* strategy; 3) records of novel classes emerge, which requires an efficient *novel class detection* method.

In this paper, to address these challenges simultaneously, we propose a social stream classification method CODES (Classification Over Drifting and Evolving Stream), which consists of: 1) an incremental semi-supervised classification module based on an extremely fast learning method named Extreme Learning Machine (ELM) [8], [9]; 2) and an efficient novel class detection module.

To the best of our knowledge, this is the first paper to address the problem of *concept drift and evolution* in social streams using *incremental semi-supervised* extreme learning machine to provide real-time learning ability. To summarize, our contributions are as follows.

1) Following the principle of stream data management, we propose an ensemble based supervised algorithm as a baseline method.
2) We propose an incremental semi-supervised social stream method CODES, which requires no labeling information of emerging records, and efficiently handles both concept drift and evolution with real-time update ability.
3) The performance of CODES is verified in a simulated application of social stream classification using several real-world datasets. The results indicate a higher learning ability against state-of-the-art rival methods.

The remainder of the paper is organized as follows. Section II studies recent related work. Section III presents problem definitions and ELM theories. Section IV proposes the ensemble based supervised learning method. Section V proposes our incremental semi-supervised learning algorithm. We present and discuss our experimental results in Section VI, and draw conclusions in Section VII.

II. RELATED WORK

Social stream classification has been extensively studied. In earlier studies, some researchers perform *supervised learning* methods to classify social records. Support Vector Machine (SVM) [10] is used as classifier to learn the analyzed social features in the social streams [3]. Naive Bayes Model is applied for event identification from social media feeds [2]. MuENL [4] applies regularized SVMs and two update strategies. But the tree-structured detector causes a high computation cost. Furthermore, emerging records in the social stream have *no label information* for further supervised learning, and detection of concept drifting and evolving. Applying supervised learning strategy requires labeling each record before update manually, which is *not feasible* in practice.

Thus, *unsupervised learning* techniques are then applied intuitively for this real-world application scenario. ECSMiner [1] applies k-Means clustering method to identify outliers to realize novel class by time constraints for delayed classification. Topic modeling problem is studied in [5], which applies clustering method to group documents into clusters based on document similarity and co-occurrence term patterns. Augmented class is proposed in [6] based on the assumption of the availability of an unlabeled dataset. In [7], completely-random trees are applied as the unsupervised learning basis. In order to better represent a category of stream data, a metric named prototype is proposed [11] instead of cluster centroid, along with a deep neural network and update mechanism. However, these instance-based unsupervised learning consumes tremendous *online computation*, so that they cannot provide *real-time* social stream learning ability.

To address the issue of concept drift and evolution, some researchers have proposed several solutions. Han, et al. in [12] continuously update the training dataset with samples of novel classes. But this method requires *large number of parameters* and extra storage space. In [13], [14], outputs of a convolutional neural network [15] are utilized to detect novel classes from training set. But the novelty threshold have to be decided manually as a hyper-parameter. In [16], the novel class detection ability is improved by a proposed temperature scaling function. Completely-random trees are applied in [17] to ensemble an instance-based learner and a label-based learner. In [18], another ensemble framework is proposed using genetic algorithm optimization techniques. A framework SAND is proposed in [19] to detect concept drift by a change detection mechanism. But a small number of labeled data is still required to update the classifier. The detection method is exhaustive invoked, which causes *high computation cost*. The authors further improve the framework in [20] by selectively invoking the detection module using dynamic programming.

Some studies focus on *other problems* of learning over data streams. In [21], a class-based ensemble technique is proposed to distinguish between a recurring class and a novel one. In [22], matrix sketches is applied by measuring the distances in the global sketch. In [23], the event detection

problem is explored by constructing a clustering system on a large stream with a dynamic update mechanism. In [24], the personal life event prediction problem is explored to predict users' next personal life event based on their historically reported data. In [25], the changes of influential spreaders interests during event evolution are studied. A user-interest model-based event evolution model is proposed considering both user interest distribution and short text data in social networks. In [26], the feature-level changes in data streams are also studied by proposing a dynamic feature mask clustering method.

Different from these related work, in this paper, we focus on solving the social stream classification problem by a incremental semi-supervised learning method, which simultaneously provides the abilities of *semi-supervised learning* for feasibility in real-world application environment, *incremental and extremely fast update* for concept drifting, and *efficient novel class detection* for class evolution.

III. PRELIMINARIES

A. PROBLEM DEFINITION

In the social stream classification problem, each record in the social stream will be assigned to an event topic. The identification process can be considered as a classification problem. Thus, we first present the definition of a social stream, followed by the definition of the social stream classification problem.

Definition 1 (Social Stream): Given a time-sequential data stream of social records, a social stream S is noted as an infinite sequence of content entries $\{r_1, r_2, \ldots, r_i, \ldots\}$, in which r_i is a record of social media. Stream S arrives with time stamps $\{t_1, t_2, \ldots, t_i, \ldots\}$, and for any $j < k$, we have $t_j < t_k$.

In the scope of this paper, we introduce *sliding window* to denote a finite collection of social records. Then, we define record linkage task in event identification as a classification problem.

Definition 2 (Social Stream Classification): Given a social stream S, assuming there is a set $S^Y \in S$, each $r_i^Y \in S^Y$ is related to a known event topic from label space $\mathcal{L}$. For the set $S^N = S \setminus S^Y$, the problem of social stream classification is to learn a function $\varphi : S^N \to \mathcal{L}$ using a classification algorithm, so that each social record $r_i^N \in S^N$ will be assigned with an event topic from $\mathcal{L}$.

B. BRIEF OF EXTREME LEARNING MACHINE

Extreme Learning Machine (ELM) [8], [9] achieves extremely fast learning speed and good generalization performance. Many variants of ELM have been developed aiming at different training strategies and scenarios, e.g., graph data learning [27], online sequential learning [28], medical data learning [29], kernelized learning [30], [31], text classification [32], distributed learning based on MapReduce [33]–[35].

An ELM network consists of three layers of neurons. Given n input layer nodes, the hidden layer maps the input samples from the n-dimensional input data space first into the L-dimensional ELM feature mapping space, and then into the m-dimensional output data space. Figure 1 presents the ELM network structure.

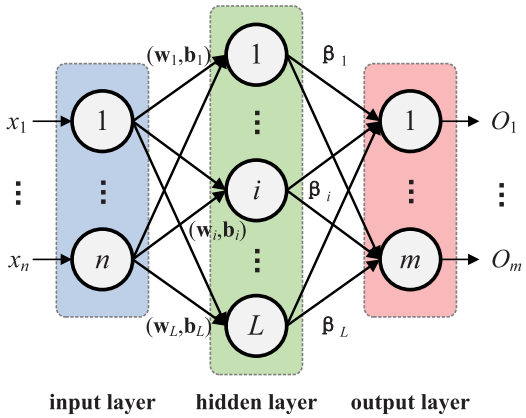


FIGURE 1. Network structure of extreme learning machine.

In order to achieve extremely fast learning speed, ELM generates parameters of the single hidden layer randomly to avoid iteratively tuning. As a generalized feedforward neural network, given N arbitrary samples $(x_i, t_i) \in R^{n\times m}$, ELM is mathematically modeled as

$$\sum_{i=1}^{L} \beta_i \, \mathrm{G}(w_i, b_i, x) = \boldsymbol{\beta}\, \mathrm{h}(\mathbf{x}) \tag{1}$$

where L is the number of hidden layer nodes, $\mathbf{w}_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^{\mathrm{T}}$ is the input weight vector from input nodes to the ith hidden node, b_i is the bias of ith hidden node, $\boldsymbol{\beta}_i$ is the output weight from the ith hidden node to the output node. $\mathrm{G}(w_i, b_i, x)$ is the activation function to generate mapping neurons, which can be any nonlinear piecewise continuous functions [36].

The ELM feature mapping denoted as $\mathbf{H}$ is calculated as

$$\mathbf{H} = \begin{bmatrix} \mathrm{G}(w_1, b_1, x_1) & \cdots & \mathrm{G}(w_L, b_L, x_1) \\ \vdots & \cdots & \vdots \\ \mathrm{G}(w_1, b_1, x_N) & \cdots & \mathrm{G}(w_L, b_L, x_N) \end{bmatrix}_{N\times L} \tag{2}$$

ELM aims to minimize the training error and the norm of the output weights, that is

$$\text{Minimize:} \quad \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|^2 \text{ and } \|\boldsymbol{\beta}\| \tag{3}$$

Therefore, the output weight $\boldsymbol{\beta}$ can be calculated as

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger} T \tag{4}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose Inverse of $\mathbf{H}$.

To gain better stability and generalization performance, the output function of ELM can be rewritten as

$$\mathrm{f}(\mathbf{x}) = \mathrm{h}(\mathbf{x})\boldsymbol{\beta} = \mathrm{h}(\mathbf{x})\mathbf{H}^{\mathrm{T}} \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^{\mathrm{T}} \right)^{-1} \mathbf{T} \tag{5}$$

In the case that the number of training samples is much larger than the dimensionality of the feature space, to reduce calculation cost, the output function can also be rewritten as

$$\mathrm{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} = \mathbf{h}(\mathbf{x})\left(\frac{\mathbf{I}}{C} + \mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{T} \tag{6}$$

IV. THE ENSEMBLE BASED SUPERVISED METHOD

In this section, we first propose an ensemble based supervised learning algorithm as the baseline method. The ensemble based method maintains an ensemble of n classifiers, and utilizes update strategy to avoid performance degradation caused by concept drift. Each time the sliding window slides, we first use the records in the current sliding window to evaluate the classifiers. If the performance of any classifier does not meet the criterion, this classifier will be replaced by a new classifier trained using emerging records.

The key features of this baseline method are as follows.

1) The sliding step size is set to the size of the sliding window, so that more stream elements can participate the learning phase in a batch.
2) The ensemble strategy combined with mechanisms such as voting theory improves the overall classification performance.
3) Lazy update strategy is applied. Whether the classifiers should be updated is determined by some evaluation criteria of classification performance.

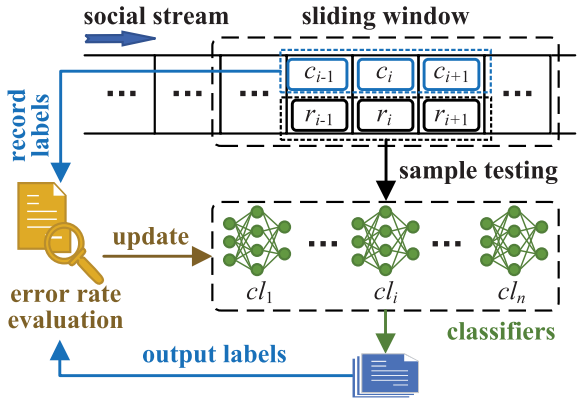


FIGURE 2. The ensemble based method.

The flow chart of the ensemble based method is presented in Figure 2. We assume that initial n classifiers have already been trained, namely $cl_1, cl_2, \ldots cl_n$, in the initiate phase. With the current sliding window, all the elements are tested by each of the n ensemble classifiers. For each classifier cl_i, we calculate its error rate as the evaluation criterion of classifiers elimination.

Definition 3 (Error Rate η): For a social stream record r_i and a classifier cl_j, if r_i is correctly classified by classifier cl_j, we set $flag_{cl_j}^{r_i} = 0$, otherwise, $flag_{cl_j}^{r_i} = 1$. Thus, for s records in the current sliding window, the error rate of classifier cl_j is

$$\eta_j = \frac{\Sigma_{i=1}^{s} flag_{cl_j}^{r_i}}{s} \tag{7}$$

If the error rate η_j of a classifier cl_j is higher than the threshold ε_η, this classifier cl_i will be eliminated. Then a new classifier is trained using all the records in the current sliding window to replace cl_i. If all the error rates are lower than ε_η, we keep all these n classifiers without update. Then the sliding window continues to slide.

Algorithm 1 The Ensemble Based Method

Input: record stream S, a initial sliding window W_{init} of size s, number of classifiers n, error rate threshold ε_η
Output: assigned topics of emerging records

```
1  R_init = {r_1, ..., r_s | r_i ∈ W_init};
2  Train n initial classifiers using R_init;
3  W_current = W_init.slide(step = s);
4  while !S.end() do
5      R_current = {r_i | r_i ∈ W_current};
6      for i = 1 to n do
7          for j = 1 to s do
8              calculate the output O_j^i of record r_j using cl_i;
9          calculate error rate η_i of cl_i;
10         if η_i > ε then
11             train a new classifier cl_i' using R_current to
               replace cl_i;
12     for i = 1 to s do
13         assign the topic label with the most votes to r_i;
14     W_current = W_current.slide(step = s);
```

The procedure of the ensemble based method is described as Algorithm 1. The algorithm accepts inputs of a feeds stream S, a initial sliding window W_{init}, the sliding step size s, the number of classifiers n, and the error rate threshold ε_η, we first initiate n classifiers using the set R_{init}, which contains the s records in the initial sliding window W_{init} (Lines 1,2). After the first slide (Line 3), for each classifier cl_i (Lines 6-11), the outputs of each record r_j in the current sliding window $W_{current}$ are calculated (Lines 7, 8). Then the error rate η_i of each classifier cl_i is calculated (Line 8). If the error rate η_i of classifier cl_i is larger than threshold ε_η, cl_i is eliminated and a new classifier is trained using all the records $R_{current}$ in the current sliding window $W_{current}$ (Line 11). With all the outputs of the s elements from the n classifiers, voting mechanism is utilized to identify the event topics with the most votes to each record (Lines 12, 13). The whole procedure will be executed iteratively along with the social stream over time (Line 14).

V. THE PROPOSED METHOD

In this section, we propose our method CODES (Classification over Drifting and Evolving Stream), which consists of a learning module and a novel class detection module. CODES applies: 1) incremental learning strategy to learn the emerging records in combination with historical learning

experience; 2) semi-supervised learning strategy to provide practical learning ability of the unlabeled emerging records in the real-world social stream; 3) real-time update mechanism based on an extreme fast learning method.

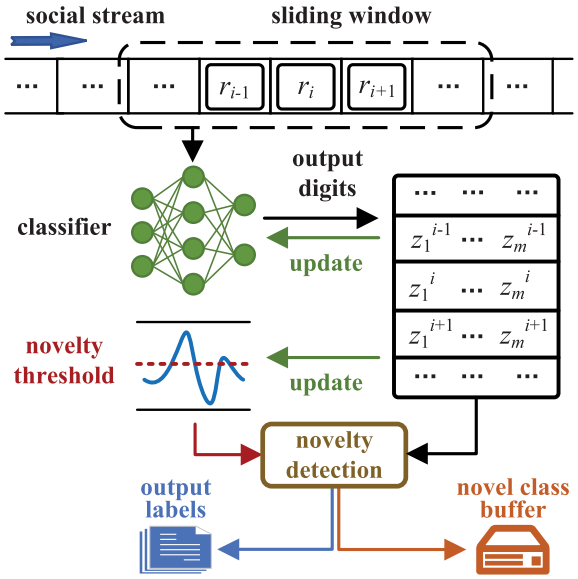


FIGURE 3. The incremental semi-supervised method CODES.

The major procedure of CODES is demonstrated in Figure 3. The current classifier first calculate the output digits of each emerging record r_i in the current sliding window. These emerging records are first utilized to incrementally update the current classifier, which will be presented in detail in the following Section V-A. Then CODES executes novel class detection to determine whether the emerging records belong to a novel class according to the output digits. The novelty threshold is also updated utilizing the output digits. This novel class detection module will be presented in detail in Section V-B.

A. INCREMENTAL SEMI-SUPERVISED LEARNING

Aiming at the setting of training set containing both labeled and unlabeled samples, the semi-supervised learning strategy based on extreme learning machine is able to maintain the ELM feature mapping space without label information of emerging records in the social stream [36].

Given l labeled data $\{\mathbf{x}_i, \mathbf{t}_i\}_{i=1}^{l}$ and u unlabeled data $\{\mathbf{x}_i\}_{i=1}^{u}$, the optimization problem of semi-supervised extreme learning machine is written as

$$\begin{aligned} \text{minimize: } & \frac{1}{2}||\beta||^2 + \frac{1}{2}\sum_{i=1}^{N}\mathbf{C}_i||\xi_i||^2 + \frac{\lambda}{2}\text{Tr}(\mathbf{F})^\top\mathbf{L}\mathbf{F} \\ \text{subject to: } & \text{h}(\mathbf{x}_i)\beta = \mathbf{t}_i^\top - \xi_i^\top, \quad i = 1, \ldots, l \\ & \text{f}(\mathbf{x}_i) = \text{h}(\mathbf{x}_i)\beta, \quad i = 1, \ldots, l+u \end{aligned} \tag{8}$$

where $\mathbf{L} \in \mathbf{R}^{(l+u)\times(l+u)}$ is the Laplacian matrix which is generated by both labeled and unlabeled training samples. $\mathbf{F} \in \mathbf{R}^{(l+u)\times m}$ is the ELM output with its ith row equal to $\text{f}(\mathbf{x}_i)$ and $\text{Tr}(\mathbf{F})$ denotes the trace of matrix $\mathbf{F}$ based on manifold regularization framework. λ is a tradeoff parameter. Therefore, the output weight β is calculated as

$$\boldsymbol{\beta} = (\mathbf{I}_L + \mathbf{H}^\top\mathbf{C}\mathbf{H} + \lambda\mathbf{H}^\top\mathbf{L}\mathbf{H})^{-1}\mathbf{H}^\top\mathbf{C}\mathbf{T} \tag{9}$$

when the number of samples N is larger than the number of hidden layer nodes L, or

$$\boldsymbol{\beta} = \mathbf{H}^\top(\mathbf{I}_L + \mathbf{C}\mathbf{H}\mathbf{H}^\top + \lambda\mathbf{L}\mathbf{H}\mathbf{H}^\top)^{-1}\mathbf{C}\mathbf{T} \tag{10}$$

when L is larger than N.

Given a chunk of records, which will be viewed as the set of the elements in the current sliding window, the output weight $\boldsymbol{\beta}_{k+1}$ is calculated by a transition matrix $\mathbf{P}_{k+1}$ as

$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \mathbf{P}_{k+1}\mathbf{H}_{k+1}^\top(\mathbf{T}_{k+1} - \mathbf{H}_{k+1}\boldsymbol{\beta}_k) \tag{11}$$

where

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \mathbf{P}_k\mathbf{H}_{k+1}^\top(\mathbf{I} + \mathbf{H}_{k+1}\mathbf{P}_k\mathbf{H}_{k+1}^\top)^{-1}\mathbf{H}_{k+1}\mathbf{P}_k \tag{12}$$

For convenience, we define a matrix

$$\mathbf{Q} = \mathbf{I}_L + \mathbf{H}^\top\mathbf{C}\mathbf{H} + \lambda\mathbf{H}^\top\mathbf{L}\mathbf{H} \tag{13}$$

then Equation 9 can be rewritten as

$$\boldsymbol{\beta} = \mathbf{Q}^\top\mathbf{H}^\top\mathbf{C}\mathbf{T} \tag{14}$$

For the first chunk $\mathbb{N}_0$ of emerging records, the initial output weight $\boldsymbol{\beta}_0$ can be calculated using 14. When the second chunk $\mathbb{N}_1$ of records emerge in the sliding window, the updated output weight $\boldsymbol{\beta}_1$ is calculated as

$$\boldsymbol{\beta}_1 = \mathbf{Q}_1^\top\begin{bmatrix}\mathbf{H}_0\\ \mathbf{H}_1\end{bmatrix}^\top\begin{bmatrix}\mathbf{C}_0 & 0\\ 0 & \mathbf{C}_1\end{bmatrix}\begin{bmatrix}\mathbf{T}_0\\ \mathbf{T}_1\end{bmatrix} \tag{15}$$

where

$$\begin{aligned} \mathbf{Q}_1 &= \mathbf{I}_L + \begin{bmatrix}\mathbf{H}_0\\ \mathbf{H}_1\end{bmatrix}^\top\begin{bmatrix}\mathbf{C}_0 & 0\\ 0 & \mathbf{C}_1\end{bmatrix}\begin{bmatrix}\mathbf{H}_0\\ \mathbf{H}_1\end{bmatrix} \\ &\quad + \lambda\begin{bmatrix}\mathbf{H}_0\\ \mathbf{H}_1\end{bmatrix}^\top\mathbf{L}_{\mathbb{N}_0+\mathbb{N}_1}\begin{bmatrix}\mathbf{H}_0\\ \mathbf{H}_1\end{bmatrix} \\ &= \mathbf{I}_L + \mathbf{H}_0^\top\mathbf{C}_0\mathbf{H}_0 + \mathbf{H}_1^\top\mathbf{C}_1\mathbf{H}_1 + \lambda\mathbf{H}_0^\top\mathbf{L}_{\mathbb{N}_0}\mathbf{H}_0 \\ &\quad + \lambda\mathbf{H}_1^\top\mathbf{L}_{\mathbb{N}_1}\mathbf{H}_1 \\ &= \mathbf{Q}_0 + \mathbf{H}_1^\top(\mathbf{C}_1 + \lambda\mathbf{L}_{\mathbb{N}_1})\mathbf{H}_1 \end{aligned} \tag{16}$$

Substituting 16 into 15, we have the final calculation equation of output weight matrix as

$$\boldsymbol{\beta}_1 = \boldsymbol{\beta}_0 + \mathbf{Q}_1^{-1}\mathbf{H}_1^\top[\mathbf{C}_1\mathbf{T}_1 - (\mathbf{C}_1 + \lambda\mathbf{L}_{\mathbb{N}_1})\mathbf{H}_1\boldsymbol{\beta}_0] \tag{17}$$

According to this derivation, we have the general calculation of the $(k+1)th$ chunk of records as

$$\mathbf{Q}_{k+1} = \mathbf{Q}_k + \mathbf{H}_{k+1}^\top(\mathbf{C}_{k+1} + \lambda\mathbf{L}_{\mathbb{N}_1})\mathbf{H}_{k+1} \tag{18}$$

and the output weight

$$\begin{aligned} \boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \mathbf{Q}_{k+1}^{-1}\mathbf{H}_{k+1}^\top[&\mathbf{C}_{k+1}\mathbf{T}_{k+1} \\ &- (\mathbf{C}_{k+1} + \lambda\mathbf{L}_{\mathbb{N}_{k+1}})\mathbf{H}_{k+1}\boldsymbol{\beta}_k] \end{aligned} \tag{19}$$

To simplify the equation, we define $\mathbf{P}_{k+1} = \mathbf{Q}_{k+1}^{-1}$, where

$$\begin{aligned}\mathbf{P}_{k+1} = \mathbf{P}_k - \mathbf{P}_k\mathbf{H}_{k+1}^T \\ (\mathbf{I}_L + (\mathbf{C}_{k+1} + \lambda\mathbf{L}_{\mathbb{N}_{k+1}})\mathbf{H}_{k+1}\mathbf{P}_k\mathbf{H}_{k+1}^\top)^{-1} \\ (\mathbf{C}_{k+1} + \lambda\mathbf{L}_{\mathbb{N}_{k+1}})\mathbf{H}_{k+1}\mathbf{P}_k \end{aligned} \tag{20}$$

Thus, the $(k+1)$th output weight matrix is finally calculated as

$$\begin{aligned}\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \mathbf{P}_{k+1}\mathbf{H}_{k+1}^\top[\mathbf{C}_{k+1}\mathbf{T}_{k+1} \\ - (\mathbf{C}_{k+1} + \lambda\mathbf{L}_{\mathbb{N}_{k+1}})\mathbf{H}_{k+1}\boldsymbol{\beta}_k] \end{aligned} \tag{21}$$

Algorithm 2 Incremental Semi-Supervised Learning of CODES

Input: social stream S, a sliding window W of size s, an initial record set R_{init}
Output: updated output weight $\boldsymbol{\beta}$ of the classifier, assigned topics of emerging records

1 init the incremental indicator $k = 0$;
2 randomly generate **w** and bias **b**;
3 generate initial mapping matrix $\mathbf{H}_k$ of R_{init};
4 $W_{current} = W_{init}.slide(step = s)$;
5 **while** *!S.end()* **do**
6 $R_{current} = \{r_i \mid r_i \in W_{current}\}$;
/* Label the emerging records: */
7 **for** $i = 1$ *to* s **do**
8 calculate output of r_i as $\mathbf{O}_i$;
9 assign $argmax(\mathbf{O}_i)$ as the topic label to record r_i;
/* Update the classifier: */
10 calculate matrix $\mathbf{H}_{k+1}$;
11 calculate matrix $\mathbf{P}_{k+1}$;
12 calculate output weight $\boldsymbol{\beta}_{k\mathcal{C}1}$;
/* The stream continues: */
13 $W_{current} = W_{current}.slide(step = s)$;
14 $k = k + 1$;

Algorithm 2 describes the pseudocode of CODES. The initial classifier is first trained (Line 2) using the initial record set R_{init} (Line 3). After the initiate phase, the sliding window starts to slide with a step size of s (Line 4). We update the current chunk set by the emerging records in the current sliding window (Line 6). Then, for each record r_i in the $R_{current}$ (Lines 7-9), we first calculate the output $\mathbf{O}_i$ (Line 8) and assign a topic label to r_i (Line 9). Then we update the classifier (Lines 10-12) by calculating the $(k+1)$th feature mapping matrix $\mathbf{H}_{k+1}$, transitive matrix $\mathbf{P}_{k+1}$, and output weight matrix $\boldsymbol{\beta}_{k+1}$. The whole procedure of CODES is executed iteratively along with the social stream over time (Line 13). In each round of sliding, the indicator k is incremented by one (Line 14).

B. NOVEL CLASS DETECTION

Traditional classification problems assume a fixed label space of all the training samples, testing samples, and even input instances in the practical applications. However, this assumption is unpractical in the social stream learning tasks. New hot topics and trending events keep emerging over time in the social stream.

To solve this concept evolution problem, in this section, a novel class detection method is proposed. Whether a record r in the social stream belongs to a novel class or a class known a priori depends on two measurements: 1) the topic novelty of r; 2) principles of distinguishing between a novel class and known classes according to the topic novelty of r.

1) NOVELTY METRIC

The final outcomes of original ELM, which are distributed in the interval [0, 1] by the *sigmoid* function, do not sum to one. Although this distribution has no effect on the decision of output class label using *argmax* operation, the performance decreases when the outcomes are utilized as an absolute metric directly. Therefore, we evaluate the classification output of r according to the *softmax* value. Each logit of a corresponding class $i \in \mathcal{L}$ is calculated as

$$z_i = \frac{e^{o_i}}{\sum_{j=1}^{m} e^{o_j}} \tag{22}$$

where e^{o_i} is the exponential (e-power) of the outcome probability o_i of class i, $\sum_{j=1}^{m} e^{o_j}$ is the sum of exponential values of all the classes. Then we measure the class novelty of record r using the maximum logit as

$$P^r = \max_{i \in \mathcal{L}}(z_i^r) \tag{23}$$

2) DISTINGUISHING PRINCIPLE

The distinguishing principle is formulated by our defined novelty threshold.

Definition 4 (Novelty Threshold ε): Given an emerging record r and its output vector of the classifier $\mathbf{O}_r = \{o_1, o_2, \cdots, o_m\}$, where m is the current number of classes, a novelty threshold ε is utilized as a criterion for novel class detection. $\mathbf{r}_i$ is considered as an instance of a novel class if $softmax(\mathbf{O}_r) < \varepsilon$.

According to the concept of concept evolution, whether a record should be assigned to a novel class should be determined by two spaces: 1) the current classification outcome space; 2) the known class label space. By evaluating the ELM outcomes with a range of float, we calculate the novelty threshold statistically as

$$\varepsilon = \min_{i \in \mathcal{L}}(\{\overline{P_i} + \sigma_i\}) \tag{24}$$

where $\mathcal{L}$ is the current class label space, $\overline{P_i}$ is the mean novelty value of all the samples belonging to class $i \in \mathcal{L}$, σ_i is the standard deviation of class i.

$\overline{P_i}$ and σ_i are respectively calculated as

$$\overline{P_i} = mean\{z_i^k, k = 1, \cdots, N_i\} = \frac{1}{N_i}\sum_{k=1}^{N_i} z_i^k \tag{25}$$

$$\sigma_i = \sqrt{\frac{1}{N_i}\sum_{k=1}^{N_i}(z_i^k - \overline{P}_i)} \tag{26}$$

where z_i^k is the logit of the kth sample to the class i, N_i is the sample number of class i.

3) DETECTION METHOD

Based on the calculations of novelty metric and novelty threshold, the procedure of novel class detection is described as Algorithm 3.

Algorithm 3 Novel Class Detection of CODES

Input: a new chunk of records $\mathcal{N}_{new}$, the classifier C, current threshold ε, label space $\mathcal{L}$
Output: class labels of $\mathcal{N}_{new}$, updated threshold ε'

```
/* Generate output labels:                                  */
1  Generate outputs O_{N_new} of N_new from C;
2  foreach record r ∈ N_new do
3      calculate P^r = max_{i∈L}(z_i^r);
4      if P_r < ε then
5          add r to the buffer of novel class;
6      else
7          assign argmax(O_r) as the result label;
/* Update the classifier:                                   */
8  update output weight β of C using Algorithm 2;
/* Update the threshold:                                    */
9  foreach i ∈ L do
10     calculate P_i' using N_new by equation 25;
11     calculate σ_i' using N_new by equation 26;
12     update ε_i' = P_i' + σ_i';
13 update ε = min_{i∈L}{ε_i'};
```

Given the current classifier C and threshold ε, and a new chunk of records $\mathcal{N}_{new}$ generated by the current sliding window over the social stream, our novel class detection method first generates the outputs of $\mathcal{N}_{new}$ using the current classifier C (Line 1). For each record r in the $\mathcal{N}_{new}$, we determine the class label according to the *softmax* value of the output $\mathbf{O}_r$ and the current threshold ε (Lines 2-7). If the *softmax* value is smaller than the threshold ε, we assign r to the novel class temporarily (Line 5); otherwise, we assign the corresponding class with the maximum outcome probability as the output label of r (Line 7).

As to the records in buffer of temporary novel class, an unsupervised learning method can be utilized to automatically decide how many different topics do these records involve. Semantic labels or topics should be assigned manually, since there is no prior knowledge of these emerging social records, which is beyond the discussion scope of this paper.

As the incremental semi-supervised learning continues, we update both the current classifier C and threshold ε according to the outputs $\mathbf{O}_{\mathcal{N}_{new}}$ of $\mathcal{N}_{new}$. The classifier C is updated using Algorithm 2 (Line 8). The threshold ε is updated (Line 13) after average output scores $\overline{P_i}$ and standard deviations σ_i are updated (Lines 10-12).

VI. EXPERIMENTS

In this section, we first introduce our experiments setup. Then we present the evaluation results of classification performance and novel class detection performance, respectively.

A. EXPERIMENTS SETUP

1) DATASETS

We obtain the dataset of social stream from a Twitter corpus[1] using our modified data fetching scripts. After filtering out a number of invalid tweets, we have a Twitter dataset of 3600 tweets under four topic labels. Each tweet record is assigned one topic label.

We also fetch RSS feeds from IBM DeveloperWorks[2] and ABC News.[3] Each semi-structured document in the feeds is composed of several elements of title, author, summary, publish information, etc. We use the article channels as the event topics. Each of these two datasets consists of 3600 feeds entries. Each record is categorized into one of six event topics. IBM and ABC both publish content feed in the format of an XML file. Each XML file contains several content entries. The records have to be transformed into a representation space before taken as inputs to the classifiers. XML files contain not only semantic information, but also structural information. Thus, in our experiments, we utilize our previously proposed Distribution based Structured Vector Model (DSVM) [37] to further strengthen the ability of representation.

Finally, all the records of these three datasets are further mapped into embeddings using pretrained word2vec[4] model.

2) RIVAL METHODS

We choose four state-of-the-art methods of stream classification as our rival methods, including two highly-cited classical methods and two deep learning based methods.

1) **ECSMiner** [1], which detects novel classes using an ensemble of k-Means clustering.
2) **SAND** [20], which improves ECSMiner by applying semi-supervised learning methods.
3) **HG-CNN** [14], which implements a deep convolutional neural network for novel class detection.
4) **ODIN-CNN** [16], which is an improvement over HG-CNN.

All the methods are realized using MATLAB R2018b, Python 3.6. The deep learning based methods are implemented using Keras 2.2.0 with TensorFlow 1.8.0 as backend. Experiments are conducted on a PC with Intel Core i7 8700K

[1] http://www.sananalytics.com/lab/twitter-sentiment/
[2] http://www.ibm.com/developerworks
[3] http://abcnews.go.com
[4] https://code.google.com/archive/p/word2vec/

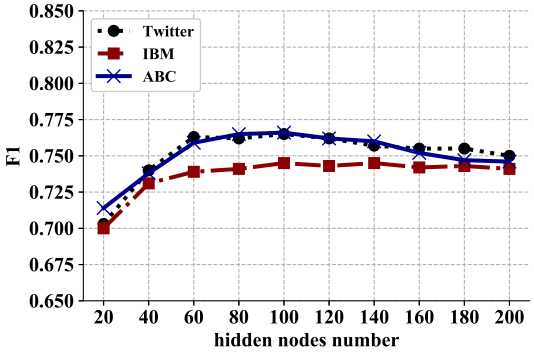
CPU, NVIDIA GeForce GTX 1080 Ti GPU, 32GB RAM, and 64-bit Windows 10 as operating system.

3) EVALUATION SETUP

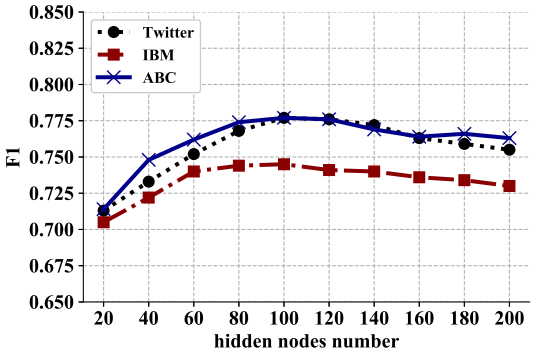
We evaluate our algorithms in two different scenarios.

1) *Classification performance evaluation*, which focus on concept drift. The influences of four hyper-parameters are observed, including the number of hidden layer nodes L, the number of ensemble n, the error rate threshold ε_η, and the size of sliding window s.
2) *Novel class detection evaluation*, which focuses on novel class detection. We simulate the streaming data environment to evaluate the performance of novel class detection.

Two evaluation metrics are used. *Training time* is used to evaluate the learning efficiency. *F-measure* (F1) is defined as the harmonic mean of the accuracy and recall.



(a) Ensemble based method



(b) CODES method

FIGURE 4. F1 evaluation of the hidden nodes number.

B. CLASSIFICATION PERFORMANCE EVALUATION

1) NUMBER OF HIDDEN LAYER NODES

This set of experiments evaluates the influence of hidden nodes number. The results presented in Figure 4 indicate that both the ensemble based method and CODES have the maximum F1 value when their networks contains around 100 hidden nodes. Thus, we set the number of hidden layer nodes to 100 in the following experiments.

2) NUMBER OF ENSEMBLE CLASSIFIERS

The ensemble based method maintains n classifiers as an ensemble. Since ensemble number has no influence on training time, we evaluate the influence of the ensemble number n on F1 performance. The ensemble number ranges from 20 to 70 with a step size of 10. The results are presented in Figure 5.

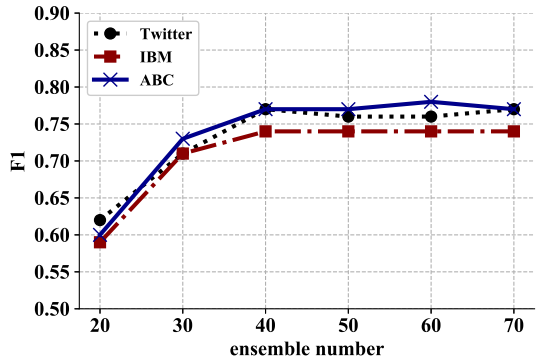


FIGURE 5. F1 evaluation of the ensemble number.

The result demonstrates that although training more classifiers costs more learning time, a larger number of ensemble classifiers leads to a better classification performance. The F1 performance stops increasing when the ensemble number grows to 40. Thus, we set the ensemble number n to 40 in our following experiments.

3) ERROR RATE THRESHOLD

In this set of experiments, we evaluate the influence of error rate threshold ε_η on F1 performance. The error rate threshold increases from 0.4 to 0.9 with a step size of 0.1. The results are presented in Figure 6.

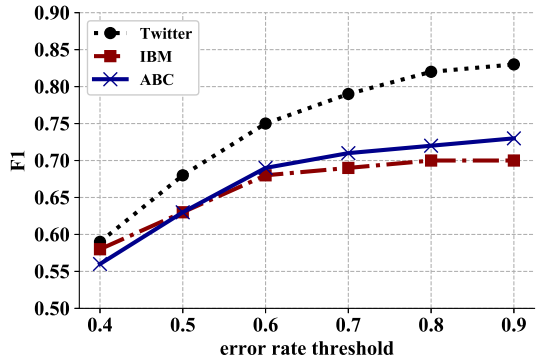
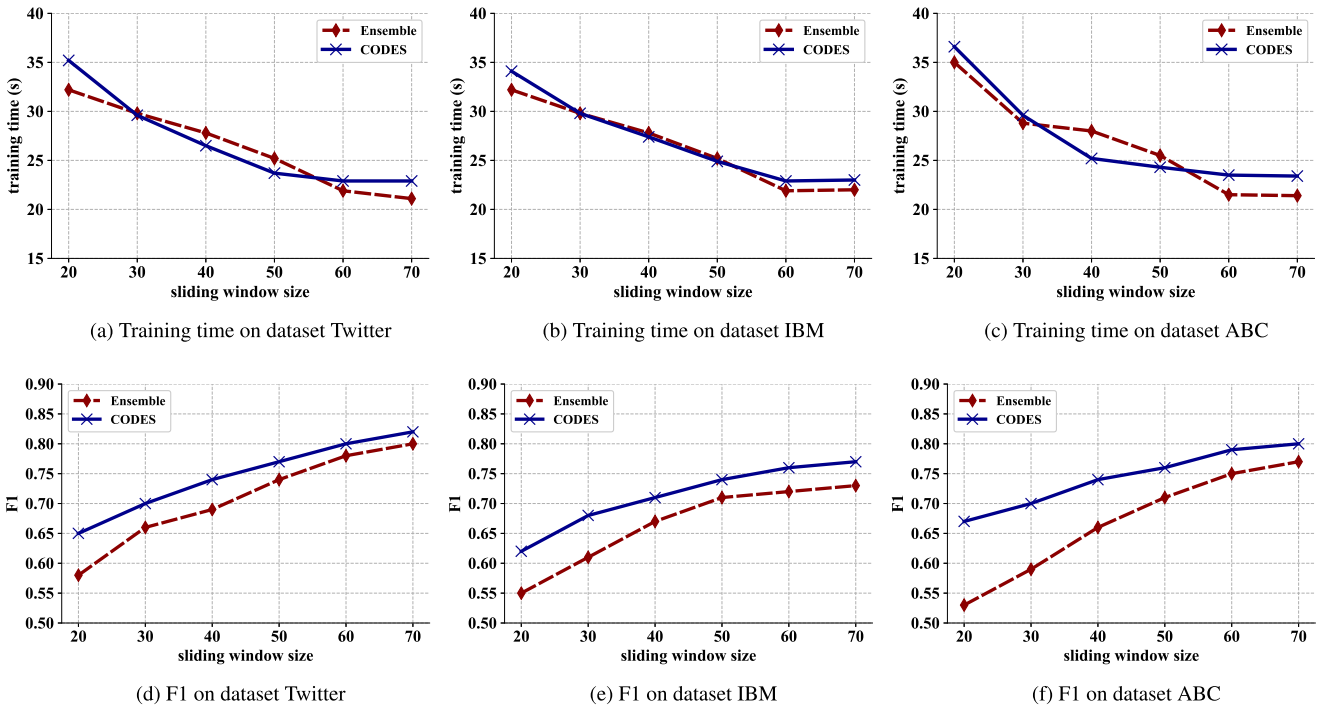


FIGURE 6. F1 evaluation of the error rate threhold.

The results demonstrate that as the threshold grows, the F1 values increases as well. A larger error rate threshold indicates a more strict criterion. To avoid frequent updates in the ensemble based method, we set the error rate ε_η to 0.7 in our following experiments.

4) SIZE OF SLIDING WINDOWS

In this set of experiments, we evaluate the influence of the sliding window size s on both training time and



(a) Training time on dataset Twitter (b) Training time on dataset IBM (c) Training time on dataset ABC

(d) F1 on dataset Twitter (e) F1 on dataset IBM (f) F1 on dataset ABC

FIGURE 7. Evaluation of the sliding window size.

F1 performance. The size of sliding window increases from 20 to 70 with a step size of 10.

Figure 7 demonstrates the influence of sliding window size on training time and F1 performance. When the size of sliding window is larger than 60, the ensemble based method takes less training time than the CODES method. As to the ensemble based method, the lazy update strategy reduces the frequency of classifier retraining. In other words, a larger sliding window size leads to less classifier updates. On the other hand, as to CODES, relatively more complex matrix operations takes more training time. Thus, when the size of sliding window is larger than 60, the ensemble based method has a higher training speed than CODES.

In the aspect of F1 performance, all the methods achieve higher performance as the sliding window increases. Although CODES outperforms the ensemble based method in all the parameter settings, we notice that the gap between CODES and the ensemble based method is decreasing as the sliding window size increases. The reason is that when the sliding window size is relatively small, the algorithm learns from a mall number of training samples, so that the contribution of the ensemble strategy is very limited. It is intuitive that more training samples leads to better learning performance at the cost of more training time.

C. EVALUATION OF NOVEL CLASS DETECTION

In this set of experiments, we simulate the real-world social stream environment to evaluate the novel class detection performance. We control the emergence of records and classes by assuming that:

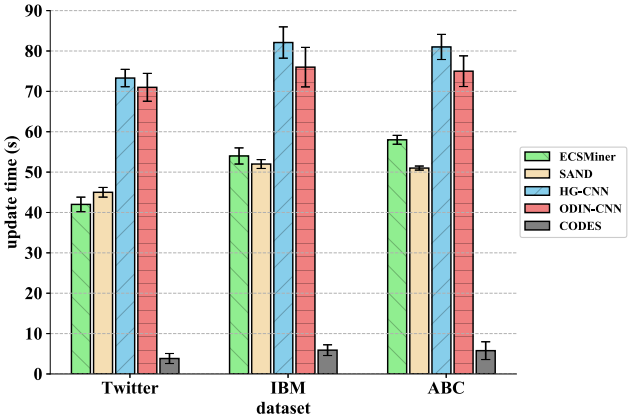


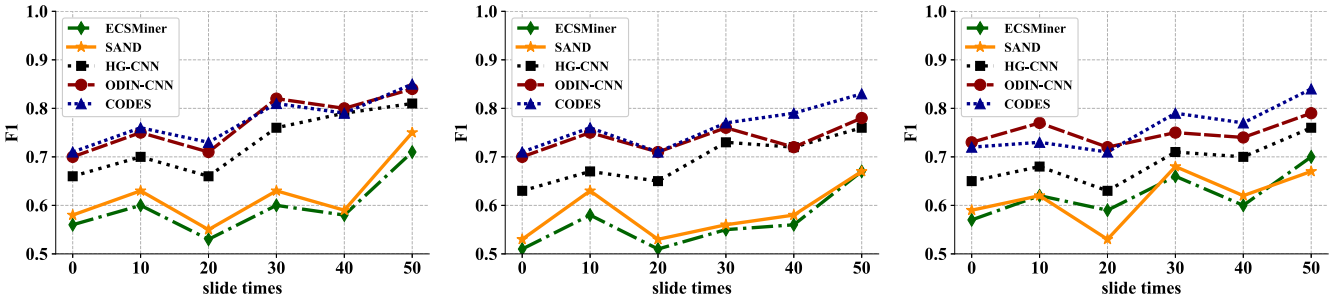
FIGURE 8. Comparison of update time.

1) all the emerging records are unlabeled;
2) the novel classes do not emerge in the initial phase.

The experiments are initiated with 600 labeled records out of total 3600 records. Then we increase the training records without labels to evaluate both training time and classification performance. The sliding window size is set to 60, so that the sliding windows slides 50 times in total. All the records emerge in the social stream are selected uniformly at random.

The emergence of novel classes are controlled manually. In each dataset, we randomly choose two classes as the novel classes. The emergence of the records belonging to these two novel class started from the 20th and 40th slide of the sliding window, respectively.

We first compare training time among all the methods in this simulated social stream application with novel classes.



(a) dataset Twitter (b) dataset IBM (c) dataset ABC

FIGURE 9. Comparison of F1 performance: two novel classes emerge at 20 and 40, respectively.

TABLE 1. Accuracy and recall of novel class classification.

Methods	Twitter		IBM		ABC	
	Accuracy	Recall	Accuracy	Recall	Accuracy	Recall
ECSMiner	0.729 ± 0.145	0.321 ± 0.224	0.701 ± 0.139	0.268± 0.250	0.718 ± 0.085	0.222 ± 0.183
SAND	0.826 ± 0.133	0.427 ± 0.124	0.756 ± 0.115	0.359 ± 0.293	0.755 ± 0.037	0.474 ± 0.175
HG-CNN	0.887 ± 0.024	0.689 ± 0.084	0.853 ± 0.023	0.589± 0.141	**0.899 ± 0.013**	0.616 ± 0.078
ODIN-CNN	**0.914 ± 0.008**	0.727 ± 0.061	**0.922 ± 0.009**	**0.699 ±0.167**	**0.908 ± 0.003**	0.700 ± 0.013
CODES	**0.913 ± 0.005**	**0.731 ± 0.025**	0.904 ± 0.011	**0.698 ±0.188**	0.902 ± 0.008	**0.715 ± 0.025**

We focus on the update time of each incremental learning or classifier retraining. The results presented in Figure 8 indicate that our CODES method has a significant advantage over all the rival methods, especially the deep learning based methods HG-CNN and ODIN-CNN. The extremely fast learning speed of ELM has a great contribution to realizing real-time update and incremental learning ability of CODES.

We also compare F1 performance among all the methods. We observe the results along with the sliding window. The results presented in Figure 9 indicate that: 1) all the methods have an overall tendency of increasing F1 values over time; 2) the performance drops when the novel classes emerge in the stream; 3) the performance decay of CODES and deep learning is less during the emergence of novel classes; 4) our CODES method and deep learning methods notably outperform ECSMiner and SAND all through the simulated stream; 5) our CODES method has a slight advantage over the deep learning methods both during the emergence of the novel classes and all through the stream.

Finally, we focus on the novel class detection performance by evaluating accuracy and recall of only the records belonging to the novel classes. Accuracy indicates the proportion of the correctly classified novel-class records to the records which are classified as novel classes. Recall indicates the proportion of the correctly classified novel-class records to the records which actually belong to novel classes.

Table 1 presents the comparison results with standard deviation. The accuracy performance is significantly higher than the recall performance. In other words, all these methods are confident of their output as long as they classify the records as novel classes, but struggle to identify the novel classes as many as possible. On the other hand, as to the overall performance of novel class detection, our proposed CODES and deep learning based methods outperform traditional methods significantly in the aspect of both accuracy and recall performance.

VII. CONCLUSION

In this paper, we address the problem of concept drift and evolution, which is the major issue of social stream classification. In order to eliminate dependency on the labeling information of emerging records for practical applications, and provide incremental learning ability, we propose an incremental semi-supervised learning method CODES, which consists of a classification module and a novel class detection module. The classification module provides fast incremental update mechanism to tackle concept drift. The novel class detection module efficiently detects novel class to tackle concept evolution. Extensive experiments are conducted on several real-world datasets. The results indicate that our method CODES achieves a higher performance of both incremental learning and novel class detection without labeling information of emerging records. The proposed method provides practical significance in real-time social stream learning applications.

REFERENCES

[1] M. Masud, J. Gao, L. Khan, J. Han, and B. M. Thuraisingham, "Classification and novel class detection in concept-drifting data streams under time constraints," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 6, pp. 859–874, Jun. 2011.

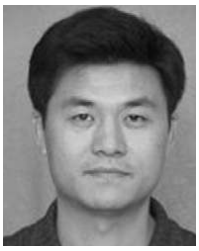
[2] C. Trabelsi and S. Yahia, "A probabilistic approach for events identification from social media RSS feeds," in *Proc. Database Syst. Adv. Appl.*, vol. 7827, 2013, pp. 139–152.

[3] A. Zubiaga, D. Spina, R. Martínez, and V. Fresno, "Real-time classification of Twitter trends," *J. Assoc. Inf. Sci. Technol.*, vol. 66, no. 3, pp. 462–473, Mar. 2015.

[4] Y. Zhu, K. M. Ting, and Z.-H. Zhou, "Multi-label learning with emerging new labels," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 10, pp. 1901–1914, Oct. 2018.
[5] D. M. Blei, "Probabilistic topic models," *Commun. ACM*, vol. 55, no. 4, pp. 77–84, Apr. 2012.
[6] Q. Da, Y. Yu, and Z. Zhou, "Learning with augmented class by exploiting unlabeled data," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, pp. 1760–1766.
[7] X. Mu, K. M. Ting, and Z.-H. Zhou, "Classification under streaming emerging new classes: A solution using completely-random trees," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 8, pp. 1605–1618, Aug. 2017.
[8] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *Proc. Int. Symp. Neural Netw.*, vol. 2, 2004.
[9] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, Dec. 2006.
[10] C. Cortes and V. N. Vapnik, "Support vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1997.
[11] Z. Wang, Z. Kong, S. Changra, H. Tao, and L. Khan, "Robust high dimensional stream classification with novel class detection," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 1418–1429.
[12] S. Han, Z. Meng, A. Khan, and Y. Tong, "Incremental boosting convolutional neural network for facial action unit recognition," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2016, pp. 109–117.
[13] A. Bendale and T. E. Boult, "Towards open set deep networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1563–1572.
[14] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *Proc. 5th Int. Conf. Learn. Represent.*, 2017.
[15] Y. Lecun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.
[16] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2018.
[17] X. Mu, F. Zhu, Y. Liu, E.-P. Lim, and Z.-H. Zhou, "Social stream classification with emerging new labels," in *Proc. Adv. Knowl. Discovery Data Mining*, 2018, pp. 16–28.
[18] H. Ghomeshi, M. M. Gaber, and Y. Kovalchuk, "RED-GENE: An evolutionary game theoretic approach to adaptive data stream classification," *IEEE Access*, vol. 7, pp. 173944–173954, 2019.
[19] A. Haque, L. Khan, and M. Baron, "SAND: Semi-supervised adaptive novel class detection and classification over data stream," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 1652–1658.
[20] A. Haque, L. Khan, M. Baron, B. Thuraisingham, and C. Aggarwal, "Efficient handling of concept drift and concept evolution over stream data," in *Proc. IEEE 32nd Int. Conf. Data Eng. (ICDE)*, May 2016, pp. 481–492.
[21] T. Al-Khateeb, M. M. Masud, L. Khan, C. Aggarwal, J. Han, and B. Thuraisingham, "Stream classification with recurring and novel class detection using class-based ensemble," in *Proc. IEEE 12th Int. Conf. Data Mining*, Dec. 2012, pp. 31–40.
[22] X. Mu, F. Zhu, J. Du, E. Lim, and Z. Zhou, "Streaming classification with emerging new class by class matrix sketching," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 2373–2379.
[23] M. Fedoryszak, B. Frederick, V. Rajaram, and C. Zhong, "Real-time event detection on social data streams," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2019, pp. 2774–2782.
[24] M. Khodabakhsh, H. Fani, F. Zarrinkalam, and E. Bagheri, "Predicting personal life events from streaming social content," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2018, pp. 1751–1754.
[25] L.-L. Shi, L. Liu, Y. Wu, L. Jiang, and J. Hardy, "Event detection and user interest discovering in social media data streams," *IEEE Access*, vol. 5, pp. 20953–20964, 2017.
[26] C. Fahy and S. Yang, "Dynamic feature selection for clustering high dimensional data streams," *IEEE Access*, vol. 7, pp. 127128–127140, 2019.
[27] X. Bi, X. Zhao, H. Huang, D. Chen, and Y. Ma, "Functional brain network classification for alzheimer's disease detection with deep features and extreme learning machine," *Cognit. Comput.*, pp. 1–15, 2019.
[28] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006.
[29] X. Bi, H. Ma, J. Li, Y. Ma, and D. Chen, "A positive and unlabeled learning framework based on extreme learning machine for drug-drug interactions discovery," *J. Ambient Intell. Hum. Comput.*, pp. 1–12, Aug. 2018.
[30] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 513–529, Apr. 2012.
[31] X. Bi, X. Zhao, G. Wang, P. Zhang, and C. Wang, "Distributed extreme learning machine with kernels based on MapReduce," *Neurocomputing*, vol. 149, pp. 456–463, Feb. 2015.
[32] X.-G. Zhao, G. Wang, X. Bi, P. Gong, and Y. Zhao, "XML document classification based on ELM," *Neurocomputing*, vol. 74, no. 16, pp. 2444–2451, Sep. 2011.
[33] Q. He, T. Shang, F. Zhuang, and Z. Shi, "Parallel extreme learning machine for regression based on MapReduce," *Neurocomputing*, vol. 102, pp. 52–58, Feb. 2013.
[34] J. Xin, Z. Wang, L. Qu, G. Yu, and Y. Kang, "A-ELM*: Adaptive distributed extreme learning machine with MapReduce," *Neurocomputing*, vol. 174, pp. 368–374, Jan. 2016.
[35] B. Wang, S. Huang, J. Qiu, Y. Liu, and G. Wang, "Parallel online sequential extreme learning machine based on MapReduce," *Neurocomputing*, vol. 149, pp. 224–232, Feb. 2015.
[36] G. Huang, S. Song, J. N. D. Gupta, and C. Wu, "Semi-supervised and unsupervised extreme learning machines," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2405–2417, Dec. 2014.
[37] X. Zhao, X. Bi, and B. Qiao, "Probability based voting extreme learning machine for multiclass XML documents classification," *World Wide Web*, vol. 17, no. 5, pp. 1217–1231, Sep. 2014.

XIN BI received the M.S. and Ph.D. degrees in computer science from Northeastern University, Shenyang, in 2011 and 2016, respectively. He is currently a Lecturer with Northeastern University. His main research interests include machine learning, knowledge graph, brain networks, and XML data management.

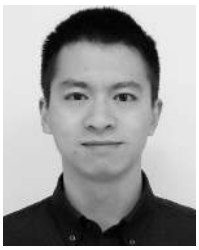
CHAO ZHANG received the bachelor's degree in the Internet of Things from Northeastern University, Shenyang, in 2019, where he is currently pursuing the master's degree. His main research interests include time series analysis and machine learning.



XIANGGUO ZHAO received the Ph.D. degree in computer science from Northeastern University, Shenyang, in 1996. He is currently an Associate Professor with Northeastern University. His main research interests include XML data management, LBSN, and machine learning.

DONGHANG LI is currently pursuing the bachelor's degree in biomedical engineering with the College of Medicine and Biomedical Information Engineering, Northeastern University, China. Her research interests include time series analysis, ECG data management, and deep learning.

YONGJIAO SUN received the B.S., M.S., and Ph.D. degrees in computer science from Northeastern University, China, in 2006, 2008, and 2012, respectively. He is currently an Associate Professor with the College of Computer Science and Engineering, Northeastern University. His research interests include probabilistic database, distributed data management, cloud database, and data privacy.



YULIANG MA received the B.S. degree in computer science from the College of Computer Science and Engineering, Northeastern University, Shenyang, in 2013, where he is currently pursuing the Ph.D. degree. His main research interests include graph databases, location-based social networks, and social network analysis.