Nearest Neighbor Ensembles: An Effective Method for Difficult Problems in Streaming Classification with Emerging New Classes

Xin-Qiang Cai[1,2], Peng Zhao[1,2], Kai-Ming Ting[3], Xin Mu[1,2], Yuan Jiang[1,2]
[1]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
[2]Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 210023, China
[3]School of Engineering and Information Technology, Federation University, Australia
Email: [1,2]{caixq, zhaop, mux, jiangy}@lamda.nju.edu.cn [3]kaiming.ting@federation.edu.au

***Abstract*—This paper re-examines existing systems in streaming classification with emerging new classes (SENC) problems, where new classes that have not been used to train a classifier may emerge in a data stream. We identify that existing systems have an unspecified assumption that emerging new classes are *geometrically far* from known classes, or instances of known classes are *densely distributed*, in the feature space. Using a class separation indicator α, we refine the SENC problem into an α-SENC problem, where α indicates a geometric distance between two classes in the feature space. We show that while most existing systems work well in high-α SENC problems (i.e., a new class is geometrically far from a known class or instances of known classes are densely distributed), they perform poorly in low-α SENC problems. To solve low-α SENC problems effectively, we propose an approach using nearest neighbor ensembles or SENNE. We demonstrate that SENNE is able to handle both the low-α and high-α SENC problems which can appear at different times in a single data stream.**

***Index Terms*—classification, data stream, new class, ensemble method, open and dynamic environments**

I. Introduction

In many real-world applications, data are often gathered over time, and thus one cannot access the whole dataset and then train the model. Meanwhile, in open and dynamic environments, there may continuously emerge instances belonging to previously unseen new classes in the data stream [1], [2], [3], [4]. Therefore, the Streaming classification with Emerging New Classes (SENC) problem has drawn much attention recently [5], [6]. In the literature, there are many studies that propose different algorithms to handle SENC problems, and some representatives are ECSMiner [5], SENCForest [6], and SENCMaS [7].

While existing approaches have shown promising results, one key weakness is that the SENC problem is assumed to have instances of new classes which are geometrically far from those of known classes, and instances of known classes are densely distributed—thus easy to detect. In this paper, we show that the SENC problem has different degrees of difficulty. If the emerging new class is significantly different from known classes, then detecting the new class would be easy. On the contrary, it would be much harder if the new class is similar to known classes (in terms of feature characteristics).

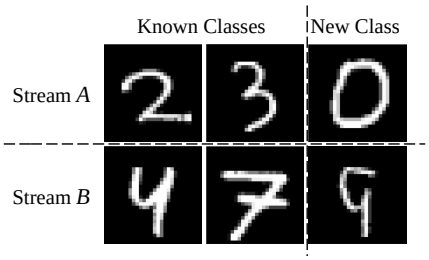

Fig. 1: Illustrative examples in detecting new emerging digits in data streams A and B using images in the MNIST dataset.

Figure 1 gives a digit identification example of two streams. For the known classes of digits 2 and 3 in stream A, detecting the emerging new class of digit 0 is easy because the feature characteristics of new class and known classes are significantly different. By contrast, for known classes of digits 4 and 7 in stream B, it is much harder to detect the emerging new class of digit 9 because there are a lot of similarities within each of the two pairs: 4-9 and 7-9.

To measure the difficulty of SENC problems, we propose a new indicator α, and refine the SENC problem into an α-SENC problem (α is defined in Section II-B). We remark that both low-α and high-α new classes may occur in the same data stream in different time periods, where low-α new classes are more similar to known classes than high-α new classes.

The emerging low-α new classes poses two challenges: First, it becomes more difficult to distinguish low-α new classes from known classes. This makes it challenging for both new class detection and known classes classification. Second, the occurrence of both low-α and high-α new classes in the same stream imposes complications in updating models along the stream that can lead to poor performance.

To address these issues, we propose to handle the α-SENC problem by using a nearest neighbor ensemble approach, or SENNE. It has dual functions: a classifier for the known classes and a detector for the new class; and it is capable of detecting both high-α new class and low-α new class which may appear in close succession in a data stream.

The main contributions are summarized as follows:

1) Introducing a *class separation indicator* α to measure the degree of difficulty in detecting an emerging new class; and leading to the α-SENC problem;
2) Proposing an effective approach to address the α-SENC problem by using nearest neighbor ensembles;
3) Conducting comprehensive experiments to validate the effectiveness of the proposed approach in both low-α and high-α SENC scenarios.

The rest of the paper is organized as follows. Section II introduces the α-SENC problem. Section III proposes our approach. Section IV provides key differences between the proposed approach and closely related methods. Sections V describes experimental settings and Section VI reports the results. Finally, Section VII concludes the paper.

II. SENC Problem and α-SENC Problem

We first review preliminary concepts in SENC [6]. Then, we refine it as the α-SENC problem.

A. SENC Problem

Definition 1 (Streaming Emerging New Classes, SENC)**.** Given the training data $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$, where $\mathbf{x}_i \in \mathbb{R}^d$ is an instance and $y_i \in \mathcal{Y} = \{1, 2, \dots K\}$ is the associated class label; data stream $S = \{(\mathbf{x}'_t, y'_t)\}_{t=1}^{\infty}$, where $\mathbf{x}'_t \in \mathbb{R}^d$ and $y'_t \in \mathcal{Y}' = \{1, 2, \dots, K, K+1, \dots, M\}$ with $M > K$. The goal is to learn an initial model f from the training data D; then f is used as a detector for emerging new class and a classifier for known classes. The model f is updated timely such that it maintains accurate predictions for known and emerging new classes on the data stream S.

Most of existing approaches for the SENC problem [5], [6] assume that the data space can be partitioned into three regions, i.e., region of known classes $\mathcal{A}$, region of anomalies of known classes $\mathcal{B}$, and region of new classes $\mathcal{Q}$. An illustration of these regions is given in Figure 2(a), where the intuition is that differences between new classes and known classes are larger than that between anomalies and known classes.

B. α-SENC Problem

Here we refine the SENC problem as the α-SENC problem, where α is an indicator reflecting the separation between known classes and a new class of the problem.

For an instance $\mathbf{x}$, let $\eta_{\mathbf{x}}$ be its nearest neighbor in the same class, and let $\tau(\mathbf{x})$ denote their distance, i.e., $\tau(\mathbf{x}) = \|\mathbf{x} - \eta_{\mathbf{x}}\|$. We define the *class separation indicator* α as follows.

Definition 2 (Class Separation Indicator)**.** The class separation indicator α of a known class dataset D_K and the new class dataset D_N is the ratio of the following two terms: $M(D_K, D_N)$: the minimum distance of the two datasets; and $C(D_K)$: the compactness of the known class dataset D_K, namely,

$$\alpha(D_K, D_N) = \frac{M(D_K, D_N)}{C(D_K)} \tag{1}$$

where $M(D_K, D_N) = \min_{\mathbf{x} \in D_K,\ \mathbf{x}' \in D_N} \|\mathbf{x} - \mathbf{x}'\|$ and $C(D) = \frac{1}{|D|} \sum_{\mathbf{x} \in D} \tau(\mathbf{x})$.

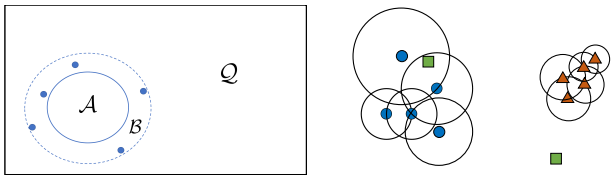

Fig. 2: The left figure shows three regions of data space: region of the known class $\mathcal{A}$, region of the anomalies of known classes $\mathcal{B}$, and region of the new classes $\mathcal{Q}$; and the right figure represents an example SENNE model with two known classes (blue circle and orange triangle) and its predictions on two instances (green rectangles) in the data stream.

The indicator $\alpha \in (0, \infty)$ essentially measures the separation between a known class and the new class: it has small value if the new class is close to the known class (i.e., $M(D_N, D_K)$ is small); or instances of the known class are sparsely distributed (i.e., $C(D_K)$ is large). Therefore, α quantifies the level of ease in detecting a new class: the smaller (larger) the value of α is, the harder (easier) the α-SENC problem will be.

Previous methods ignore scenarios that have (a) close proximity between new class and known classes (when $M(D_N, D_K)$ is small); and (b) sparsely distributed instances of known classes (when $C(D_K)$ is large). Thus, they are only suitable for α-SENC problems with a large value of α, and behave poorly in the low-α scenarios.

To handle new classes with high-α as well as low-α in a data stream, we introduce a new approach called SENNE based on nearest neighbor ensembles. Note that α is an indicator to denote the level of difficulty of the SENC problem only, and it is not used in the proposed approach.

III. The Proposed Approach: SENNE

SENNE has three major steps:

(1) Building Model: The procedure is based on nearest neighbor ensembles from a given training set. The same model is used for both new classes detection and known classes classification.
(2) Prediction in a Data Stream: Detect the new class and classify known classes according to the *new-class score* and *known-class score*. A buffer is used to temporarily store instances detected as the new class.
(3) Model Update: The current model is updated using instances in the buffer when the buffer is full.

A. Building SENNE Model

For the k-th known class dataset $D^{(k)}$ with $k = 1, \dots, K$ (where K is the number of known classes), we randomly subsample ψ instances, and repeat p times to generate subsets $\mathcal{D}_i^{(k)}$ satisfying $|\mathcal{D}_i^{(k)}| = \psi$ with $i = 1, \dots, p$.

SENNE follows the same procedure as that of iNNE [8] in building a model of K ensembles. Specifically, an *ensemble* is defined as p sets of isolation heperspheres derived from $D^{(k)}$, and each set consists of ψ isolation heperspheres. The rigor definitions are presented as follows.

Definition 3 (Isolation Hypersphere)**.** Given the k-th known class data subset $\mathcal{D}_i^{(k)}$, an isolation hypersphere $B(\mathbf{c})$ centered at $\mathbf{c}$ with radius $\tau(\mathbf{c}) = \|\mathbf{c}-\eta_{\mathbf{c}}\|$ is defined to be $\{\mathbf{x} : \|\mathbf{x}-\mathbf{c}\| \leq \tau(\mathbf{c})\}$, where $\mathbf{x} \in \mathbb{R}^d$; $\mathbf{c}, \eta_{\mathbf{c}} \in \mathcal{D}_i^{(k)}$; $\eta_{\mathbf{c}}$ is the nearest neighbor of $\mathbf{c}$ in the same class.

Definition 4 (Ensemble for Known Class)**.** Given the k-th known class dataset $D^{(k)}$, an ensemble $\mathbb{B}^{(k)}$ contains p sets, and each set consists of ψ hyperspheres. $\mathbb{B}^{(k)}$ is defined by

$$\mathbb{B}^{(k)} = \left\{\{B(\mathbf{c}) : \mathbf{c} \in \mathcal{D}_i^{(k)}\} : i = 1, \ldots, p\right\} \tag{2}$$

A SENNE model f initially consists of K ensembles after training from a dataset of K known classes. An example is shown in Figure 2(b): the SENNE model with two known classes is in a two-dimensional scenario, where $\psi = 5$ for each class. Note that the hyperspheres are large in sparse regions and they are small in dense regions—a characteristic of this model. In this example, the SENNE model will predict the top left test instance (green rectangle) as belonging to one of the known classes, i.e., the blue circle class; and the bottom right test instance is predicted as a new class instance.

Note that though the model building of SENNE and iNNE are the same, they differ in the scoring functions, which will be described later. (The differences are shown in Section IV.)

B. Prediction in Data Stream

The predictive function for each instance in a data stream has two components: one is to detect whether an instance belongs to the new class; and the other is to classify it into one of the known classes if it is not a new class instance.

We propose the following *new-class score* $N^{(k)}$ and *known-class score* $P^{(k)}$, where $k = 1, \ldots, K$.

SENNE classifies a test instance as belonging to a new class (NC) if all of its new-class scores with respect to all known classes are larger than a threshold t; otherwise, it is categorized as one of the known classes $1, \ldots, K$, namely,

$$f(\mathbf{x}) = \begin{cases} \text{NC}, & \text{if } N^{(k)}(\mathbf{x}) \geq t, \forall k = 1, \ldots, K \\ \underset{k=1,\ldots,K}{\arg\max} P^{(k)}(\mathbf{x}) & \text{otherwise} \end{cases} \tag{3}$$

Definitions of new-class score $N^{(k)}$ and known-class score $P^{(k)}$ are given as follows.

Definition 5 (New-Class Score)**.** For the k-th known class dataset $D^{(k)}$, the new-class score is defined by

$$N^{(k)}(\mathbf{x}) = \frac{1}{p}\sum_{i=1}^{p} N_i^{(k)}(\mathbf{x}), \tag{4}$$

where $N_i^{(k)}(\mathbf{x})$ is the new-class score of point $\mathbf{x}$ based on subset $\mathcal{D}_i^{(k)}$, and is defined by

$$N_i^{(k)}(\mathbf{x}) = \begin{cases} \left[1 - \frac{\tau(\eta_{cnn(\mathbf{x})})}{\|\mathbf{x}-cnn(\mathbf{x})\|}\right]_+, & \text{if } \mathbf{x} \in \bigcup_{\mathbf{c}\in\mathcal{D}_i^{(k)}} B(\mathbf{c}) \\ 1, & \text{otherwise} \end{cases} \tag{5}$$

Algorithm 1 `SENNE.Detector`

Input: Data stream $S = \{\mathbf{x}_1, \ldots\}$; number of known classes K; buffer $\mathcal{B}$ with size s; new-class score threshold t.
Output: Prediction for each new instance in the data stream.

function PREDICT($S, \mathcal{B}, K, t$)
  Build model f by the dataset with K known classes
  $\mathcal{B} \leftarrow \varnothing$ ▷ Initialize buffer
  while HasNext(S) **do**
    Calculate $N^{(k)}(\mathbf{x})$ by Eq. (4) for $k = 1, \ldots, K$
    if $N^{(k)}(\mathbf{x}) > t, \forall k = 1, \ldots, K$ **then**
      $y \leftarrow$ NC ▷ New Class
      $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathbf{x}\}$
      if $|\mathcal{B}| = s$ **then**
        $f \leftarrow$ UPDATEMODEL(K, $\mathcal{B}$, f)
        $\mathcal{B} \leftarrow \varnothing$ and $K \leftarrow K + 1$
      end if
    else
      $y \leftarrow$ PREDICTCLASS($\mathbf{x}$, K, f, t)
    end if
  end while
  return $y \in \{1, 2, \ldots, K, \text{NC}\}$
end function

Algorithm 2 `SENNE.PredictClass`

Input: Instance $\mathbf{x}$; number of known classes K; new-class score threshold t; model f.
Output: Prediction y.

function PREDICTCLASS($\mathbf{x}$, K, t, f)
  Calculate $P^{(k)}(\mathbf{x})$ for $k = 1, \ldots, K$ by Eq. (6)
  $y \leftarrow \arg\max_{k=1,\ldots,K} P^{(k)}(\mathbf{x})$
  return y
end function

where $cnn(\mathbf{x}) = \arg\min_{\mathbf{c}\in\mathcal{D}_i^{(k)}}\{\tau(\mathbf{c}) : \mathbf{x} \in B(\mathbf{c})\}$ denotes the center of the smallest hypersphere covering point $\mathbf{x}$. Besides, $[a]_+$ equals a if $a \geq 0$; and 0 otherwise.

Definition 6 (Known-Class Score)**.** The known-class score is defined as the probability of $N_i^{(k)}(\mathbf{x}) < t$:

$$P^{(k)}(\mathbf{x}) = \frac{1}{p}\sum_{i=1}^{p} \mathbb{I}[N_i^{(k)}(\mathbf{x}) < t], \text{ for } k = 1, \ldots, K, \tag{6}$$

where $\mathbb{I}[\cdot]$ is the indicator function which takes 1 if $\cdot$ is true, and 0 otherwise.

The final prediction of test instance $\mathbf{x}$ is assigned to the known class k with the highest known-class score $P^{(k)}$.

The procedures of SENNE and classification of known classes are given in Algorithms 1 and 2, respectively.

C. Model Update

We set a buffer to store the test instances that are classified as new class. When the buffer is full, we need to update the model to accommodate the new class in the buffer as a new known class. To this end, we utilize the data therein

Algorithm 3 `SENNE.UpdateModel`

Input: Number of known classes K; buffer $\mathcal{B}$ with size s; model f.
Output: Updated model f; number of known classes K.

function UPDATEMODEL(K, $\mathcal{B}$, f)
  Build $\mathbb{B}^{(K+1)}$, an ensemble of p sets of ψ hyperspheres with instances in $\mathcal{B}$, as described in Section III-A
  Label the class of instances in $\mathcal{B}$ as $K+1$
  $f \leftarrow f \cup \mathbb{B}^{(K+1)}$
  return f
end function

to construct an ensemble of p sets of hyperspheres for the new class, and then incorporate this ensemble into the existing model. After that the buffer will be emptied, and the updated model is ready for the next instance. The procedure of model update is described in Algorithm 3.

IV. Key Differences with Closely Related Work

In this section, we discuss relationships and differences with related work iNNE [8].

Although both SENNE and iNNE [8] share the same training process, they employ individual scores due to different aims. Specifically, iNNE employs the *isolation score* since it has the sole purpose of detecting anomalies. On the contrary, SENNE uses *new-class score* (cf. Eq. (5)) which is designed to differentiate emerging new classes from known classes in the SENC context. The new class detection accuracy would be low if the isolation score is naively used to detect new classes. This is because anomalies of known classes could be mistaken as emerging new classes in a data stream. For example, iNNE yields 0.723, 0.703, 0.764, 0.701, 0.744 accuracy on the Forest Cover, HAR, MNIST, Fashion-MNIST, and NYTimes datasets; but SENNE achieves 0.871, 0.804, 0.784, 0.737, and 0.768 respectively (see Section VI-B). The win/tie/loss compares with iNNE is 5/0/0. This is because the isolation score categorizes anomalies of known classes into new class (leading to false positives) in this SENC context. In contrast, the proposed new-class score is designed to reduce this kind of false positives.

A similar phenomenon occurs in SENClof (which simply employs *local outlier factor* or lof [9] in the SENC context), when lof—an anomaly detection approach—is naively extended to identify new classes. This approach tends to misclassify anomalies of known classes as belonging to new class, especially in low-α scenarios. The empirical results, shown in Section VI, validate that SENClof is not suitable for the α-SENC problem.

V. Experimental Settings

In this section, we describe experimental settings, including datasets, simulation, evaluation, and contenders. All experiments are implemented in MATLAB on Intel Core CPU machine with 24 GB memory and 3.2 GHz clock speed.

Datasets. We adopt three kinds of datasets, i.e., synthetic, benchmark, and real-world textual data stream.

For synthetic datasets, we simulate data stream with different degrees of difficulty based on various values of α. Each dataset contains three classes with 2000 two dimensional instances. For benchmark datasets, we adopt 4 datasets: Forest Cover [10], HAR [11], MNIST, and Fashion-MNIST [12]. In addition, we include a real-world data stream, which is crawled news from the website between 2014 and 2017 using the New York Times API[1], and contains 35,000 latest news items. Each news item is classified into six categories, namely, 'Arts', 'Business Day', 'Sports', 'U.S.', 'Technology', and 'World'. Each news story is converted into an 100-dim vector using the word2vec technique [13].

Data characteristics of benchmark and real-world textual datasets are summarized in the first four columns of Table I.

Stream Simulation. In the training stage, an initial training set consists of m_1 instances per known class. In the testing stage, the simulation is conducted in two periods for all datasets, except the synthetic datasets which are conducted in one period. Each period contains one new class, and each class consists of m_2 instances. For each of these instances, the model is to make a prediction, either classifying a known class or identifying the new class.

There are two setups, including the varying and fixed α-SENC scenarios, on benchmark and real-world datasets. Also we set the buffer for all approaches in the comparison with the *same* buffer size.

In the varying α-SENC scenario, as the new class is randomly selected from the whole class set, the data stream will have a mixture of high and low-α new classes. The initial dataset has two classes, except the NYTimes dataset contains five classes. For all the benchmark datasets, $m_1 = 2000$ ($m_1 = 800$ for HAR because it has a small data size), $m_2 = 500$, and the buffer size $s = 300$ are conducted. For real-world dataset NYTimes, $m_1 = 5000$, $m_2 = 5000$, and $s = 3000$ are set to simulate a long data stream.

In each of the α-SENC experiments with a fixed α, there is only one period in the whole data stream. We calculate the values of α between every two classes, and select two known classes and one new class with the lowest/highest value of α to simulate the lowest/highest-α SENC problems. The values of m_1, m_2 and s are the same as that used in the varying α-SENC experiments. We conduct 10 trials of stream simulation for each dataset, and report the average result.

Evaluation metric. We use the *accuracy* to evaluate the performance of each approach, namely, $\text{Acc} = (n_1 + n_2)/n$, where n_1 is the number of known classes instances classified correctly; n_2 denotes the number of instances belonging to the new class which are detected successfully in the data stream; and n is the total number of instances in data stream.

Contenders. We compare `SENNE` with four contenders: (a) `SENClof`, an adaptation of local outlier factor algorithm [9] from the original anomaly detection algorithm to

[1] http://developer.nytimes.com/

the SENC setting; (b) `ECSMiner` [5], an approach based on k-nearest neighbors; (c) `SENCForest` [6] is based on completely random trees; (d) `SENC-MaS` [7] approximates original information by a dynamic low-dimensional structure via matrix sketching. For compared approaches, we set default parameters as suggested in their papers or codes.

VI. Experimental Results

We present results in three subsections: (a) experiments on synthetic datasets scenario; (b) experiments on benchmark and real-world datasets; and (c) runtime comparisons.

A. Results on Synthetic Datasets

This experiment highlights performances of different approaches in α-SENC scenarios with a decreasing value of α.

Figure 3(a) shows accuracies of four approaches as α decreases. We can observe that SENNE is the most robust algorithm among all the contenders, in the sense that its accuracy does not drop much with the decrease of α. In particular, SENNE significantly outperforms other methods in the scenarios with the lowest α. ECSMiner is the second best, followed by SENClof; and SENCForest has an undesired performance. Note that since SENCMaS is designed for high-dimensional data, it is not included on 2D synthetic datasets.

The superiority of SENNE comes from the *new-class score*, which is a relative score using test point's local neighborhood information, and thus it can detect the appearance of low-α new class more effectively. On the contrary, *depth of the trees* employed by SENCForest, which does not consider test point's local neighborhood, so it fails to detect low-α new classes. Also it is invalid to apply a similar score to SENCForest due to different basic models—SENNE employs isolation hyperspheres while SENCForest uses isolation trees.

B. Results on Real-world Datasets

In this section, we present results on real-world datasets on both low-α and high-α SENC scenarios.

Table I reports performance comparisons on 5 real-world datasets. The result shows that SENNE achieves the best performance on 3 out of 5 datasets. On the other two datasets, it ranks the second and is competitive with the best one (SENCForest).

As NYTimes dataset is used as a long data stream simulation, we also plot the accuracy curves regarding time stamps of different approaches, as reported in Figure 3(b).

The results illustrate that ECSMiner behaves poorly. Meanwhile, SENCForest and SENCMaS achieve a comparably satisfying performance, and SENNE is always better than the other approaches. SENClof is a naively extended method from LOF [9] to solve SENC problems, whose performance is too poor to present since it tends to treat anomalies of known classes as new class, so we omit its results here. The empirical studies on NYTimes dataset validates the efficacy of SENNE in real-world data stream applications.

In order to examine the effect of high-α and low-α on the performances of all approaches, and to understand the reason why SENNE outperforms its contenders, we select specific known classes and new class in each dataset to simulate high-α and low-α SENC scenarios.

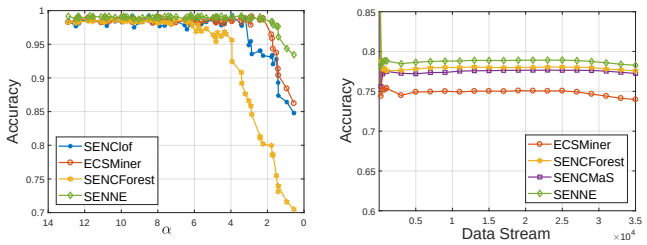

Fig. 3: The left figure reports the accuracy of four approaches as α decreases.; and the right one denotes accuracy curves on NYTimes dataset of different approaches. We omit results of SENClof, which are too poor to be presented.

Table II demonstrates that most approaches are effective in high-α SENC problems. However, when it comes to the low-α scenarios, the accuracy of all the methods decreases in different degrees. SENNE evidently has the least accuracy degradation and achieves the best performance.

The experiments using two subsets of digits $\{2, 3, 0\}$ and $\{4, 7, 9\}$ on the MNIST dataset are shown in Table II, which is used as an example of low-α and high-α SENC problems in Section I. The results validate that the class separation indicator α indeed has the ability to measure the difficulty of SENC problems, since the α value of $\{2, 3, 0\}$ is 1.01, which is larger than 0.75, the α value of $\{4, 7, 9\}$. This also accords to the intuition that detecting the new class of digit 0 is easy (when known classes are 2 and 3); while detecting digit 9 is much harder (when known classes are 4 and 7).

Note that SENNE achieves the best performance on MNIST dataset in the high-α situation. This is mainly because the highest α on the MNIST dataset is still lower than that on other datasets. This means that MNIST dataset is more difficult than others even in the high-α SENC scenario. On NYTimes dataset, the accuracies of all approaches do not decrease much regardless of the α value (SENClof is the exeption). The reason is that the lowest α value on NYTimes dataset is 1.53, much higher than those of other datasets. Thus, this dataset is not that hard even in the low-α scenario.

C. Runtime Comparison

Figure 4 reports runtime comparisons. SENClof and SENNE run faster than the other approaches, and SENNE is the only one that ran one order of magnitude faster than SENCForest on the large-scale dataset (NYTimes). We remark that SENNE is fast due to the *one-step* characteristics in the sense that it directly performs new class detection. This is in contrast with previous studies including SENCForest and SENCMaS, as they require detecting anomalies of known classes first. On the other hand, the procedures of detection and classification through isolation hyperspheres (the basic model of SENNE) are faster than that through other models. Note that the y-axis in Figure 4 is in the logarithmic scale.

TABLE I: Data characteristics and accuracy in the *varying* α-SENC problem, where new classes are randomly selected from the whole class set. • (○) indicates that SENNE is significantly better (worse) than the compared method (paired t-tests at 95% significance level).

Dataset	#class	#attribute	#instance	SENClof	ECSMiner	SENCForest	SENCMaS	SENNE
Forest Cover	7	10	581,012	0.724 ± 0.070 •	0.787 ± 0.076 •	0.791 ± 0.045 •	0.752 ± 0.059 •	**0.871 ± 0.066**
HAR	6	561	10,299	0.601 ± 0.024 •	0.654 ± 0.055 •	0.755 ± 0.043 •	0.736 ± 0.044 •	**0.804 ± 0.032**
MNIST	10	784	60,000	0.678 ± 0.019 •	0.722 ± 0.060 •	0.757 ± 0.025 •	0.757 ± 0.036 •	**0.784 ± 0.032**
Fashion-MNIST	10	784	60,000	0.710 ± 0.016 •	0.737 ± 0.047	**0.745 ± 0.035** ○	0.716 ± 0.047 •	0.737 ± 0.027
NYTimes	6	100	156,683	0.398 ± 0.023 •	0.741 ± 0.005 •	**0.772 ± 0.004**	0.767 ± 0.023	0.768 ± 0.025
SENNE `w/t/l`	–	–	–	5/ 0/ 0	4/ 1/ 0	3/ 1/ 1	4/ 1/ 0	rank first 3/ 5

TABLE II: Accuracy in *fixed* α-SENC problems. The top and bottom blocks are for high-α and low-α SENC problems on each real-world dataset, respectively. In the "classes" column, the three digits denote the selected classes, in which the first two digits represent known classes and the third one is new class. We omit Fashion-MNIST since α values of Fashion-MNIST are similar to those of MNIST.

Dataset	classes	α	SENClof	ECSMiner	SENCForest	SENCMaS	SENNE
HAR	1, 2, 6	2.10	0.769 ± 0.013 •	0.813 ± 0.017 •	**0.921 ± 0.006**	0.880 ± 0.011 •	0.919 ± 0.015
Forest Cover	3, 4, 7	19.0	0.719 ± 0.034 •	0.905 ± 0.009 •	0.946 ± 0.010	**0.952 ± 0.017**	0.947 ± 0.050
MNIST	2, 3, 0	1.01	0.805 ± 0.012 •	0.793 ± 0.010 •	0.839 ± 0.008 •	0.847 ± 0.011 •	**0.895 ± 0.010**
NYTimes	5, 3, 6	2.75	0.698 ± 0.014 •	0.912 ± 0.010	**0.936 ± 0.007** ○	0.924 ± 0.007	0.916 ± 0.018
HAR	5, 6, 4	0.89	0.563 ± 0.018 •	0.564 ± 0.024 •	0.668 ± 0.002 •	0.670 ± 0.014 •	**0.718 ± 0.014**
Forest Cover	3, 4, 6	0.58	0.689 ± 0.013 •	0.828 ± 0.010 •	0.766 ± 0.012 •	0.743 ± 0.009 •	**0.840 ± 0.013**
MNIST	4, 7, 9	0.75	0.642 ± 0.013 •	0.651 ± 0.019 •	0.625 ± 0.014 •	0.736 ± 0.016 •	**0.787 ± 0.009**
NYTimes	1, 2, 4	1.53	0.341 ± 0.009 •	0.875 ± 0.016 •	0.904 ± 0.009 •	0.903 ± 0.012 •	**0.912 ± 0.030**
SENNE `w/t/l`	–	–	8/ 0/ 0	7/ 1/ 0	5/ 2/ 1	6/ 2/ 0	rank first 5/ 8

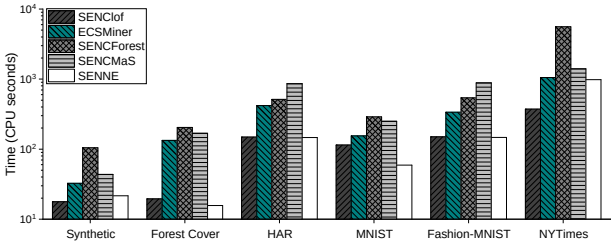

Fig. 4: The runtime comparisons.

VII. Conclusion

In this paper, we re-examine the SENC problem, namely, the streaming classification with emerging new classes problem. We identify that SENC problems have different degrees of difficulties, and thus refine it as the α-SENC problem by proposing a *class separation indicator* α to measure the difficulty. Then, we introduce a new approach called SENNE to address this problem effectively by utilizing nearest neighbor ensembles. The effectiveness and superiority of SENNE are validated on both synthetic and real-world datasets. We show that α-SENC problems with high degree of difficulty are common in real-world applications; and SENNE has comparable or higher accuracy than three existing state-of-the-art methods (SENCForest, SENCMaS, and ECSMiner), and it runs up to one order of magnitude faster.

In reality, both new class and distribution change may occur simultaneously in the data stream [14], [15], [16], and we will investigate this more challenging problem in the future work.

Acknowledgment: This research was supported by the NSFC (61673201).

References

[1] T. Tommasi, F. Orabona, and B. Caputo, "Learning categories from few examples with multi model knowledge transfer," *IEEE TPAMI*, vol. 36, no. 5, pp. 928–941, 2014.

[2] Y. Zhu, K. M. Ting, and Z.-H. Zhou, "Multi-label learning with emerging new labels," in *ICDM*, 2016, pp. 1371–1376.

[3] Q. Da, Y. Yu, and Z.-H. Zhou, "Learning with augmented class by exploiting unlabeled data," in *AAAI*, 2014, pp. 1760–1766.

[4] Y. Zhu, K. M. Ting, and Z.-H. Zhou, "Discover multiple novel labels in multi-instance multi-label learning," in *AAAI*, 2017, pp. 2977–2984.

[5] M. 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 TKDE*, vol. 23, no. 6, pp. 859–874, 2011.

[6] X. Mu, K. M. Ting, and Z.-H. Zhou, "Classification under streaming emerging new classes: A solution using completely-random trees," *IEEE TKDE*, vol. 29, no. 8, pp. 1605–1618, 2017.

[7] X. Mu, F. Zhu, J. Du, E.-P. Lim, and Z.-H. Zhou, "Streaming classification with emerging new class by class matrix sketching," in *AAAI*, 2017, pp. 2373–2379.

[8] T. R. Bandaragoda, K. M. Ting, D. Albrecht, F. T. Liu, Y. Zhu, and J. R. Wells, "Isolation-based anomaly detection using nearest-neighbor ensembles," *Computational Intelligence*, vol. 34, no. 4, pp. 968–998, 2018.

[9] M. M. Breunig, H. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *SIGMOD*, 2000, pp. 93–104.

[10] J. Gama, R. Rocha, and P. Medas, "Accurate decision trees for mining high-speed data streams," in *KDD*, 2003, pp. 523–528.

[11] C. C. Aggarwal, "A survey of stream clustering algorithms," in *Data Clustering: Algorithms and Applications*, 2013, pp. 231–258.

[12] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *CoRR*, vol. abs/1708.07747, 2017.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR*, 2013.

[14] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, pp. 44:1–44:37, 2014.

[15] Z.-H. Zhou, "Learnware: on the future of machine learning," *Frontiers Computer Science*, vol. 10, no. 4, pp. 589–590, 2016.

[16] P. Zhao, X. Wang, S. Xie, L. Guo, and Z.-H. Zhou, "Distribution-free one-pass learning," *IEEE TKDE*, 2019.