Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Streaming Classification with Emerging New Class by Class Matrix Sketching*

Xin Mu,[1,2] Feida Zhu,[3] Juan Du,[3] Ee-Peng Lim,[3] Zhi-Hua Zhou[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 Information Systems, Singapore Management University, Singapore, 178902
{mux, zhouzh}@lamda.nju.edu.cn, {fdzhu, juandu, eplim}@smu.edu.sg

Abstract

Streaming classification with emerging new class is an important problem of great research challenge and practical value. In many real applications, the task often needs to handle large matrices issues such as textual data in the bag-of-words model and large-scale image analysis. However, the methodologies and approaches adopted by the existing solutions, most of which involve massive distance calculation, have so far fallen short of successfully addressing a real-time requested task. In this paper, the proposed method dynamically maintains two low-dimensional matrix sketches to 1) detect emerging new classes; 2) classify known classes; and 3) update the model in the data stream. The update efficiency is superior to the existing methods. The empirical evaluation shows the proposed method not only receives the comparable performance but also strengthens modelling on large-scale data sets.

Introduction

The problem of classification under Streaming Emerging New Class (SENC), which aims to maintain the predictive accuracy for identifying the novel class and the known class in the stream, has recently been attracting an increasing amount of attention and effort due to both the significant research challenges (Masud et al. 2011; Parker and Khan 2015; Haque, Khan, and Baron 2016) and the immense practical value (Abdallah et al. 2016). A common example in industry is the topic categorization in the news stream, where a new topic of news may arise when a new event occurred.

There are three main challenges when we explore solutions for the SENC problem:

- The distinguishing challenge is that, under no instances from new classes observed in the training set, the high performance of three tasks including detection, classification and update needs to be ensured simultaneously. It is also different from traditional classification problem or novel class detection (or anomaly detection) because those problems are equivalent to one of the three tasks, without addressing classification or model update.
- In real-time streaming tasks, the solution should not only increase classification quality but also decrease computational cost when utilizing large-scale data. In the previous works (Haque, Khan, and Baron 2016; Al-Khateeb et al. 2012; Masud et al. 2011), data with high dimension is still a big issue when deploying in real-time application because most of them are based on distance calculation or tree structure.
- Concept drift is very common in the SENC problem, and an efficient updating strategy needs to be adopted in the systems or algorithms to overcome this issue.

The matrix sketching approach is one way to model large-scale matrix as low-dimension approximation, and has been used in many data mining tasks such as textual data mining and large-scale image analysis (Achlioptas and McSherry 2007; Metwally, Agrawal, and El Abbadi 2006). In particular, when the SENC problem meets large-scale data, we should try to explore a low-dimensional space for data analysis so that the model can be implemented on streaming within limited time and space cost.

Based on the technique Frequent-Directions (Liberty 2013), which can produce low-dimensional matrix from large-scale matrix approximations in streaming model, we propose *SENC-MaS*, a framework for Streaming classification with Emerging New Class emerging by class Matrix Sketching modelling. The main idea is to maintain a low-dimensional structure dynamically in the long data stream to approximate original information, and build the classifier and the detector by using this structure. The proposed framework contains two low-dimensional matrix sketches for approximating original global and local information. The former matrix sketching is produced on the whole data set for new class detection, whereas the latter sketching is built on each class as local information for classification. We conduct empirical studies on the benchmark data sets and a real-world topic of news data set to validate the effectiveness and efficiency of our approach.

Our main contributions are (1) the proposed framework is able to identify new class and classify known classes in stream; (2) the model update is efficient due to the low-dimension characteristic of class matrix sketching; and (3)

*This research was supported by the NSFC (61333014, 61321491); the National Research Foundation, Prime Minister's Office, Singapore under its International Research Centres in Singapore Funding Initiative and Pinnacle lab for analytics at Singapore Management University.

we conduct experiments on both simulated and real-world streams to comprehensively evaluate the performance.

The rest of this paper starts with an introduction of the related work. Then the *SENC-MaS* framework is presented, followed by the discussion section. The paper ends with empirical studies and conclusion.

Related Work

Class-incremental learning(C-IL) (Zhou and Chen 2002) is a branch of incremental learning which strengthens a previously trained classifier to deal with emerging new class. It has attracted much attention in machine learning and data mining community recent years. The problem of Streaming Classification Under Emerging New Class(SENC) (Masud et al. 2011; Mu, Ting, and Zhou 2016) is a C-IL problem in the data stream context. The ECSMiner (Masud et al. 2011) tackles the novel class detection and classification problems by introducing time constraints for delayed classification. ECSMiner assumes that true labels of new emerging class can be obtained after some time delay. Learning with Augmented Class (LAC) (Da, Yu, and Zhou 2014) is proposed for identifying emerging new classes under assuming an unlabeled dataset is available to help identify emerging new class, and the framework of LACU-SVM works in an SVM regularization style, which can assign a test instance to either one of the known classes or emerging new class. SAND (Haque, Khan, and Baron 2016) is a semi-supervised framework that uses change detection on classifier confidence to detect concept drift, and to determine chunk boundaries dynamically. These approaches achieved good performance, though they were not designed for large-scale streaming data set, as most of them are based on distance or tree structure.

Another line of study, mostly in computer vision, is zero-shot learning (Larochelle, Erhan, and Bengio 2008; Palatucci et al. 2009), which aims to recognise instances from novel categories that were not presented during training, and has been found useful in applications, e.g., detecting bots (Chen, Ranjan, and Tan 2011), face recognition (Huang et al. 2007) and video concept detection (Yang, Yan, and Hauptmann 2007). Most of these studies focus on batch-mode tasks, and requires all training data available.

In large-scale data analysis, low-rank approximations are popularly used in various tasks. Liberty (2013) introduced Frequent Directions which achieves the best bounds on the covariance error, and is further analyzed by (Ghashami and Phillips 2014). This technique has been adopted in several tasks, e.g., one-pass AUC optimization (Gao et al. 2016), streaming hashing (Leng et al. 2015), and streaming anomaly detection (Huang and Kasiviswanathan 2015). In this paper, we will extend this idea to the SENC task.

Other relevant works include novel class detection (Spinosa and de Carvalho 2004) and anomaly detection, such as by iForest (Liu, Ting, and Zhou 2008) and one-class SVM (Ma and Perkins 2003), they can identify new data that have not been previously seen during training. However, they just address subproblem in SENC, without addressing the classification and model update issues. The only way to solve the whole SENC problem is that combine with other classification framework.

Recently Zhou (2016) proposed the new concept of *learnware*, with properties of *reusability*, *evolvability* and *comprehensibility*. The evolvability emphasizes the ability of getting accustomed to environment changes, whereas the solution to SENC problem can be viewed as a preliminary step.

Preliminaries

The SENC task. Given training data set $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$, where $\boldsymbol{x}_i \in R^d$ is a training instance and $y_i \in Y = \{1, 2, \ldots, c\}$ is the associated class label, streaming data $S = \{(\boldsymbol{x}'_t, y'_t)\}_{t=1}^{\infty}$, where $\boldsymbol{x}' \in R^d$ and $y' \in Y' = \{1, 2, \ldots, c, c+1, \ldots, o\}$ with $o > c$, the goal is first to learn an initial model f with D, and $f(\boldsymbol{x}') \to Y'$. For every test instance in the data stream, f is able to determine whether it belongs to a known class. If yes, f will produce a class prediction; Otherwise the instance is placed in a buffer $\mathcal{B}$ which stores candidates of previously unseen class. When the number of candidates reaches the buffer size, they will be used to update model f. The overall aim of the task is to maintain high classification accuracy continuously in a data stream (Mu, Ting, and Zhou 2016).

Frequent Directions A sketch of the matrix A is another matrix B which is significantly smaller than A, but can approximates it well. Finding such sketches efficiently is an important building block in algorithms for approximating. In particular, the streaming algorithm urgently needs a lightweight matrix for computing. Liberty (2013) has presented a well-known streaming algorithm for approximating item frequencies to the matrix sketching setting. The algorithm receives m rows of a large matrix $A \in R^{m \times d}$ one after the other, in a streaming fashion. It maintains a sketch $B \in R^{l \times d}$ containing only $l \ll m$ rows but still guarantees that $A^T A \approx B^T B$.

The SENC-MaS Framework

Overview

Inspired by (Liberty 2013), we introduce "Class Matrix Sketching" (CMS). The main idea is to maintain two low-dimensional matrix sketches for approximating original global information and local information in the data stream, named by "*Global Sketching*" (GS) and "*Local Sketching*" (LS), respectively.

In the proposed *SENC-MaS*, the two sketches are trained on known classes before stream coming, and then they are used to identify the novel class and the known class in the stream. Ideally, we would like instances of new class to be detected as soon as they emerge in the data stream; and only instances that are likely to belong to known classes are passed to the classifier for prediction. We first introduce GS and LS as follows.

Class Matrix Sketching

Firstly, a matrix sketching which can well represented global information is built on whole known data. We call this matrix Global Sketching:

Definition 1. *Global Sketching (GS): Given training data set $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$, where $\boldsymbol{x}_i \in R^d$ is a training instance*

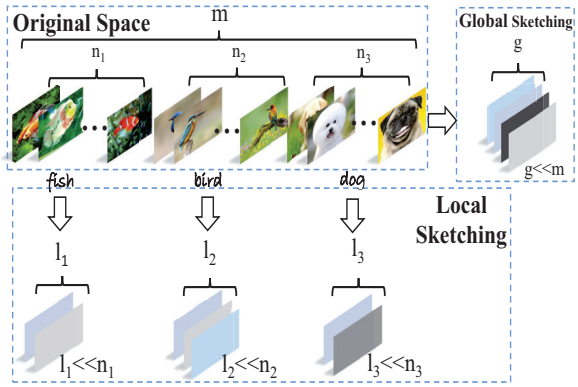


Figure 1: Building class matrix sketching

and $y_i \in Y = \{1, 2, \ldots, c\}$ is the associated class label. The Global Sketching is $G \in R^{g\times d}$, $g \ll m$, and the matrix G is significantly smaller than the whole data matrix $D \in R^{m\times d}$, but still approximates it well as $G^T G \approx D^T D$.

Based on the definition, the known class data should be more naturally approximated well by GS; on the contrary, the new class data which is unseen before may not be represented well by GS.

Secondly, multiple matrix sketches are built for discriminating the category of known class as the local information. A testing data belongs to one known class only if this class of sketching can best represent for it. We call these sketches Local Sketching:

Definition 2. *Local Sketching (LS): Given c classes training data set $\{D_1, D_2, \ldots, D_c\}$, where $D_i = \{(\boldsymbol{x}, y)|y = i\}^{n_i}$, $i \in \{1, 2, \ldots, c\}$. The Local Sketching is $\mathcal{L} = \{L_1, L_2, \cdots, L_c\}$ where $L_i \in R^{l_i\times d}$, $l_i \ll n_i$. The local sketching matrix L_i is significantly smaller than $D_i \in R^{n_i\times d}$, but approximates it well as $(L_i)^T L_i \approx (D_i)^T D_i$.*

An illustrative example to build class matrix sketching is provided in Figure 1. We assume that there are three initial classes. The global sketching is a smaller matrix with g dimension, which is built on the whole m original data sample, $g \ll m$. Similarly, local sketching is built on each class data, e.g., the 1st sketching is produced by using the 1st-class data, $l_1 \ll n_1$.

SENC-MaS

Initialization of class matrix sketching. The training procedure to build the *SENC-MaS* is detailed in Algorithms 1. The steps 1∼4 show the main process of building class matrix sketching: In line 1, an unsupervised procedure on whole data set D is implemented for building GS; in line 4, we produce multiple local sketches for known classes by considering the supervised information. The function Construct Sketching is to build a matrix sketching which is similar with building the Frequent-directions in (Liberty 2013). The function Construct Sketching shows that the algorithm keeps a low-dimensional sketch matrix that is updated every time when the sketch C has none zero-valued rows. During steps 8∼11, the sketch is first rotated (from the left) using its SVD such that its rows are orthogonal and in descending magnitude order. Then the sketch rows norms are "shrunk" so that at least half of them are set to zero. In the algorithm, Σ is a non-negative diagonal matrix $\Sigma = diag([\delta_1, \ldots, \delta_p])$, $\delta_1 \geq \ldots \geq \delta_p \geq 0$. Finally, at least half of the rows of C are all zero. Following assumption in (Liberty 2013), we assume that $p/2$ is an integer.

Algorithm 1 Initialize Class Matrix Sketching

Input: $D \in R^{m\times d}$ - input gobal data. $D_1, D_2, \ldots, D_c \in R^{n_i\times d}$ - input local data. $G \in R^{g\times d}$, $L_i \in R^{l_i\times d}$ - all zeros matrix
Output: G and $\mathcal{L} = \{L_1, L_2, \ldots, L_c\}$

1: $G \leftarrow$ Construct Sketching(D,G)
2: **for** $i = 1, \ldots, c$ **do**
3: $\quad L_i \leftarrow$ Construct Sketching(D_i, L_i)
4: **end for**
$\quad$ Construct Sketching(A, C)
$\quad$ # $A \in R^{w\times d}, C \in R^{v\times d}$
5: **for** $i = 1, \ldots, w$ **do**
6: $\quad$ Insert A_i into a zero valued row of C
7: $\quad$ **if** C has no zero valued rows **then**
8: $\quad\quad [U, \Sigma, V] \leftarrow \text{SVD}(C)$
9: $\quad\quad \delta \leftarrow \delta^2_{p/2}$
10: $\quad\quad \check{\Sigma} \leftarrow \max\sqrt{(\Sigma^2 - I_p\delta, 0)}$
11: $\quad\quad C \leftarrow \check{\Sigma}V^T$
12: $\quad$ **end if**
13: **end for**
14: **return** C

Deployment in data stream. Algorithm 2 describes the deployment of our proposed method in the data stream. Given a test instance $\boldsymbol{x}$, *C-MaS*($\boldsymbol{x}$) produces a class label $y \in \{1, \ldots, c, new\ class\}$, where $new\ class$ is the label given for an emerging new class. Note that *C-MaS*($\boldsymbol{x}$) can detect instances of any number of emerging new classes, though they are grouped into one meta-class.

In line 3 of Algorithm 2, the function *C-MaS* is defined as follows:

$$\textit{C-MaS}(\boldsymbol{x}) = \begin{cases} new\ class, \text{if } \psi(\boldsymbol{x}) < \textit{threshold} & (1) \\ j,\ j = \max\limits_j \phi_j(\boldsymbol{x}), \text{ otherwise} & (2) \end{cases}$$

where $j \in \{1, 2, \ldots, c\}$, $\psi(\cdot)$ is the function for new class detection, $\phi(\cdot)$ is the function for classification on known classes, a *threshold* is introduced to determine if new class is emerging. The detail is as follows:

- New class detection. Let $[\cdot]_{i,:}$ be the vector at i_{th} row in a matrix. The function $\psi()$ is defined as:

$$\psi(\boldsymbol{x}) = \max\langle \boldsymbol{x}, [G]_{i,:}\rangle,\ \forall i\ \ i \in \{1, 2, ..., g\} \qquad (3)$$

where $\langle\cdot\rangle$ is the inner product of two vectors, $[G]_{i,:}$ is i_{th} row in matrix G. This is a very straightforward idea that we employ inner product as building relationship between a test data and the sketching space. The inner product can be defined either algebraically or geometrically. Algebraically, it is the sum of the products of the corresponding entries of the two sequences of numbers. Geometrically, it is positive

for acute angles and negative for obtuse angles. Therefore, the larger inner product means that the two vectors are more similar. In function $\psi(\cdot)$, the group of inner product is calculated between testing instance $\boldsymbol{x}$ and every row of G, and then the maximum value of this group is used to illustrate how similar the between $\boldsymbol{x}$ and G is.

- Threshold determination. We employ a *threshold* to measure new class emerging in (1). Different from previous manual setup, it is determined automatically. The intuition is that new class has the different property from known classes, e.g., it has been assumed that instances of any emerging new class are far from the known classes in the data space (Masud et al. 2011; Haque, Khan, and Baron 2016). In this paper, we assume that the inner product between new class and GS should be smaller than between known classes and global sketching. Thus, the definition of *threshold* is as follows:

$$\textit{threshold} = \arg\min\ Q \quad (4) \quad \text{or} \quad \tfrac{1}{m}\textstyle\sum_{k=1}^{m} Q \quad (5)$$

where $Q = \{\psi(\boldsymbol{x}_1), \psi(\boldsymbol{x}_2), \ldots, \psi(\boldsymbol{x}_m)\}$. We produce a list Q by Eqn.(3) on every training instacne $\boldsymbol{x}$. Ideally, we assume that there is none outlier in training set, the minimum value of Q such as Eqn.(4) can be used to distinguish the new class from the known classes, where the former should have smaller value than the latter according to our assumption. However, outlier usually exists in common practise, and thus, the minimum value of Q does not always give the better representation on the known property. Therefore, the average of Q in Eqn.(5) is used for the threshold which would be more robust. In this paper, we use the Eqn.(5) for *threshold*, which is also recommended to use in practise.

if a testing instance leads to a smaller value than this threshold, it is determined as new class. Otherwise, it will be passed to the classifier in Eqn.(2).

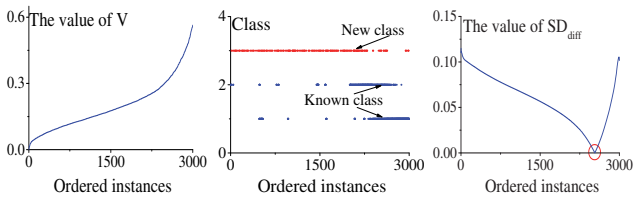
Note that using of the Eqn.(5) will produce the *candidate of new class* in the buffer including novel classes and anomaly of known classes. For handling the problem of determining new class, a simple yet effective solution called SD_{diff} (Mu, Ting, and Zhou 2016) curve for separating these two types of instances is used when buffer is full in our framework. The main idea is to take advantage of the fact that, anomaly instances are more similar to the normal instances than instances from emerging new classes . Therefore, let V be a ascending order list of the result of Eqn.(3) on data in the buffer. The new class instances are expected to have smaller value in list V than the anomalies, the split point $\hat{\tau}$ in this list would yield two sub-lists V^{left} and V^{right}, where the former is regarded as new class with small value and the latter is regarded as the anomalies with the large value. The best split point is used to separate the novel class part from the anomaly part. We use the criterion which minimises the difference in standard deviations $\sigma(\cdot)$: $\hat{\tau} = \arg\min_{\tau}\ |\sigma(V^{left}) - \sigma(V^{right})|$. The $\hat{\tau}$ is the best point to separate new class and known class.

Figure 2 provides an illustration on this strategy. Firstly, we assume that 3000 candidates of new class are in the buffer $\mathcal{B}$, and a list V(ascending order) according to these candidates can be obtained as mentioned before. The X-axis of all three figures is the ordered 3000 instances with respect to the value of V. Figure 2(b) shows the true label distribution of those ordered instances. The red is the new class and the blue is the known class. We can see that two kinds of the known classes mostly distribute on the right side, that means they have large value in list G. On the contrary, the new class with small value in list V distributes mostly on the left side; Figure 2(c) shows the SD_{diff} curve calculated by formula mentioned before. As can be seen from Figure 2(c), the point marked which yields the minimum SD_{diff} can be chosen as the threshold to separate the novel class and the known classes in the buffer.

Algorithm 2 Deploying *SENC-MaS* in data stream

Input: G, L_i, $\mathcal{B}$ - empty buffer of size s
Output: y - class label for each $\boldsymbol{x}$ in a data stream

```
1:  while not end of data stream do
2:    for each x do
3:      y ← C-MaS(x)   # using (1) and (2)
4:      if y = new class then
5:        B ← B ∪ {x}
6:        if |B| ≥ s then
7:          Update (G, L, B)
8:          B ← NULL
9:        end if
10:     end if
11:     Output y ∈ {1, ..., c, new class}
12:   end for
13: end while
```



(a) V curve. (b) Class distribution. (c) SD_{diff} curve.

Figure 2: An illustration on determining the new class and the known classes in the buffer.

- Classification on known classes. We calculate the inner product between a testing instance and every local matrix sketching L_i. We define:

$$\phi_i(\boldsymbol{x}) = \max\langle \boldsymbol{x}, [L_i]_{j,:}\rangle,\ \forall i, j \\ i \in \{1, 2, ..., c\},\ j \in \{1, 2, ..., n_j\} \quad (6)$$

Finally, Eqn.(2) produces the index of maximum value of the list $\{\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \cdots, \phi_c(\boldsymbol{x})\}$ as class number.

If *C-MaS*($\boldsymbol{x}$) outputs *new class*, $\boldsymbol{x}$ is placed in buffer $\mathcal{B}$ which stores the candidates of the previously unseen class (line 5). When the number of candidates has reached the buffer size s, the candidates are used to update both the global and local sketching (line 7). Once these updates are completed, the buffer is reset and the new model is ready for the next test instance in the data stream.

Update sketching. The update process is described in Algorithm 3. Before starting, similar to (Haque, Khan, and

Algorithm 3 Update

Input: G, $\mathcal{L}$ - existing sketching, $\mathcal{B}$ - input data
Output: the new G, $\mathcal{L}$, *threshold'*

1: **initialize:** All instances in $\mathcal{B}$ are assigned the true class $Y_{\mathcal{B}}$, $\mathcal{B}'$ is a sub set of $\mathcal{B}$ including k new class instances, Q' is a list like in Eqn.(5) with respect to $\mathcal{B}'$. $Y'_{\mathcal{B}}$ is a set of r new class types in $Y_{\mathcal{B}}$, n - the number of instances is used to compute the old *threshold*
2: $G \leftarrow$ Construct Sketching($\mathcal{B}'$,G)
3: *threshold'* $\leftarrow \frac{\textit{threshold} \times n + \sum^{k} Q'}{n+k}$
4: **for** $i = 1, \ldots, r$ **do**
5: $L_{new} \leftarrow$ Construct Sketching ($\{(x, y)|x \in \mathcal{B}, y = Y'_{\mathcal{B}}(i)\}$,$L_{new}$) # L_{new} - all zeros matrix
6: $\mathcal{L} \leftarrow \mathcal{L} \cup L_{new}$
7: update the number of known class c
8: **end for**

Baron 2016; Mu, Ting, and Zhou 2016), we assume all instances in $\mathcal{B}$ are assigned the true class information.

For global sketching G, it is still an unsupervised procedure by using function Construct Sketching in line 2. This process leads to the global sketching with new class information. For local matrix sketching $\mathcal{L}$, a newly grown sketching L_{new} for new class is produced to update $\mathcal{L}$ (in line 4∼8). Threshold will be upgraded by using the equation in line 3. Note that if one new class is with small amount of instances, it will not be used for updating temporarily and stay in buffer to wait for more instances.

Discussion

Concept drift. As mentioned before, there are some known classes instances in the buffer, they are not used for updating model in algorithm 3. From another perspective, it is possible that the appearance of these known classes instances is due to concept drift. We can transfer them to another memory and wait for more instances, then existing concept drift solutions can be adopted to detect concept drift for each known class. If the concept drift occurs, the relevant sketching will be updated by using function Construct Sketching.

The error of approximation and complexity. In (Liberty 2013), the bound of difference between original matrix $A \in R^{m \times d}$ and sketching $B \in R^{l \times d}$ is given as: $\forall x, ||x|| = 1, 0 \leq ||Ax||^2 - ||Bx||^2 \leq 2||A||_f^2/l$. Or $B^T B \prec A^T A$ and $||A^T A - B^T B|| \leq 2||A||_f^2/l$. Furthermore, Ghashami et. al. (2015) provided more detail of theoretical analysis of this technique. In this paper, our core technique is extension of Frequent-Direction method so that the global sketching matrix and the local sketching matrix also can be offered theoretical bound.

In training and updating procedure, the main time consumption is on the SVD. A basic matrix sketching runs in $O(mdl)$ time to produce a sketch of size $l \times d$ from m instances. The total running time in our framework is therefore bounded by $O(mdg + \sum n_i d l_i)$. An update efficiency analysis will be demonstrated in experimental section.

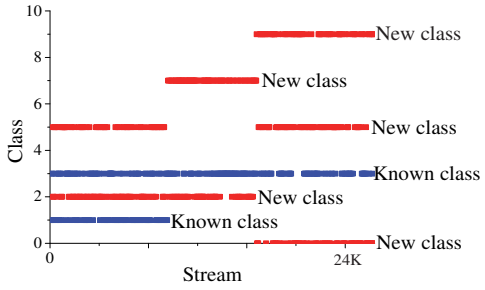


Figure 3: The class distribution of one simulated stream on MNIST data set. The X-axis is data streaming and the Y-axis is class information.

Experiment

Experimental setup

Data sets. Three benchmark data sets are used to assess the performance of all methods, including KDDCup99[1], Forest Cover[2], MNIST[3]. In particular, we use the 10 percent version of the KDDCup99 data set, and it contains 12 largest classes. An initial training set with two known classes is available to train the model, We simulate a data stream on each benchmark data set including both instances of the two known classes and multiple new classes, note that those new classes appear in the different periods in this simulated data stream with uniform distribution. When buffer is full, model will be updated immediately. After updating, model can continuously deal with another new classes in the streaming. The experiments on each data set are repeated for 10 times with different simulated streams and both the mean and the standard variance of the performance are reported.

In addition, a real news summary stream is used to evaluate performance. it is crawled over a period of time by using the New York Times API[4]. There are 10k news items categorised into 8 classes. The most common 5 classes, i.e., "World", "U.S.", "Sport", "Business Day" and "Arts", are used as the initial known classes to train the initial model. The class "N.Y./Region", "Fashion & Style" and "Technology" are regarded as new classes which occur in the streaming. Each item is preprocessed using the "word2vec" technique[5] to produce a 1000-dimension feature vector. In order to illustrate distribution of experimental stream clearly, Two examples will be presented in Figure 3 and Figure 4.

Competing algorithms. We compare with: **iForest + KNN**: iForest (Liu, Ting, and Zhou 2008) is an unsupervised anomaly detector which can be treated as new class detector by isolating new class data. We combine it with KNN as classifier; **LACU-SVM** (Da, Yu, and Zhou 2014): this method trains c binary classifiers $f_c(\cdot)$ for each known class. LACU-SVM makes a prediction for the known class

[1] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[2] https://kdd.ics.uci.edu/databases/covertype/covertype.data.html
[3] http://cis.jhu.edu/ sachin/digit/digit.html
[4] http://developer.nytimes.com/
[5] https://radimrehurek.com/gensim/index.html

Table 1: Comparisons of different methods on simulated streams.

	KDD Cup 99		Forest Cover		MNIST	
Algorithm	Accuracy	F-measure	Accuracy	F-measure	Accuracy	F-measure
iForest+KNN	0.867±0.01	0.882±0.04	0.707± 0.02	0.742± 0.08	0.720 ± 0.06	0.593 ± 0.05
LACU-SVM	0.853±0.05	0.827± 0.02	0.702± 0.07	0.752± 0.11	0.752 ± 0.01	0.697 ± 0.04
SAND-F	0.880±0.03	0.892± 0.03	0.799± 0.02	0.771± 0.03	0.725 ± 0.04	0.622 ± 0.03
ECSMiner	0.857± 0.03	0.852±0.06	**0.823± 0.05**	**0.794± 0.02**	0.745 ± 0.07	0.641 ± 0.03
SENC-non-MaS	0.792± 0.08	0.722±0.13	0.633± 0.02	0.651± 0.05	0.701 ± 0.07	0.652 ± 0.06
SENC-MaS	**0.897± 0.03**	**0.902±0.02**	0.792± 0.04	0.752± 0.07	**0.817 ± 0.05**	**0.710 ± 0.09**

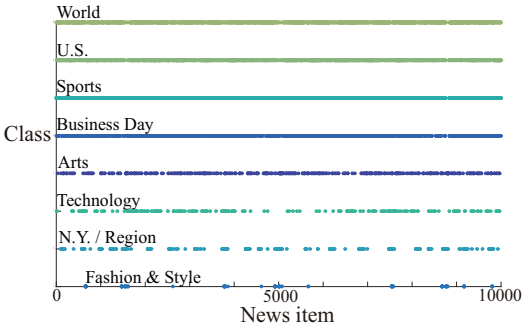


Figure 4: New York Times news stream. The X-axis is data streaming and the Y-axis is class information of each item.

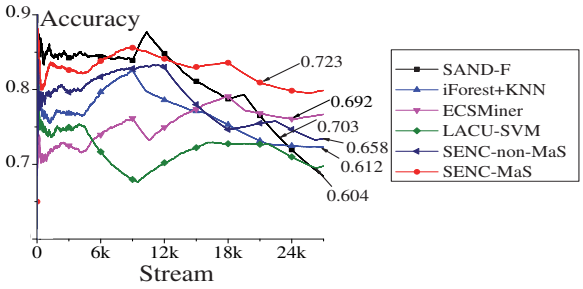


Figure 5: An illustration result on the MNIST data set. Average F-measure for each method is marked at the end of each line.

if $\max_k f_k(x) > 0$; otherwise x is predicted as belonging to the emerging new class; **SAND-F** (Haque, Khan, and Baron 2016): This is a framework that maintains an ensemble of clustering-based classifier models, each trained on a different dynamically determined partially labeled chunk of data; **ECSMiner** (Masud et al. 2011): ECSMiner is an ensemble framework for SENC problem, and a new measure is defined to decide whether they are emerging new classes. K nearest neighbor is used as the classifier to make predictions for instances of known classes; **SENC-non-MaS**: our proposed framework works on original space without building any sketches, KNN is as classifier.

Algorithms settings and evaluation metrics. All methods are executed in the MATLAB environment. The following implementations are used: LACU-SVM and iForest were the codes as released by the corresponding authors; The ECSMiner and SAND-F code are completed based on the authors' paper (Masud et al. 2011; Haque, Khan, and Baron 2016). Number of trees in iForest is set to 50 and $\psi = 200$. Parameters in LACU-SVM are set by $ramp_s = -0.3, \eta = 1.3, \lambda = 0.1, max\_iter = 10$ according to authors paper. ECSMiner employs K-means and K is set to 5. In SAND-F, ensemble size t is set to 6, $q = 50$ and τ= 0.4. In *SENC-MaS*, the buffer of size $s = 3000$, $L = N * 0.8, l_i = n_i * 0.8$. The data size of the initial training set D is 2000 per class.

Two measurements are used in this paper. One is *Accuracy*, $Accuracy = \frac{A_{new}+A_{known}}{m}$, where m is the total number of instances, A_{new} is the number of emerging class instances identified correctly, A_{known} is number of known class instances classified correctly; the other is the *F-measure* for new class detection, which is calculated when buffer is full.

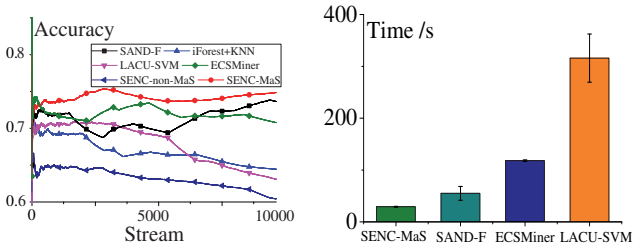
Results

Simulated stream. The results of simulated stream are shown in Figure 5 and Table 1. Figure 5 shows one simulated stream result on the MNIST data set. Note that different methods have different update points, and the marked *F-measure* is average of all results in the stream. We will run ten independent experiments with different simulated streams on each data set and both the mean and the standard variance of the performance are reported in Table 1.

SENC-MaS maintains the good performance in data streams with new classes continually emerging. The closest contenders are ECSMiner and SAND-F, both of them employ an ensemble framework. While ECSMiner needs to provide with all true labels in order to train a new classifier; and accuracy of SAND-F depends on base classifier. It is difficult to tune appropriate parameters for LACU-SVM. iForest + KNN using two frameworks still performed worse. SENC-non-MaS is different type of our method, its result also shows our proposed algorithm efficient.

Real-world stream. Figure 6(a) shows *SENC-MaS* performs better than other methods. The performance of SAND-F and ECSMiner are unstable in the stream as demonstrated in Figure 6(a). *SENC-non-MaS* results in the worst performance although it is the simplest. The remaining two methods hinge heavily upon parameter settings.

Update efficiency analysis. Four algorithms with better performance are chosen to analyze update efficiency on real-world stream, and Figure 6(b) shows the average of whole update time in the stream. There is no doubt that *SENC-MaS*



(a) Accuracy result. (b) Updating time analysis.

Figure 6: Result on New York Times news stream.

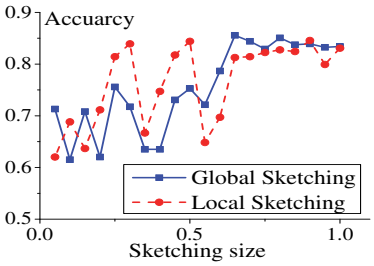


Figure 7: The influence of size of sketching.

achieves faster update time, and it should be the best choice for a real-time task.

Parameter analysis. We study the influence of two parameters in *SENC-MaS* including the size of sketch g and l_i. The initial conditions are same as before. We test performance accuracy with different sketch size of GS and LS on MNIST data set. When testing one parameter, another is fixed. As can be seen from Figure 7, we can use greater than 50 percent of original data size to guide the setup of sketching size in practice. Note that similar results can be concluded on other data sets.

Conclusion

This paper tackles the SENC problem. We propose *SENC-MaS* which uses two low-dimensional matrix sketches for detecting new class and classifying known classes. These two sketches are continuously updated in the stream. Empirical evaluation shows that *SENC-MaS* outperforms existing methods. Future work includes improving further the efficiency and scalability, and exploring infinite stream and concept drift. Theoretical guarantee is a future issue.

Acknowledgement: We would like to thank anonymous reviewers for comments and suggestions, and Chenghao Liu, Mina Ghashami for discussions.

References

Abdallah, Z. S.; Gaber, M. M.; Srinivasan, B.; and Krishnaswamy, S. 2016. Anynovel: detection of novel concepts in evolving data streams. *Evolving Systems* 7(2):73–93.

Achlioptas, D., and McSherry, F. 2007. Fast computation of low-rank matrix approximations. *JACM* 54(2):9.

Al-Khateeb, T.; Masud, M. M.; Khan, L.; Aggarwal, C.; Han, J.; and Thuraisingham, B. 2012. Stream classification with recurring and novel class detection using class-based ensemble. In *ICDM*, 31–40.

Chen, F.; Ranjan, S.; and Tan, P. 2011. Detecting bots via incremental LS-SVM learning with dynamic feature adaptation. In *SIGKDD*, 386–394.

Da, Q.; Yu, Y.; and Zhou, Z.-H. 2014. Learning with augmented class by exploiting unlabeled data. In *AAAI*, 1760–1766.

Gao, W.; Wang, L.; Jin, R.; Zhu, S.; and Zhou, Z.-H. 2016. One-pass AUC optimization. *Artificial Intelligence* 236:1–29.

Ghashami, M., and Phillips, J. M. 2014. Relative errors for deterministic low-rank matrix approximations. In *SDM*, 707–717.

Ghashami, M.; Liberty, E.; Phillips, J. M.; and Woodruff, D. P. 2015. Frequent directions: Simple and deterministic matrix sketching. *CoRR* abs/1501.01711.

Haque, A.; Khan, L.; and Baron, M. 2016. Sand: Semi-supervised adaptive novel class detection and classification over data stream. In *AAAI*, 1652–1658.

Huang, H., and Kasiviswanathan, S. P. 2015. Streaming anomaly detection using randomized matrix sketching. *Proceedings of the VLDB Endowment* 9(3):192–203.

Huang, C.; Ai, H.; Yamashita, T.; Lao, S.; and Kawade, M. 2007. Incremental learning of boosted face detector. In *ICCV*, 1–8.

Larochelle, H.; Erhan, D.; and Bengio, Y. 2008. Zero-data learning of new tasks. In *AAAI*, 646–651.

Leng, C.; Wu, J.; Cheng, J.; Bai, X.; and Lu, H. 2015. Online sketching hashing. In *CVPR*, 2503–2511.

Liberty, E. 2013. Simple and deterministic matrix sketching. In *SIGKDD*, 581–588.

Liu, F. T.; Ting, K. M.; and Zhou, Z.-H. 2008. Isolation forest. In *ICDM*, 413–422.

Ma, J., and Perkins, S. 2003. Time-series novelty detection using one-class support vector machines. In *IJCNN*, 1741–1745.

Masud, M.; Gao, J.; Khan, L.; Han, J.; and Thuraisingham, B. M. 2011. Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE TKDE* 23(6):859–874.

Metwally, A.; Agrawal, D.; and El Abbadi, A. 2006. An integrated efficient solution for computing frequent and top-*k* elements in data streams. *ACM TODS* 31(3):1095–1133.

Mu, X.; Ting, K. M.; and Zhou, Z.-H. 2016. Classification under streaming emerging new classes: A solution using completely random trees. *CoRR* abs/1605.09131.

Palatucci, M.; Pomerleau, D.; Hinton, G. E.; and Mitchell, T. M. 2009. Zero-shot learning with semantic output codes. In *NIPS*, 1410–1418.

Parker, B. S., and Khan, L. 2015. Detecting and tracking concept class drift and emergence in non-stationary fast data streams. In *AAAI*, 2908–2913.

Spinosa, E. J., and de Carvalho, A. C. P. L. F. 2004. SVMs for novel class detection in bioinformatics. In *III Brazilian Workshop on Bioinformatics*, 81–88.

Yang, J.; Yan, R.; and Hauptmann, A. G. 2007. Cross-domain video concept detection using adaptive svms. In *ACM Multimedia*, 188–197.

Zhou, Z.-H., and Chen, Z.-Q. 2002. Hybrid decision tree. *Knowledge Based Systems* 15(8):515–528.

Zhou, Z.-H. 2016. Learnware: On the future of machine learning. *Frontiers of Computer Science* 10(4):355–384.