New Class Adaptation via Instance Generation in One-Pass Class Incremental Learning

Yue Zhu[1], Kai-Ming Ting[2], Zhi-Hua Zhou[1]
[1]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
[2]School of Engineering and Information Technology, Federation University, Australia
Email: [1]{zhuy, zhouzh}@lamda.nju.edu.cn [2]kaiming.ting@federation.edu.au

***Abstract*—One pass learning updates a model with only a single scan of the dataset, without storing historical data. Previous studies focus on classification tasks with a fixed class set, and will perform poorly in an open dynamic environment when new classes emerge in a data stream. The performance degrades because the classifier needs to receive a sufficient number of instances from new classes to establish a good model. This can take a long period of time. In order to reduce this period to deal with any-time prediction task, we introduce a framework to handle emerging new classes called *One-Pass Class Incremental Learning* (`OPCIL`). The central issue in `OPCIL` is: how to effectively adapt a classifier of existing classes to incorporate emerging new classes. We call it the *new class adaptation* issue, and propose a new approach to address it, which requires only one new class instance. The key is to generate pseudo instances which are optimized to satisfy properties that produce a good discriminative classifier. We provide the necessary properties and optimization procedures required to address this issue. Experiments validate the effectiveness of this approach.**

I. Introduction

In an open dynamic environment, new instances arrive successively, where new classes may emerge at any time. Class incremental learning [2], [9], [12] is a learning paradigm designed for this kind of dynamic environment where classes appear incrementally. It focuses on enabling a learning system to incorporate instances of previously unseen classes in a continuous training process as new instances emerge. Most existing studies on class incremental learning require multiple scans of a training set. In real applications with limited resources (e.g., storage, energy consumption, etc. in mobile applications), it is imperative that a satisfying performance can be produced with a minimum number of scans. Therefore, an ideal solution is one that each instance needs to be scanned only once, and it is discarded and not stored for future training—one-pass learning.

In contrast to the traditional batch learning paradigm which generates a model based on the entire training data, one-pass learning [6], [17] deals with one instance at each time step without accessing any historical data. This enables an efficient strategy to build a large-scale learning system that would otherwise be computationally infeasible using the entire dataset. However, existing one-pass learning methods [1], [4], [6], [7], [14], [17] assume that the class set is fixed; thus they cannot handle class incremental learning in a data stream.

In this work, we extend both one-pass learning and class incremental learning to create *One-Pass Class Incremental Learning (*`OPCIL`*)*. The new learning framework enables any-time predictions as new classes emerge in data streams. No existing methods in one-pass learning or class incremental learning are able to perform effective any-time predictions under the stated scenario, as far as we know.

A naive extension of one-pass learning to address the above-mentioned issue is to enlarge the model for the new class with random initialization. This method will degrade the accuracy substantially every time a new class emerges, and can stay low for an extended period of time until a sufficient number of instances from the new class are received. As a result, in order to produce a robust learning system, dealing with any-time prediction tasks, the central issue in `OPCIL` is to adapt the previously updated classifier for existing classes to incorporate emerging new classes under a stringent condition that all historical data are unavailable at all times. We call it the new class adaptation issue.

In this paper, we propose an effective method to the new class adaptation issue in `OPCIL` to maintain a robust learning system for any-time predictions. Having only one instance when a new class first appears, the key is to generate pseudo instances which are optimized to satisfy properties that produce a good performing discriminative classifier.

The contribution of this work is summarized as follows:

1) Introducing a new type of learning called one-pass class incremental learning (`OPCIL`);
2) Identifying the central issue in `OPCIL`, i.e., how to effectively adapt an existing classifier to incorporate a new class when the new class first appears in a stream.
3) Proposing an effective method to address the new class adaptation issue by generating pseudo instances which are optimized to satisfy properties that produce a good performing discriminative classifier using one observed instance only.

There are several other lines of related works. Learning with emerging new classes [11], [18] focuses on discovering emerging novel classes in a data stream; one-class learning [8], [13], [15] aims to build a classification model from a training set which mainly consists of positive class only; one-shot learning [5], [10] focuses on learning from very few training examples. All these works can handle new classes, but they have to access historical data several passes. In contrast, our work focuses on *new class adaptation* problem in one-pass learning setting, in which the classifier is updated immediately

```
Algorithm 1: OPCIL
Input: Data stream: ${\boldsymbol{x}_t, \boldsymbol{y}_t}_{t=1}^T$; number of existing classes at
       time step: $t = 0$ - $K_0$
Output: Weight $\tilde{W}_t$ for prediction
Initialize: $\mathcal{Y}_0 = \{1, \cdots, K_0\}$, $\tilde{W}_0$ with random initialization,
       $t = 1$;
1  repeat
2      Receive a labeled instance $(\boldsymbol{x}_t, y_t)$;
3      Update center $C$ for each class;
4      $\tilde{\boldsymbol{x}}_t \leftarrow [\boldsymbol{x}_t; 1]$;
5      if $y_t \notin \mathcal{Y}_{t-1}$ then            // $y_t$ is a new class
6          $K_t \leftarrow K_{t-1} + 1$;
7          $\mathcal{Y}_t \leftarrow \mathcal{Y}_{t-1} \cup \{K_t\}$;
8          $\tilde{W} \leftarrow$ GPI$(\tilde{W}_{t-1}, C)$;
9      else        // $y_t$ is an existing known class
10         $K_t \leftarrow K_{t-1}$;
11         $\mathcal{Y}_t \leftarrow \mathcal{Y}_{t-1}$;
12         $\tilde{W} \leftarrow \tilde{W}_{t-1}$;
13     $\tilde{W}_t \leftarrow$ SGDUpdate$(\tilde{\boldsymbol{x}}_t, y_t, \tilde{W})$ [1] ;
14     $t \leftarrow t + 1$;
15 until $t = T$;
```

after a training instance is observed in the data stream and no historical data are accessible.

The rest of this paper is organized as follows: Section II describes the OPCIL framework and the new class adaptation issue; Section III provides the proposed method; experimental results are presented in Sections IV and V; and the conclusions are provided in the last section.

II. One-Pass Class Incremental Learning

A. Framework

Let $\mathcal{X} = \mathbb{R}^d$ be the instance space, and $\mathcal{Y}_t = \{1, 2, \cdots, K_t\}$ be the class set, where K_t is the number of classes at time t.

In the context of one-pass class incremental learning, we are given a sequence $S_T = \{(\boldsymbol{x}_t, y_t)\}_{t=1}^T$, where $\boldsymbol{x}_t \in \mathcal{X}$ is the instance observed at time t; and $y_t \in \mathcal{Y}_t$ is the class label of $\boldsymbol{x}_t$. If $y_t \notin \mathcal{Y}_{t-1}$, then $\boldsymbol{x}_t$ belongs to a new class $\boldsymbol{y}_t = K_t = K_{t-1} + 1$, and the class set will be augmented with K_t: $\mathcal{Y}_t = \{1, 2, \cdots, K_{t-1}, K_t\}$; otherwise, $K_t = K_{t-1}$.

The goal for one-pass class incremental learning is to learn a mapping $f_t : \mathcal{X} \to \mathcal{Y}_t$, so as to minimize the loss over the sequence S_T: $\sum_{t=1}^T \mathbb{I}(f_t(\boldsymbol{x}_t) \neq y_t)$, where $\mathbb{I}(\cdot)$ is an indicator function which returns 1 when the argument is true, and returns 0 otherwise; and f_t predicts the labels of unseen instances.

Because 0-1 loss is hard to optimize, the softmax transformation of a linear function for an instance $\boldsymbol{x}_t$ is considered instead in practice:

$$g_k(\boldsymbol{x}_t, W_t, \boldsymbol{b}_t) = \frac{\exp(\boldsymbol{w}_{t,k}^\top \boldsymbol{x}_t + b_{t,k})}{\sum_{j=1}^{K_t} \exp(\boldsymbol{w}_{t,j}^\top \boldsymbol{x}_t + b_{t,j})}, k \in \{1, \cdots, K_t\}, \quad (1)$$

where $W_t = [\boldsymbol{w}_{t,1}, \cdots, \boldsymbol{w}_{t,K_t}] \in \mathbb{R}^{d \times K_t}$ is the linear weight matrix, $\boldsymbol{b}_t = [b_1, \cdots, b_{K_t}]^\top$ is the bias vector.

The learning becomes minimizing the following log loss function as a surrogate:

$$\ell(\boldsymbol{x}_t, y_t, W_t, \boldsymbol{b}_t) = -\sum_{k=1}^{K_t} \mathbb{I}(y_t = k) \ln g_k(\boldsymbol{x}_t, W_t, \boldsymbol{b}_t);$$

and the prediction function is

$$f(\boldsymbol{x}_t, W_t, \boldsymbol{b}_t) = \arg\max_k g_k(\boldsymbol{x}_t, W_t, \boldsymbol{b}_t), k \in \{1, \cdots, K_t\};$$

In order to simplify both the optimization and the description, we add an additional dimension to each instance $\tilde{\boldsymbol{x}}_t = [\boldsymbol{x}_t; 1]$, and let $\tilde{W}_t = [W_t; \boldsymbol{b}_t^\top]$. Then, the bias term in g_k, f and ℓ can be dropped by applying $\tilde{\boldsymbol{x}}_t$ and $\tilde{W}_t$.

The optimization is typically conducted using stochastic gradient descent (SGD) [2]. The gradient of $\ell(\tilde{\boldsymbol{x}}_t, y_t, \tilde{W}_t)$ w.r.t. $\tilde{W}_t$ is given by $\nabla_{\tilde{W}_t} = -\tilde{\boldsymbol{x}}_t(\boldsymbol{e}_{y_t} \circ (\mathbf{1} - \boldsymbol{g}))^\top$, where $\boldsymbol{e}_{y_t}$ is the unit vector whose y_t-th element is 1, the other elements are 0; "$\circ$" is the element-wise product; $\mathbf{1}$ is an all-one vector; and $\boldsymbol{g} = [g_1(\tilde{\boldsymbol{x}}_t, \tilde{W}_t), \cdots, g_k(\tilde{\boldsymbol{x}}_t, \tilde{W}_t)]^\top$ [3].

The framework of One-Pass Class Incremental Learning (OPCIL) is summarized in Algorithm 1.

B. New Class Adaptation

Under the constraint that all historical data are unavailable at all times, the central issue in OPCIL is defined as follows:

Definition 1. *New class adaptation in OPCIL aims to adapt a classifier of known classes to a good performing classifier for both the known classes and the new class, using only one instance of the new class.*

Because there is no access to historical data and there is only a single instance from the emerging new class, it appears to be a mission impossible to upgrade to a good performing model for all classes. Fortunately, two types of information are available:

1) Centers of existing classes. In one-pass learning setting, each class center maintains the first-moment information of historical data for each class; and it is easy to update as new instances are observed. Let $C_t = [\boldsymbol{c}_{t,1}, \cdots, \boldsymbol{c}_{t,K_t}]$ be the center vector estimated for all K classes at time t [4]. In order to estimate $\boldsymbol{c}_{t,k}$, we maintain a counter $n_{t,k}$ for each class k. If $y_t = k$, $n_{t,k} = n_{t-1,k} + 1$, otherwise $n_{t,k} = n_{t-1,k}$; then the center is updated as $\boldsymbol{c}_{t,k} = (\boldsymbol{c}_{t-1,k} \times n_{t-1,k} + \boldsymbol{x}_t)/n_{t,k}$.
2) Current classification model for existing classes ($\tilde{W}_{t-1}$). This model is updated as instances of existing classes are observed in the data stream.

The next section describes the proposed method which makes use of these two pieces of information to update the existing model when a new class emerges.

[1]The general form for SGDUpdate is: $\tilde{W}_t \leftarrow \tilde{W} - \eta_t \nabla_{\tilde{W}}$, where η_t is the learning rate. It may slightly differ in different SGD approaches.

[2]Specifically, the implementation used in this paper is ADAGRAD [3].

[3]To simplify the notation, we drop the arguments for g hereafter, i.e., g_i represents $g_i(\tilde{\boldsymbol{x}}_t, \tilde{W}_t)$.

[4]Note that $\boldsymbol{c}_{t,K_t}$ is instantiated with the new instance $\boldsymbol{x}_t$ when a new class emerges at time t.

III. Learning to generate pseudo instances (GPI)

To address the *new class adaptation* issue, we propose a learning to Generate Pseudo Instances (GPI) approach such that the generated instances can be used effectively to upgrade the current classifier of existing classes to a classifier for both the existing classes and the new class. This process learns a good performing classifier $\tilde{W}$ (Line 8 in Algorithm 1) from the only instance of a new class when it first appears.

Specifically, $\tilde{W}$ is initialized based on the class centers, including the currently observed instance of the new class, and the current classifier of existing classes $\tilde{W}_{t-1}$. The initialization of $\tilde{W}$ is done as follows: 1) generate pseudo instances for both new and existing classes; 2) train a classifier based on the generated instances to produce $\tilde{W}$.

Then, $\tilde{W}$ is used to adapt the current classifier $\tilde{W}_{t-1}$ for existing classes to incorporate the new class. This is done iteratively by generating pseudo instances for all classes and updating $\tilde{W}$ to further refine the model. The pseudo instance generation process is the key contribution of this paper.

A. Initialization of $\tilde{W}$

The initialization process of upgrading the current model $\tilde{W}_{t-1}$ to incorporate the new class involves generating pseudo instances for existing classes and the new class. They are described in the following two subsections.

A1) Pseudo Instance Generation for Existing Classes

A pseudo instance of an existing class must have three properties in order to produce a good discriminative model. The properties are defined in the following paragraphs.

Property 1. *A pseudo instance $\hat{\boldsymbol{x}}$ of class i with respect to class j shall be in the class i region bordering class j region.*

Note that we have no access to all historical data; but we have access to the class centers C and the current classification model $\tilde{W}$. To produce pseudo instances which have Property 1, we propose to simultaneously minimize (i) the distance between pseudo instance ($\hat{\boldsymbol{x}}$) and the center of class j ($\boldsymbol{c}_j$): $\|\hat{\boldsymbol{x}}-\boldsymbol{c}_j\|^2$; and (ii) log loss in class i: $-\ln(g_i)$ [5]:

$$\min_{\hat{\boldsymbol{x}}} -\ln(g_i) + \frac{\lambda}{2}\|\hat{\boldsymbol{x}}-\boldsymbol{c}_j\|^2, \tag{2}$$

where λ is the trade off parameter.

Optimizing Eqn. (2) via gradient descent produces (intermediate) pseudo instances which will gradually approach the decision boundary between i and j.

However, Property 1 is not sufficient because it only takes the current classifier into consideration without the knowledge of the new class. It can produce instances which are well inside the region of the new class rather than at the boundary.

The second property is proposed to tightly bound the region of each class.

Property 2. *Pseudo instances of class i must be within a fixed distance from its class center c_i.*

[5]We focus on $\hat{\boldsymbol{x}}$ for class i, and omit constant terms in the original log loss

This property can be acquired by having a bounded norm constraint:

$$\|\hat{\boldsymbol{x}}-\boldsymbol{c}_i\|^2 \le \delta^2, \tag{3}$$

where δ is a threshold.

In addition, pseudo instances should cover a large area along the decision boundary. Thus the third property is required and it is given as follows.

Property 3. *Pseudo instances of class i must be highly disperse around the region of class i.*

The dispersion of two pseudo instances $\hat{\boldsymbol{x}}_1$ and $\hat{\boldsymbol{x}}_2$, generated for class i (centered at $\boldsymbol{c}_i$), is measured by the angle (θ) between $\hat{\boldsymbol{x}}_1-\boldsymbol{c}_i$ and $\hat{\boldsymbol{x}}_2-\boldsymbol{c}_i$:

$$\mathrm{dis}_{\boldsymbol{c}_i}(\hat{\boldsymbol{x}}_1,\hat{\boldsymbol{x}}_2) = 1 - \frac{(\hat{\boldsymbol{x}}_1-\boldsymbol{c}_i)^\top(\hat{\boldsymbol{x}}_2-\boldsymbol{c}_i)}{\|\hat{\boldsymbol{x}}_1-\boldsymbol{c}_i\|\cdot\|\hat{\boldsymbol{x}}_2-\boldsymbol{c}_i\|}.$$

The larger the angle is between $\hat{\boldsymbol{x}}_1-\boldsymbol{c}_i$ and $\hat{\boldsymbol{x}}_2-\boldsymbol{c}_i$, the larger dispersion will be between $\hat{\boldsymbol{x}}_1$ and $\hat{\boldsymbol{x}}_2$.

We adopt a greedy strategy to generate pseudo instances for each class i, so as to maximize the pairwise dispersion. Let $P_i=\{\hat{\boldsymbol{x}}_1,\cdots,\hat{\boldsymbol{x}}_m\}$ be m pseudo instances generated for class i thus far. A new pseudo instance $\hat{\boldsymbol{x}}$ for the same class, which maximizes the dispersion, is formulated as follows:

$$\min_{\hat{\boldsymbol{x}}} \frac{1}{m}\sum_{\hat{\boldsymbol{x}}'\in P_i} \frac{(\hat{\boldsymbol{x}}-\boldsymbol{c}_i)^\top(\hat{\boldsymbol{x}}'-\boldsymbol{c}_i)}{\|\hat{\boldsymbol{x}}-\boldsymbol{c}_i\|\cdot\|\hat{\boldsymbol{x}}'-\boldsymbol{c}_i\|}. \tag{4}$$

According to [16], under the bounded norm constraint Eqn. (3), minimizing Eqn. (4) can be relaxed to

$$\min_{\hat{\boldsymbol{x}}} \frac{1}{2m}\sum_{\hat{\boldsymbol{x}}'\in P_i}\|\hat{\boldsymbol{x}}-\boldsymbol{c}_i+\hat{\boldsymbol{x}}'-\boldsymbol{c}_i\|^2. \tag{5}$$

Combining Eqns. (2), (3) and (5), we have our optimization formulation for generating pseudo instances for class i, given the class center $\boldsymbol{c}_j$ and previously generated instances P_i of size m:

$$\begin{aligned}&\min_{\hat{\boldsymbol{x}}} -\ln(g_i) + \frac{\lambda}{2}\|\hat{\boldsymbol{x}}-\boldsymbol{c}_j\|^2 + \frac{1}{2m}\sum_{\hat{\boldsymbol{x}}'\in P_i}\|\hat{\boldsymbol{x}}+\hat{\boldsymbol{x}}'-2\boldsymbol{c}_i\|^2\\ &\text{s.t.}\quad \|\hat{\boldsymbol{x}}-\boldsymbol{c}_i\|^2\le\delta^2.\end{aligned} \tag{6}$$

The third term $\frac{1}{m}\sum_{\hat{\boldsymbol{x}}'\in P_i}\|\hat{\boldsymbol{x}}+\hat{\boldsymbol{x}}'-2\boldsymbol{c}_i\|^2$ is set to 0, when P_i is empty, to avoid the division-by-zero error.

To optimize Eqn. (6), the gradient descent is applied. We define

$$\mathcal{L} = -\ln(g_i) + \frac{\lambda}{2}\|\hat{\boldsymbol{x}}-\boldsymbol{c}_j\|^2 + \frac{1}{2m}\sum_{\hat{\boldsymbol{x}}'\in P_i}\|\hat{\boldsymbol{x}}+\hat{\boldsymbol{x}}'-2\boldsymbol{c}_i\|^2, \tag{7}$$

then the gradient w.r.t. $\mathcal{L}$ is given by

$$\nabla_{\hat{\boldsymbol{x}}} = -(1-g_i)\boldsymbol{w}_i + \lambda(\hat{\boldsymbol{x}}-\boldsymbol{c}_j) + \frac{1}{m}\sum_{\hat{\boldsymbol{x}}'\in P_i}(\hat{\boldsymbol{x}}+\hat{\boldsymbol{x}}'-2\boldsymbol{c}_i),$$

where $\boldsymbol{w}_i = \tilde{W}_{1:d,i}$ is the first d rows of the i-th column of $\tilde{W}$. The update rule for $\hat{\boldsymbol{x}}$ is $\hat{\boldsymbol{x}} \leftarrow \hat{\boldsymbol{x}} - \eta\nabla_{\hat{\boldsymbol{x}}}$, where η is the learning rate.

After $\hat{\boldsymbol{x}}$ is updated, we check whether the constraint specified in Eqn. (3) is satisfied. If $\|\hat{\boldsymbol{x}}-\boldsymbol{c}_i\|^2 > \delta^2$, we project $\hat{\boldsymbol{x}}$ on the surface of the ball with center $\boldsymbol{c}_i$ having radius δ:

$$\hat{\boldsymbol{x}} \leftarrow \frac{\hat{\boldsymbol{x}}-\boldsymbol{c}_i}{\|\hat{\boldsymbol{x}}-\boldsymbol{c}_i\|}\delta + \boldsymbol{c}_i. \tag{8}$$

A2) Pseudo Instance Generation for the New Class

Having seen only the first instance of the new class, no knowledge of its decision boundary is available. As a result, Property 1, which is valid for existing classes only, cannot be applied. The replacement property is given as follows:

Property 4. *Each pseudo instance of a new class j is predicted to belong to any of the existing classes $i \neq j$ with the highest uncertainty by the model $\tilde{W}_{t-1}$ for all existing classes $i = 1, \ldots, K$.*

Entropy is a measure that quantifies the amount of uncertainty. Maximizing the entropy will obtain the largest uncertainty.

Note that g_k, in Eqn. (1), has a probabilistic interpretation: it is the probability that $\hat{\boldsymbol{x}}$ is classified to class k: $p(y=k|\hat{\boldsymbol{x}})=g_k$. The joint probability is given by $p(y=k,\hat{\boldsymbol{x}})=p(y=k|\hat{\boldsymbol{x}})p(\hat{\boldsymbol{x}})$. The joint entropy can then be given as:

$$\begin{aligned} H(\hat{\boldsymbol{x}},y) &= -\sum_{k=1}^{K} p(y=k,\hat{\boldsymbol{x}})\,\ln(p(y=k,\hat{\boldsymbol{x}})) \\ &= -p(\hat{\boldsymbol{x}})\,\ln(p(\hat{\boldsymbol{x}}))\sum_{k=1}^{K} g_k - p(\hat{\boldsymbol{x}})\sum_{k=1}^{K} g_k\,\ln(g_k), \end{aligned} \tag{9}$$

where K is the number of existing classes.

Because $\sum_{k=1}^{K} g_k = 1$ and $p(\hat{\boldsymbol{x}})$ is an unknown constant, maximizing Eqn. (9) w.r.t. $\hat{\boldsymbol{x}}$ is equivalent to minimizing $\sum_{k=1}^{K} g_k \ln(g_k)$.

Because Properties 2 and 3 are still valid for the new class, the optimization can be done similarly by simply replacing the first term in Eqn. (6) with $\sum_{k=1}^{K} g_k \ln g_k$. We therefore have the optimization formulation for the new class as follows:

$$\begin{aligned} &\min_{\hat{\boldsymbol{x}}} \; \sum_{k=1}^{K} g_k \ln(g_k) + \frac{\lambda}{2}\|\hat{\boldsymbol{x}}-\boldsymbol{c}_j\|^2 + \frac{1}{2m}\sum_{\hat{\boldsymbol{x}}'\in P_i}\|\hat{\boldsymbol{x}}+\hat{\boldsymbol{x}}'-2\boldsymbol{c}_i\|^2 \\ &\text{s.t.} \quad \|\hat{\boldsymbol{x}}-\boldsymbol{c}_i\|^2 \le \delta^2. \end{aligned} \tag{10}$$

Define $\mathcal{L}'$ as

$$\mathcal{L}' = \sum_{k=1}^{K} g_k \ln(g_k) + \frac{\lambda}{2}\|\hat{\boldsymbol{x}}-\boldsymbol{c}_j\|^2 + \frac{1}{2m}\sum_{\hat{\boldsymbol{x}}'\in P_i}\|\hat{\boldsymbol{x}}+\hat{\boldsymbol{x}}'-2\boldsymbol{c}_i\|^2.$$

The optimization of Eqn. (10) can also be conducted by the gradient descend. The gradient of $\mathcal{L}'$ w.r.t. $\hat{\boldsymbol{x}}$ is given by

$$\nabla'_{\hat{\boldsymbol{x}}} = \sum_{k=1}^{K}(1+\ln(g_k))g_k(1-g_k)\boldsymbol{w}_k + \lambda(\hat{\boldsymbol{x}}-\boldsymbol{c}_j) + \frac{1}{m}\sum_{\hat{\boldsymbol{x}}'\in P_i}(\hat{\boldsymbol{x}}+\hat{\boldsymbol{x}}'-2\boldsymbol{c}_i).$$

B. Iterative Model Update

After training a classifier with generated pseudo instances as the initialization for $\tilde{W}$, the decision boundaries are available for all classes, including the new class. The subsequent process to refine the model or the decision boundaries is an iterative process called `GPI_update` which iterates the following two steps until it converges (i.e., $\tilde{W}$ does not change):

(i) Generate pseudo instances for all classes based on the centers;
(ii) Update $\tilde{W}$ (obtained in the previous iteration) using the generated pseudo instances in step (i).

The pseudo instances generation process in step (i) is the same as that for existing classes in the initialization of $\tilde{W}$ (Section III-A1), since the decision boundary for the new class has been established.

C. Set λ and δ automatically

We describe how to set λ and δ for Eqn (6) for the existing classes, and then followed by that for Eqn (6) and Eqn (10) for the new class.
(a) **`AdapSelect`λ** for Eqn. (6). Minimizing $\mathcal{L}(\hat{\boldsymbol{x}}, \tilde{W}, C, P_i, \lambda)$ in Eqn. (7) with an appropriate λ, $\hat{\boldsymbol{x}}$ will gradually approach the decision boundary. If λ is too large, the solution will cross the boundary. The automatic process begins by setting λ initially to a large value (say, $\lambda = 4$); and then check whether the optimized solution $\hat{\boldsymbol{x}}$ is correctly classified. If $\hat{\boldsymbol{x}}$ is classified as another class, λ is reduced by half. We repeat this process, until an appropriate λ is found.
(b) **`AdapSelect`δ**. The idea to set δ automatically for Eqn. (6) is similar to that of setting λ, i.e., it aims to generate pseudo instances which are correctly classified by the classifier. If δ is too large, due to the bounded norm constraint in Eqn. (6), the generated instances may be located within the region of another class. Given $\hat{\boldsymbol{x}}$, if $\|\hat{\boldsymbol{x}}-c_i\| > \delta$, then $\hat{\boldsymbol{x}}$ is projected via Eqn. (8). After the projection, if $\hat{\boldsymbol{x}}$ is still mistaken for another class by the classifier, then δ is reduced by half. We repeat this process, until an appropriate δ is found.

Both `AdapSelect`λ and `AdapSelect`δ examine whether a generated instance is correctly classified. Note that the pseudo instance generation procedure is related to a class pair. In the initialization of $\tilde{W}$, when the class pair involves a new class, `AdapSelect`λ and `AdapSelect`δ cannot be applied, since no decision boundary is available for the new class. Alternatives are used in the following two situations:

(a) Generate instances for an existing class i w.r.t. the new class. First, λ and δ for Eqn. (6) are determined based on `AdapSelect`λ and `AdapSelect`δ mentioned above for all class pairs $\{(i,j), j \in \{1,\cdots,K\} \setminus \{i\}\}$, where K is the number of existing classes.

Algorithm 2: `GPI`

```
Input: current weight: $\tilde{W}_{t-1}$; class centers: $C$; the number of
       pseudo instances for each class pair: $m$.
Output: Weight $\tilde{W}_t$ for prediction
1 $\tilde{W} \leftarrow$ GPI_init($\tilde{W}_{t-1}, C, m$);
2 repeat
3     $\tilde{W} \leftarrow$ GPI_update($\tilde{W}, C, m$);
4 until convergence;
```

Algorithm 3: `GPI_init`

```
Input: current weight: $\tilde{W}_{t-1}$; class centers: $C$; the number of
       pseudo instances for each class pair: $m$.
Output: Weight $\tilde{W}$
1  for $i = 1 : K$ do
2      $P_i \leftarrow \emptyset$;
3      for $j \in \{1,\cdots,K\} \setminus \{i\}$ do
4          for $k = 1 : m$ do
5              $\lambda_{i,j} \leftarrow$ AdapSelect$\lambda$;
6              $\delta_{i,j} \leftarrow$ AdapSelect$\delta$;
7              $\hat{x} \leftarrow$solve Eqn.(6), with $\lambda = \lambda_{i,j}$, $\delta = \frac{1}{2}\|c_{K+1} - c_j\|$;
8              add $\hat{x}$ to $P_i$;
9      $\lambda_{i,K+1} \leftarrow \min\{\lambda_{i,j}, j = 1,\cdots,K\}$;
10     $\delta_{i,K+1} \leftarrow \min\{\delta_{i,j}, j = 1,\cdots,K\} \cup \{\frac{1}{2}\|c_i - c_{K+1}\|\}$;
11     for $k = 1 : m$ do
12         $\hat{x} \leftarrow$solve Eqn. (6), with $\lambda = \lambda_{i,K+1}$, $\delta = \delta_{i,K+1}$;
13         add $\hat{x}$ to $P_i$;
14 $P_{K+1} \leftarrow \emptyset$;
15 for $j = 1 : K$ do
16     for $k = 1 : m$ do
17         $\hat{x} \leftarrow$solve Eqn. (10), with $\lambda = 1$, $\delta = \frac{1}{2}\|c_{K+1} - c_j\|$;
18         add $\hat{x}$ to $P_{K+1}$;
19 Randomly initialize $w_{K+1}$, then $\tilde{w}_{K+1} = [w_{K+1}; 1]$,
   $\tilde{W}_{init} = [\tilde{W}_{t-1}, \tilde{w}_{K+1}]$;
20 Build training set $\mathcal{T}$: $P_i$ labeled as class $i$, $i \in \{1,\cdots,K+1\}$;
21 $\tilde{W} \leftarrow$ train a classifier, given $\mathcal{T}$ and initial $\tilde{W}_{init}$.
```

```
Algorithm 4: GPI_update
Input: Initial weight: $\tilde{W}_{init}$; class centers: $C$; the number of
       pseudo instances for each class pair: $m$.
Output: Weight $\tilde{W}$
1  for $i = 1 : K+1$ do
2      $P_i \leftarrow \emptyset$;
3      for $j \in \{1, \cdots, K+1\} \setminus \{i\}$ do
4          for $k = 1 : m$ do
5              $\lambda_{i,j} \leftarrow$ AdapSelect$\lambda$;
6              $\delta_{i,j} \leftarrow$ AdapSelect$\delta$;
7              $\hat{\boldsymbol{x}} \leftarrow$solve Eqn. (6), with $\lambda = \lambda_{i,K+1}$, $\delta = \delta_{i,K+1}$;
8              add $\hat{\boldsymbol{x}}$ to $P_i$;
9  Build training set $\mathcal{T}$: $P_i$ labeled as class $i$, $i \in \{1, \cdots, K+1\}$;
10 $\tilde{W} \leftarrow$ train a classifier, given $\mathcal{T}$ and initial $\tilde{W} = \tilde{W}_{init}$.
```

Let $\lambda_{i,j}$ and $\delta_{i,j}$ denote the parameter settings for the pair (i, j). Then, pseudo instances of an existing class i w.r.t. the new class is generated via Eqn. (6) using the minimum $\lambda_{i,j}$ and $\delta_{i,j}$: $\lambda \leftarrow \min\{\lambda_{i,1}, \cdots, \lambda_{i,K}\}$ and $\delta \leftarrow \min\{\delta_{i,1}, \cdots, \delta_{i,K}, \frac{1}{2}\|\boldsymbol{c}_i - \boldsymbol{c}_{K+1}\|\}$, where $\boldsymbol{c}_{K+1}$ (which is the new class instance) is the new class center, and $\frac{1}{2}\|\boldsymbol{c}_i - \boldsymbol{c}_{K+1}\|$ is an estimate that δ is at the midpoint of two class centers, having no other information.

(b) Generate instances for the new class w.r.t. an existing class. Here, no adaptive selections could be made for Eqn. (10) since the decision boundary for the new class is unknown. As a heuristic, $\lambda = 1$ which gives the first and second terms in Eqn. (10) the same weighting; and $\delta = \frac{1}{2}\|\boldsymbol{c}_{K+1} - \boldsymbol{c}_j\|$.

Based on the descriptions in Sections III-A, III-B and III-C, three procedures are presented: (a) Algorithm 2 summarizes the entire GPI approach; (b) the initialization of $\tilde{W}$ is in Algorithm 3: GPI_init; and (c) iterative model update of $\tilde{W}$ is in Algorithm 4: GPI_update. Only one parameter m which needs to be set. In the following experiments, $m = 1$ is used as the default.

IV. Experiments on Synthetic Data

A. Configuration

Synthetic Data. The 2-dimensional dataset has the following characteristics: Each of the 10 classes has 1,000 instances drawn from a multivariate normal distribution. Then, 20% instances are randomly sampled as the test set, and the rest are used for a stream simulation.
Stream Simulation. To simulate a data stream, we randomly select 5 classes as new classes, and shuffle the training data to build a data stream such that each new class emerges after 1,000 instances are observed. Only 5 existing classes appear in the first 1000 instances, labeled as section 1-1000. Then, Classes 6-10 appear successively as new classes in the next 5000 instances in sections 1001-6000. In the section 6001-8000, there are no other new classes.
Evaluation. To evaluate any-time predictions, the accuracy of the classifier is assessed using the test set in each time step, after the classifier has been updated. All instances in the test set which contains all known classes are used in the assessment.
Baseline. We compare OPCIL-GPI with OPCIL-Rnd, where OPCIL-GPI uses GPI to obtain the classifier to incorporate a new class (Line 8 in Algorithm 1), and OPCIL-Rnd adopts a random initialization, so as to illustrate the effectiveness of our approach. For SGDUpdate in Line 13 Algorithm 1, we adopt ADAGRAD [3].

B. Results

The comparison result of OPCIL-GPI and OPCIL-Rnd is exhibited in Figure. 1. It is a vivid illustration of the cold start phenomenon exhibited by OPCIL-Rnd, i.e., the accuracy degrades significantly immediately after a new class emerges. The accuracy recovers to a good level only after a sufficient number of instances of the new class have been observed. In contrast, the accuracy degradation of OPCIL-GPI is more subdued; and notice that the performance gap between OPCIL-GPI and OPCIL-Rnd is huge in the initial time steps every time a new class first emerges. The performance gap can remain large for more than 500 instances in some sections, e.g., 2001-4000 and 5001-6000.

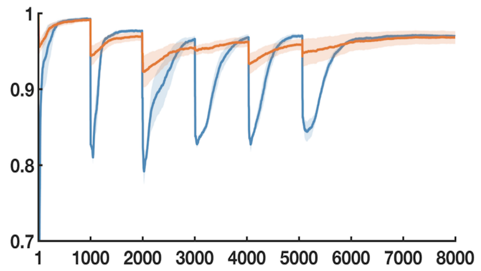

Figure 1. Comparison results between OPCIL-GPI and OPCIL-Rnd, where the orange line represents the average accuracy of OPCIL-GPI; and the blue line shows the average accuracy of OPCIL-Rnd; the shadow area is the variance on 5 independent runs. Performance drops significantly using OPCIL-Rnd when new classes appear; whereas this is more subdue using OPCIL-GPI.

V. Experiments on Real Data

A. Configuration

Datasets. We conduct experiments on 10 multi-class benchmark datasets, i.e., *dna, mnist, pendigits, satimage, segment, svmguide4, usps, vehicle, vowel, wine* [6]. As in Section IV, we take 80% of the dataset for training, and the remaining 20% for testing.
Stream Simulation. The simulation is same as that in Section IV, except that the same time interval between two new classes cannot be maintained because each real dataset has a limited number of instances and the number of instances for each class varies from one class to another. In addition, in order to check whether our approach is sensitive to the order of classes, we randomly shuffle the class set in each dataset 5 times. Thus for each dataset, we simulate 5 data streams with different new classes with different orders; and report the average result over the 5 data streams.
Evaluation. As in Section IV, we assess the any-time prediction accuracy of the trained classifier on the current class set $\mathcal{Y}_t$ at every time step t in the data stream using the test set. Because a new class emerges at different time steps in different data streams of the same dataset, we report (i) the overall accuracy over one entire stream, averaged over five streams; and (ii) the accuracy at each time step in which a new class first emerges, averaged over all new classes that emerge in a data stream and over five streams. We call this measure *First-emergence accuracy*.
Contenders. In addition to OPCIL-Rnd, we compare OPCIL-GPI with two other variants:

- OPCIL-CC uses the class centers only to establish $\tilde{W}$. CC is short for building $\tilde{W}$ using Class Centers;
- OPCIL-CCR uses the class centers and randomly generated instances around centers to build $\tilde{W}$. CCR is short for building $\tilde{W}$ using Class Centers and Random points. Specifically, those randomly generated instances for each class i are limited within a ball centered at the class center $\boldsymbol{c}_i$ having a radius $\min\{\frac{1}{2}\|\boldsymbol{c}_i - \boldsymbol{c}_j\|, j \in \{1, \cdots, K_t\}, j \neq i\}$. For a fair comparison, the number of generated instances for each class is set to be the same as that used in OPCIL-GPI.

[6]Those datasets are available on https://www.csie.ntu.edu.tw/ cjlin/libsvmtools /datasets/multiclass.html. For mnist, we randomly sample 20,000 instances.

B. Results

Figure 2 exhibits the overall performance of all methods using the performance of OPCIL-Rnd as the base. OPCIL-GPI is the best performer which is better than Rnd, CC, and CCR in all datasets. Specifically, it is better than Rnd on 9 of 10 datasets, significantly better than CC on 9 of 10 datasets, and significantly better than CCR on 6 of 10 datasets (under paired t-tests at 95% significance level).

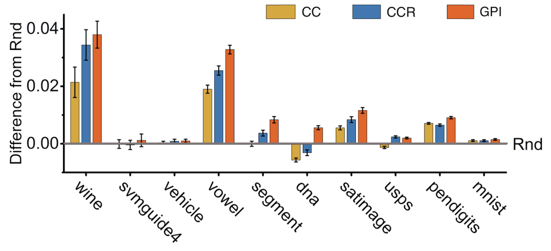

Figure 2. Overall accuracy with reference to OPCIL-Rnd. The gray baseline is OPCIL-Rnd. To simplify notation, prefix "OPCIL" is dropped.

Note that when the time gap between different new classes is large, the improvement is not obvious (see *usps, pendigits, and mnist*), since all those OPCIL algorithms will converge to a similar accuracy given enough instances; even though each new class model has a different initialization.

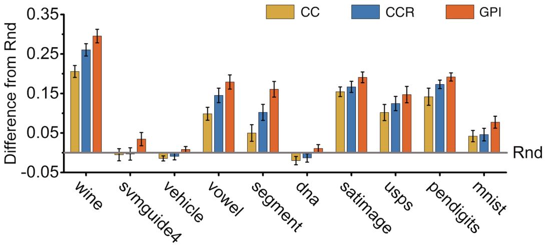

Figure 3. First-emergence accuracy with reference to OPCIL-Rnd.

Figure 3 shows the First-emergence accuracy with reference to OPCIL-Rnd. OPCIL-GPI achieves much better performance than OPCIL-Rnd, OPCIL-CC and OPCIL-CCR. It is significantly better than all other contenders on 9 out of 10 datasets. This validates the effectiveness of the new class adaptation approach.

To examine whether the performance difference between any two algorithms is significant, we perform a post-hoc Nemenyi test. The result shown in Figure 4 reveals that the proposed new class adaptation with GPI is significantly better than other contenders.

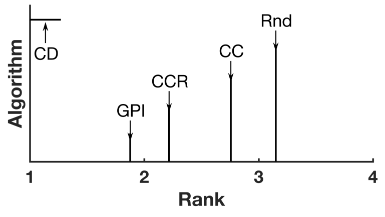

Figure 4. Critical difference (CD) diagram of the post-hoc Nemenyi test ($\alpha = 0.05$) for the comparison results in Figure 3. The difference between two algorithms is significant if the gap between their ranks is larger than the CD. There is a line between two algorithms if the rank gap between them is smaller than the CD.

VI. Conclusion

We introduce a new framework to deal with emerging new classes called *One-Pass Class Incremental Learning (OPCIL)*. It is an extension of both the one-pass learning and the class incremental learning to enable any-time predictions as new classes emerge in data streams. The central issue in OPCIL is how to effectively adapt a classifier of existing classes to incorporate emerging new classes. We call it the *new class adaptation* issue, and propose an effective method to address it. Having only one instance of the new class, the key is to generate pseudo instances which are optimized to satisfy properties that produce a good performing discriminative classifier. We provide the necessary properties and optimization procedures required to address this issue. Experiments on synthetic data and real datasets validate the effectiveness of our approach.

Acknowledgment

This research was supported by the NSFC (61333014) and the 111 Program (B14020), and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

[1] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.
[2] Q. Da, Y. Yu, and Z.-H. Zhou. Learning with augmented class by exploiting unlabeled data. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 1760–1766, 2014.
[3] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7):2121–2159, 2011.
[4] J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934, 2009.
[5] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.
[6] W. Gao, R. Jin, S. Zhu, and Z.-H. Zhou. One-pass auc optimization. In *Proceedings of the 30th International Conference on Machine Learning*, pages 906–914, 2013.
[7] E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.
[8] S. S. Khan and M. G. Madden. One-class classification: Taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(03):345–374, 2014.
[9] I. Kuzborskij, F. Orabona, and B. Caputo. From n to n+ 1: multiclass transfer incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3358–3365, 2013.
[10] B. M. Lake, R. Salakhutdinov, J. Gross, and J. B. Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, pages 2568–2573, 2011.
[11] X. Mu, F. Zhu, J. Du, E.-P. Lim, and Z.-H. Zhou. Streaming classification with emerging new class by class matrix sketching. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 2373–2379, 2017.
[12] M. D. Muhlbaier, A. Topalis, and R. Polikar. Learn++ nc: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes. *IEEE Transactions on Neural Networks*, 20(1):152–168, 2009.
[13] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, J. C. Platt, et al. Support vector method for novelty detection. In *Proceedigns of Advances in Neural Information Processing Systems 12*, pages 582–588, 1999.
[14] S. Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2012.
[15] D. M. Tax and R. P. Duin. Uniform object generation for optimizing one-class classifiers. *Journal of machine learning research*, 2(12):155–173, 2001.
[16] Y. Yu, Y.-F. Li, and Z.-H. Zhou. Diversity regularized machine. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 1603–1608, 2011.
[17] Y. Zhu, W. Gao, and Z.-H. Zhou. One-pass multi-view learning. In *Proceedings of the 6th Asian Conference on Machine Learning*, pages 407–422, 2016.
[18] Y. Zhu, K.-M. Ting, and Z.-H. Zhou. Multi-label learning with emerging new labels. In *Proceedings of the 16th IEEE International Conference on Data Mining*, pages 1371–1376, 2016.