Data-efficient Online Classification with Siamese Networks and Active Learning

Kleanthis Malialis[a], Christos G. Panayiotou[a, b] and Marios M. Polycarpou[a, b]
[a] *KIOS Research and Innovation Center of Excellence*
[b] *Department of Electrical and Computer Engineering*
University of Cyprus
Nicosia, Cyprus
Email: {malialis.kleanthis, christosp, mpolycar}@ucy.ac.cy
ORCID: {0000-0003-3432-7434, 0000-0002-6476-9025, 0000-0001-6495-9171}

***Abstract*—An ever increasing volume of data is nowadays becoming available in a streaming manner in many application areas, such as, in critical infrastructure systems, finance and banking, security and crime and web analytics. To meet this new demand, predictive models need to be built online where learning occurs on-the-fly. Online learning poses important challenges that affect the deployment of online classification systems to real-life problems. In this paper we investigate learning from limited labelled, nonstationary and imbalanced data in online classification. We propose a learning method that synergistically combines siamese neural networks and active learning. The proposed method uses a multi-sliding window approach to store data, and maintains separate and balanced queues for each class. Our study shows that the proposed method is robust to data nonstationarity and imbalance, and significantly outperforms baselines and state-of-the-art algorithms in terms of both learning speed and performance. Importantly, it is effective even when only 1% of the labels of the arriving instances are available.**

***Index Terms*—online active learning, siamese neural networks, nonstationary environments, concept drift, class imbalance.**

I. Introduction

Traditionally, predictive models are built from historical data consisting of examples annotated with class labels (i.e. the ground truth). This paper is concerned with online learning, with a focus on the following key challenges:

- **One-by-one online learning**: We focus on online learning i.e. as data is arriving in a streaming fashion. Contrary to the majority of online learning work, we focus on one-by-one learning, where only a single instance (rather than a batch) is observed at each time step.
- **Limited labelled data**: Acquiring labels at every step is expensive / impractical. A potential solution that makes such an assumption, may not be practical in real applications. In this paper, we consider limited labelled data.
- **Nonstationary data**: This paper focuses on online learning where the distribution of data is unknown. It is also concerned with cases where the data distribution is nonstationary; i.e., it evolves or "drifts" over time.
- **Imbalanced data**: Class imbalance, in conjunction with the aforementioned challenges, causes one-by-one online learning to become significantly more challenging.

The contributions of this paper are as follows. We provide new insights into learning from limited labelled, nonstationary and imbalanced in one-by-one online classification, a largely unexplored area. We propose a novel learning approach for one-by-one online learning which utilises active learning and siamese neural networks. Active learning is a paradigm in which the classifier selectively queries an oracle (typically, a human expert) to provide class labels according to an allocated budget [1]. Several industrial large-scale classification systems, such as, Google's method for labeling malicious advertisements, have been realised through active learning [2].

Siamese networks enable learning when only a few examples per class are available, commonly referred to as *few-shot learning*, and have recently achieved state-of-the-art results in image recognition [3]. To our knowledge, this is the first time that an approach is developed, which synergistically brings together siamese networks and one-by-one active learning. The proposed synergy enables the effective learning from limited labelled data in nonstationary and imbalanced settings.

The organisation of this paper is as follows. Section II provides the background material necessary to understand the contributions made in the paper. Section III provides a review of related work. The proposed learning approach is presented in Section IV. Our experimental setup is described in Section V while the experimental results are presented and discussed in Section VI. Lastly, concluding remarks and directions for future work are provided in Section VII.

II. Background

In **online** learning we consider a data generating process that provides at each time step t a sequence of examples or instances $S^t = \{(x_i^t, y_i^t)\}_{i=1}^M$ from an unknown probability distribution $p^t(x, y)$, where $x^t \in \mathbb{R}^d$ is a d-dimensional input vector belonging to input space $X \subset \mathbb{R}^d$, $y^t \in [1, K]$ is the class label, $K \geq 2$ is the number of classes and M is the number of instances arriving at each step.

When the observed sequence S^t consists only of a single instance (i.e. $M = 1$), it is termed **one-by-one online** learning, otherwise it is termed **batch-by-batch online** learning [4]. The design of batch-by-batch algorithms differs significantly from that of one-by-one algorithms as they are designed to process chunks of data, possibly by utilising an offline

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

learning algorithm [5]. Therefore, the majority of batch-by-batch algorithms are typically not suitable for one-by-one tasks [5]. This work focuses on one-by-one online learning, which is important for real-time monitoring and control.

Active learning is concerned with strategies to selectively query for labels from an *oracle* (typically, a human expert) according to a set of available resources [1]. Typically, the available resources are modelled by an allocated budget $B \in [0, 1]$ where it is expressed as a fraction of the number of arriving examples e.g. $B = 0.1$ means that 10% of the arriving instances can be labelled [6]. A budget spending mechanism must ensure that the labelling spending $b \in [0, 1]$ does not exceed the allocated budget.

In one-by-one online classification, a classifier is built that receives a new example x^t at time t and makes a prediction $\hat{y}^t$ based on a concept $h : X \to Y$ such that $\hat{y}^t = h(x^t)$. A given active learning strategy $\alpha : X \to \{0, 1\}$ determines if the true label y^t is required, which is assumed that the oracle will provide. The classifier is evaluated using a loss function and is then trained, i.e., its parameters are updated accordingly based on the loss incurred. This process is repeated at each step and, depending on the application, new examples do not necessarily arrive at regular and pre-defined intervals. If learning occurs on the most recent single instance (or batch) only, without taking into account previously labelled data, it is termed **incremental** (or *one-pass*) learning [4]. Specifically, the cost J at time t is calculated using the loss function l as follows $J = l(y^t, \hat{y}^t)$.

Learning in **nonstationary environments** is a major challenge in some applications. Nonstationarity is caused by **concept drift**, which represents a change in the joint probability. Drift can be characterised by type, severity, speed, predictability, frequency and recurrence [7]. Hence, in practise, it is very difficult to characterise concept drift. Our focus is on *learning the concept drift* without its explicit characterisation and detection. **Class imbalance** [8] is another challenge that occurs when at least one data class is under-represented compared to others, thus constituting a minority class.

III. Related Work

Algorithms that are capable of learning from imbalanced and nonstationary data in one-by-one online classification typically fall into two categories: *(i)* resampling algorithms (e.g. *QBR* [9], *OOB* [10]) and *(ii)* cost-sensitive learning algorithms (e.g. *CSOGD* [11]). These have been shown to perform well provided that the data label becomes available at each time step. This may be a key limitation in some applications since acquiring data labels is expensive or impractical to do at every single time step. This work focuses on active learning to address this problem. We provide a description of existing active learning strategies and budget spending mechanisms.

A. Active learning strategies

The most common active learning strategy is **uncertainty sampling**, where the learner queries the most uncertain instances, which are typically found around the decision boundary. One way to measure uncertainty is to query the instance whose best labelling is the least confident [1]. The majority of active learning work assumes the availability of all training examples (offline active learning) [12] while some work considers batch-by-batch online active learning [13].

Recently the community started focusing on one-by-one active learning [6]. The arriving x^t is queried if it satisfies:

$$p(y^*|x^t) < \theta, \tag{1}$$

where $y^* = \text{argmax}_y \, p(y|x^t)$ and θ is a threshold which is typically fixed. This is known as a *fixed uncertainty* strategy

In [6] the authors introduce a *randomised variable uncertainty* strategy. A fixed uncertainty strategy may fail if the threshold is set incorrectly, or if the classifier learns enough so that the uncertainty remains above the fixed threshold most of the time. The threshold is modified as follows:

$$\theta = \begin{cases} \theta(1-s) & \text{if } p(y^*|x^t) < \theta_{rdm} \\ \theta(1+s) & \text{if } p(y^*|x^t) \geq \theta_{rdm} \end{cases} \tag{2}$$

where s is a step size parameter, $\theta_{rdm} = \theta * \eta \sim N(1, \delta)$ and δ is another parameter. Randomisation ensures that the probability of labelling an instance is not zero. This strategy has been shown to work very well.

Uncertainty sampling has been criticised as being prone to outliers [1] and, for this reason, **density sampling** has been proposed. Its central idea is that informative queries are not only those that are uncertain, but those which lie in high density regions. As with uncertainty sampling, the majority of work is on offline active learning where the set of all unlabelled instances U is already available. In [14] the instance queried for labelling is selected by its average similarity to other instances in U as follows:

$$\underset{x}{\text{argmax}} \frac{1}{U} \sum_{u=1}^{U} sim(x, x_u), \tag{3}$$

where sim is the cosine similarity. Density sampling has also been applied in batch-by-batch online active learning where, using the Mahalanobis distance, an informative instance is the one which is similar to other unlabelled instances in the most recent batch [15]. In our work, we use density sampling in one-by-one active learning. The interested reader is directed towards [1] for a survey on active learning strategies.

B. Budget spending mechanisms

A budget spending mechanism must ensure that the label spending $b \in [0, 1]$ does not exceed the allocated budget $B \in [0, 1]$ over infinite time. The most common approach is to count the *exact* labelling spending [6]. The labelling expenses at any time t is given by:

$$b^t = \frac{u^t}{t}, \tag{4}$$

where u^t is the number of instances queried until time t. The drawback of this mechanism is that the contribution of every next label will diminish over infinite time.

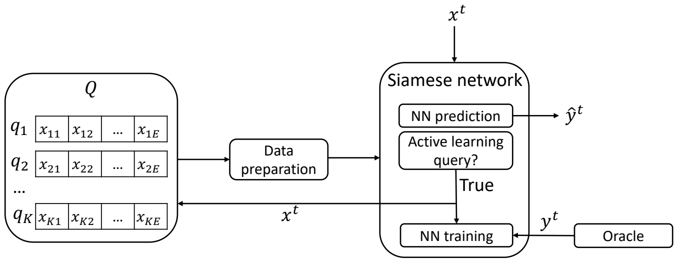


Fig. 1. Overview of the proposed method which uses a multi-sliding window approach to data storage, a siamese neural network (NN) and active learning to enable online learning from limited, nonstationary and imbalanced data.

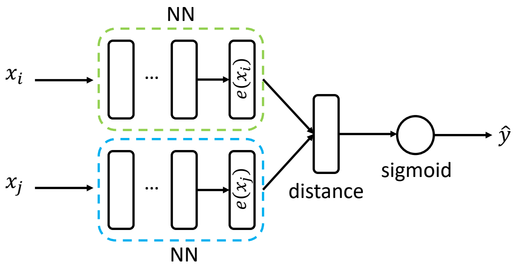


Fig. 2. A siamese neural network (NN)

One way to solve the aforementioned problem is to count the *exact* labelling spending over a sliding window w:

$$b^t = \frac{u_w^t}{w}, \tag{5}$$

where u_w^t is the number of instances queried within the sliding window. This, however, defies the requirements of incremental learning. The authors in [6] propose to *approximate* the number of instances queried within the sliding window:

$$\hat{u}_w^t = \lambda \hat{u}_w^{t-1} + a(x^t), \tag{6}$$

where $\lambda = \frac{w-1}{w}$. The authors prove that $\hat{b}$ is an unbiased estimate of b.

IV. Proposed approach

The overview of the proposed learning approach is shown in Fig. 1. The learning approach uses a multi-sliding window approach to store data, denoted by Q in the figure. A data preparation phase is in place before feeding the data to a siamese neural network. For each arriving example x^t, the siamese network makes a prediction $\hat{y}^t$. Notice that, only if the active learning strategy decides to send a query, the new example is stored in the Q and the siamese network is then trained. A detailed description of each component is provided below, followed by a discussion on the advantages and limitations of the proposed approach.

A. Detailed Description

Data storage: Our approach assumes the initial availability of E examples per class. Importantly, to avoid a potential deployment issue, we restrict E to be a very small number, e.g., up to five. We argue that for the vast majority of applications this assumption is realistic.

The initial labelled examples and those which will be queried by the active learning strategy will be stored using a multi-sliding window approach. Each sliding window is implemented as a queue and at any time t, we maintain a collection of K queues for each class:

$$Q^t = \{q_c^t\}_{c=1}^K, \tag{7}$$

where $K \geq 2$ is the number of classes. All queues have the same capacity and each queue is defined as follows:

$$q_c^t = \{x_{ci}\}_{i=1}^E, \tag{8}$$

where for any two $x_{ci}, x_{cj} \in q_c^t$ such that $j > i$, x_{cj} arrived more recently in time. The multi-sliding window approach is depicted by Q in the figure. Notice that we have set the capacity of each queue to $|q_c^t| = E$. The advantage of this is two-fold. Firstly, the number of examples required for storage is small and secondly, the queues will always remain balanced thus avoiding any bias towards majority classes.

Siamese network & training: At the heart of the proposed approach lies a siamese neural network [16]. A siamese network consists of two identical neural networks (the 'twins') as shown in Fig. 2. The central idea is to learn a function (denoted by e on the figure) that maps an input pattern into a target space (the 'embedding') in such a way that a simple distance in the target space approximates the neighbourhood relationships in the input space. For this reason, siamese networks have been shown to significantly outperform traditional models such as k-nearest neighbour (k-NN) algorithms in high-dimensional spaces e.g. for image recognition [3]. We use the *L1* or *Manhattan* distance as in [3].

$$\begin{aligned} d(x_1, x_2) =& ||e(x_1) - e(x_2)||_1 \\ =& \sum_i |e(x_1)_i - e(x_2)_i| \end{aligned} \tag{9}$$

where the embeddings are of equal size $|e(x_1)| = |e(x_2)|$.

This is then given to a single sigmoidal output unit. Ideally, the siamese network $h : X \times X \rightarrow Y$ will learn to output $\hat{y} = 1$ for any pair of inputs that belongs to the same class and $\hat{y} = 0$ if the pair of inputs is of different class. There exists a process which transforms the examples in Q^t into the training pairs Q_{train}^t (described in the next section). The cost function used is then as follows:

$$J = \frac{1}{|Q_{train}^t|} \sum_{(x_1,x_2) \in Q_{train}^t} l(y^t, h(x_1, x_2)) \tag{10}$$

where the loss function l used is the binary cross-entropy.

Data preparation: Given the data in Q^t at time t, we generate all possible combinations C_2^t of size two. Three subsets of C_2^t are then generated as follows.

The first one (Q_{id}) contains all pairs in which the two examples are identical and, hence, belong to the same class. The second one (Q_{same}) contains all (non-identical) pairs in which the two examples belong to the same class. The two are then joined ($Q^t_{id\_same}$) to indicate the positive pairs. These are defined as follows:

$$Q^t_{id} = \{(x_{c_1,i_1}, x_{c_2,i_2}) \in C^t_2 | c_1 = c_2, i_1 = i_2\} \quad (11)$$

$$Q^t_{same} = \{(x_{c_1,i_1}, x_{c_2,i_2}) \in C^t_2 | c_1 = c_2, i_1 \neq i_2\} \quad (12)$$

$$Q^t_{id\_same} = Q^t_{id} \cup Q^t_{same} \quad (13)$$

It is essential that Q^t_{same} is not empty; in other words, the approach expects that the initial labelled set consists of at least two examples per class i.e. $E \geq 2$.

The third subset (Q_{diff}) contains all pairs in which the two examples belong to different queues:

$$Q^t_{diff} = \{(x_{c_1,i_1}, x_{c_2,i_2}) \in C^t_2 | c_1 \neq c_2\} \quad (14)$$

Importantly, resizing is performed to ensure balance between positive and negative pairs. The training set is thus formed as follows:

$$Q^t_{train} = Q^t_{id\_same} \cup Q^t_{diff} \quad (15)$$

Class prediction: The siamese network predicts the class of each arriving instance x^t by taking into consideration all examples in the queues Q^t. For each queue, we find the average similarity of x^t to its elements. We then choose the queue with the highest average similarity as follows:

$$\underset{c \in [1,K]}{\operatorname{argmax}} \frac{1}{|E|} \sum_{i=1}^{E} p(y|x^t, x^t_{ci}) \quad (16)$$

Active learning strategy: One of the advantages of this approach is that it is highly flexible with respect to the active learning strategy as it does not rely upon any specific strategy. Since the proposed approach is intended to address one-by-one online classification tasks, however, it is expected that a one-by-one active learning strategy will be used.

This work uses a *randomised variable similarity* strategy, inspired from the *randomised variable uncertainty* strategy. In fact, the equation is the same as the one described in Eq. 2, although, the selection criterion is not $p(y^*|x^t)$ but the maximum similarity in the predicted class:

$$\max_i p(y|x^t, x^t_{ci}) \quad (17)$$

where c is the class selected using Eq. 16. The budget spending mechanism used is the one shown in Eq. 6. The pseudocode of the proposed learning approach is presented in Algorithm 1.

B. Discussion

Class imbalance. The proposed approach is robust to class imbalance. This is the result of three mechanisms 'embedded' in the approach. Firstly, the use of *separate* and *balanced* queues alleviates the problem as propagating past examples in the most recent training set can be viewed as a form of oversampling. This concept has been applied in [9] for binary one-by-one online classification tasks. This work extends this to a multi-sliding window approach. Secondly, the data preparation phase creates $|Q^t_{train}|$ training pairs, as opposed to the $K \times E$ examples in the Q^t. Depending on the values of K and E, the number of training pairs can be considerably larger thus constituting another form of oversampling. Thirdly, we always balance the number of positive and negative pairs. As we will demonstrate in our experimental work, the learning approach can perform well even in extreme imbalanced scenarios.

Algorithm 1 Proposed learning method

```
Input:
a: active learning strategy
B: labelling budget
$Q^0$: initial labelled examples
$h^0$: siamese network
$b^0 = 0$: budget expenses
for each time step $t$ do
    receive example $x^t \in \mathbb{R}^d$
    predict class using Eq.16
    $h^t = h^{t-1}$
    $Q^t = Q^{t-1}$
    if $b^{t-1} < B$ then                ▷ expenses within budget
        calculate query criterion value using Eq.17
        if $a(x^t, v) == True$ then      ▷ label request
            receive true label $y^t$
            append $x^t$ to relevant queue in $Q^t$
            prepare training pairs $Q^t_{train}$ using Eq. 15
            calculate cost $J$ using Eq. 10
            update classifier $h^t = h^{t-1}.train()$
    update budget expenses $b^t$ using Eq. 6
```

Concept drift. As it will be illustrated, the learning approach is robust to drift too. As the examples are carried over a series of time steps, this allows the classifier to 'remember' old concepts. The classifier needs to also be able to 'forget' old concepts. This is achieved by the algorithm's memory-based nature i.e. by bounding the length of queues, these are behaving like sliding windows.

Fixed memory. The proposed learning algorithm is not incremental. An incremental learning algorithm would receive instance x^t, its active learning strategy would decide if training will be performed, and then would discard x^t. The proposed algorithm, despite not being incremental, always uses a fixed amount of memory that contains $K \times E$ examples. Additionally, the storage requirements are low since the number of examples per class is kept to a minimum; e.g. $|q^t_c| = E \in \{2, 3, 4, 5\}$ Most importantly, however, we will demonstrate that an incremental learning algorithm performs significantly worse compared to algorithms that utilise the examples in Q^t.

V. Experimental Setup

A. Data

Synthetic datasets provide us with the flexibility to control various parameters of the approach; e.g., the severity of class imbalance, when to introduce concept drift and the drift characteristics. Synthetic datasets enable us to stress test the proposed approach. We will use the following datasets.

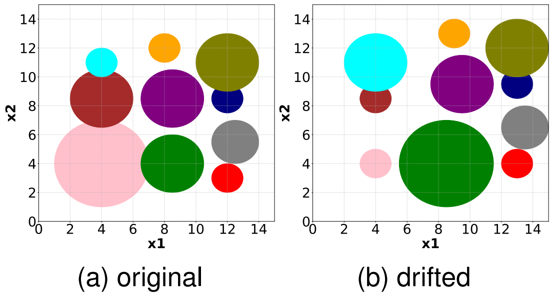


Fig. 3. The *circles10* dataset which consists of ten classes.

sea4 [17]: It has two features $x_1, x_2 \in [0, 10]$ and four classes. The decision boundaries are as follows:

$$\begin{aligned} 0 \leq x_1 + x_2 < \theta_1 &\longrightarrow \mathit{class}\ 1 \\ \theta_1 \leq x_1 + x_2 < \theta_2 &\longrightarrow \mathit{class}\ 2 \\ \theta_2 \leq x_1 + x_2 < \theta_3 &\longrightarrow \mathit{class}\ 3 \\ \theta_3 \leq x_1 + x_2 \leq 10 &\longrightarrow \mathit{class}\ 4 \end{aligned} \tag{18}$$

We choose the thresholds as follows $\theta_1 = 3.0$, $\theta_2 = 5.0$ and $\theta_3 = 7.0$. When concept drift occurs, the thresholds are changed abruptly to $\theta_1 = 2.0$, $\theta_2 = 6.0$ and $\theta_3 = 8.0$. Data normalisation is afterwards applied so that $x_1, x_2 \in [0, 1]$.

Initially, the dataset is balanced i.e. the probability of an arriving instance belonging to any class is $p(y) = 0.25$. Also, we conducted experiments in a multi-minority scenario where the probability of an arriving instance belonging to a specific class is $p(y) = 0.97$, while for the other three is $p(y) = 0.01$. Notice that this constitutes a case of severe imbalance as our aim is to stress test our learning approach.

circles10 [18]: It has two features $x_1, x_2 \in [0, 15]$ and ten classes as shown in Fig. 3a. Each class function is a circle given by $(x_1 - x_{1c})^2 + (x_2 - x_{2c})^2 = r_c^2$ where (x_{1c}, x_{2c}) is its centre and r_c its radius. Data normalisation is applied so that $x_1, x_2 \in [0, 1]$.

The same dataset under concept drift is presented in Fig. 3b. The radius of the three vertical circles on the left (pink, brown, cyan) and the green circle has been changed, while all the remaining circles have been drifted by $+1$ in both dimensions. Concept drift affects all the classes *simultaneously* and *immediately* (abruptly) in our experiments.

Initially, the dataset is balanced i.e. the probability of an arriving instance belonging to any class is $p(y) = 0.1$. We also conducted experiments in a multi-minority scenario where the probability of an arriving instance belonging to a specific class (pink circle) is $p(y) = 0.955$, while for the rest is $p(y) = 0.005$. This constitutes a case of extreme imbalance, which creates significant problems for most learning algorithms.

Lastly, the *circles10* is more challenging than *sea4*, not only because it has a larger number of classes, but also because the data is noisy. We can observe from Fig. 3a the overlap between some circles, in other words, examples with the same inputs may not have the same class label.

B. Compared methods

For fairness, all the approaches share the same base classifier, which is a fully-connected neural network of three hidden 32-neuron layers with parameters as follows: *He Normal* [19] weight initialisation, learning rate of 0.01, the *Rectified Adam* [20] optimisation algorithm, *LeakyReLU* [21] as the activation function of the hidden neurons and mini-batch size of 64. Note that the classifier is only trained once per time step (i.e. $num\_epochs = 1$) as, in practise, this would allow learning in high-speed data applications. For the siamese network, the *sigmoid* activation and the *binary cross-entropy* loss function are used, while for a fully-connected network, the *softmax* activation and the *categorical cross-entropy* loss function. The following algorithms are compared in our study:

incremental: The state-of-the-art incremental learning algorithm [6] which initially proposed the active learning strategy in Eq. 2 and the budget spending mechanism in Eq. 6. We use the following parameters as recommended by [6]: step size parameter $s = 0.01$, randomisation threshold $\delta = 1.0$ and sliding window $w = 300$.

ActiQ: It uses the proposed learning approach but instead of a siamese network, it uses the fully-connected neural network described earlier. It is similar to the previous one but it is not incremental as it makes use of older examples in Q. In all experiments $E = 5$.

ActiSiamese: This is the proposed approach as discussed in Section IV and its pseudocode is given in Algorithm 1.

To make this comparison as fair as possible, in addition to the fact that all approaches share the same active learning strategy and base classifier, we do not allow any offline learning. In other words, learning starts at time $t = 0$ for all compared methods, even if the *ActiQ* and *ActiSiamese* have access to the initial labelled set of E examples.

C. Performance metrics

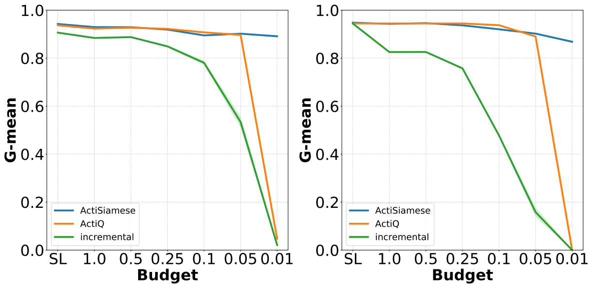
Classifiers are typically evaluated using the overall accuracy metric. When class imbalance exists, however, this metric becomes problematic as it is biased towards the majority class(es) [8]. Hence, it is necessary to use a metric which is not sensitive to imbalance. The geometric mean is such a metric which is defined as follows [22]:

$$G\text{-}mean = \sqrt[K]{\prod_{c=1}^{K} R_c}, \tag{19}$$

where R_c is the recall for class c.

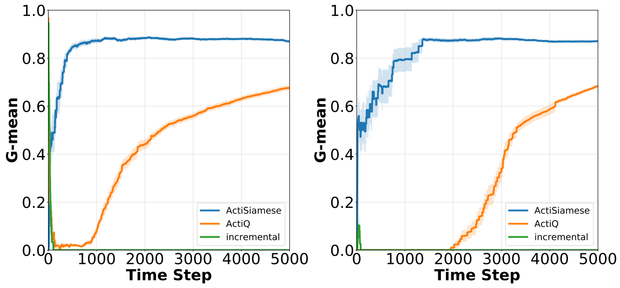
D. Evaluation method

To evaluate predictive sequential learning algorithms, we adopt the *prequential error with fading factors* method. It has been proven that for learning algorithms in stationary data this method converges to the Bayes error [23]. This method does not require a holdout set and the predictive model is always tested on unseen data. We have set the fading factor to $\theta = 0.99$. In all simulations we plot the prequential *G-mean* in every step averaged over 30 repetitions, including the error bars displaying the standard error around the mean.



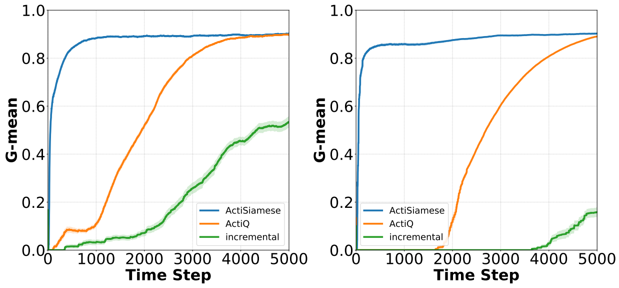
(a) *sea4* (b) *circles10*

Fig. 4. Role of the budget in the final performance ($E = 5$)



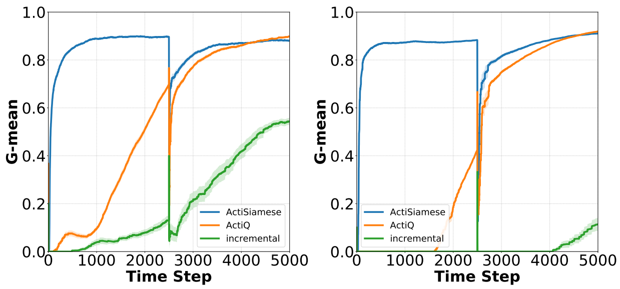
(a) *sea4* (b) *circles10*

Fig. 6. Comparative study in imbalanced settings ($B = 0.05$)



(a) *sea4* (b) *circles10*

Fig. 5. Comparative study in stationary settings ($B = 0.05$)



(a) *sea4* (b) *circles10*

Fig. 7. Comparative study in nonstationary settings ($B = 0.05$)

VI. Experimental Results

A. *Role of the budget*

The first experimental series examines how various budget values affect the *final performance* (i.e. the prequential $G\text{-}mean$ at the final time step of a simulation). Figs 4a and 4b show the results for *sea4* and *circles10* respectively. SL refers to fully supervised online learning. The general trend is that as the budget is reduced, the final performance declines. The difference lies in how rapidly or slowly this decline occurs.

When the budget is $B = 0.01$, the *ActiQ* and *incremental* obtain a score of $G\text{-}mean = 0$. The *ActiSiamese* approach significantly outperforms the rest. In fact, by having access to 1% of the labels, the proposed approach sacrifices only 0.5% of its performance. We consider this to be a significant advantage of the proposed approach as it can potentially enable the realistic deployment of an online classifier.

For greater values of budget ($B > 0.01$) the *ActiSiamese* and *ActiQ* obtain almost identical final performance scores. As we will discuss in the next section, however, their learning speed is significantly different. The *incremental* approach almost always performs significantly worse. We do acknowledge, of course, that this approach does not store or use any older examples, contrary to the other two.

B. *Stationary data*

The previous experiments only consider the algorithms' final performance. This section examines another important characteristic, that is, their *learning speed*. We focus on the interesting case of $B = 0.05$ as *ActiQ* and *ActiSiamese* appear to achieve a similar final performance. Figs. 5a and 5b show a comparison for $B = 0.05$ in the *sea4* and *circles10* dataset respectively. Despite the fact that the two approaches obtain a similar final G-mean score, it can be observed that the *ActiQ* requires about 4000 and 5000 time steps respectively to equalise the *ActiSiamese*'s performance. This is another major advantage of the proposed learning approach since in *online* learning, speed is a crucial performance measure. If a proposed solution is slow, it may be practically useless even if it eventually achieves the correct result.

C. *Imbalanced data*

Fig. 6a depicts the learning curves for imbalanced data for *sea4*. The *ActiSiamese* approach significantly outperforms the rest as it is robust to imbalance. The *ActiQ*'s performance is severely affected by the imbalance. The *incremental* obtains a score of $G\text{-}mean = 0$ as it fails to do well on the minority classes, and hence it is not visible in the figure. This is similar for *circles10* in Fig. 6b where the *ActiSiamese* is only hindered until about $t = 1300$. In both figures, the *ActiQ*'s final performance is significantly worse even after 5000 time steps, something that wasn't the case in Figs. 5a and 5b.

D. *Nonstationary data*

Fig. 7a depicts the performance in nonstationary data for *sea4*, specifically, when drift occurs abruptly at $t = 2500$. The *ActiSiamese* approach is unable to fully recover, however, it

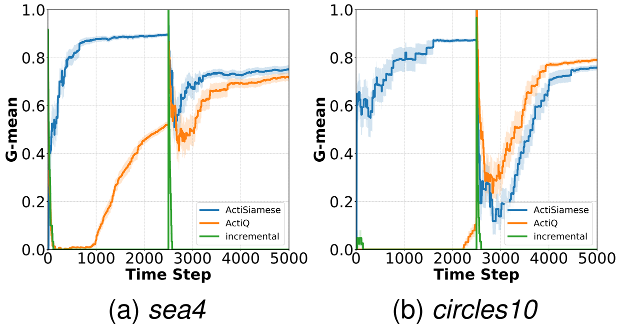


Fig. 8. Comparative study in imbalanced & nonstationary settings ($B = 0.05$)

does fully recover in Fig. 7b in the *circles10* dataset. Interestingly, the proposed *ActiQ* approach slightly outperforms the *ActiSiamese* by time $t = 5000$. This preliminary study reveals that there may be evidence to suggest that *ActiSiamese* has a more difficult time to 'forget' old concepts than *ActiQ*.

E. Imbalanced and nonstationary data

Fig. 8a depicts the performance when data is both imbalanced and nonstationary for the *sea4* dataset. After the concept drift, the *ActiSiamese* approach cannot fully recover from it but performs better than *ActiQ*. In Fig. 8b, the *ActiQ* performs better than the *ActiSiamese* after the drift. The *ActiSiamese*'s poor performance under these conditions is attributed to its inability to fully recover from the drift, thus reinforcing our previous finding that *ActiSiamese* may have a more difficult time to 'forget' old concepts. Moreover, these results indicate that when imbalance and drift co-exist and are both severe, this still remains an open and challenging problem.

VII. Conclusion and Future Work

We have proposed an online learning approach that combines active learning and siamese networks to address the challenges of limited labelled, nonstationary and imbalanced data. The proposed approach significantly outperforms strong baselines and state-of-the-art algorithms in terms of both learning speed and performance and has been shown to be effective even when only 1% of the labels of the arriving instances is available, something which is not unrealistic in deployed settings. For future work, we will enrich our study with real-world datasets. The problem of learning from nonstationary and imbalanced data still remains open. We have shown that when imbalance and drift co-exist and are both severe, all compared algorithms are severely affected. We plan to make the proposed algorithms more robust to these conditions.

Acknowledgment

This work has been supported by the European Union's Horizon 2020 research and innovation programme under grant agreements No. 867433 (FAULT-LEARNING) and No. 739551 (KIOS CoE), and from the Republic of Cyprus through the Directorate General for European Programmes, Coordination and Development.

References

[1] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.

[2] D. Sculley, M. E. Otey, M. Pohl, B. Spitznagel, J. Hainsworth, and Y. Zhou, "Detecting adversarial advertisements in the wild," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 274–282.

[3] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learning Workshop*, vol. 2, 2015.

[4] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[5] S. Wang, L. L. Minku, and X. Yao, "A systematic study of online class imbalance learning with concept drift," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4802–4821, 2018.

[6] I. Žliobaitė, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with drifting streaming data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 27–39, 2013.

[7] L. L. Minku, A. P. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 5, pp. 730–742, 2010.

[8] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, no. 9, pp. 1263–1284, 2008.

[9] K. Malialis, C. G. Panayiotou, and M. M. Polycarpou, "Queue-based resampling for online class imbalance learning," in *International Conference on Artificial Neural Networks (ICANN)*. Springer, 2018, pp. 498–507.

[10] S. Wang, L. L. Minku, and X. Yao, "Resampling-based ensemble methods for online class imbalance learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1356–1368, 2015.

[11] J. Wang, P. Zhao, and S. C. H. Hoi, "Cost-sensitive online classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 10, pp. 2425–2438, 2014.

[12] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Machine Learning*, vol. 15, no. 2, pp. 201–221, 1994.

[13] X. Zhu, P. Zhang, X. Lin, and Y. Shi, "Active learning from data streams," in *Seventh IEEE International Conference on Data Mining (ICDM)*. IEEE, 2007, pp. 757–762.

[14] B. Settles and M. Craven, "An analysis of active learning strategies for sequence labeling tasks," in *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2008, pp. 1070–1079.

[15] R. Capo, K. B. Dyer, and R. Polikar, "Active learning in nonstationary environments," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2013, pp. 1–8.

[16] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 539–546.

[17] W. N. Street and Y. S. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *Proceedings of the seventh ACM SIGKDD International —Conference on Knowledge Discovery and Data Mining*. ACM, 2001, pp. 377–382.

[18] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Brazilian Symposium on Artificial Intelligence*. Springer, 2004, pp. 286–295.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[20] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," *arXiv preprint arXiv:1908.03265*, 2019.

[21] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[22] Y. Sun, M. S. Kamel, and Y. Wang, "Boosting for learning multiple classes with imbalanced class distribution," in *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 2006, pp. 592–602.

[23] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Machine Learning*, vol. 90, no. 3, pp. 317–346, 2013.