A Simple Unlearning Framework for Online Learning under Concept Drifts

Sheng-Chi You and Hsuan-Tien Lin

Department of Computer Science and Information Engineering,
National Taiwan University, No.1, Sec. 4, Roosevelt Road, Taipei, Taiwan
{r02922068,htlin}@csie.ntu.edu.tw

Abstract. Real-world online learning applications often face data coming from changing target functions or distributions. Such changes, called the concept drift, degrade the performance of traditional online learning algorithms. Thus, many existing works focus on detecting concept drift based on statistical evidence. Other works use sliding window or similar mechanisms to select the data that closely reflect current concept. Nevertheless, few works study how the detection and selection techniques can be combined to improve the learning performance. We propose a novel framework on top of existing online learning algorithms to improve the learning performance under concept drifts. The framework detects the possible concept drift by checking whether forgetting some older data may be helpful, and then conduct forgetting through a step called unlearning. The framework effectively results in a dynamic sliding window that selects some data flexibly for different kinds of concept drifts. We design concrete approaches from the framework based on three popular online learning algorithms. Empirical results show that the framework consistently improves those algorithms on ten synthetic data sets and two real-world data sets.

Keywords: online learning, concept drift

1 Introduction

Online learning is a machine learning setup where the learning algorithm needs to learn from and make predictions on streaming data efficiently and effectively [3, 4, 9]. The setup enjoys many potential applications, such as predicting the weather, customer preferences, or stock prices [16].

Traditional online learning algorithms, such as the passive-aggressive algorithm (PA) [3], the confidence weighted algorithm (CW) [4] and the adaptive regularization of weight algorithm (AROW) [9], are designed under the assumption that the target function to be learned is fixed. In many applications, however, change in the underlying environment can result in change of the target function (concept) as time goes by. That is, the concept can be drifting [17] instead of fixed. For example, the best popular-cloth predictor (concept) is consistently drifting as the fashion trend evolves [10]. The drifting concept possesses

difficulty for traditional online learning algorithms and are studied by two families of works. One family of works focuses on the detection of concept drift from the data stream [5, 1, 12, 15]. Those works generally conduct statistical analysis on the data distribution and set up an alert threshold to reliably detect concept drift. The other family tries to construct learning models from selected instances of the data stream, with the hope that such instances match the drifting concept better [13, 2]. The simplest approach of this family is to use a sliding window to capture the newest instances for learning [13]. While the two kinds both deal with concept-drifting data, it is not fully clear on how they could be combined to improve the learning performance and will be the main focus of this work.

In particular, we propose a framework on top of existing online learning algorithms to improve the learning performance under concept drifts. The framework detects the possible concept drift by checking whether forgetting some older data may be helpful, where the detection is motivated by the confidence terms used in modern online learning algorithms. Then, it conducts forgetting by unlearning older data from the current model. By greedily repeating the detection and unlearning steps along with online learning, the framework effectively results in a dynamic sliding window that can suit different concept drifts. We design concrete approaches of the framework based on PA [3], AROW [9] and CW [4]. Our empirical results demonstrate that the framework can reach better accuracy on artificial and real-world data. The results justify the usefulness of the framework.

The paper is organized as follows. Section 2 establishes the setup and lists related online learning algorithms. Section 3 introduces the proposed framework. Section 4 discusses the experimental results and Section 5 concludes the paper.

2 Preliminaries

In this paper, we consider the online learning problem for binary classification. In each round of this problem, the learning algorithm observes a coming instance and predicts its label to be $+1$ or -1. After the prediction, the true label is revealed and the algorithm can then take the new instance-label pair to improve its internal prediction model. The goal of the algorithm is to make as few prediction errors as possible.

We shall denote the instance-label pair in round t as $(\mathbf{x}_t, y_t)$, where $t \in \{1, 2, \cdots, T\}$. Each $\mathbf{x}_t \in \mathbb{R}^n$ represents the instance (feature vector) and $y_t \in \{+1, -1\}$ indicates the label. The prediction in the t-th round is denoted as $\hat{y}_t$, and the error refers to the zero-one loss $\ell_{01}(y_t, \hat{y}_t)$, which is 1 if and only if $y_t \neq \hat{y}_t$, and 0 otherwise.

In this work, we consider the linear model for online learning, where some linear weight vector $\mathbf{w}_t \in \mathbb{R}^n$ is maintained within round t and $\hat{y}_t = \text{sign}(\mathbf{w}_t \cdot \mathbf{x}_t)$ with $\cdot$ denoting an inner product. The linear model generally enjoys efficiency in online learning and is often the focus of study in many online learning works [14, 3, 4, 9]. For the linear model, improving would then mean updating from $\mathbf{w}_t$ to $\mathbf{w}_{t+1}$, and we denote the difference as $\Delta\mathbf{w}_t = \mathbf{w}_{t+1} - \mathbf{w}_t$. The core of different online learning algorithms is then to design a proper update function

Algorithm 1 the linear model for online learning

1: initialize $\mathbf{w}_1 \leftarrow (0, 0, ..., 0)$
2: **for** $t = 1$ **to** T **do**
3: receive instance $\mathbf{x}_t \in R^n$ and predict $\hat{y}_t \leftarrow \text{sign}(\mathbf{w}_t \cdot \mathbf{x}_t)$
4: receive label: $y_t \in \{-1, +1\}$
5: $\Delta\mathbf{w}_t \leftarrow \textsc{Update}(\mathbf{w}_t, \mathbf{x}_t, y_t)$ and $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \Delta\mathbf{w}_t$
6: **end for**

$\textsc{Update}(\mathbf{w}_t, \mathbf{x}_t, y_t)$ that calculates $\Delta\mathbf{w}_t$. The details steps of the linear model for online learning is shown in Algorithm 1, where we assume $\mathbf{w}_1$ to be the zero vector for simplicity.

One of the most popular algorithms for online learning with the linear model is the Passive-Aggressive algorithm (PA) [3]. PA calculates the signed margin of the labeled instance by $y_t(\mathbf{w}_t \cdot \mathbf{x}_t)$, which indicates how confident the prediction $\hat{y}_t = \text{sign}(\mathbf{w}_t \cdot \mathbf{x}_t)$ was. PA then aims to adjust the weights $\mathbf{w}_t$ to the closest $\mathbf{w}_{t+1}$ (passive) in terms of the Euclidean distance, such that the hinge loss $\ell_h(\mathbf{w}; (y_t, \mathbf{x}_t)) = \max(0, 1 - y_t(\mathbf{w} \cdot \mathbf{x}_t))$ is decreased to $\ell_h(\mathbf{w}_{t+1}; (y_t, \mathbf{x}_t)) = 0$ (aggressive). The aim leads to the following $\textsc{Update}(\mathbf{w}_t, \mathbf{x}_t, y_t)$ for PA:

$$\Delta\mathbf{w}_t = \frac{\ell_h\big(\mathbf{w}_t; (y_t, \mathbf{x}_t)\big)}{\|\mathbf{x}_t\|^2} y_t \mathbf{x}_t. \tag{1}$$

The Confidence weighted (CW) algorithm [4] is extended from PA. Instead of considering a single weight vector $\mathbf{w}_t$, the algorithm considers the weight *distribution*, modeled as a Gaussian distribution with mean $\mathbf{w}_t$ and covariance Σ_t. During each $\textsc{Update}$ for CW, both $\mathbf{w}_t$ and Σ_t are taken into account, and updated to $\mathbf{w}_{t+1}$ and Σ_{t+1}. The updating step adjusts $(\mathbf{w}_t, \Sigma_t)$ to the closest $(\mathbf{w}_{t+1}, \Sigma_{t+1})$ (passive) in terms of the KL divergence, such that the probabilistic zero-one loss under the new Gaussian distribution is smaller than some $(1 - \eta)$ (aggressive).

An extension of CW is called adaptive regularization of weight (AROW) [9], which improves CW by including more regularization. In particular, the updating step of AROW solves an *unconstrained* optimization problem that calculates $(\mathbf{w}_{t+1}, \Sigma_{t+1})$ by

$$\underset{\mathbf{w}, \Sigma}{\text{argmin}}\, \mathrm{D}_{\mathrm{KL}}(\mathcal{N}(\mathbf{w}, \Sigma) || \mathcal{N}(\mathbf{w}_t, \Sigma_t)) + \lambda_1 \ell_h^2(\mathbf{w}; (y_t, \mathbf{x}_t)) + \lambda_2 \mathbf{x}_t^T \Sigma \mathbf{x}_t. \tag{2}$$

The first term is exactly the KL divergence that the passive part of CW considers; the second term embeds the aggressive part of CW with the squared hinge loss (similar to PA); the third term represents the confidence on $\mathbf{x}_t$ that should generally grow as more instances have been observed. In particular, the confidence term represents how different $\mathbf{x}_t$ is from the current estimate of Σ. The confidence term acts as a regularization term to make the learning algorithm more robust. In this work, we set the parameters λ_1 and λ_2 by $\lambda_1 = \lambda_2 = 1/(2\gamma)$ as the original paper suggests [9].

One special property of the three algorithms above, which is also shared by many algorithms for the linear model of online learning, is that $\Delta\mathbf{w}_t$ is a scaled version of $y_t\mathbf{x}_t$, as can be seen in (1) for PA. Then, by having $\mathbf{w}_1$ as the zero vector, each $\mathbf{w}_t$ is simply a *linear combination* of the previous data $y_1\mathbf{x}_1, y_2\mathbf{x}_2, \cdots, y_{t-1}\mathbf{x}_{t-1}$. We will use this property later for designing our framework.

The three representative algorithms introduced above do not specifically focus on concept-drifting data. For example, when concept drift happens, being passive like the algorithms do may easily lead to slow adaptation to the latest concept. Next, we illustrate more on what we mean by concept drift in online learning. [16] defines concept drift to mean the change of "property" within the data. Some major types of concept drifts that will be considered here are abrupt concept drift, gradual concept drift and virtual concept drift. The first two entail the change of the relation between instances and labels. Denote the relation as the ideal target function f such that $y_t = f(\mathbf{x}_t) + \text{noise}$, *abrupt concept drift* means that the ideal target function can change from f to a very different one like $(-f)$ at some round t_1, and *gradual concept drift* means f is slowly changed to some different f' between rounds t_1 and t_2.

Virtual concept drift, unlike the other two, is generally a consequence of the change of some hidden context within the data [6]. The change effectively causes the distribution of $\mathbf{x}_t$ to vary. While the target function that characterizes the relation between $\mathbf{x}_t$ and y_t may stay the same for virtual concept drift, the change of distribution places different importance on different parts of the feature space for the algorithm to digest.

Two families of methods in the literature focus on dealing with concept-drifting data for online learning. One family [5, 1, 12, 15] is about drift detection based on different statistical property of the data. [5] proposes the drift detection method (DDM) that tracks the trend of the zero-one loss to calculate the drift level. When the drift level reaches an alert threshold, the method claims to detect the concept drift and resets the internal model. While the idea of DDM is simple, it generally cannot detect gradual concept drift effectively. [1] thus proposes the early drift detection method (EDDM) to cope with gradual concept drift, where the distribution of errors instead of the trend is estimated for detection. Some other popular detection criteria include the estimated accuracy difference between an all-data model and a recent-data model [12], and the estimated performance difference between models built from different chunks of data [15]. Generally, similar to [5], after detecting the concept drift, the methods above reset the internal model. That is, all knowledge about the data received before detection are effectively forgotten. Nevertheless, forgetting all data before the detection may not be the best strategy for gradual concept drift (where the earlier data may be somewhat helpful) and virtual concept drift (where the earlier data still hint the target function).

The other family [13, 2] makes the internal model adaptive to the concept drift by training the model with selected instances only. The selected instances are often within a sliding window, which matches the fact that the latest instances

should best reflect the current concept. Most of the state-of-the-art methods consider dynamic sliding windows. For instance, [13] takes the leave-one-out error estimate of the support vector machine to design a method that computes the best dynamic sliding window for minimizing the leave-one-out error. [2] proposes a general dynamic sliding window method by maintaining a sliding window such that the "head" and "tail" sub-windows are of little statistical difference. The sliding-window methods naturally trace concept drifts well, especially gradual concept drifts. Nevertheless, calculating a good dynamic sliding window is often computationally intensive. It is thus difficult to apply the methods within this family to real-world online learning scenario where efficiency is highly demanded.

In summary, drift-detection methods are usually simple and efficient, but resetting the internal model may not lead to the best learning performance under concept drifts; sliding-windows methods are usually effective, but are at the expense of computation. We aim to design a different framework for better online learning performance under the concept drift. Our framework will include a simple detection scheme and directly exploits the detection scheme to efficiently determine a dynamic sliding window. In addition, the framework can be flexibly coupled with existing online learning algorithms with linear models.

3 Unlearning Framework

The idea of our proposed unlearning framework is simple. Between steps 5 and 6 of Algorithm 1, we add a procedure UnlearningTest to check if forgetting some older instance can be beneficial for learning. In particular, the decision of "beneficial" is done by comparing a regularized objective function before and after the forgetting, where the regularized objective function mimics that being used by AROW. If forgetting is beneficial, a new $\mathbf{w}'_{t+1}$ (and its accompanying Σ'_{t+1} in the case of CW or AROW) replaces the original $\mathbf{w}_{t+1}$. There are then two issues in describing the framework concretely: what the regularized objective function and unlearning step are, and which "older" instance to check? We will clarify the issues in the next subsections.

3.1 Unlearning Test

Denote $(\mathbf{x}_k, y_k)$, $k \in \{1, 2, \cdots, t-1\}$ as the selected instance for UnlearningTest. Recall that in round t, each $\mathbf{w}_t$ is simply a linear combination of the previous data $y_1\mathbf{x}_1, y_2\mathbf{x}_2, \cdots, y_{t-1}\mathbf{x}_{t-1}$. That is, every old instance has its (possibly 0) footprint within $\mathbf{w}_{t+1}$ if we record $\Delta\mathbf{w}_k$ along with the online learning process. Then, one straightforward step to unlearn $(\mathbf{x}_k, y_k)$ is to remove it from $\mathbf{w}_{t+1}$. That is,

$$\mathbf{w}'_{t+1} \leftarrow \mathbf{w}_{t+1} - \Delta\mathbf{w}_k.$$

The Σ'_{t+1} accompanying $\mathbf{w}_{t+1}$ can also be calculated similarly by recording $\Delta\Sigma_k$ along with the online learning process.

Now, $\mathbf{w}'_{t+1}$ represents the weight vector after removing some older instance, and $\mathbf{w}_{t+1}$ represents the original weight vector. Our task is to pick the better one

for online learning with concept drift. A simple idea is to just compare their loss, such as the squared hinge loss used by AROW. That is, unlearning is conducted if and only if

$$\ell_h^2(\mathbf{w}'_{t+1}; (\mathbf{x}_t, y_t)) \leq \ell_h^2(\mathbf{w}_{t+1}; (\mathbf{x}_t, y_t)).$$

We can even make the condition more strict by inserting a parameter $\alpha \leq 1.0$ that controls the demanded reduction of loss from the original weight vector. That is, unlearning is conducted if and only if

$$\ell_h^2(\mathbf{w}'_{t+1}; (\mathbf{x}_t, y_t)) \leq \alpha \ell_h^2(\mathbf{w}_{t+1}; (\mathbf{x}_t, y_t)).$$

Then, $\alpha = 0.0$ makes unlearning happen only if $\mathbf{w}'_{t+1}$ is fully correct on $(\mathbf{x}_t, y_t)$ in terms of the hinge loss, and the original online learning algorithms are *as if* using $\alpha < 0$.

In our study, we find that only using ℓ_h^2 as the decision objective makes the unlearning procedure rather unstable. Motivated by AROW, we thus decide to add two terms to the decision objective. One is the confidence term used by AROW, and the other is the usual squared length of $\mathbf{w}$. The first term regularizes against unwanted update of Σ, much like AROW does. The second term regularizes against unwanted update of $\mathbf{w}$ to a long vector, much like the usual ridge regression regularization. That is, given $(\mathbf{x}_t, y_t)$, the framework considers

$$\text{obj}(\mathbf{w}, \Sigma) = \ell_h^2(\mathbf{w}; (\mathbf{x}_t, y_t)) + \beta \mathbf{x}_t^T \Sigma \mathbf{x}_t + \gamma \|\mathbf{w}\|^2 \tag{3}$$

and conduct unlearning if and only if $\text{obj}(\mathbf{w}'_{t+1}, \Sigma'_{t+1}) \leq \alpha \text{obj}(\mathbf{w}_{t+1}, \Sigma_{t+1})$. The parameters β and γ balances the influence of each term.

The final missing component is how to specify β and γ. To avoid making the framework overly complicated, we only consider using those parameters to balance the numerical range of the terms. In particular, we let β be the average of

$$\frac{1}{2}\left(\frac{\ell_h^2(\mathbf{w}_{\tau+1}, \mathbf{x}_\tau, y_\tau)}{\mathbf{x}_\tau^T \Sigma_{\tau+1} \mathbf{x}_\tau} + \frac{\ell_h^2(\mathbf{w}'_{\tau+1}, \mathbf{x}_\tau, y_\tau)}{\mathbf{x}_\tau^T \Sigma'_{\tau+1} \mathbf{x}_\tau}\right). \tag{4}$$

for $\tau \in \{1, 2, \ldots, t\}$ so $\beta \mathbf{x}_t^T \Sigma \mathbf{x}_t$ can be of a similar numerical range to ℓ_h^2. Similarly, we let γ be the average of

$$\frac{1}{2}\left(\frac{\ell_h^2(\mathbf{w}_{\tau+1}, \mathbf{x}_\tau, y_\tau)}{\|\mathbf{w}_{\tau+1}\|^2} + \frac{\ell_h^2(\mathbf{w}'_{\tau+1}, \mathbf{x}_\tau, y_\tau)}{\|\mathbf{w}'_{\tau+1}\|^2}\right). \tag{5}$$

The details of UnlearningTest is listed in Algorithm 2.

3.2 Instance for Unlearning Test

Unlearning is completed by the unlearning test at a certain selected instance $(\mathbf{x}_k, y_k)$. But how to determine the k from all previous processed instances? We proposed three possible unlearning strategies to deciding the instance $(\mathbf{x}_k, y_k)$.

Algorithm 2 Unlearning test for some instance $(\mathbf{x}_k, y_k)$

input parameter: $\alpha \in [0.0, 1.0]$
procedure UNLEARNINGTEST($\mathbf{w}_{t+1}, \Sigma_{t+1}, \mathbf{x}_k, y_k$)
    $\Delta\mathbf{w}_k$, $\Delta\Sigma_k \leftarrow$ UPDATEHISTORY($\mathbf{x}_k, y_k$) ▷ previous updated status on $(\mathbf{x}_k, y_k)$
    $\mathbf{w}'_{t+1} \leftarrow \mathbf{w}_{t+1} - \Delta\mathbf{w}_k$, $\Sigma'_{t+1} \leftarrow \Sigma_{t+1} - \Delta\Sigma_k$
    set β, γ as the average of (4) and (5), respectively
    if obj($\mathbf{w}'_{t+1}, \Sigma'_{t+1}$) $\leq \alpha$obj($\mathbf{w}_{t+1}, \Sigma_{t+1}$) **then** ▷ see (3)
        return $\mathbf{w}'_{t+1}$, Σ'_{t+1}
    else
        return $\mathbf{w}_{t+1}$, Σ_{t+1}
    end if
end procedure

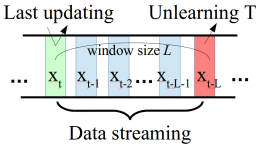

Fig. 1: Forwarding

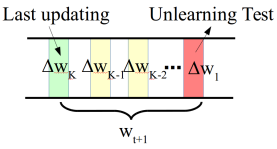

Fig. 2: Queue

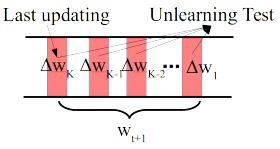

Fig. 3: Selecting

Forwarding-removing: Traditional sliding window technique tries to maintain a window that keeps the recent accessed examples, and drops the oldest instance according to some set of rules [2]. Here, the unlearning test is substituted for the rules. Forward-removing considers $(\mathbf{x}_{t-L}, y_{t-L})$ subject to a fixed window size L as the as the selected instance for unlearning test. The strategy is illustrated by Fig. 1, where the older instances are at the right-hand-side of the data stream. After updating on $\mathbf{x}_t$ is done, the unlearning test examines the red instance $\mathbf{x}_{t-L}$.

With some studies on parameter $L = \{1, 10, 100, 1000\}$, $L = 100$ is sufficiently stable and will be used to demonstrate this strategy in Section 4.

Queue-removing: Instead of considering the instance that is L rounds away, this strategy selects the oldest one within the current model $\mathbf{w}_{t+1}$. Recall that the current model $\mathbf{w}_{t+1}$ is a combination of some updated parts $\Delta\mathbf{w}_i$ on previous updated instance $(\mathbf{x}_i, y_i)$. We record those $\Delta\mathbf{w}_i$ like a data list, as illustrated in Fig. 2.

$$\mathbf{w}_{t+1} = \sum_{i=1}^{K} \Delta\mathbf{w}_i = \sum_{i=1}^{K} \tau_i \mathbf{x}_i y_i \quad \text{where } \tau_i \neq 0. \tag{6}$$

Take $\mathbf{w}_{t+1}$ as a queue, unlearning test will be executed at the red updated part $\Delta\mathbf{w}_1$, which is the oldest updated instance in model. As $(\mathbf{x}_i, y_i)$ are added and removed from $\mathbf{w}_t$, the size of the queue can change dynamically, resulting in a dynamic sliding window effectively.

Table 1: The properties of the ten data sets

Data set	Properties		
	Features	Drift type	Drifting details
SINE1	2 real	Abrupt	Reversed wave: $y = sin(x)$
SINE2	2 real	Abrupt	Reversed wave: $0.5 + 0.3sin(3\pi x)$
SINIRREL1	2 real + 2 irrelevant	Abrupt	Same as SINE1 function
SINIRREL2	2 real + 2 irrelevant	Abrupt	Same as SINE2 function
MIXED	2 real + 2 boolean	Abrupt	Reversed 1 function with 1 boolean condition
STAGGER	3 boolean	Abrupt	Switching between 3 boolean conditions
GAUSS	2 real	Virtual	Switching between 2 distributions
CIRCLES	2 real	Gradual	Switching between 4 circles [5]
LINES	2 real	Gradual	changing line functions: shift and rotate
MULTILINES	4 - 15 real	Gradual	changing hyperplanes: $\Sigma_i^d \mathbf{w_i x_i} = w_0$ [8]

Selecting-removing: Above strategies both select one particular instance under different structure. However, those strategies neither consider all candidates in their window nor find out the best unlearned weight $\mathbf{w}'_{t+1}$ for current instance $(\mathbf{x}_t, y_t)$. Illustrated by Fig. 3, Selecting-removing will test all K instances and take the instance that can decrease obj the most as the instance to be unlearned.

4 Empirical Evaluation

We take these three unlearning strategies in Section 3 with PA [3], AROW [9] and CW [4]. In those algorithms, we set $a = 1.0$, $\phi = 0.0001$ in CW and $r = 0.1$ in AROW. The parameter α in unlearning test is individually selected from $\{0.1, 0.2, \ldots, 0.9\}$ due to the different properties on these algorithms.

All ten synthetic data sets contain different concept drifts described in Table 1. The first eight data sets are used by [5]. Because most of them are about abrupt concept drift, we construct two more data sets, LINES and MULTILINES, whose drifting type is gradual. The target function of LINES is changed by shifting and rotating gradually in 2D, and MULTILINES is a d dimensional version defined in [8].

Previous works [1, 5] assume every concept contains a fixed number of instances, and examine on small size data sets. Here we construct these artificial data sets with three differences to make the data sets more realistic. First, the number and the timing of concept drifts are randomly assigned and all drift events are recorded so that we could simulate a perfect drifting detection, Concept-removing, which resets $\mathbf{w}_{t+1}$ immediately after a concept drift happens. We take Concept-removing as an upper bound benchmark for using the ideal drifting information. Second, at least 1,000,000 instances are generated in each data set for the robustness. Finally, we inject noise made by flipping binary labels under different probabilities to check the robustness of the proposed

Table 2: Ranking all unlearning strategies under three types of drifting data

drifting type / strategy	Abrupt	Gradual	Virtual
None	4.031 ± 0.347	3.047 ± 0.402	3.333 ± 0.890
Forwarding-removing	4.325 ± 0.308	3.809 ± 0.471	5.476 ± 0.534
Queue-removing	$\mathbf{3.373 \pm 0.235}$	$\mathbf{2.984 \pm 0.387}$	$\mathbf{2.190 \pm 0.499}$
Selecting-removing	3.769 ± 0.254	3.666 ± 0.517	3.714 ± 0.730
EDDM	$\mathbf{3.309 \pm 0.264}$	3.174 ± 0.385	3.000 ± 0.427
Concept-removing	1.269 ± 0.138	3.809 ± 0.501	2.666 ± 0.930

framework. All artificial data sets are generated under different flipping level within $\{0.00, 0.05, \cdots, 0.30\}$.

For each data, a simple second-order polynomial transform is applied to improve the accuracy. Two evaluation criteria are considered, *ranking performance* and *cumulative classification accuracy*. A smaller average rank (along with standard deviation) indicates that an higher classification accuracy performed among compared methods.

4.1 Results and Discussion

In addition to the three proposed strategies within the framework, and the ideal Concept-removing strategy, we also compare the proposed framework with EDDM [1]. Our experimental results are summarized in following tables with different control variables. Table 2 compares all unlearning strategies under three kinds of concept-drifting data. Table 3 compares the relation between different unlearning strategies and each online learning algorithm individually. Table 4 evaluates the influence on the best unlearning strategy with different noise level. The individual accuracy performances for each data set are recorded in Table 5.

Table 2 makes comparison by different kinds of concept-drifting data. The ideal Concept-removing strategy performs very well for abrupt drifting and virtual drifting, as expected. But the immediate resetting cannot work for gradual drifting data, and the ideal detection is not realistic anyway. Our proposed framework, on the other hand, performs well on all kinds of data when using Queue-removing.

Table 3 is evaluated under individual learning algorithms. On the strategy side, Queue-removing preforms the best ranking on average in four unlearning strategies. Note that Selecting-removing is worse than Queue-removing, which indicates that overly searching for the "best" instance to unlearn is not necessary. On the algorithm sides, a significant ranking gap between Concept-removing and the others is presented in AROW. All four unlearning strategies show the smaller ranking than original AROW. For the other two algorithms, only Queue-removing and EDDM gets smaller ranking on PA. But almost unlearning approaches do not have great advantage in CW. The cause of non-improving is

Table 3: Ranking all unlearning strategies under each learning algorithm

strategy \ algorithm	PA	AROW	CW	Average
None	3.257 ± 0.325	5.642 ± 0.336	2.100 ± 0.215	3.666 ± 0.263
Forwarding-removing	5.128 ± 0.352	5.371 ± 0.239	2.357 ± 0.236	4.285 ± 0.245
Queue-removing	$\mathbf{3.100 \pm 0.348}$	3.485 ± 0.335	2.528 ± 0.284	$\mathbf{3.138 \pm 0.195}$
Selecting-removing	4.228 ± 0.352	4.457 ± 0.411	2.514 ± 0.235	3.733 ± 0.228
EDDM	3.342 ± 0.394	$\mathbf{2.200 \pm 0.152}$	4.171 ± 0.279	3.238 ± 0.200
Concept-removing	2.242 ± 0.467	1.242 ± 0.140	3.028 ± 0.483	2.171 ± 0.248

Table 4: Ranking three main unlearning strategies under different bias data sets

Unlearning strategy	Noise level					
	0.05	0.10	0.15	0.20	0.25	0.30
None	2.22 ± 0.17	2.16 ± 0.16	2.11 ± 0.17	2.18 ± 0.16	2.06 ± 0.16	2.02 ± 0.16
Queue-removing	1.90 ± 0.08	1.97 ± 0.11	1.97 ± 0.11	1.97 ± 0.11	2.01 ± 0.12	1.97 ± 0.14
Concept-removing	1.28 ± 0.12	1.38 ± 0.14	1.36 ± 0.13	1.34 ± 0.13	1.40 ± 0.14	1.44 ± 0.14

their individual updating rules, which does not consider confidence term in PA and squared hinge-loss in CW.

We study Queue-removing more in Table 4, which shows the ranking performance under different noise levels. From lowest to highest bias, Concept-removing is still the best in three strategies but Queue-removing shows its effectiveness in all noise levels. When the noise becomes larger, Queue-removing is closer to the ideal Concept-removing strategy.

Table 5 explains whether unlearning framework reflects the significant difference from original algorithms. We conducted the t-test experiment by its cumulative classification accuracy at each data set 30 times for all artificial data and directly evaluated two real data, MNIST[1] and ELEC2[2]. For two real data, we directly compare the accuracy performance with EDDM and Queue-removing.

The t-test is evaluated in three different strategies. Queue-removing shows better accuracy than no-unlearning, and those p-value(N-Q) are mostly smaller than 0.01, which indicates the performance gap is significant enough. Concept-removing reveal the upper bound accuracy and the nearly 0 on p-value(Q-C) comparing with the Queue-removing in all data sets except for CIRCLES.

MNIST [11] is a handwritten digits data. Although it is not a concept-drifting data, we test whether our unlearning framework will deteriorate the classifying performance. We use one versus one to evaluate 45 binary classifications for those digits under online learning scenario. To handle all classifications quickly,

[1] handwritten digits: `http://yann.lecun.com/exdb/mnist`
[2] electricity price data: `http://www.inescporto.pt/~jgama/ales/ales.html`

Table 5: Cumulative accuracy and t-test on ten artificial and two real-world data

Properties	Average accuracy among three algorithms			P-value	
Strategy	None	Queue-removing	Concept-removing	N-Q	Q-C
SINE1	0.6696 ± 0.0232	$\mathbf{0.6816 \pm 0.0244}$	0.7541 ± 0.0267	0.0352	0.0000
SINE2	0.6373 ± 0.0161	$\mathbf{0.6422 \pm 0.0154}$	0.6984 ± 0.0166	0.0117	0.0000
SINIRREL1	0.6819 ± 0.0212	$\mathbf{0.7202 \pm 0.0199}$	0.7687 ± 0.0215	0.0000	0.0000
SINIRREL2	0.6395 ± 0.0181	$\mathbf{0.6660 \pm 0.0175}$	0.7071 ± 0.0169	0.0000	0.0000
MIXED	0.6792 ± 0.0220	$\mathbf{0.6938 \pm 0.0214}$	0.7469 ± 0.0211	0.0011	0.0000
STAGGER	0.7476 ± 0.0219	$\mathbf{0.7517 \pm 0.0216}$	0.7996 ± 0.0223	0.0001	0.0000
GAUSS	0.6452 ± 0.0189	$\mathbf{0.6676 \pm 0.0188}$	0.6871 ± 0.0182	0.0001	0.0000
CIRCLES	0.7179 ± 0.0194	$\mathbf{0.7262 \pm 0.0208}$	0.7244 ± 0.0217	0.0000	0.5950
LINES	0.7557 ± 0.0244	$\mathbf{0.7783 \pm 0.0246}$	0.7970 ± 0.0230	0.0002	0.0002
MULTILINES	0.7687 ± 0.0222	0.7566 ± 0.0240	0.7865 ± 0.0239	0.0002	0.0000
Strategy	None	Queue-removing	EDDM	N-Q	Q-C
MNIST	0.9774 ± 0.0032	0.9774 ± 0.0032	0.9758 ± 0.0033	NA	NA
ELEC2	0.8423 ± 0.0765	$\mathbf{0.8742 \pm 0.0575}$	0.8342 ± 0.1312	NA	NA

we scale each image by 25% and take its pixel as feature. Because MNIST data does not contain significant drifting and the nearly same accuracies are presented, it implies our unlearning framework can work well in the normal data set.

ELEC2 [7] is the collection of the electricity price. Those prices are affected by demand and supply of the market, and the labels identify the changing prices related to a moving average. It is a widely used for concept-drifting. We predict the current price rises or falls by its all 8 features. The result shows that Queue-removing preforms better than no-unlearning and EDDM.

5 Conclusion

We present an unlearning framework on top of PA-based online algorithms to improve the learning performance under different kinds of concept-drifting data. This framework is simple yet effective. In particular, the queue-removing strategy, which is the best-performing one, results in a dynamic sliding window on the *mistaken* data and dynamically unlearns based on a simple unlearning test as the drift detection. Future work includes more sophisticated ways to balance between loss and regularization for the unlearning test.

6 Acknowledgment

The work arises from the Master's thesis of the first author [18]. We thank Profs. Yuh-Jye Lee, Shou-De Lin, the anonymous reviewers and the members of the NTU Computational Learning Lab for valuable suggestions. This work is partially supported by the Ministry of Science and Technology of Taiwan

(MOST 103-2221-E-002-148-MY3) and the Asian Office of Aerospace Research and Development (AOARD FA2386-15-1-4012).

References

1. Baena-García, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavaldà, R., Morales-Bueno, R.: Early drift detection method. International Workshop on Knowledge Discovery from Data Streams pp. 77–86 (2006)
2. Bifet, A., Gavalda, R.: Learning from time-changing data with adaptive windowing. SIAM International Conference on Data Mining pp. 443–448 (2007)
3. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. Journal of Machine Learning Research 7, 551–585 (2006)
4. Dredze, M., Crammer, K., Pereira, F.: Confidence-weighted linear classification. Proceedings of the 25th International Conference on Machine Learning pp. 264–271 (2008)
5. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. Advances in Artificial Intelligence–SBIA 2004 3171, 286–295 (2004)
6. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Computing Surveys (CSUR) 46, 44:1–44:37 (2014)
7. Harries, M., Wales, N.S.: Splice-2 comparative evaluation: Electricity pricing (1999)
8. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining pp. 97–106 (2001)
9. Koby Crammer, Alex Kulesza, M.D.: Adaptive regularization of weight vectors. Machine Learning 91, 155–187 (2013)
10. Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: An ensemble method for drifting concepts. Journal of Machine Learning Research 8, 2755–2790 (2007)
11. LeCun, Y., Cortes, C.: Mnist handwritten digit database. AT&T Labs (2010)
12. Nishida, K., Yamauchi, K.: Detecting concept drift using statistical testing. Discovery Science 4755, 264–269 (2007)
13. Ralf, K., Joachims, T.: Detecting concept drift with support vector machines. Proceedings of the Seventeenth International Conference on Machine Learning pp. 487–494 (2000)
14. Shalev-Shwartz, S.: Online learning and online convex optimization. Foundations and Trends in Machine Learning 4, 107–194 (2012)
15. Sobhani, P., Beigy, H.: New drift detection method for data streams. Adaptive and Intelligent Systems 6943, 88–97 (2011)
16. Tsymbal, Alexey: The problem of concept drift: definitions and related work. Technical Report, Computer Science Department, Trinity College Dublin (2004)
17. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning 23, 69–101 (1996)
18. You, S.C.: Dynamic unlearning for online learning on concept-drifting data. Masters thesis, National Taiwan University (2015)