IEEE INFOCOM 2021 - IEEE Conference on Computer Communications | 978-1-6654-0325-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/INFOCOM42981.2021.9488890

Robust Online Learning against Malicious Manipulation with Application to Network Flow Classification

Yupeng Li*, Ben Liang*, Ali Tizghadam†
*Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada
†Technology Strategy and Business Transformation, TELUS Communications, Toronto, Canada
yupeng.li@utoronto.ca, liang@ece.utoronto.ca, ali.tizghadam@telus.com

***Abstract*—Malicious data manipulation reduces the effectiveness of machine learning techniques, which rely on accurate knowledge of the input data. Motivated by real-world applications in network flow classification, we address the problem of robust online learning with delayed feedback in the presence of malicious data generators that attempt to gain favorable classification outcome by manipulating the data features. We propose online algorithms termed `ROLC-NC` and `ROLC-C` when the malicious data generators are non-clairvoyant and clairvoyant, respectively. We derive regret bounds for both algorithms and show that they are sub-linear under mild conditions. We further evaluate the proposed algorithms in network flow classification via extensive experiments using real-world data traces. Our experimental results demonstrate that both algorithms can approach the performance of an optimal static offline classifier that is not under attack, while outperforming the same offline classifier when tested with a mixture of normal and manipulated data.**

I. Introduction

We consider the problem of robust online learning against malicious manipulation with delayed feedback. In this problem, each data sample belongs to a specific class, and the task of an online classifier is to estimate the classes of a sequence of data samples that arrive over time. Malicious data generators may exist, which can manipulate their data features to best respond to the classification model used by the classifier, to obtain some specific benefit while incurring some manipulation cost. Our objective is to decide a sequence of classification models over time, given some delayed feedback on the true class labels of the data samples, such that the classification accuracy is maximized, even in the presence of malicious data generators.

Many real-world applications motivate this problem, e.g., traffic flow classification in computer networking and credit card fraud in banking. We take network flow classification as an example, which is essential to modern network management [1]. The success of machine learning (ML) techniques for flow classification depends on having reliable knowledge of the flow feature values [1]–[3]. However, malicious network flow generators may manipulate their flow features to game the classifier, e.g., to be prioritized for network resource allocation [4]–[7], or to evade detection [8]. Such malicious behavior can render the conventional statistics-based methods ineffective.

To counter the attacks against ML systems, defensive strategies have been proposed, generally termed adversarial learning [9]. Most of them are offline strategies [10]. However, in many applications the training data are generated over time (e.g., traffic flows in computer networks), so online learning is often desirable [11]. Furthermore, since the malicious data generators can manipulate the data features to best respond to the latest classification model employed by the classifier, it is important to update the classification model over time to counter such malicious attacks [12], [13]. Finally, online learning can limit the amount of memory required to store the training data, and it allows dynamic adaptation to new malicious behavior patterns.

It is challenging to address the problem of robust online learning against malicious feature manipulation. First, online optimization implies that the algorithms make decision with no information about data in the future. Second, the feature manipulation by a malicious data generator may be the best response to the classification model, so that the feature presented might be a function of the model, which further complicates the design space. Third, the classifier depends on the feedback of the true labels to adjust the classification model to dynamically adapt to malicious behavior patterns as well as the normal ones. However, such feedback often is delayed, making learning difficult.

In this work, we propose two online classification algorithms for the cases where the malicious generators are *non-clairvoyant* and *clairvoyant*. A non-clairvoyant data generator is one that can observe the classification model committed by the classifier only after some delay, while, representing the worst-case challenge, a clairvoyant data generator is one that can observe the classification model as soon as it is changed. To evaluate the performance of these two algorithms, we analyze their *regret*. The regret is the cumulative cost difference between the considered algorithm and an offline oracle that chooses the best static classification model given all information of the future [14]. Specifically, the major contributions of our work are summarized as follows:

- We study the problem of robust online learning against malicious feature manipulation with delayed feedback.

This work has been funded by grants from TELUS and the Natural Sciences and Engineering Research Council (NSERC) of Canada.

We consider a detailed system model of online linear classification, which captures practical issues including the delay for the data generators to observe the classifier's classification model (denoted by ξ) and the delay for the classifier to receive the feedback of the true label (denoted by τ).

- We propose an robust online classification algorithm termed ROLC-NC when the malicious data generators are non-clairvoyant (i.e., $\xi > 0$) and another algorithm termed ROLC-C when the malicious data generators are clairvoyant (i.e., $\xi = 0$). ROLC-NC requires knowledge only of the observed features of arriving data samples (regardless whether they are manipulated) and the delayed feedback of true labels, while ROLC-C requires further knowledge of the cost function of the malicious data generators. We prove that both algorithms have a regret bound that is *sub-linear in time* under mild conditions, which implies that the largest possible difference between the *average* costs of the classifier and the offline oracle vanishes as the time horizon goes to infinity
- We consider the application of network flow classification and conduct extensive experiments with real-world data traces. Our results demonstrate the effectiveness of the proposed algorithms. For example, with 50% clairvoyant malicious flow generators and feedback delay $\tau = 10$, ROLC-NC can achieve 0.91 accuracy and F_1 score, while ROLC-C can achieve 0.93 accuracy and F_1 score, in steady state. Furthermore, under a wide range of experiment settings, our algorithms can approach the performance of an optimal static offline classifier that is not under attack (Offline-Norm), and they outperform the same offline classifier when tested with a mixture of normal and manipulated flows (Offline). Our evaluation also sheds light on how to choose an appropriate loss function and tune the parameters in the classification model.

The rest of the paper is organized as follows. We first discuss the related work in Sec. II. We propose the system model and formulate the robust online classification problem in Sec. III. Then, we develop ROLC-NC and ROLC-C and derive their regret bounds in Sec. IV. This is followed by performance evaluation with an application to robust online network flow classification based on real-world data traces in Sec.V. Finally, we make concluding remarks in Sec. VI.

II. Related Work

There are three main types of adversarial attacks in ML: data poisoning (DP), reverse engineering (RE), and test-time evasion (TTE) [15]–[20]. The malicious feature manipulation in our work belongs to the family of TTE attacks. Most TTE attacks seek to simply degrade a classifier's accuracy without additional benefit to the data generator. Correspondingly, robust classification aims to correctly classify the samples generated under TTE, which is usually achieved by modifying the training process, through, for example, feature obfuscation [21], [22] and robust training [23], [24]. Anomaly detection of the TTE attacks is also a common defense strategy [25]. All of the above works study offline strategies, while our work considers an online setting.

There is a paucity of prior art on online adversarial learning. Abramson [12] discussed, in general, the scenarios where an attacker seeks to evade a classifier or modify an online learning algorithm. This position paper did not present a formulated problem or a detailed solution approach. Barreno et al. [13] introduced a DP attack in online learning with an aim to alter the training process through influence over the training data on the fly. They did not propose any defense against such attacks. Kloft and Laskov [26] studied online centroid anomaly detection in the presence of DP attacks, where the goal of the considered attack was to force the learning model to accept a set of targeted data samples as normal. Different from [26], we consider a kind of TTE attack. Specifically, the malicious data generators, after observing the committed learning model, manipulates the data features aiming at increase the likelihood of certain classification outcome. Sethi and Kantardzic [27] proposed an unsupervised drift detection approach against TTE through adversarial drift in streaming data. The main idea of their approach was to detect the drift according to the disagreements between two orthogonal classifiers, each trained on a disjoint subset of the features of the data. Different from [27], our proposed methods require no detection of malicious data before classifying the data sample. Furthermore, in addition to the differences in problem setting, our proposed algorithms have provable performance guarantee in terms of their regret bounds, which is not available in [26] or [27].

There is another line of research using the word "adversarial" to mean an internal non-malicious mechanism to challenge and improve the design of a learning system, e.g., Generative Adversarial Networks (GANs) [28] and other works related to online learning [29], [30]. These works have no relevance to our work.

III. System Model and Problem Formulation

In this section, we present the system model and formulate the problem of robust online learning against malicious feature manipulation with delayed feedback.

A. Online Classification Model

We consider a learning model where the data samples each has M features, represented by $\mathbf{x} \in \mathbb{R}^M$, and each sample has a truthful class $k \in \mathcal{K} = \{1, 2, \cdots, K\}$. For example, in the application of network flow classification presented in Sec. V, the features may include packet lengths, packet inter-arrival times, etc., and k may represent the level of delay sensitivity. Let $\mathbf{y} \in \mathbb{R}^K$ be a one-hot vector denoting the data sample's true class. We denote the k-th element of $\mathbf{y}$ by $\mathbf{y}^{(k)}$. Suppose the true label is k^*, then $\mathbf{y}^{(k^*)} = 1$ and $\mathbf{y}^{(k)} = 0, \forall k \neq k^*$. Thus, each data sample is associated with a tuple $(\mathbf{x}, \mathbf{y})$.

Time is slotted by rounds. In each round $t \in \{0, 1, ..., T\}$, the classifier decides its classification model, and then some data sample $(\mathbf{x}_t, \mathbf{y}_t)$ arrives. The classifier observes $\mathbf{x}_t$ and needs to estimate $\mathbf{y}_t$. We consider a linear classifier with a

decision function $h_c(\mathbf{x}_t) = \arg\max_k \hat{\mathbf{y}}_t^{(k)}$, where $\hat{\mathbf{y}}_t = \mathbf{A}_t^T \mathbf{x}_t$, and matrix $\mathbf{A}_t \in \mathcal{H} \subseteq \mathbb{R}^{M \times K}$. In cases where data are not linearly separable, non-linear feature transformation can be applied, so that the resulting transformed data features become linearly separable [31], [32]. That is, in case $\{(\mathbf{x}_t, \mathbf{y}_t)\}$ is not linearly separable we can use some non-linear function $\phi : \mathbb{V}^M \rightarrow \Phi$ to transform $\mathbf{x}_t \in \mathbb{R}^M$ to some $\phi(\mathbf{x}_t) \in \Phi$ so that the data $\{(\phi(\mathbf{x}_t), \mathbf{y}_t)\}$ is linearly separable. Thus, our consideration of linear classification can be generalized. Furthermore, linear classifiers are beneficial, especially for online classification, as they can be executed efficiently. They are commonly used in practice for network traffic classification [33], [34], since they can quickly process a large volume of traffic. Our evaluation results of network flow classification based on real-world data, presented in Sec. V, further demonstrate that a linear classifier can achieve high classification accuracy.

Let $L_c(\hat{\mathbf{y}}_t, \mathbf{y}_t)$ be the loss function of the classifier with respect to the data sample $(\mathbf{x}_t, \mathbf{y}_t)$. The loss function in this work is general and may include, for example, the commonly used *Hinge Loss* (HL) and *Categorical Cross-Entropy Loss* (CEL) [31]. We refer the readers to Sec. V for more details. To lessen the chance of overfitting, we use a regularizer $R_c(\mathbf{A}_t) = \sum_i \sum_j A_{i,j}^2$, which is a standard technique when training a classification model. Therefore, the classifier's cost function is $\mathcal{C}_c(\mathbf{A}_t, \mathbf{y}_t, \hat{\mathbf{y}}_t) = L_c(\hat{\mathbf{y}}_t, \mathbf{y}_t) + \gamma_c R_c(\mathbf{A}_t)$, where $\gamma_c \geq 0$ controls the trade-off of the regularizer against the loss. Since $\hat{\mathbf{y}}_t$ is a function of $\mathbf{x}_t$, we can write $\mathcal{C}_c(\mathbf{A}_t, \mathbf{y}_t, \hat{\mathbf{y}}_t) = \mathcal{C}_c(\mathbf{A}_t, \mathbf{y}_t, \mathbf{x}_t)$ when the context is clear.

We assume the feasible set $\mathcal{H}$ is convex and has Frobenius norm bounded by $Z_{\mathcal{H}}$.

B. Feature Manipulation

The generator of data sample $(\mathbf{x}_t, \mathbf{y}_t)$ receives some utility based on the classifier's decision $h_c(\mathbf{x}_t)$. For example, in network traffic classification, a flow may be allocated network resource based on the classifier's estimation of its delay sensitivity. Without loss of generality, we assume that the data generator has some preference for the output of the classifier, e.g., to be classified as high-priority traffic, as expressed by the general loss function $L_g(\hat{\mathbf{y}}) = \mathbf{b}^T \hat{\mathbf{y}}$ where $\mathbf{b} \in \mathbb{R}^K$. A malicious data generator has the incentive to manipulate the features of its data, in order to game the classifier and receive higher utility. In this work, we allow the simultaneous existence of both normal and malicious data generators.

Feature manipulation incurs some cost to the malicious data generator. For example, in Sec. V-A, we discuss several options for feature manipulation and their associated cost in network flow classification. Here, we consider a general model where the malicious data generator suffers a manipulation cost $C_m(\hat{\mathbf{x}}_t, \mathbf{x}_t) = \|\alpha \circ (\hat{\mathbf{x}}_t - \mathbf{x}_t)\|^2$, when the feature is changed from $\mathbf{x}_t$ to $\hat{\mathbf{x}}_t$, where α is a vector representing the weights of manipulating each feature.[1] Without loss of generality, in our analysis of the proposed algorithms, we normalize the feature values such that α is an all-ones vector. Thus, the cost function corresponding to the data sample $(\mathbf{x}_t, \mathbf{y}_t)$ is $\mathcal{C}_g(\mathbf{A}_t, \mathbf{x}_t, \mathbf{y}_t, \hat{\mathbf{x}}_t, \hat{\mathbf{y}}_t) = L_g(\hat{\mathbf{y}}_t) + \gamma_g C_m(\hat{\mathbf{x}}_t, \mathbf{x}_t)$, where g denotes the data generator, while $\gamma_g \geq 0$ controls the trade-off of the manipulation cost against the loss.

[1]Here, $\|\cdot\|$ is the Euclidean norm and "$\circ$"denotes element-by-element multiplication.

C. Clairvoyant and Non-Clairvoyant Data Generators

We assume that a malicious data generator can gain knowledge about the classification model. For example, the generator may observe the classification result from the past and use it to detect the classification model through training a local model on its own side, i.e., via reverse engineering attacks [17], [22]. Since detecting the classification model takes time, we assume that, at time t, the generator t can only observe the model $\mathbf{A}_{t'}$ for $t' \leq t - \xi$, for some non-negative integer ξ. This setting is general. Note that it does not rule out the possibility of $\xi = 0$, which represents the case where the data generator at any time t can accurately predict the classification model $\mathbf{A}_t$. In this case, we say that the data generator is *clairvoyant*. In the case $\xi > 0$, the data generator is called *non-clairvoyant*. Note that $\mathbf{A}_{t-\xi}$ is the most up-to-date classification model that the data generator in round t can observe. A malicious data generator can then manipulate its features from $\mathbf{x}_t$ to $\hat{\mathbf{x}}_t = \arg\min_{\mathbf{x}} \mathcal{C}_g(\mathbf{A}_{t-\xi}, \mathbf{x}_t, \mathbf{y}_t, \mathbf{x}, \hat{\mathbf{y}}_t)$ which best responds to $\mathbf{A}_{t-\xi}$. For a normal data generator, we have $\hat{\mathbf{x}}_t = \mathbf{x}_t$.

D. Model Updating Based on Delayed Feedback

At time t, after receiving a data sample with observed features $\hat{\mathbf{x}}_t$, which may be different from the true features $\mathbf{x}_t$, the online classifier computes the predicted scores $\hat{\mathbf{y}}_t = \mathbf{A}_t^T \hat{\mathbf{x}}_t$. Under a general online learning setup, the classifier receives delayed feedback of the true lable $\mathbf{y}_t$, which it can use to estimate the accuracy of $\hat{\mathbf{y}}_t$ and adaptively update the classification model over time [11]. Detecting the true label for data samples takes time. For example, in Sec. V-A, we discuss how various techniques can be used to obtain the true labels in online network flow classification. Initially, for clarity of presentation, we assume the classifier can observe the true label only after τ rounds, for some $\tau \geq 0$. That is, $\mathbf{y}_t$ is known to the classifier after $t' \geq t + \tau$.

E. Robust Online Learning Problem Formulation

The classifier aims at minimizing its *cumulative* cost $\sum_{t=1}^T \mathcal{C}_c(\mathbf{A}_t, \mathbf{y}_t, \hat{\mathbf{y}}_t)$ to learn an accurate classification model while making prediction on the fly. During this process, the classifier has to be *robust* to feature manipulation by the malicious data generators. Such a learning process typically incurs an increase in the obtained cumulative cost compared with an *offline oracle*, defined as $\arg\min_{\mathbf{A}} \sum_{t=1}^T \mathcal{C}_c(\mathbf{A}, \mathbf{y}_t, \hat{\mathbf{y}}_t)$. Note that the offline oracle chooses the best static classifier with full knowledge of the whole sequence of data, including their features $\hat{\mathbf{x}}_t$ and their true labels $\mathbf{y}_t$. The *regret* at time T is thus defined as [14]

$$\mathrm{Reg}(T) = \sum_{t=1}^{T} \mathcal{C}_c(\mathbf{A}_t, \mathbf{y}_t, \hat{\mathbf{y}}_t) - \min_{\mathbf{A}} \sum_{t=1}^{T} \mathcal{C}_c(\mathbf{A}, \mathbf{y}_t, \hat{\mathbf{y}}_t). \quad (1)$$

Our objective is to design an online algorithm to choose the classification parameters $\mathbf{A}_t, \forall t \in \{0, 1, ..., T\}$, such that the regret is minimized. Note that minimizing the regret is equivalent to minimizing the cumulative cost. We call the regret *sub-linear* if it grows sub-linearly with the number of rounds T, which implies that the difference between the *average* cost $\frac{1}{T}\sum_{t=1}^{T} \mathcal{C}_c(\mathbf{A}_t, \mathbf{y}_t, \hat{\mathbf{x}}_t)$ of the classifier and the average cost of the offline oracle vanishes as T goes to infinity.

IV. Robust Online Classification with Delayed Feedback

We present algorithms ROLC−NC and ROLC−C for the cases where the malicious data generators are non-clairvoyant (Sec. IV-A) and clairvoyant (Sec. IV-B) respectively. We show that both of them have provable performance guarantee in terms of a sub-linear regret bound.

A. Non-Clairvoyant Malicious Data Generators

Non-clairvoyant malicious data generators have $\xi > 0$. For this case, we propose the ROLC−NC algorithm, whose pseudo code is shown in Algorithm 1. The algorithm uses gradient descent, which is widely adopted in machine learning [31], [35], and combines it with delayed feedback information. A unique property of ROLC−NC is that it decides the classification model in each round, by leveraging *only* the data features presented in each round (regardless whether they are manipulated) and the delayed feedback of the true labels of some previously arrived data sample. Note that ROLC−NC does not require any additional information, e.g., whether the data sample is from a malicious or normal generator, the cost function of the malicious data generators, or the generator's delay ξ in observing the classification model.

1) Algorithm Design: ROLC−NC learns from the history and deploys a classification model parameterized by $\mathbf{A}_t$ in each round $t \in \{1, \cdots, T\}$ to predict the label for the corresponding data sample $(\mathbf{x}_t, \mathbf{y}_t)$.

As the feedback is delayed for τ rounds, the algorithm first initializes $\mathbf{A}_t$, for $t \in \{1, \cdots, \tau+1\}$, to some randomly picked $\mathbf{A}_0$ (Line 1). We recall that the parameters $\mathbf{A}_t$ are known to the data generators only after ξ rounds; that is, at time t, each data generator only knows the matrix $\mathbf{A}_{t'}$ for $t' \leq t - \xi$, for some positive integer ξ. Therefore, in rounds $t = 1, \cdots, \xi$, the malicious data generators manipulate the features arbitrarily. In each round t, ROLC−NC first commits to $\mathbf{A}_t$. Then a data sample $(\mathbf{x}_t, \mathbf{y}_t)$ arrives. If the data generator is malicious (but non-clairvoyant), it manipulates its data features to $\hat{\mathbf{x}}_t$ to obtain the best response to the latest model it observes $\mathbf{A}_{t-\xi}$; otherwise, $\hat{\mathbf{x}}_t = \mathbf{x}_t$. The classifier then observes $\hat{\mathbf{x}}_t$ and predicts a label $\hat{\mathbf{y}}_t$ with the model $\mathbf{A}_t$ (Line 3). In this round, the delayed true label $\mathbf{y}_{t-\tau}$ is known to the classifier, based on which the cost $\mathcal{C}_c(\mathbf{A}_{t-\tau}, \mathbf{y}_{t-\tau}, \hat{\mathbf{x}}_{t-\tau})$ from the prediction in that round is revealed (Line 4). This cost function is used to update the classification model in the next round. Then, ROLC−NC computes the (sub)gradient $\mathbf{g}_{t-\tau}$ of the cost function chosen at $\mathbf{A} = \mathbf{A}_{t-\tau}$ (Line 5). Finally, the algorithm updates the classification parameters $\mathbf{A}_{t+1} = \mathbf{A}_t - \eta \mathbf{g}_{t-\tau}$ for the next round, i.e., by taking a step proportional to the negative of the (sub)gradient, where η is the step size set by the algorithm (Line 6).

Algorithm 1: ROLC-NC: Robust OnLine Classification with Non-Clairvoyant Malicious Data Generators

Input: Data $\{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{T}$, step size η, and delay τ for receiving feedback
Output: Models parameterized by $\{\mathbf{A}_t\}_{t=1}^{T}$
1 Initialization: Select $\mathbf{A}_0 \in \mathcal{H}$ randomly and set $\mathbf{A}_1, ..., \mathbf{A}_\tau, \mathbf{A}_{\tau+1} = \mathbf{A}_0$.
2 **for** $t = \tau + 1$ to T **do**
3 Commit to $\mathbf{A}_t$, observe $\hat{\mathbf{x}}_t$, predict $\hat{\mathbf{y}}_t$.
4 Observe $\mathbf{y}_{t-\tau}$ (and suffer the cost $\mathcal{C}_c(\mathbf{A}_{t-\tau}, \mathbf{y}_{t-\tau}, \hat{\mathbf{x}}_{t-\tau})$).
5 Set $\mathbf{g}_{t-\tau} \in \partial_{\mathbf{A}} \mathcal{C}_c(\mathbf{A}, \mathbf{y}_{t-\tau}, \hat{\mathbf{x}}_{t-\tau})$ at $\mathbf{A}_{t-\tau}$.
6 $\mathbf{A}_{t+1} = \arg\min_{\mathbf{A} \in \mathcal{H}} \|\mathbf{A} - (\mathbf{A}_t - \eta \mathbf{g}_{t-\tau})\|^2$.

2) Performance Analysis: In this part, we analyze the performance of ROLC−NC. Specifically, we show in Theorem 1 that ROLC−NC has a sub-linear regret bound.

We first note that a function $f(\mathbf{x}) : \mathcal{X} \to \mathcal{Y}$ is called Q-Lipschitz over $\mathbf{x}$ in $\mathcal{X}$ if there exists a real constant $Q \geq 0$ such that, for all $\mathbf{x}$ and $\mathbf{x}'$ in $\mathcal{X}$, $\|f(\mathbf{x}) - f(\mathbf{x}')\| \leq Q\|\mathbf{x} - \mathbf{x}'\|$.

Theorem 1. *Assume that, at any round t, the cost function $\mathcal{C}_c(\mathbf{A}, \mathbf{y}_t, \hat{\mathbf{x}}_t)$ is convex over $\mathbf{A}$ and Q-Lipschitz over $\mathbf{A}$ in $\mathcal{H}$. With step size $\eta = \frac{Z_{\mathcal{H}}}{Q}\frac{1}{\sqrt{(\frac{1}{4}+\tau)T}}$, ROLC−NC has regret bound $Reg_{ROLC-NC}(T) \leq \rho\sqrt{T}$, where $\rho = 2QZ_{\mathcal{H}}\sqrt{1+4\tau}$.*

Proof. A matrix can be converted to a column vector through vectorization.[2] For ease of presentation, we assume $\mathbf{A}$, $\mathbf{A}_t$ and $\mathbf{g}_t$, for all t, are vectors in this proof.

Let $\mathbf{A}^* \in \arg\min_{\mathbf{A}} \sum_{t=1}^{T} \mathcal{C}_c(\mathbf{A}, \mathbf{y}_t, \hat{\mathbf{x}}_t)$. Then, due to the convexity of $\mathcal{C}_c$, we have

$$\begin{aligned}\mathrm{Reg}_{\mathrm{ROLC-NC}}(T) &= \sum_{t=1}^{T} [\mathcal{C}_c(\mathbf{A}_t, \mathbf{y}_t, \hat{\mathbf{x}}_t) - \mathcal{C}_c(\mathbf{A}^*, \mathbf{y}_t, \hat{\mathbf{x}}_t)] \\ &\leq \sum_{t=1}^{T} \mathbf{g}_t^T (\mathbf{A}_t - \mathbf{A}^*),\end{aligned}$$

where $\mathbf{g}_t \in \partial_{\mathbf{A}} \mathcal{C}_c(\mathbf{A}, \mathbf{y}_t, \hat{\mathbf{x}}_t)$ at $\mathbf{A}_t$. In this paper, we denote the transpose of any vector $\mathbf{p}$ as $\mathbf{p}^T$.

For $t > \tau$, since $\mathbf{A}_{t+1}$ is the result of projecting $\mathbf{A}_t - \eta \mathbf{g}_{t-\tau}$ onto the convex set $\mathcal{H}$, we have

$$\begin{aligned}&\|\mathbf{A}_{t+1} - \mathbf{A}^*\|^2 - \|\mathbf{A}_t - \mathbf{A}^*\|^2 \\ \leq &\eta^2 \|\mathbf{g}_{t-\tau}\|^2 - 2\eta \mathbf{g}_{t-\tau}^T [(\mathbf{A}_{t-\tau} - \mathbf{A}^*) + (\mathbf{A}_t - \mathbf{A}_{t-\tau})]. \quad (2)\end{aligned}$$

[2]The vectorization of a matrix is a linear transformation which converts the matrix into a column vector. Specifically, the vectorization of a $m \times n$ matrix $\mathbf{W}$ is the $mn \times 1$ column vector obtained by stacking the columns of the matrix $\mathbf{W}$ on top of each other.

Now we expand $\mathbf{g}_{t-\tau}^T(\mathbf{A}_t - \mathbf{A}_{t-\tau})$ as follows.

$$\begin{aligned}&\mathbf{g}_{t-\tau}^T(\mathbf{A}_t - \mathbf{A}_{t-\tau})\\ =&\sum_{j=1}^{\tau}\mathbf{g}_{t-\tau}^T(\mathbf{A}_{t-(j-1)} - \mathbf{A}_{t-j})\\ =&\sum_{j=1}^{\min(t-(\tau+1),\tau)}[-\eta\mathbf{g}_{t-\tau-j}^T\mathbf{g}_{t-\tau}-c_j],\end{aligned}\tag{3}$$

where $c_j = \mathbf{g}_{t-\tau}^T[(\mathbf{A}_{t-j} - \eta\mathbf{g}_{t-\tau-j}) - \mathbf{A}_{t-(j-1)}]$.

The proof will continue after the following lemma.

Lemma 1. $\|(\mathbf{A}_{t-j} - \eta\mathbf{g}_{t-\tau-j}) - \mathbf{A}_{t-(j-1)}\| \leq \|\eta\mathbf{g}_{t-\tau-j}\|$.

Proof. Since $\mathcal{H}$ is convex and $\mathbf{A}_{t-j}, \mathbf{A}_{t-(j-1)} \in \mathcal{H}$, we have $(1-\alpha)\mathbf{A}_{t-j} + \alpha\mathbf{A}_{t-(j-1)} \in \mathcal{H}$, for all $0 \leq \alpha \leq 1$. By the fact that $\mathbf{A}_{t-(j-1)}$ is the result of projecting $(\mathbf{A}_{t-j} - \eta\mathbf{g}_{t-\tau-j}^T)$ onto $\mathcal{H}$, we have, for all $0 \leq t \leq 1$, $\|(\mathbf{A}_{t-j} - \eta\mathbf{g}_{t-\tau-j}) - \mathbf{A}_{t-(j-1)}\| \leq \|(\mathbf{A}_{t-j} - \eta\mathbf{g}_{t-\tau-j}) - [(1-\alpha)\mathbf{A}_{t-j} + \alpha\mathbf{A}_{t-(j-1)}]\|$. Set $\alpha = 0$, and we have $\|(\mathbf{A}_{t-j} - \eta\mathbf{g}_{t-\tau-j}) - \mathbf{A}_{t-(j-1)}\| \leq \|\eta\mathbf{g}_{t-\tau-j}\|$. □

The Lipschitz property of $\mathcal{C}_c(\mathbf{A}, \mathbf{y}_t, \hat{\mathbf{x}}_t)$ gives $\|\mathbf{g}_t\| \leq Q, \forall t$. Thus, by Lemma 1, we have

$$c_j \leq \|(\mathbf{A}_{t-j} - \eta\mathbf{g}_{t-\tau-j}) - \mathbf{A}_{t-(j-1)}\| \cdot \|\mathbf{g}_{t-\tau}\| \leq \eta Q^2.$$

By (3), we have

$$\mathbf{g}_{t-\tau}^T(\mathbf{A}_t - \mathbf{A}_{t-\tau}) \geq \sum_{j=1}^{\min(t-(\tau+1),\tau)}[-\eta\mathbf{g}_{t-\tau-j}^T\mathbf{g}_{t-\tau}-\eta Q^2].$$

Substituting the above into (2), we have

$$\begin{aligned}&\mathbf{g}_{t-\tau}^T(\mathbf{A}_{t-\tau} - \mathbf{A}^*)\\ \leq&\frac{\eta}{2}\|\mathbf{g}_{t-\tau}\|^2+\frac{\|\mathbf{A}_t - \mathbf{A}^*\|^2-\|\mathbf{A}_{t+1} - \mathbf{A}^*\|^2}{2\eta}\\ &+\sum_{j=1}^{\min(t-(\tau+1),\tau)}[\eta\mathbf{g}_{t-\tau-j}^T\mathbf{g}_{t-\tau}+\eta Q^2].\end{aligned}\tag{4}$$

Interestingly, the last term characterizes the correlation between successive subgradients. Due to the Lipschitz properties of $\mathcal{C}_c(\mathbf{A}, \mathbf{y}_t, \hat{\mathbf{x}}_t)$, we have $\|\mathbf{g}_t\| \leq Q$. Furthermore, by the Cauchy-Schwarz inequality, we have $\mathbf{g}_{t-\tau-j}^T\mathbf{g}_{t-\tau} \leq \|\mathbf{g}_{t-\tau-j}\| \cdot \|\mathbf{g}_{t-\tau}\| = Q^2$. Hence we have

$$\begin{aligned}&\sum_{t=\tau+1}^{T+\tau}\sum_{j=1}^{\min(t-(\tau+1),\tau)}\eta\mathbf{g}_{t-\tau-j}^T\mathbf{g}_{t-\tau}\\ \leq&\sum_{t=\tau+1}^{T+\tau}\min(t-(\tau+1),\tau)\eta Q^2\\ =&(T\tau - \frac{\tau^2}{2} - \frac{1}{2})\eta Q^2.\end{aligned}$$

Summing up (4) over $t = \tau+1$ to $T+\tau$ yields $\sum_{t=\tau+1}^{T+\tau}\mathbf{g}_{t-\tau}^T(\mathbf{A}_{t-\tau} - \mathbf{A}^*) \leq \frac{\eta}{2}\sum_{t=\tau+1}^{T+\tau}Q^2 + \frac{\|\mathbf{A}_{\tau+1} - \mathbf{A}^*\|^2}{2\eta} + \sum_{t=\tau+1}^{T+\tau}\sum_{j=1}^{\min(t-(\tau+1),\tau)}\eta\mathbf{g}_{t-\tau-j}^T\mathbf{g}_{t-\tau} + \sum_{t=\tau+1}^{T+\tau}\tau\eta Q^2$.

Therefore, $\text{Reg}_{\texttt{ROLC-NC}}(T) \leq \frac{\eta}{2}TQ^2 + \frac{1}{2\eta}4Z_{\mathcal{H}}^2 + (T\tau - \frac{\tau^2}{2} - \frac{1}{2})\eta Q^2 + \tau TQ^2\eta \leq (\frac{1}{2} + \tau)\eta TQ^2 + \frac{2}{\eta}Z_{\mathcal{H}}^2 + \tau TQ^2\eta = (\frac{1}{2} + 2\tau)TQ^2\eta + \frac{2}{\eta}Z_{\mathcal{H}}^2$. We pick the step size $\eta = \frac{Z_{\mathcal{H}}}{Q}\frac{1}{\sqrt{(\frac{1}{4}+\tau)T}}$ so that $\text{Reg}_{\texttt{ROLC-NC}}(T) \leq 2QZ_{\mathcal{H}}\sqrt{1+4\tau}\sqrt{T}$. □

B. Clairvoyant Malicious Data Generators

In this section, we consider the case where data generators are clairvoyant (i.e., $\xi = 0$). This represents the worst-case challenge to the classifier. Note that the proposed ROLC−NC can be applied to classify the data when $\xi = 0$. However, to obtain guaranteed sub-linear regret in this case, the classifier requires more information. Thus, we extend ROLC−NC to propose ROLC−C. The pseudo code of ROLC−C is shown in Algorithm 2.

1) Malicious and Normal Rounds: We recall here that normal and malicious data co-exist. Let o_t indicate whether the data sample $(\mathbf{x}_t, \mathbf{y}_t)$ is normal (i.e., if normal, $o_t = 1$; otherwise, $o_t = 0$). We call a round *malicious round* (resp. *normal round*) if $o_t = 0$ (resp. $o_t = 1$).

We first analyze how a clairvoyant malicious data generator affects the classifier. Different from the non-clairvoyant data generators, a clairvoyant data generator can predict the classification model adopted by the classifier in the current round accurately. Therefore, after observing $\mathbf{A}_t$ committed by the classifier, the clairvoyant malicious generator manipulates its data features from $\mathbf{x}_t$ to $\hat{\mathbf{x}}_t$ that gives the best response to $\mathbf{A}_t$. Thus, $\hat{\mathbf{x}}_t$ is a solution of $\min_{\mathbf{x}}\mathcal{C}_g(\mathbf{A}_t, \mathbf{x}_t, \mathbf{y}_t, \mathbf{x}, \hat{\mathbf{y}}_t)$. That is, $\min_{\mathbf{x}} L_g(\hat{\mathbf{y}}_t(\mathbf{x})) + \gamma_g C_m(\mathbf{x}, \mathbf{x}_t)$, where $\mathbf{x}_t$ and $\mathbf{y}_t$ are the data features and true label of the data sample respectively. We observe that, for any fixed $\mathbf{A}_t$ and $\mathbf{x}_t$, the cost function $\mathcal{C}_g(\mathbf{A}_t, \mathbf{x}_t, \mathbf{y}_t, \mathbf{x}, \hat{\mathbf{y}}_t)$ is convex in $\mathbf{x}$. According to the KarushKuhnTucker (KKT) conditions, we have $\nabla_{\hat{\mathbf{x}}_t}\mathcal{C}_g(\mathbf{A}_t, \mathbf{x}_t, \mathbf{y}_t, \hat{\mathbf{x}}_t, \hat{\mathbf{y}}_t) = \mathbf{A}_t\mathbf{b} + 2\gamma_g(\hat{\mathbf{x}}_t - \mathbf{x}_t) = 0$. Therefore, $\hat{\mathbf{x}}_t$ is uniquely defined as

$$\hat{\mathbf{x}}_t = \mathbf{x}_t - \frac{1}{2\gamma_g}\mathbf{A}_t\mathbf{b}.\tag{5}$$

In the same round, the classifier aims to minimize its cost function $\mathcal{C}_c(\mathbf{A}_t, \mathbf{y}_t, \hat{\mathbf{x}}_t)$ by choosing an $\mathbf{A}_t$.

By substituting $\hat{\mathbf{x}}_t = \mathbf{x}_t - \frac{1}{2\gamma_g}\mathbf{A}_t\mathbf{b}$ into the classifier's cost function, we can identify the actual cost function that the classifier should optimize in the malicious round, which we denote as $\mathcal{C}_c^{(\text{m})}(\mathbf{A}_t, \mathbf{y}_t, \mathbf{x}_t)$. That is, $\mathcal{C}_c^{(\text{m})}(\mathbf{A}_t, \mathbf{y}_t, \mathbf{x}_t) = L_c(\hat{\mathbf{y}}_t(\hat{\mathbf{x}}_t), \mathbf{y}_t) + \gamma_c R_c(\mathbf{A}_t) = L_c(\hat{\mathbf{y}}_t(\mathbf{x}_t - \frac{1}{2\gamma_g}\mathbf{A}_t\mathbf{b}), \mathbf{y}_t) + \gamma_c R_c(\mathbf{A}_t) = L_c(\hat{\mathbf{y}}_t(\mathbf{x}_t) - \frac{1}{2\gamma_g}\mathbf{A}_t^T\mathbf{A}_t\mathbf{b}, \mathbf{y}_t) + \gamma_c R_c(\mathbf{A}_t)$. In contrast, in the normal round, the classifier's cost function is $\mathcal{C}_c^{(\text{n})}(\mathbf{A}_t, \mathbf{y}_t, \mathbf{x}_t) = L_c(\hat{\mathbf{y}}_t(\hat{\mathbf{x}}_t), \mathbf{y}_t) + \gamma_c R_c(\mathbf{A}_t) = L_c(\hat{\mathbf{y}}_t(\mathbf{x}_t), \mathbf{y}_t) + \gamma_c R_c(\mathbf{A}_t)$.

2) Algorithm Design: As discussed above, to compute the actual cost function of the classifier requires additional information on the data generator, including the cost function of data generators $\mathcal{C}_c^{(\text{n})}(\mathbf{A}_t, \mathbf{y}_t, \mathbf{x}_t)$, $\mathcal{C}_c^{(\text{m})}(\mathbf{A}_t, \mathbf{y}_t, \mathbf{x}_t)$, and o_t. However, we recall that, because of feedback delay, the true label $\mathbf{y}_t$, and hence also o_t, are known to the classifier only

Algorithm 2: ROLC-C: Robust OnLine Classification with Clairvoyant Malicious Data Generators

Input: Data $\{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^T$, step size η and delay τ
Output: Models parameterized by $\{\mathbf{A}_t\}_{t=1}^T$

1 Initialization: Select $\mathbf{A}_0 \in \mathcal{H}$ randomly and set $\mathbf{A}_1, ..., \mathbf{A}_\tau, \mathbf{A}_{\tau+1} = \mathbf{A}_0$.
2 **for** $t = \tau + 1$ to T **do**
3 Commit to $\mathbf{A}_t$, observe $\hat{\mathbf{x}}_t$, predict $\hat{\mathbf{y}}_t$.
4 Observe $\mathbf{y}_{t-\tau}$, $o_{t-\tau}$ and the cost $\mathcal{C}_c(\mathbf{A}_{t-\tau}, \mathbf{y}_{t-\tau}, \hat{\mathbf{x}}_{t-\tau})$.
5 **if** $o_{t-\tau} = 1$ **then**
6 Set $\mathbf{x}_{t-\tau} = \hat{\mathbf{x}}_{t-\tau}$ and
7 $\mathbf{g}_{t-\tau} \in \partial_{\mathbf{A}} \mathcal{C}_c^{(\mathrm{n})}(\mathbf{A}, \mathbf{y}_{t-\tau}, \mathbf{x}_{t-\tau})$ at $\mathbf{A}_{t-\tau}$.
8 **else**
9 Set $\mathbf{x}_{t-\tau} = \hat{\mathbf{x}}_{t-\tau} + \frac{1}{2\gamma_g}\mathbf{A}_{t-\tau}\mathbf{b}$ and
10 $\mathbf{g}_{t-\tau} \in \partial_{\mathbf{A}} \mathcal{C}_c^{(\mathrm{m})}(\mathbf{A}, \mathbf{y}_{t-\tau}, \mathbf{x}_{t-\tau})$ at $\mathbf{A}_{t-\tau}$.
11 $\mathbf{A}_{t+1} = \arg\min_{\mathbf{A}\in\mathcal{H}} \|\mathbf{A} - (\mathbf{A}_t - \eta \mathbf{g}_{t-\tau})\|^2$.

in round $t' \geq t + \tau$. Therefore, ROLC−C is designed to utilize such delayed information.

ROLC−C first initializes $\mathbf{A}_t, \forall t \in \{1, \cdots, \tau + 1\}$, to some randomly picked $\mathbf{A}_0$ (Line 1). In each round t, ROLC−C first commits to $\mathbf{A}_t$. Then a data sample $(\mathbf{x}_t, \mathbf{y}_t)$ arrives, and the features presented to the classifier is $\hat{\mathbf{x}}_t$. If it is a malicious data sample, $\hat{\mathbf{x}}_t$ is the best response to $\mathbf{A}_t$; otherwise, $\hat{\mathbf{x}}_t = \mathbf{x}_t$. The classifier observes $\hat{\mathbf{x}}_t$ and predicts a label $\hat{\mathbf{y}}_t$ with the model $\mathbf{A}_t$ (Line 3). In the same round t, the true label $\mathbf{y}_{t-\tau}$ and the indicator $o_{t-\tau}$ become known to the classifier, based on which the cost $\mathcal{C}_c(\mathbf{A}_{t-\tau}, \mathbf{y}_{t-\tau}, \hat{\mathbf{x}}_{t-\tau})$ from the prediction in that round is revealed (Line 4). Then the classifier chooses the cost function and computes its (sub)gradient to update the classification model in the next round (Line 5 to 10) according to $o_{t-\tau}$. If round $t - \tau$ is a normal round (Line 5), the algorithm sets the feature $\mathbf{x}_{t-\tau} = \hat{\mathbf{x}}_{t-\tau}$ as observed at time $t - \tau$ (Line 6) and computes the gradient $\mathbf{g}_{t-\tau}$ of the cost function $\mathcal{C}_c^{(\mathrm{n})}(\mathbf{A}, \mathbf{y}_{t-\tau}, \mathbf{x}_{t-\tau})$ at $\mathbf{A} = \mathbf{A}_{t-\tau}$ (Line 7); otherwise, it sets $\mathbf{x}_{t-\tau} = \hat{\mathbf{x}}_{t-\tau} + \frac{1}{2\gamma_g}\mathbf{A}_{t-\tau}\mathbf{b}$ according to (5) (Line 9), and computes the gradient $\mathbf{g}_{t-\tau}$ of the cost function $\mathcal{C}_c^{(\mathrm{m})}(\mathbf{A}, \mathbf{y}_{t-\tau}, \mathbf{x}_{t-\tau})$ at $\mathbf{A} = \mathbf{A}_{t-\tau}$ (Line 10). Finally, the algorithm computes the classification parameters $\mathbf{A}_{t+1} = \mathbf{A}_t - \eta \mathbf{g}_{t-\tau}$ for next round (Line 11).

3) Performance Analysis: In this part, we analyze the performance of ROLC−C. Specifically, we show in Theorem 2 that ROLC−C has a sub-linear upper bound on regret.

When the context is clear, we write $\mathcal{C}_c^{(\mathrm{n})}(\mathbf{A}_t, \mathbf{y}_t, \mathbf{x}_t)$ (resp. $\mathcal{C}_c^{(\mathrm{m})}(\mathbf{A}_t, \mathbf{y}_t, \mathbf{x}_t)$) as $\mathcal{C}_c^{(\mathrm{n})}$ (resp. $\mathcal{C}_c^{(\mathrm{m})}$) for simplicity.

Theorem 2. *Assume that, in any round t, the cost functions of $\mathbf{A}$, $\mathcal{C}_c^{(n)}(\mathbf{A}, \mathbf{y}_t, \mathbf{x}_t)$ and $\mathcal{C}_c^{(m)}(\mathbf{A}, \mathbf{y}_t, \mathbf{x}_t)$, are convex over $\mathbf{A}$ in $\mathcal{H}$, and Q_n-Lipschitz and Q_m-Lipschitz over $\mathbf{A}$ in $\mathcal{H}$ respectively. With step size $\eta = \frac{2Z_{\mathcal{H}}}{\sqrt{T\kappa Q_n^2 + T(1-\kappa)Q_m^2 + 4T\tau B}}$, ROLC−C has regret bound $\mathrm{Reg}_{\text{ROLC−C}}(T) \leq \rho\sqrt{T}$, where $\rho = 2Z_{\mathcal{H}}\sqrt{\kappa Q_n^2 + (1-\kappa)Q_m^2 + 4\tau B}$, κ is the fraction of normal data over the time interval from 1 to T, and $B = \max(Q_n^2, Q_n Q_m, Q_m^2)$.*

Proof. Similar to the proof of Theorem 1, for ease of presentation, we assume $\mathbf{A}$, $\mathbf{A}_t$ and $\mathbf{g}_t$, for all t, are vectors in this proof.

For a fixed $\mathbf{A}$, we have

$$\sum_{t=1}^{T}[\mathcal{C}_c(\mathbf{A}_t, \mathbf{y}_t, \hat{\mathbf{x}}_t) - \mathcal{C}_c(\mathbf{A}, \mathbf{y}_t, \hat{\mathbf{x}}_t)] = \sum_{t:o_t=1}[\mathcal{C}_c^{(\mathrm{n})}(\mathbf{A}_t) - \mathcal{C}_c^{(\mathrm{n})}(\mathbf{A})] + \sum_{t:o_t=0}[\mathcal{C}_c^{(\mathrm{m})}(\mathbf{A}_t) - \mathcal{C}_c^{(\mathrm{m})}(\mathbf{A})].$$

By convexity, for $\mathbf{g}_t' \in \partial\mathcal{C}_c^{(\mathrm{n})}(\mathbf{A}, \mathbf{y}_t, \mathbf{x}_t)$ at $\mathbf{A}_t$ in round t, we have $\mathcal{C}_c^{(\mathrm{n})}(\mathbf{A}_t) - \mathcal{C}_c^{(\mathrm{n})}(\mathbf{A}) \leq \mathbf{g}_t{}'^T(\mathbf{A}_t - \mathbf{A})$, and, for $\mathbf{g}_t'' \in \partial\mathcal{C}_c^{(\mathrm{m})}(\mathbf{A}, \mathbf{y}_t, \mathbf{x}_t)$ at $\mathbf{A}_t$ in round t, we have $\mathcal{C}_c^{(\mathrm{m})}(\mathbf{A}_t) - \mathcal{C}_c^{(\mathrm{m})}(\mathbf{A}) \leq \mathbf{g}_t{}''^T(\mathbf{A}_t - \mathbf{A})$.

Let $\mathbf{A}^* \in \arg\min_{\mathbf{A}} \sum_{t=1}^T \mathcal{C}_c(\mathbf{A}, \mathbf{y}_t, \hat{\mathbf{x}}_t)$. Then, by (1), we have $\mathrm{Reg}_{\text{ROLC−C}}(T) \leq \sum_{t=1}^T \mathbf{g}_t^T(\mathbf{A}_t - \mathbf{A}^*)$, where $\mathbf{g}_t \in \partial_{\mathbf{A}}\mathcal{C}_c^{(\mathrm{n})}(\mathbf{A}, \mathbf{y}_t, \hat{\mathbf{x}}_t)$ at $\mathbf{A}_t$ if $o_t = 1$; and $\mathbf{g}_t \in \partial_{\mathbf{A}}\mathcal{C}_c^{(\mathrm{m})}(\mathbf{A}, \mathbf{y}_t, \hat{\mathbf{x}}_t)$ at $\mathbf{A}_t$ if $o_t = 0$. For $t > \tau$, since $\mathbf{A}_{t+1}$ is the result of projecting $\mathbf{A}_t - \eta\mathbf{g}_{t-\tau}$ onto the convex set $\mathcal{H}$, we have

$$\begin{aligned}&\|\mathbf{A}_{t+1} - \mathbf{A}^*\|^2 - \|\mathbf{A}_t - \mathbf{A}^*\|^2 \\ \leq&\|\mathbf{A}_t - \eta\mathbf{g}_{t-\tau} - \mathbf{A}^*\|^2 - \|\mathbf{A}_t - \mathbf{A}^*\|^2 \\ =&\eta^2\|\mathbf{g}_{t-\tau}\|^2 - 2\eta\mathbf{g}_{t-\tau}^T[(\mathbf{A}_{t-\tau} - \mathbf{A}^*) + (\mathbf{A}_t - \mathbf{A}_{t-\tau})]. \quad (6)\end{aligned}$$

Now we expand $\mathbf{g}_{t-\tau}^T(\mathbf{A}_t - \mathbf{A}_{t-\tau})$ in (6) as follows.

$$\begin{aligned}&\mathbf{g}_{t-\tau}^T(\mathbf{A}_t - \mathbf{A}_{t-\tau}) \\ =&\sum_{j=1}^{\min(t-(\tau+1),\tau)} \mathbf{g}_{t-\tau}^T(\mathbf{A}_{t-(j-1)} - \mathbf{A}_{t-j}) \\ =&\sum_{j=1}^{\min(t-(\tau+1),\tau)} -\eta\mathbf{g}_{t-\tau-j}^T\mathbf{g}_{t-\tau} - c_j. \quad (7)\end{aligned}$$

where $c_j = [(\mathbf{A}_{t-j} - \eta\mathbf{g}_{t-\tau-j}^T) - \mathbf{A}_{t-(j-1)}]\mathbf{g}_{t-\tau}$.

Note that the Lipschitz property of $\mathcal{C}_c^{(\mathrm{n})}(\mathbf{A}, \mathbf{y}_t, \mathbf{x}_t)$ and $\mathcal{C}_c^{(\mathrm{m})}(\mathbf{A}, \mathbf{y}_t, \mathbf{x}_t)$ give $\|\mathbf{g}_t\| \leq Q_{\mathrm{n}}, \forall t: o_t = 1$, and $\|\mathbf{g}_t\| \leq Q_{\mathrm{m}}, \forall t: o_t = 0$. Thus, we have $\|\mathbf{g}_{t-\tau-j}^T\| \cdot \|\mathbf{g}_{t-\tau}\| \leq B$. Due to Lemma 1 and the Lipschitz property of $\mathcal{C}_c^{(\mathrm{n})}(\mathbf{A}, \mathbf{y}_t, \mathbf{x}_t)$ and $\mathcal{C}_c^{(\mathrm{m})}(\mathbf{A}, \mathbf{y}_t, \mathbf{x}_t)$, we have

$$c_j \leq \|(\mathbf{A}_{t-j} - \eta\mathbf{g}_{t-\tau-j}) - \mathbf{A}_{t-(j-1)}\| \cdot \|\mathbf{g}_{t-\tau}\| \leq \eta B.$$

The above inequation and (7) give

$$\mathbf{g}_{t-\tau}^T(\mathbf{A}_t - \mathbf{A}_{t-\tau}) \geq \sum_{j=1}^{\min(t-(\tau+1),\tau)} [-\eta\mathbf{g}_{t-\tau-j}^T\mathbf{g}_{t-\tau} - \eta B].$$

Substituting the above into (6), we have

$$\begin{aligned}&\mathbf{g}_{t-\tau}^T(\mathbf{A}_{t-\tau}-\mathbf{A}^*)\\ \leq&\frac{\eta}{2}\|\mathbf{g}_{t-\tau}\|^2+\frac{\|\mathbf{A}_t-\mathbf{A}^*\|^2-\|\mathbf{A}_{t+1}-\mathbf{A}^*\|^2}{2\eta}\\ &+\sum_{j=1}^{\min(t-(\tau+1),\tau)}[\eta\mathbf{g}_{t-\tau-j}^T\mathbf{g}_{t-\tau}+\eta B]. \quad (8)\end{aligned}$$

Summing up (8) over $t=\tau+1$ to $T+\tau$ yields

$$\begin{aligned}&\sum_{t=\tau+1}^{T+\tau}\mathbf{g}_{t-\tau}^T(\mathbf{A}_{t-\tau}-\mathbf{A}^*)\\ \leq&\frac{\eta}{2}\sum_{t=\tau+1}^{T+\tau}\|\mathbf{g}_{t-\tau}\|^2+\frac{\|\mathbf{A}_{\tau+1}-\mathbf{A}^*\|^2-\|\mathbf{A}_{T+\tau+2}-\mathbf{A}^*\|^2}{2\eta}\\ &+\sum_{t=\tau+1}^{T+\tau}\sum_{j=1}^{\min(t-(\tau+1),\tau)}\eta\mathbf{g}_{t-\tau-j}^T\mathbf{g}_{t-\tau}+\sum_{t=\tau+1}^{T+\tau}\tau\eta B\\ \leq&\frac{\eta}{2}\sum_{t=\tau+1}^{T+\tau}\|\mathbf{g}_{t-\tau}\|^2+\frac{\|\mathbf{A}_{\tau+1}-\mathbf{A}^*\|^2}{2\eta}\\ &+\sum_{t=\tau+1}^{T+\tau}\sum_{j=1}^{\min(t-(\tau+1),\tau)}\eta\mathbf{g}_{t-\tau-j}^T\mathbf{g}_{t-\tau}+\sum_{t=\tau+1}^{T+\tau}\tau\eta B.\end{aligned}$$

Next we bound the second last term in the right hand side of the above inequation. Since $\mathbf{g}_{t-\tau-j}^T\mathbf{g}_{t-\tau} \leq \|\mathbf{g}_{t-\tau-j}^T\|\cdot\|\mathbf{g}_{t-\tau}\|\leq B$, we have

$$\begin{aligned}&\sum_{t=\tau+1}^{T+\tau}\sum_{j=1}^{\min(t-(\tau+1),\tau)}\eta\mathbf{g}_{t-\tau-j}^T\mathbf{g}_{t-\tau}\\ \leq&\sum_{t=\tau+1}^{T+\tau}\min(t-(\tau+1),\tau)\eta B\\ =&(T\tau-\frac{\tau^2}{2}-\frac{1}{2})\eta B.\end{aligned}$$

Therefore, we have

$$\begin{aligned}&\mathrm{Reg}_{\texttt{ROLC-C}}(T)\\ \leq&\frac{\eta}{2}[T\kappa Q_{\mathrm{n}}^2+T(1-\kappa)Q_{\mathrm{m}}^2]+\frac{2}{\eta}Z_{\mathcal{H}}^2+2\tau TB\eta.\end{aligned}$$

We pick the step size $\eta=\frac{2Z_{\mathcal{H}}}{\sqrt{T\kappa Q_{\mathrm{n}}^2+T(1-\kappa)Q_{\mathrm{m}}^2+4T\tau B}}$. Then, we have $\mathrm{Reg}_{\texttt{ROLC-C}}(T) \leq Z_{\mathcal{H}}\sqrt{\kappa Q_{\mathrm{n}}^2+(1-\kappa)Q_{\mathrm{m}}^2+4\tau B}\sqrt{T}$. □

V. Application to Network Flow Classification

In this section, we present an application of the proposed robust online learning algorithms to network flow classification. We evaluate their performance by experimenting with real-world traffic data traces.

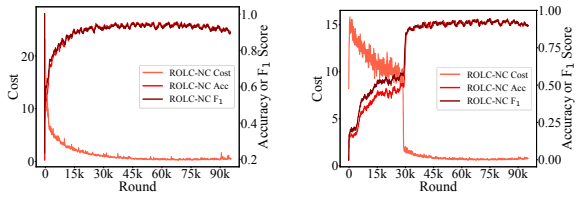
A. Robust Online Network Flow Classification

Network traffic classification enables proper assignment of network resource (e.g., bandwidth) to applications with different service requirements (e.g., delay). Thus, it is essential to modern network management [1]. To this end, machine learning techniques have become prominent in recent years due to the ineffectiveness of traditional port-based or payload-based approaches, especially for encrypted traffic [1]–[4], [36], [37]. Online traffic classification methods have been proposed to generate timely networking decisions on incoming flows [33], [38]. However, neither of these online solutions consider malicious feature manipulation. To the best of our knowledge, no existing work studies the problem of robust online traffic classification.

Here we consider the scenario where each traffic flow has a specific required quality of service (QoS) level [39]. The task is to classify flows that arrive online into multiple classes corresponding to different QoS requirements. The classifier inspects the sequence of flows one-by-one as they arrive and decides a classification model parameterized by $\mathbf{A}_t$ in each round. When a flow arrives, it observes the presented features and assigns to the arriving flow a predicted label denoting its QoS level.

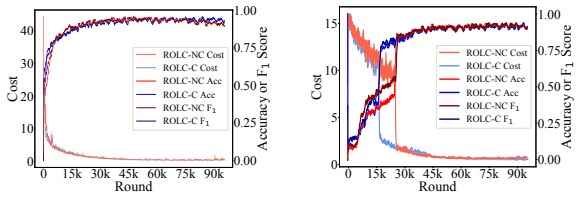
A flow generator may be an application, a user, or some other entity that is concerned about the QoS level of the flow. Each flow is allocated network resource based on the classifier's estimation of its QoS level. *Malicious flow generators* may exist, which can manipulate their flow features to best respond to the classification model to increase the likelihood of a certain outcome so as to increase their own utility. However, there is a cost to manipulate flow features. In a high-speed network, it is generally expensive to build hardware for packet flow manipulation at line rate. On the other hand, software-based packet flow manipulation requires frequent interaction with the memory (e.g., read and write operations) [5]. Furthermore, some manipulation such as changing header fields requires more complex operation [7]. Therefore, software-based manipulation at line rate is non-trivial and often reduces the performance of the flow. Here we adopt the general cost model as explained in Sec. III-B. In addition, often it may take some time for a malicious flow generator to detect the classification model through reverse engineering attacks [17], [22]. We consider both the *non-clairvoyant* (i.e., $\xi>0$) and *clairvoyant* (i.e., $\xi=0$) cases as defined in Sec. III-C.

An online learning approach requires comparison between the true and predicted labels of a previously arrived and classified flow. For online network flow classification, the true label can be obtained, for example, through deep packet inspection (DPI), or by a sophisticated robust classifier residing in some powerful remote server, which is trained using sufficient data to give accurate prediction in spite of the altered features. Both options incur computation or communication delay. State-of-the-art DPI systems for encrypted traffic has high runtime overhead and cannot process packets in real-time [1], [40]–[42]. For example, BlindBox requires minutes for every new



(a) HL, $\xi = 10$ (b) CEL, $\xi = 10$

Fig. 1. Learning process of ROLC-NC with $\xi = 10$.



(a) HL, $\xi = 0$ (b) CEL, $\xi = 0$

Fig. 2. Learning process of ROLC-NC and ROLC-C with $\xi = 0$.

end-to-end connection [40]. Embark enables a cloud provider (e.g., Amazon EC2) to outsource DPI processing, which incurs communication delay [41].

Fortunately, as discussed in Sec. IV, the proposed algorithms ROLC-NC and ROLC-C can accommodate both malicious flow generators and feedback delay, while providing performance guarantee in terms of sub-linear regret bounds. Next, we present further experimental details and results when these algorithms are applied to online network flow classification.

B. Trace-Driven Experiments

1) Data Trace: To apply ROLC-NC and ROLC-C to online network flow classification, we use packet traces from [43] and [44]. All packets in the trace dataset are TCP packets, and each TCP connection corresponds to a flow. The dataset contains 377,526 flows in total. In each round of an experiment, a randomly selected flow from this set is sent to the classifier. The record of each flow contains a variety of characteristic features. We consider 100 standard features including the minimum, mean, maximum, and standard deviation of packet lengths and packet inter-arrival times, number of packets and bytes, and the duration of the network connection. They are features numbered 3-9, 195-208, 10-30, 153-194, 210-215, 31-40 in [45].[3]

Each flow belongs to 12 application types: www, mail, ftp-control, ftp-pasv, attack, p2p, database, ftp-data, multimedia, services, interactive, and games. We map these application types into four different QoS labels roughly based on the application's delay requirement. The label assignment is as follows. $k = 1$: multimedia, interactive, games, and ftp-control; $k = 2$: attack, www, and p2p; $k = 3$: database, ftp-data, and services; $k = 4$: mail and ftp-pasv. Thus, the lower k is, the more sensitive to delay the corresponding flow is.

2) Performance Metrics and Benchmarks: We take the classification *accuracy* as our primary metric, which is more useful than the regret in practice. In addition, since the class distribution is imbalanced in the dataset, we also take the F_1 *score* as a second performance metric [31], [46].

We compare the proposed algorithms with two offline benchmarks as follows. We first train a static classification model by minimizing its cumulative cost $\sum_{t=1}^{T} \mathcal{C}_c(\mathbf{A}, \mathbf{y}_t, \mathbf{x}_t)$, assuming knowledge of the true features and true labels of all flows for $t = 1, 2, \cdots, T$. The optimizer we use is Sequential Least Squares Programming (SLSQP) [47]. Then, the two benchmarks are the performance metrics obtained by this classification model under two different test datasets as follows.

- **`Offline`**: Performance obtained with test dataset $(\hat{\mathbf{x}}'_t, \mathbf{y}_t), \forall t$, which contains the observed flow features (manipulated version if the flow comes from a malicious generator) of the same sequence of flows presented to the online algorithms, except that the manipulated flow features $\hat{\mathbf{x}}'_t$ best respond to the above offline classification model.
- **`Offline-Norm`**: Performance obtained with test dataset $(\mathbf{x}_t, \mathbf{y}_t), \forall t$, the normal flows containing unmanipulated original features.

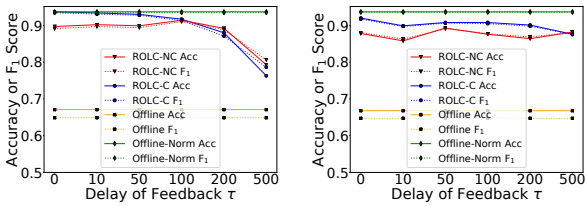
3) Experiment Setting: The loss function of the classifier is general in this work (see Sec. III). In our experiments, we use two common loss functions, Hinge Loss (HL) and Categorical Cross-Entropy Loss (CEL) defined as follows [31]:

- HL: $L_c(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{k \neq k^*} \max(0, 1 + \hat{\mathbf{y}}^{(k)} - \hat{\mathbf{y}}^{(k^*)})$, where k^* is the true QoS level;
- CEL: $L_c(\hat{\mathbf{y}}, \mathbf{y}) = -\log \frac{e^{\hat{\mathbf{y}}^T \mathbf{y}}}{\sum_k e^{\hat{\mathbf{y}}^{(k)}}}$.

In each experiment, ROLC-NC and ROLC-C are run using either HL or CEL for $T = 100,000$ rounds, which is sufficient for them to go into the steady state. Over the T rounds, a fraction κ (resp. $1 - \kappa$) of rounds are randomly assigned as normal (resp. malicious) rounds. In each round, a flow is uniformly randomly chosen from the data trace. We evaluate the performance of ROLC-NC and ROLC-C in different settings by varying the value of key parameters around the default setting $(\kappa, \tau, \gamma_c, \gamma_g, \mathbf{b}) = (0.5, 10, 0.1, 0.8, [1, 2, 4, 8])$. The comparison benchmarks are calculated based on the same test data as those of the online algorithms. All experiments are run on a machine with two Intel(R) Xeon(R) CPU E5-2650 v4 2.20GHz with 32GB memory and 1.8TB hard drive.

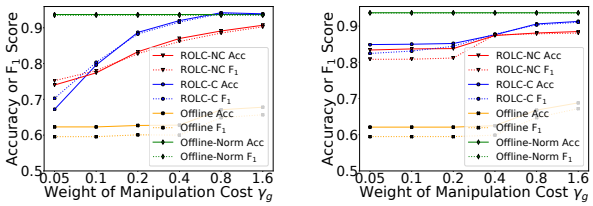
4) Accuracy and F_1 Score over Time: We evaluate the performance of ROLC-NC and ROLC-C in terms of the cost of each round, the accuracy, and the F_1 score, over a sliding window of size 2000 rounds during the learning process. For the first 2000 rounds, the performance metrics are calculated over all arrived flows. Fig. 1 shows the learning process of ROLC-NC when the delay for a malicious flow generator to detect the classification model is $\xi = 10$. Fig. 2 shows the

[3]These features are chosen since they can be manipulated by the flow generators in practice.



(a) HL (b) CEL

Fig. 3. The impact of delay of feedback τ.



(a) HL (b) CEL

Fig. 4. The impact of weight of manipulation cost γ_g.

learning process of ROLC−NC and ROLC−C when $\xi = 0$.[4] We observe that, in all experiments, the cost value decreases quickly at the beginning and converges to its steady state after 15k (resp. 30k) rounds for ROLC−NC using HL (resp. CEL) and after 15k rounds for ROLC−C for both HL and CEL. The accuracy and F_1 score at convergence are both greater than 0.9. We also find that both algorithms learn faster and have higher accuracy and F1 score using HL than using CEL. After 45k rounds, ROLC−NC using HL achieves 0.91 accuracy and 0.91 F_1 score when $\xi = 10$, and ROLC−C using HL achieves 0.93 accuracy and 0.93 F_1 score when $\xi = 0$.

When $\xi = 0$, we can further compare the performance of ROLC−NC and ROLC−C. Fig. 2 shows that, ROLC−C converges faster than ROLC−NC does, and performs slightly better than ROLC−NC in classification accuracy and F_1 score. This is because ROLC−C uses the real cost function $C_c^{(\mathrm{m})}$ in a malicious round to update the model, while ROLC−NC does not. This, as well as similar results presented in Figs. 3 and 4, suggests that ROLC−C can be a better choice in the case when the flow generators are clairvoyant.

We also evaluate the convergence time for these algorithms to reach steady state in the learning process. We find that, with our machine, the average convergence time is between 75 and 150 seconds. To further reduce this time in practical implementation, one may use more powerful computing hardware and/or fewer features, e.g., only features from the initial packets of a flow [48].

5) Impact of Feedback Delay τ: We further evaluate the impact of the feedback delay τ on the performances of ROLC−NC and ROLC−C in steady state and present the results in Fig. 3. We calculate the accuracy and F_1 score by averaging over 2000 rounds in steady state. Keeping the default parameter setting described previously and ξ=0, we vary τ from 0 to 500. For each data point in the figure, we take the average of 10 repeated experiments, where we fix the sequences of the arriving flows $\{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^T$ and indicators $\{o_t\}_{t=1}^T$ while varying the random initialization of the algorithms (Line 1 of Algorithms 1 and 2).

Fig. 3 presents the accuracy and F_1 score of ROLC−NC, ROLC−C, and the offline benchmarks. Note that the offline benchmarks are not affected by τ. When using HL (Fig. 3(a)), both ROLC−NC and ROLC−C have stable performance when $\tau \leq 100$, and ROLC−C achieves 0.92 accuracy and F_1 score on average, which is only 0.01 lower than those of Offline−Norm, while ROLC−NC achieves 0.90 accuracy and F_1 score on average. When the delay τ becomes longer, both ROLC−NC and ROLC−C yield lower accuracy and F_1 score. When $\tau \geq 200$, ROLC−NC performs slightly better than ROLC−C. When using CEL (Fig. 3(b)), ROLC−NC and ROLC−C are less sensitive to the parameter τ, achieving above 0.88 accuracy and F_1 score even when τ is large. We note that, although not shown in this figure, a larger τ incurs a longer convergence time for both algorithms, which is expected for online learning.

6) Impact of Manipulation Cost Weight γ_g: We vary the value of γ_g to study its impact on the performance of ROLC−NC and ROLC−C. We use the default parameter setting with $\xi = 0$. Fig. 4 presents the accuracy and F_1 score of the proposed algorithms and the offline benchmarks. Note that Offline−Norm is not affected by γ_g as expected. With either HL or CEL, the proposed algorithms and Offline all yield higher accuracy and F_1 score when the manipulation cost γ_g increases. This is because a larger γ_g encourages the malicious flow generators to stay closer to the original features.

We have also conducted experiments to evaluate the fraction of normal flows κ, the impact of the delay for the flow generator to observe the classification model ξ, the weight of regularizer γ_c, and the weighting vector of flow loss $\mathbf{b}$. We omit them due to the space constraint.

VI. Conclusion

In this work, we address the problem of robust online learning against malicious manipulation. The data features may be manipulated by malicious generators to best respond to the classification models committed by the classifier. Practical issues such as delayed feedback (for the classifier) and delayed observation of the classification model (for malicious data generators) are captured in the problem. We propose online algorithms ROLC−NC and ROLC−C, respectively, for when the data generators are non-clairvoyant and clairvoyant. Our theoretical analysis shows that both algorithms have a sub-linear regret bound when the classifier's cost function is convex. We further evaluate the performance of the proposed algorithms in network flow classification via experiments using real-world data traces. We observe that the proposed algorithms are effective, in terms of steady-state accuracy and the F_1 score, and they compare favorably with an optimal static offline classification strategy under different testing scenarios.

[4]Recall that ROLC−NC is designed for $\xi > 0$ but it can also be applied to the case when $\xi = 0$, while ROLC−C is used only when $\xi = 0$.

REFERENCES

[1] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: an overview," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 76–81, 2019.

[2] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust network traffic classification," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1257–1270, 2015.

[3] T. T. Nguyen, G. Armitage, P. Branch, and S. Zander, "Timely and continuous machine-learning-based classification for interactive IP traffic," *IEEE/ACM Transactions on Networking*, vol. 20, no. 6, pp. 1880–1894, 2012.

[4] Y. Li, B. Liang, and A. Tizghadam, "Robust network flow classication against malicious feature manipulation," in *Proc. of IEEE ICC*, 2020.

[5] S. Pontarelli, M. Bonola, and G. Bianchi, "Smashing SDN 'built-in' actions: programmable data plane packet manipulation in hardware," in *Proc. of IEEE NetSoft*, 2017.

[6] M. Gadelrab, A. A. El Kalam, and Y. Deswarte, "Manipulation of network traffic traces for security evaluation," in *Proc. of IEEE AINA*.

[7] M. Meitinger, R. Ohlendorf, T. Wild, and A. Herkersdorf, "A programmable stream processing engine for packet manipulation in network processors," in *Proc. of IEEE ISVLSI*, 2007.

[8] M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli, "Tunnel hunter: Detecting application-layer tunnels with statistical fingerprinting," *Computer Networks*, vol. 53, no. 1, pp. 81–97, 2009.

[9] D. J. Miller, Z. Xiang, and G. Kesidis, "Adversarial learning in statistical classification: a comprehensive review of defenses against attacks," *arXiv preprint arXiv:1904.06292*, 2019.

[10] B. Biggio and F. Roli, "Wild patterns: ten years after the rise of adversarial machine learning," in *Proc. of ACM CCS*, 2018.

[11] S. Shalev-Shwartz, "Online learning and online convex optimization," *Foundations and Trends in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2011.

[12] M. Abramson, "Toward adversarial online learning and the science of deceptive machines," in *Proc. of AAAI Fall Symposium Series*, 2015.

[13] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?" in *Proc. of ACM AsiaCCS*, 2006.

[14] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. of ICML*, 2003.

[15] D. J. Miller, Z. Xiang, and G. Kesidis, "Adversarial learning targeting deep neural network classification: a comprehensive review of defenses against attacks," *Proceedings of the IEEE*, vol. 108, no. 3, pp. 402–433, 2020.

[16] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar, "Adversarial machine learning," in *Proc. of ACM AISec*, 2011.

[17] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *Proc. of USENIX Security*, 2016.

[18] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Proc. of ECMLPKDD*, 2013.

[19] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Proc. of ICLR*, 2014.

[20] M. Brückner, C. Kanzow, and T. Scheffer, "Static prediction games for adversarial learning problems," *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 2617–2654, 2012.

[21] J. Lin, C. Gan, and S. Han, "Defensive quantization: when efficiency meets robustness," in *Proc. of ICLR*, 2019.

[22] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. of ACM AsiaCCS*, 2017.

[23] A. Demontis, M. Melis, B. Biggio, D. Maiorca, D. Arp, K. Rieck, I. Corona, G. Giacinto, and F. Roli, "Yes, machine learning can be more secure! a case study on android malware detection," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 4, pp. 711–724, 2017.

[24] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. of IEEE SP*, 2016.

[25] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: bypassing ten detection methods," in *Proc. of ACM AISec*, 2017.

[26] M. Kloft and P. Laskov, "Online anomaly detection under adversarial impact," in *Proc. of AISTATS*, 2010.

[27] T. S. Sethi and M. Kantardzic, "Handling adversarial concept drift in streaming data," *Expert Systems with Applications*, vol. 97, pp. 18–40, 2018.

[28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. of NIPS*, 2014.

[29] P. Grnarova, K. Y. Levy, A. Lucchi, T. Hofmann, and A. Krause, "An online learning approach to generative adversarial networks," in *Proc. of ICLR*, 2018.

[30] A. Resler and Y. Mansour, "Adversarial online learning with noise," in *Proc. of ICML*, 2019.

[31] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning from Data*. AMLBook New York, NY, USA, 2012.

[32] I. Kononenko and M. Kukar, *Machine Learning and Data Mining*. Horwood Publishing, 2007.

[33] Y. Jin, N. Duffield, J. Erman, P. Haffner, S. Sen, and Z.-L. Zhang, "A modular machine learning system for flow-level traffic classification in large networks," *ACM Transactions on Knowledge Discovery from Data*, vol. 6, no. 1, pp. 1–34, 2012.

[34] C. S. Miao, L. Meng, C. Q. Yuan, X. W. Wang, and G. R. Chang, "Traffic classification combining flow correlation and ensemble classifier," *International Journal of Wireless and Mobile Computing*, vol. 6, no. 6, pp. 556–563, 2013.

[35] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. 7, pp. 2121–2159, 2011.

[36] B. Wang, J. Zhang, Z. Zhang, W. Luo, and D. Xia, "Robust traffic classification with mislabelled training samples," in *Proc. of IEEE ICPADS*, 2015.

[37] J. Zhang, F. Li, F. Ye, and H. Wu, "Autonomous unknown-application filtering and labeling for DL-based traffic classifier update," in *Proc. of IEEE INFOCOM*, 2020.

[38] K. L. Dias, M. A. Pongelupe, W. M. Caminhas, and L. de Errico, "An innovative approach for real-time network traffic classification," *Computer Networks*, vol. 158, pp. 143–157, 2019.

[39] M. Lopez-Martin, B. Carro, J. Lloret, S. Egea, and A. Sanchez-Esguevillas, "Deep learning model for multimedia quality of experience prediction based on network flow packets," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 110–117, 2018.

[40] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, "BlindBox: deep packet inspection over encrypted traffic," in *Proc. of ACM SIGCOMM*, 2015.

[41] C. Lan, J. Sherry, R. A. Popa, S. Ratnasamy, and Z. Liu, "Embark: securely outsourcing middleboxes to the cloud," in *Proc. of USENIX NSDI*, 2016.

[42] S. Canard, A. Diop, N. Kheir, M. Paindavoine, and M. Sabt, "BlindIDS: market-compliant and privacy-friendly intrusion detection system over encrypted traffic," in *Proc. of ACM AsiaCCS*, 2017.

[43] "Nprobe: scalable network monitoring architecture," https://www.cl.cam.ac.uk/research/srg/netos/projects/archive/nprobe/.

[44] A. Moore, D. Zuev, and M. Crogan, "Discriminators for use in flow-based classification," https://www.cl.cam.ac.uk/~awm22/publication/moore2005discriminators.pdf, Tech. Rep. RR-05-13.

[45] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1, 2005, pp. 50–60.

[46] "Compute the F_1 score," https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html.

[47] J. Nocedal and S. Wright, *Numerical Optimization*. Springer Science & Business Media, 2006.

[48] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 2, pp. 23–26, 2006.