Defending against Poisoning Attacks in Online Learning Settings

Greg Collinge, Emil C. Lupu, Luis Muñoz-González [*]

Department of Computing - Imperial College London
180 Queen's Gate, SW7 2AZ, London, United Kingdom
{greg.collinge17, e.c.lupu, l.munoz}@imperial.ac.uk

Abstract. Machine learning systems are vulnerable to data poisoning, a coordinated attack where a fraction of the training dataset is manipulated by an attacker to subvert learning. In this paper we first formulate an optimal attack strategy against online learning classifiers to assess worst-case scenarios. We also propose two defence mechanisms to mitigate the effect of online poisoning attacks by analysing the impact of the data points in the classifier and by means of an adaptive combination of machine learning classifiers with different learning rates. Our experimental evaluation supports the usefulness of our proposed defences to mitigate the effect of poisoning attacks in online learning settings.

1 Introduction

Despite the advancements of machine learning and their effectiveness to automate many tasks, it has been shown that learning algorithms are vulnerable and can be compromised by attackers both at training and test time [1]. Poisoning attacks, i.e. those produced at training time, are considered one of the most relevant and emerging threats for data-driven technologies, especially in applications that rely on the collection of untrusted data [2]. Different poisoning attack strategies have shown the sensitivity of machine learning algorithms to this threat, including Support Vector Machines (SVMs) [3], logistic regression [4], or neural networks and deep learning [5, 6], among others. In standard (*offline*) settings, optimal attack strategies can be modelled as a bi-level optimization problem [3, 4, 5], where the attacker aims to optimise a particular objective (typically to maximise the error of the system) by injecting a set of malicious data points in the training set. On the other side the defender aims to minimise some loss function on a training set that includes these poisoning points. In contrast, optimal poisoning attacks against *online* learning algorithms just require to solve a (single-level) optimisation problem [7]. In this paper we present a novel poisoning attack against online learning classifiers with a compounding strategy that prevents overfitting on the attacker's side, maximising the impact of the attack at test time. We also propose two different defensive mechanisms to mitigate the effect of such attacks: First, inspired by the defence in [8], we propose a mechanism to reject samples that are suspicious to be malicious, not using them to update the parameters of the system. Second, we

[*]This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 824988.

propose an adaptive combination of learning algorithms with different learning rates, which allows to delay the effect of the attack on the overall system and to speed-up recovery after the attack. Our preliminary experiments in MNIST dataset show the benefits of our proposed defences.

2 Attack

In online learning settings the attacker's objective is to alter the state of the existing trained model when poisoning points are injected into the flow of training data to degrade system's performance on a targeted or an indiscriminate way. Thus, at iteration n, given the attacker's utility function $\mathcal{A}$ (typically a cost function) evaluated on a target dataset $\mathcal{D}_t$, the attacker's objective is to inject a poisoning point $(\mathbf{x}_p, y_p)$, defined within a feasible domain $\phi(\mathbf{x}_p)$, so that:[1]

$$\mathbf{x}_p^* \in \arg\max_{\mathbf{x}_p \in \phi(\mathbf{x}_p)} \mathcal{A}(\mathcal{D}_t, \mathbf{w}_{n+1}) \tag{1}$$

where $\mathcal{A}$ is evaluated on the parameters of the classifier at iteration $n+1$. These parameters can be computed with stochastic gradient descent as $\mathbf{w}_{n+1} = \mathbf{w}_n - \eta \nabla_{\mathbf{w}_n} \mathcal{L}(\mathbf{x}_p, \mathbf{w}_n)$, where $\mathcal{L}$ is the loss function used by the defender to train the algorithm, and η is the learning rate. We can solve (1) following a gradient ascent strategy with the update equation given by:

$$\mathbf{x}_{p_{n+1}} = \mathbf{x}_{p_n} + \alpha \nabla_{\mathbf{x}_{p_n}} \mathcal{A}(\mathcal{D}_t, \mathbf{w}_{n+1}) \tag{2}$$

where α is the learning rate for the attacker's problem in (1). We can compute the gradient in (2) by applying the chain rule:

$$\nabla_{\mathbf{x}_{p_n}} \mathcal{A}(\mathcal{D}_t, \mathbf{w}_{n+1}) = -\eta \nabla_{\mathbf{x}_{p_n}} \nabla_{\mathbf{w}_n} \mathcal{L}(\mathbf{x}_p, \mathbf{w}_n) \nabla_{\mathbf{w}_{n+1}} \mathcal{A}(\mathcal{D}_t, \mathbf{w}_{n+1}) \tag{3}$$

Direct computation of (3) is inefficient, but we can use the fast Hessian-vector product approximation in [9] to avoid the computation of the Hessian.

Assuming that the attacker can inject a concentration of poisoning points p, which are randomly distributed between genuine data points, the direct application of the greedy strategy in (3) may lead to poor results, as the attack depends on a specific parameter state. The attacker may not know the whole system's state when interleaving the poisoning points with (unknown) genuine points. In Alg. 1 we propose a compounding strategy to enhance the effectiveness of the attack and reduce overfitting on the attacker's side. First, we train the learning algorithm using the whole training set for N iterations.[2] Then, for each poisoning point we use a random subset of the training data (with n_s samples) and the poisoning points crafted in previous iterations. We run gradient descent for 1 epoch to update the state of the learning algorithm and compute the poisoning point using (3). The size of the random subset n can be set according to the number of poisoning points n_p, so that $p = n_p/(n_p + n_s)$, as a guide to a suitable value for developing an attack aimed at a specific concentration p.

[1]In line with most related work we assume that the label of the poisoning point is defined a priori and cannot change.

[2]To model worst-case scenarios we assume that the attacker has access to the defender's training set.

```
Input: Training data $\mathcal{D}_{tr}$, attacker's target data $\mathcal{D}_t$, training subset
  size $n_s$, initial training epochs $N$, number of poisoning points $n_p$,
  initial set of parameters $\mathbf{w}_0$
$\mathcal{D}_p \leftarrow \emptyset$, Initialize set to store poisoning points;
$\mathbf{w} \leftarrow$ Gradient_Descent($\mathbf{w}_0$, $\mathcal{D}_{tr}$, $N$);
for $i \leftarrow 1$ to $n_p$ do
    $\mathcal{D}_i \leftarrow$ random_subset($\mathcal{D}_{tr}$, $n_s$);
    $\mathbf{w}_i \leftarrow$ Gradient_Descent($\mathbf{w}$, $\mathcal{D}_i \cup \mathcal{D}_p$, 1);
    $(\mathbf{x}_p, y_p) \leftarrow$ compute_poisoning_point($\mathbf{w}_i$,$\mathcal{D}_i \cup \mathcal{D}_p$,$\mathcal{D}_t$) (use eq. (3));
    $\mathcal{D}_p \leftarrow \mathcal{D}_p \cup (\mathbf{x}_p, y_p)$;
end
Output: $\mathcal{D}_p$
```
Algorithm 1: Online attack compounding strategy

3 Defences

In this section we present two different defensive mechanisms to mitigate the effect of the online poisoning attacks described in previous section.

3.1 Rejection of samples

If the attacker does not impose appropriate constraints to craft the attack points, it has been shown that outlier detection can effectively detect malicious samples [10]. In online learning settings, assuming that the learning algorithm has reached a relative stable state we can apply a rejection mechanism to exclude from training *unexpected* data points to ignore suspicious poisoning points or points that have a strong negative impact on the learning algorithm at its current state. In multi-class classification we need to include one rejection mechanism for each class: Targeted attacks against particular classes may be difficult to detect as the overall effect in the error of the classifier can be relatively small. Thus, for a sample arriving at instant n, $(\mathbf{x}_n, y_n)$, given the state of the learning algorithm in $\mathbf{w}_{n-1}$, we reject the sample if:

$$f_{y_n}(\mathbf{x}_n, \mathbf{w}_{n-1}) < \beta \tag{4}$$

where $f_{y_n}(\mathbf{x}_n, \mathbf{w}_{n-1})$ is the element of the soft output of the classifier for the sample $\mathbf{x}_n$, with class label y_n, evaluated on $\mathbf{w}_{n-1}$, and $\beta < 1$ is the threshold of the rejection mechanism. If the sample is rejected, then $w_n = w_{n-1}$. If β is too low more poisoning points will be admitted, whereas if it is too high the ability of the learning algorithm to adapt to natural distribution drifts will be reduced.

3.2 Adaptive Combination of Machine Learning Models

Following a different approach, we also propose to defend against online poisoning attacks with a convex combination of two classifiers with different learning rates η_1 and η_2 where $\eta_1 > \eta_2$. We can expect higher learning rate classifiers to adapt more quickly to natural distribution drifts, providing high accuracy when the system is not under attack, whereas the latter is more resilient to burst attacks due to being slower to react to changes. We can leverage the advantages of each classifier depending on the situation (attack/no attack) by an adaptive combination of the classifiers' output. This scheme has been successfully applied in the context of adaptive filtering in non-stationary environments [11]. The soft output of the combination can be written as:

$$f_{n-1}(\mathbf{x}_n) = \lambda_{n-1} f^{(1)}_{n-1}(\mathbf{x}_n) + (1-\lambda_{n-1}) f^{(2)}_{n-1}(\mathbf{x}_n), \qquad 0 < \lambda_{n-1} < 1, \tag{5}$$

where $f^{(1)}_{n-1}(\mathbf{x}_n)$ and $f^{(2)}_{n-1}(\mathbf{x}_n)$ are the soft outputs of the two classifiers for $\mathbf{x}_n$. Similar to the parameters of the two models, the parameter that controls the adaptive combination, λ_{n-1} can be updated using a gradient descent approach:

$$\lambda_n = \lambda_{n-1} - \alpha \nabla_{\lambda_{n-1}} \mathcal{L}(\mathcal{D}_{val}, \mathbf{w}^{(1)}_{n-1}, \mathbf{w}^{(2)}_{n-1}, \lambda_{n-1}) \tag{6}$$

where α is the learning rate and $\mathcal{L}(\mathcal{D}_{val}, \mathbf{w}^{(1)}_{n-1}, \mathbf{w}^{(2)}_{n-1}, \lambda_{n-1})$ is the loss function evaluated on a separate validation set $\mathcal{D}_{val}$ with the parameters of the two classifiers, $\mathbf{w}^{(1)}_{n-1}$ and $\mathbf{w}^{(2)}_{n-1}$, and λ_{n-1}. As suggested in [11] we can use a sigmoid parametrization of λ_n to provide smoother updates, and reduce the effective range of λ_n from $[0, 1]$ to $[\delta, 1-\delta]$ (being δ a small positive constant) to avoid saturation and improve responsiveness when conditions change. For $\mathcal{D}_{val}$ we can follow different strategies such as evaluating the performance on a rolling window containing the last samples in the training set or using a clean validation dataset. In our experiments we used the latter approach.

4 Experiments

To assess the validity of our attack and defence strategies we performed two experiments with MNIST dataset. In the first experiment we crafted an *error-specific* attack [5], which aims to increase the error rate between digits 6 and 9 (*6v9*) for a multi-class logistic regression classifier, trained online using a mini-batch size of 32 samples. The poisoning attack was crafted using Alg. 1 with a concentration parameter $p = 0.1$. We compared the performance of the system when applying the rejection mechanism described in Sect. 3.1 (with $\beta = 0.1$) and when no defence is applied. The results in Fig. 1 show the effectiveness of the attack to increase the error between digits 6 and 9 from 12% up to 25%. The overall effect on the classifier is reduced, as only two digits are involved in this error-specific attack. We can also observe that the rejection strategy completely mitigates the effect of the attack, not affecting the performance of the classifier when the attacker is not present. Despite the simplicity of this approach, in practice, this strategy can be effective in cases where the attacker uses aggressive strategies for poisoning the system, forcing his/her to craft more subtle and persistent attacks to successfully degrade the system's performance.

In the second experiment we evaluated the performance of the adaptive combination of two logistic regression classifiers, with $\eta_1 = 0.1$ and $\eta_2 = 10^{-4}$,

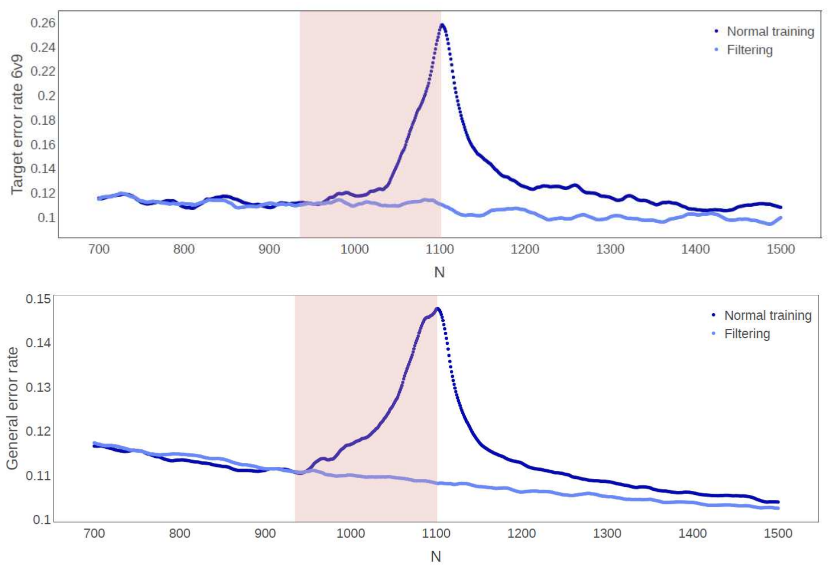

Fig. 1: Poisoning attack against (online) logistic regression, comparing the rejection mechanism in Sect. 3.1 to the case where no defence is applied. N is the cumulative number of mini-batched trained on and the red box depicts the interval where the attacker is present. The top figure represents the specific errors between digits 6 and 9. The bottom plot shows the overall classification error.

against the poisoning attack in Alg. 1 with $p = 0.17$. We performed an *error-generic* attack [5], aiming to maximally increase the overall classification error. We constrained λ to the interval $[0.08, 0.92]$. For the loss function in (6) we used a separated validation set with 200 samples. The classifiers were trained online with a mini-batch size of 32. The results in Fig. 2 show that the adaptive combination effectively mitigates the poisoning attack. Before the attack, the *fast* classifier dominates the output of the combined scheme, with $\lambda = 0.92$. However, when the attack started λ gradually shifted to 0.08, so that the *slow* classifier became dominant in the combination, reducing the overall impact of the attack. Although not shown in Fig. 2, when the fast classifier recovered from the attack, λ increased again to 0.92.

5 Conclusion

In this paper we proposed a new poisoning attack against online learning algorithms with a compounding strategy that prevents overfitting. We also developed two defensive mechanisms to mitigate the effect of such attacks by rejecting samples that are suspicious to be malicious and with an adaptive combination of classifiers with different learning rates. The preliminary experimental results on MNIST show the effectiveness of these defences to mitigate poisoning attacks and to recover faster to the system's original state after the attack.

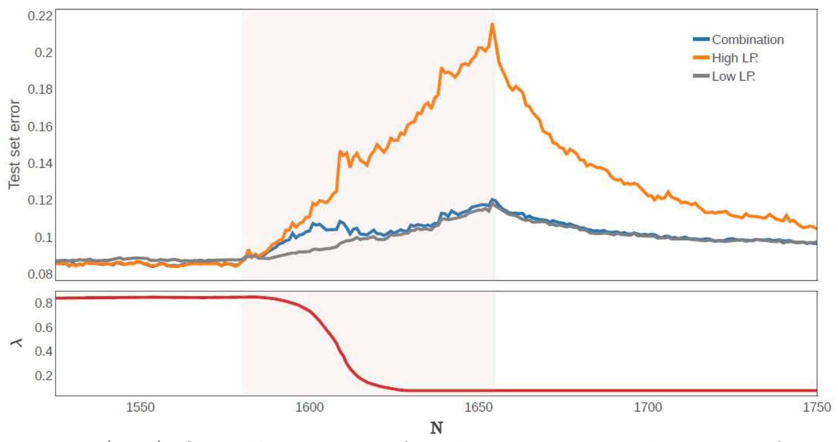

Fig. 2: (Top) Overall test error for the adaptive combination of two logistic regression classifiers. The combination was initially trained with $10,000$ samples followed by an error-generic attack. The red box represents the interval where the attack is present (with $p = 0.17$). N is the cumulative number of mini-batched trained on. (Bottom) Evolution of λ.

References

[1] L. Huang, A.D. Joseph, B. Nelson, B.I.P. Rubinstein, and J.D. Tygar. Adversarial Machine Learning. In *Workshop on AI and Security*, pages 43–58, 2011.

[2] A.D. Joseph, P. Laskov, F. Roli, J.D. Tygar, and B. Nelson. Machine Learning Methods for Computer Security (Dagstuhl Perspectives Workshop 12371). *Dagstuhl Manifestos*, 3(1), 2013.

[3] B. Biggio, B. Nelson, and P. Laskov. Poisoning Attacks against Support Vector Machines. In *ICML*, pages 1807–1814, 2012.

[4] Shike Mei and Xiaojin Zhu. Using Machine Teaching to Identify Optimal Training-Set Attacks on Machine Learners. In *AAAI*, pages 2871–2877, 2015.

[5] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E.C. Lupu, and F. Roli. Towards Poisoning of Deep Learning Algorithms with Back-Gradient Optimization. In *Workshop on AI and Security*, pages 27–38, 2017.

[6] P.W. Koh and P. Liang. Understanding Black-box Predictions via Influence Functions. In *ICML*, pages 1885–1894, 2017.

[7] Y. Wang and K. Chaudhuri. Data Poisoning Attacks against Online Learning. *arXiv preprint arXiv:1808.08994*, 2018.

[8] B. Nelson, M. Barreno, F.J. Chi, A.D. Joseph, B.I.P. Rubinstein, U. Saini, C.A. Sutton, J.D. Tygar, and K. Xia. Exploiting Machine Learning to Subvert Your Spam Filter. *LEET*, 8:1–9, 2008.

[9] B.A. Pearlmutter. Fast Exact Multiplication by the Hessian. *Neural Computation*, 6(1):147–160, 1994.

[10] A. Paudice, L. Muñoz-González, A. Gyorgy, and E.C. Lupu. Detection of Adversarial Training Examples in Poisoning Attacks through Anomaly Detection. *arXiv preprint arXiv:1802.03041*, 2018.

[11] J. Arenas-García, A.R. Figueiras-Vidal, and A.H. Sayed. Mean-square Performance of a Convex Combination of Two Adaptive Filters. *IEEE Trans. on Signal Processing*, 54(3):1078–1090, 2006.