With Great Dispersion Comes Greater Resilience: Efficient Poisoning Attacks and Defenses for Linear Regression Models

Jialin Wen, Benjamin Zi Hao Zhao, Minhui Xue🄳, Alina Oprea, and Haifeng Qian🄳

Abstract—With the rise of third parties in the machine learning pipeline, the service provider in "Machine Learning as a Service" (MLaaS), or external data contributors in online learning, or the retraining of existing models, the need to ensure the security of the resulting machine learning models has become an increasingly important topic. The security community has demonstrated that without transparency of the data and the resulting model, there exist many potential security risks, with new risks constantly being discovered. In this paper, we focus on one of these security risks – *poisoning attacks*. Specifically, we analyze how attackers may interfere with the results of regression learning by poisoning the training datasets. To this end, we analyze and develop a new poisoning attack algorithm. Our attack, termed *Nopt*, in contrast with previous poisoning attack algorithms, can produce larger errors with the same proportion of poisoning data-points. Furthermore, we also significantly improve the state-of-the-art defense algorithm, termed TRIM, proposed by Jagielsk *et al.* (IEEE S&P 2018), by incorporating the concept of probability estimation of clean data-points into the algorithm. Our new defense algorithm, termed *Proda*, demonstrates an increased effectiveness in reducing errors arising from the poisoning dataset through optimizing ensemble models. We highlight that the time complexity of TRIM had not been estimated; however, we deduce from their work that TRIM can take exponential time complexity in the worst-case scenario, in excess of *Proda*'s logarithmic time. The performance of both our proposed attack and defense algorithms is extensively evaluated on four real-world datasets of housing prices, loans, health care, and bike sharing services. We hope that our work will inspire future research to develop more robust learning algorithms immune to poisoning attacks.

I. Introduction

WITH the widespread adoption of Machine Learning (ML) algorithms, it has been elevated out of the exclusive use of high-tech companies [1]. Services such as "Machine Learning as a Service" (MLaaS) [2] can assist companies without domain expertise in ML to solve business problems with ML. However, in the MLaaS setting, there exist poisoning attacks, in which malicious MLaaS providers can either manipulate the integrity of the training data supplied by the company or compromise the integrity of the training process. Alternatively, in a collaborative setting, whereby a model holder solicits data contributions from multiple parties for *online training or retraining of an existing model*, a malicious participant may provide poisoned training samples in their submission, thereby infecting the resulting model for all parties. In such poisoning attacks, the attacker's objective may be to indiscriminately alter prediction results, create a denial of service, or cause specific targeted mis-predictions during test time. The attacker seeks to create these negative effects while preserving correct predictions on the remaining test samples to bypass detection. An inconspicuous attack may produce dire consequences, thus necessitating to study poisoning attacks on ML. A conceptual example of poisoning attacks is illustrated in Figure 1.

Many poisoning attacks have been proposed and demonstrated against different ML architectures. Specially, grey-box attacks, in which the attacker has no knowledge of the training set but has an alternative dataset with the same distribution as the training set, have been proposed against Support Vector Machines (SVMs) [3], Deep Neural Networks (DNNs) [4], Logistic Regression (LR) [5], Graph-based classification [6], and Recommender systems [7]. These works have shown that poisoning attacks are effective in interfering with the accuracy of producing classifications, or recommendations by either indiscriminately altering prediction results, or causing specific mis-predictions at test time. However, the aforementioned attacks target models producing a label prediction; in this work, we shall focus on models that perform regression, the prediction of a numerical value. Thus, with a different functional objective of regression, an attacker's objective for poisoning a regression model may also differ. For example,
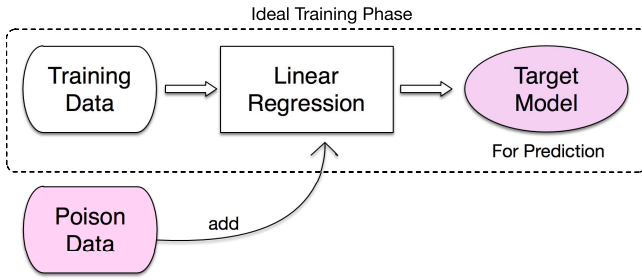
Fig. 1. The poisoning attack on linear regression.

an attacker may want to increase or decrease the predicted value, or may want to maximize the dispersion of the training set. Ma *et al.* [8] first propose a white-box poisoning attack against linear regression, aimed at manipulating the trained model by adversarially modifying the training set. Additionally, Jagielski *et al.* [1] propose several white-box and grey-box[1] poisoning attacks against linear regression, which aims to increase the loss function on the original training set.

A. Our Attack Contributions

We scrutinize poisoning attacks on linear regression by improving and redefining the attacker's objective in existing attack models and establish a new attack optimization problem for linear regression. Our new attack, termed *Nopt*, is observed to be more efficient than the state of the art [1] (IEEE S&P 2018), termed *Opt*, in maximizing the dispersion of the training set, for the same proportion of poisoned data-points. *Intuitively, the key difference between Opt [1] and Nopt is that the optimization evaluated for each subsequent poisoning point is performed on a dataset that includes all previous poisoning points, thereby creating a new poisoning point that maximizes the loss of the collective training dataset of original clean and poisoning points.*

To defend against poisoning attacks on regression learning, defense mechanisms have been proposed [9]–[13]. One such approach treats poisoning attack data-points as outliers, which can be counteracted with data sanitization techniques [14]–[16] (i.e., input validation and removal). Another approach is through robust learning [1], [10], [17], [18], as learning algorithms based on robust statistics are intrinsically less sensitive to outlying training samples; the robustness can be realized through bounded losses or specific kernel functions. Ma *et al.* [8] leverage differential privacy as a defensive measure against poisoning attacks on linear regression. Jagielski *et al.* [1] propose a defense algorithm against regression learning poisoning attacks, named TRIM. TRIM offers high robustness and resilience against a large number of poisoning attacks. *We highlight that the time complexity of TRIM had not been estimated; however, we deduce from their work that TRIM can take exponential time complexity in the worst-case scenario, in excess of our defense (termed Proda)'s logarithmic time.*

[1]The assumptions of [1] is the same as ours, albeit previously mislabelled as a black-box attack.

B. Our Defense Contributions

We define a new defense algorithm against poisoning attacks, termed *Proda*. We are the first to introduce the concept of probability estimation of unpolluted data-points into the defense algorithm. Proda demonstrates an increased effectiveness in reducing errors arising from the poisoning dataset. Additionally, the time complexity of Proda is also lower than the state-of-the-art defense of TRIM [1] (IEEE S&P 2018). *The key insight of achieving better efficacy is that we have prior knowledge that from the sets of points we randomly sample, there must be a group of points that all belong to the unpolluted dataset. Therefore, when comparing the minimum mean squared error (MSE) values of some groups of data-points, even if there are poisoning points in the group of the smallest MSE, they must conform to the original distribution and rules of the training dataset, and will have little impact on altering the regression model.*

In this paper, we systematically study the poisoning attack and its defense for linear regression models. We define a new poisoning attack on linear regression to maximize the dispersion of the training set. Additionally, we develop a new probabilistic defense algorithm against poisoning attacks, named Proda. We extensively evaluate poisoning attacks and defenses across different four regression models (Ordinary Least Squares, Ridge Regression, LASSO, Elastic-Net) trained on multiple datasets originating from different fields, including house pricing, loans, pharmaceuticals, and bike sharing services. In summary, the overall contributions of this paper are as follows.

- We develop a new grey-box poisoning attack against regression models, termed *Nopt*. *Nopt* outperforms the state-of-the-art attack, termed Opt, proposed by Jagielsk *et al.* [1] (IEEE S&P 2018).
- We prove that the state-of-the-art defense, termed TRIM [1], is estimated to have exponential time complexity in the worst-case scenario, in excess of our *Proda*'s logarithmic time.
- We further overhaul TRIM and propose to date the most effective defense against poisoning attacks through optimizing ensemble models, termed *Proda*. The performance of both our proposed attack and defense algorithms is extensively evaluated on four real-world datasets of housing prices, loans, health care, and bike sharing services.

To the best of our knowledge, we are among the *first* to systematically design, develop, and evaluate the poisoning attack and defense for linear regression models. We hope that our work will inspire future research to develop more robust learning algorithms immune to poisoning attacks.

II. PRELIMINARIES

In this section, we first introduce linear regression and then take a deep dive into defining the threat model of this paper.

A. Linear Regression

Linear Regression [1] is a supervised machine learning algorithm, frequently used to analyze complex data relationships. Linear regression uses inherent statistical features of the

training dataset to quantitatively determine mutually dependent relationships between two or more variables. By learning these relationships, the resulting linear regression model can produce a numerical output on an unseen input.

Specifically, in linear regression, the model after training is a linear function $f(x, \theta) = w^T + b$, which seeks to regress the value of y for a given input x. The real parameter vector $\theta = (w, b)$ of dimension $d + 1$ consists of the feature weights w and the bias b, of dimensionality d and 1, respectively. However, the true value is noted as $y = f(x, \theta) + e$, containing e, the error between the true value and the predicted value. Assuming that e is Independent and Identically Distributed (IID), the mean is 0, the variance is fixed, and that the noise term e satisfies the Gaussian distribution:

$$g(y_i | x_i; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(y_i - \theta^T x_i)^2}{2\sigma^2}). \quad (1)$$

Then, the maximum likelihood function of the model parameters can be obtained as the product of all training sets:

$$L(\theta) = \prod_{i=1}^{m} g(y_i | x_i); \theta))$$
$$= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(y_i - \theta^T x_i)^2}{2\sigma^2}). \quad (2)$$

The maximum value of $L(\theta)$ is maintained when the log function is applied, $\log L(\theta)$.

$$\log L(\theta) = \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(y_i - \theta^T x_i)^2}{2\sigma^2})$$
$$= m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^{m} (y_i - \theta^T x_i)^2. \quad (3)$$

In order to obtain the maximum likelihood, the latter term of Equation (3) is to be minimized.

We note that the maximum likelihood of linear regression can also be converted into the minimum value of the least squares, the most common mathematical form of the loss function:

$$L(D_{tr}, \theta) = \frac{1}{2} \sum_{i=1}^{m} (f(x_i, \theta) - y_i)^2 + \lambda \Omega(w), \quad (4)$$

where D_{tr} is the training data, $\Omega(w)$ is a regularization term penalizing large weight values, and λ is the regularization parameter used to prevent overfitting.[2]

The primary difference between popular linear regression methods is in the choice of the regularization term. In this paper, we study the following four regression models: Ordinary Least Squares (OLS), with no regularization, Ridge regression, which uses l_2-norm regularization, LASSO, which uses l_1-norm regularization, and Elastic-net regression, which uses a combination of l_1-norm and l_2-norm regularization. We elaborate on the regularization term in context of the more

common minimum least squares form of the loss function.

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^{m} (f(x_i, \theta) - y_i)^2. \quad (5)$$

Our proposed attack and defense hinge on its relative effectiveness in comparison with existing methods. We shall inspect effectiveness in two aspects. Firstly, the degree of poisoning by comparing the loss function of the poisoned model with the non-poisoned model when trained on the same dataset, as a successful poisoning attack, will have increased the dispersion of the points, and thus the resulting learned regression line. The specific metric used to quantify the effect of the poisoning attack will be the Mean Squared Error (MSE) (see Equation (5)) of the true value from the predicted value. Secondly, specific to our defense, the time complexity of the deploying defense shall be experimentally measured in seconds.

B. Threat Model

The core objective of a poisoning attack is to corrupt the learning model generated from the training phase, such that predictions on unseen data will greatly differ in the testing phase. However, depending on whether the goal is to produce predictions that greatly differ on specific subsets of input data, while preserving predictions on the remaining subsets, or if predictions are to be altered indiscriminately, the poisoning attack is categorized as either an Integrity attack, or an Availability attack. A similar deconstruction of attacks is found in backdoor poisoning attacks [19]–[23]. In this work, we consider a poisoning availability attack.

1) Attack Assumptions: There is a sliding scale of knowledge that is assumed to be available to the attack, from white-box to grey-box and black-box attacks. Under the assumptions of a white-box attack, the attacker has access to the training data D_{tr}, the learning algorithm L, and the trained parameters θ. Black-box attacks have no knowledge about the internal construction of the model, with only input and output access to the model. However, situated between white-box and black-box, in our grey-box setting,[3] the attacker has no knowledge of the training set D_{tr} but has an alternative dataset D'_{tr} that has the same distribution as the pristine training set D_{tr}. Internal to the model, the learning algorithm L is known; however, the trained parameters θ are not. We do note that an attacker, can approximate θ' by optimizing L on D'_{tr} [13], [24].

It is known that black-box attacks are more practical in real-world adversaries with less knowledge required about the model. However, in this work we adopt the grey-box setting. Under the grey-box setting, we assume the adversary has no information about the structure L or parameters θ of linear regression, and does not have access to any large training dataset.

2) Poisoning Rates: As visualized in Figure 1, a poisoning attack is performed by injecting poisoned data into the training set before the regression model is (re)trained. The influence of an attacker on the resulting model is limited by an upper bound

---

[2]Equation (4) is an instantiation of Equation (1) in [1] for our alternative loss E, as we shall explain later in Section III-A.

[3]There are many types of grey-box attacks; this is our own grey-box setting.

on the proportion $\alpha = n_p/N$ of poisoned data (D_p of size n_p) to the original clean data (D_o of size n_o) in the training dataset (D_N of size $N = n_p + n_o$) [1]. An attacker has complete control of the poisoning samples, as such input feature values and responses can be arbitrarily set within known bounds (These feature bounds may either be derived from D_{tr} or D'_{tr}, or assumed if the data is normalized.). Consistent with restrictions imposed by our settings of MLaaS, online learning, and retraining, prior works rarely consider poisoning rates larger than 20%, as the attacker is limited to being able to control only a small fraction of the training data [1]. Thus, in this paper, we shall investigate poisoning rates up to a maximum of $\alpha = 0.2$. This maximum is motivated by prior works [1], as poisoning rates higher than 20% have rarely been considered, since the attacker is assumed to be capable of controlling only a small fraction of the training data. This is motivated by application scenarios, such as crowdsourcing and network traffic analysis, in which attackers can only reasonably control a small fraction of participants and network packets, respectively. Moreover, learning a sufficiently-accurate regression function in the presence of higher poisoning rates would be an ill-posed task, as the poisoning attack would be trivial [1].

3) Defense Assumptions: We shall also be investigating defenses. To the defender, the model is a white box (as they are the model holder), the only additional item of information a defender may not know is the poisoning rate of an attacker. It is possible for the defender to derive the poisoning rate from the size of the update data provided to it. In the event of an inability to derive the poisoning rate, it has been argued that a poisoning rate of $\alpha = 0.2$ is representative of an upper limit of poisoning attacks, and can be assumed as a worst-case scenario.

III. POISONING ATTACKS BASED ON OPTIMIZATION

Previous works have discussed a poisoning attack strategy, which is applicable not only to linear regression, but also to classification algorithms. Those poisoning attacks aim to maximize the test error. However, as we have discussed, the attack objective on linear regression is different from the attack objectives for classification algorithms, with the latter seeking to only produce a specific wrong answer. Therefore, we define a new poisoning attack, *Nopt* poisoning attack. By establishing a new poisoning optimization algorithm for linear regression, this attack will force the model to receive a more dispersed training dataset. With a more dispersed training set this will result in larger losses and/or poor convergence on the regression task, which may erode confidence in the model holder, or simply result in worse prediction confidence in practice.

A. Definition of Nopt Poisoning Attack

In this section, we define a new form of linear regression attack, the *Nopt* attack. Previously in Section II-A, we observed that the loss function of linear regression is the sum of squares from each point to a regression model. When more points are added, the loss should also increase.
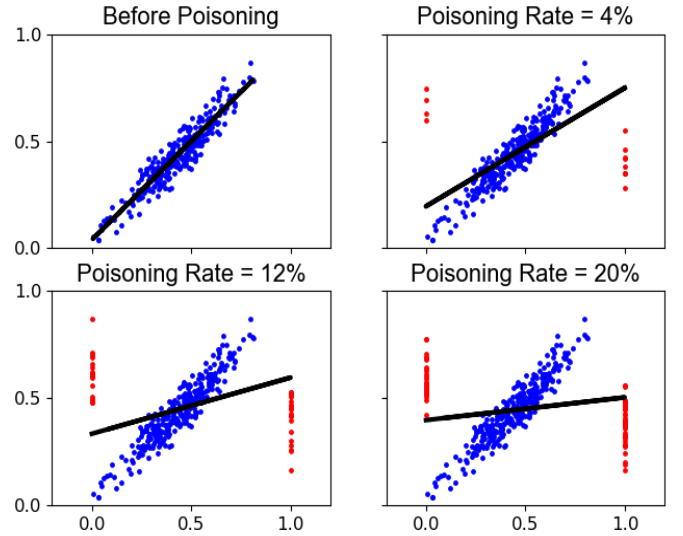

Fig. 2. Impact on learned regression line as a result of poisoning attacks at different poisoning rates. The original unpoisoned data is shown in blue, while poisoning data is shown in red.

However, when the added points are distributed in the similar manner as the pristine data, we obtain (without considering regularization):

$$\frac{L(D_N)}{L(D_o)} = \frac{N}{n_o}. \tag{6}$$

That is, the ratio of the loss function $L(D_N)$ of the new training set D_N to the loss function $L(D_o)$ of the original training set D_o should be equal to the ratio of the size of the new training set N to the size of the original training set n_o when the poisoning data is distributed in a similar way as the original training set. Additionally, as the new training set D_N is made up of the original training set D_o and poisoning data set D_p, we formulate our attack as follows:

$$E = \left| \frac{L(D_N)}{L(D_o)} - \frac{N}{n_o} \right| = \left| \frac{L(D_o \cup D_p)}{L(D_o)} - \frac{(n_o \cup n_p)}{n_o} \right|. \tag{7}$$

As we can see from Equations (6) and (7), when the added poisoning points are distributed in the same way as the original data, $E = 0$. The further a data point is added away from the original regression line, the higher the value of E. Therefore, the value of E is directly related to the rate of poisoning.

B. Application of Nopt Poisoning Attack

Previously, Jagielski *et al.* [1] established a bilevel optimization problem to find the set of poisoning points that maximize the loss function of the original data-points. In this section, we compose our new poisoning attack by optimizing the objective function:

$$\arg\max_{D_p} \ E(D_{tr} \cup D_p, \theta^{(p)}),$$
$$\text{s.t. } \theta^{(p)} \in \arg\min_{\theta} \ L(D_{tr} \cup D_p, \theta). \tag{8}$$

The Nopt poisoning attack searches for poisoning data-points D_p by maximizing E, optimizing the loss function with respect to the poisoning training dataset. Intuitively,

---

Algorithm 1 Nopt Poisoning Attack Algorithm

Input: $D = D_{tr}$ (white-box) or $D = D'_{tr}$ (grey-box), L, E, the initial poisoning attack samples $D_p^{(0)} = (x_c, y_c)_{c=1}^p$, a small positive constant ε.

1: $i \leftarrow 0$ (iteration counter);
2: $\theta^{(i)} \leftarrow \arg\min_\theta L(D \cup D_p^{(i)})$;
3: **repeat**
4:    $e^{(i)} \leftarrow E(\theta^{(i)})$;
5:    $\theta^{(i+1)} \leftarrow \theta^{(i)}$;
6:    **for** $c = 1, \ldots, p$ **do**
7:       $x_c^{(i+1)} \leftarrow$ linesearch $(x_c^{(i)}, \nabla_{x_c} E(D \cup D_p^{(i+1)}, \theta^{(i+1)}))$;
8:       $\theta^{(i+1)} \leftarrow \arg\min_\theta L(D \cup D_p^{(i+1)})$;
9:       $e^{(i+1)} \leftarrow E(\theta^{(i+1)})$;
10:   **end for**
11:   $i \leftarrow i + 1$;
12: **until** $\left| e^{(i)} - e^{(i+1)} \right| < \varepsilon$

Output: The final poisoning attack sample $D_p \leftarrow D_p^{(i)}$.
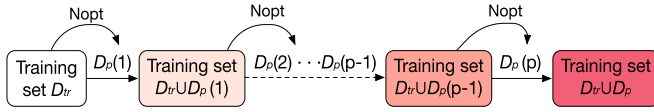

Fig. 3. The workflow of the poisoning attack algorithm.

the key difference between the work [1] and Nopt is that the optimization evaluated for each subsequent poisoning point is performed on a dataset $D_{tr} \cup D_p$ that includes all previous poisoning points, thereby creating a new poisoning point that maximizes the loss of the collective training dataset of original clean and poisoning points.

Algorithm 1 outlines the Nopt poisoning attack. As our loss function takes the same form as that of [1], the process to find the optimal points, we adopt the same gradient descent approach. In summary, vector x_c is updated through a line search along the direction of the gradient from the outer objective E (evaluated at the current iteration). The algorithm finishes when the outer objective E yields no further changes. Figure 3, illustrates the iterative process of Nopt to find out the poisoning points.

Figure 2 provides a visual representation of a contrived example in which our attack is applied with different poisoning rates. If the abscissa of the poisoning point must be located in the feasible domain, then we can still obtain an optimal solution within the feasible domain, and E will converge.

This application scenario is realistic as the attacker's goal was to create poisoning data-points to interfere with the original data-points without making the poisoning points appear abnormal, and thus compromises the secrecy of the attack. Therefore the attacker should determine the feasible domain before determining the location of the poisoning point through optimization. Through Equation (7), we obtain the set of poisoning points D_p with a specified poisoning degree E'.

C. Gradient Computation

The loss function we have defined in Equation (8) takes the same form as the loss function found in the work [1]. Consequently, the steps of the derivation and computation of the gradients are summarized below.

Algorithm 1 reduces to a gradient-ascent task with a line search. To compute the gradient ($\nabla_{x_c} E(\theta^{(p)})$), whilst capturing the relationship between θ and the poisoning point x_c, the chain rule can be used:

$$\nabla_{x_c} E = \nabla_{x_c} \theta(x_c)^T \cdot \nabla_\theta E, \tag{9}$$

where the first term captures the dependency of learned θ and the point x_c, and the second term is the derivative of the outer objective with respect to the regression parameters θ.

To solve $\nabla_{x_c} \theta(x_c)$ in the bilevel optimization problem, the inner learning problem is replaced with its Karush-Kuhn-Tucker (KKT) equilibrium condition, namely $\nabla_\theta L(D_{tr} \cup D_p, \theta) = 0$, while searching for x_c. This replacement is necessary as approximations will be required to solve the inner problem, particularly when the inner problem is not convex (when the inner problem is convex, it may be solved via its closed form expression). Imposing the derivative with respect to x_c satisfies this condition, $\nabla_{x_c} \nabla_\theta L(D_{tr} \cup D_p, \theta) = 0$. It is seen that L depends explicitly on x_c and implicitly through θ. One final application of the chain rule produces the linear system:

$$\nabla_{x_c} \nabla_\theta L + \nabla_{x_c} \theta^T \cdot \nabla_\theta^2 L = 0. \tag{10}$$

For our specific form of L given in Equation (4), the derivative follows:

$$\left(\frac{\partial \omega^T}{\partial x_c} \quad \frac{\partial b}{\partial x_c} \right) \begin{bmatrix} \Sigma + \lambda g & \mu \\ \mu^T & 1 \end{bmatrix} = -\frac{1}{n} \begin{bmatrix} M & \omega \end{bmatrix}, \tag{11}$$

where $\Sigma = \frac{1}{n} \Sigma_i x_i x_i^T$, $\mu = \frac{1}{n} \Sigma_i x_i$, and $M = \omega x_c^T + (f(x_c) - y_c) \mathbb{I} d$.

To jointly optimize the feature values x_c associated with their responses y_c, we need to consider the optimization of $z_c = (x_c, y_c)$. To do this, we replace ∇_{z_c} by ∇_{x_c} through expanding $\nabla_{x_c} \theta$ by incorporating derivatives with respect to y_c:

$$\nabla_{z_c} \theta = \begin{bmatrix} \frac{\partial \omega}{\partial x_c} & \frac{\partial \omega}{\partial y_c} \\ \frac{\partial b}{\partial x_c} & \frac{\partial b}{\partial y_c} \end{bmatrix}, \tag{12}$$

and, accordingly, we update Equation (11) as:

$$\nabla_{z_c} \theta^T = -\frac{1}{n} \begin{bmatrix} M & \omega \\ -x_c^T & -1 \end{bmatrix} \begin{bmatrix} \Sigma + \lambda g & \mu \\ \mu^T & 1 \end{bmatrix}^{-1}. \tag{13}$$

Therefore, when Algorithm 1 is used to implement this attack, both x_c and y_c are to be updated along the gradient $\nabla_{x_c} E$ (cf. Algorithm 1, line 7).

With this, we have to use tools to perform the optimization for our Nopt poisoning attack. Following the proposal of the defense algorithm shown in Section IV, in Section V we will evaluate our attack in comparison to previous poisoning attacks. We shall demonstrate that our subtle change in

Equation (8) produces larger errors compared to the previous poisoning attack on linear regression models, for the same given poisoning rates.

IV. DEFENSE AGAINST POISONING ATTACKS

In this section, we propose a new probabilistic defense algorithm, named *Proda*, which is designed to deal with regression learning poisoning attacks. We shall analyze its time complexity and efficiency. The objective of the Proda algorithm is similar to TRIM [1], which is to find the original training set through a subset solving algorithm, instead of trying to identify the poisoning set of the algorithm. If we randomly select γ points, the probability of all γ points belonging to the unpolluted training set is:

$$P_1 = (1 - \alpha)^\gamma, \qquad (14)$$

where α is the poisoning rate of the training set. If we randomly select β groups of γ points, the probability that no group of points belongs to the unpolluted training set is:

$$P = (1 - P_1)^\beta. \qquad (15)$$

When $P \leq \varepsilon$, a small positive constant, there must be a group of points all belonging to the unpolluted training set. We note that $\varepsilon = 10^{-5}$ shall be used later in our experiment after considering trade-offs between accuracy and time complexity. This means if the value of β follows the rules of Equation (16), there must be a group of γ points that all belong to the unpolluted training set.

$$(1 - (1 - \alpha)^\gamma)^\beta \leq \varepsilon, \qquad (16)$$

where α represents the poisoning rate. So the value of β is:

$$\beta = \log_{1-(1-\alpha)^\gamma} \varepsilon. \qquad (17)$$

The steps of Proda algorithm are illustrated in Figure 4, and formally detailed in Algorithm 2. However, intuitively, the Proda algorithm proceeds as follows:

1) Given α and γ, calculate the value of β.
2) Choose β groups of γ points at random. (This step can ensure that there must be a group of γ points that all belong to the unpolluted training set).
3) Each group of γ points is subjected to linear regression, resulting in β lines.
4) For each line, take the n points closest to this line.
5) Each group of n points is subjected to linear regression, resulting in β lines, and we obtain β groups of MSEs.
6) Find n points corresponding to the smallest MSE.

As Proda requires the pre-selection of γ, in Figure 5, we graphically display the effect of different selected γ on the resulting regression line.

A. Efficiency

According to Equation (7), we know that from the sets of points we randomly sample, there must be a group of points that all belong to the unpolluted dataset. Therefore, when comparing the MSE minimum values of β groups, even if there are poisoning points in the group of the smallest MSE,
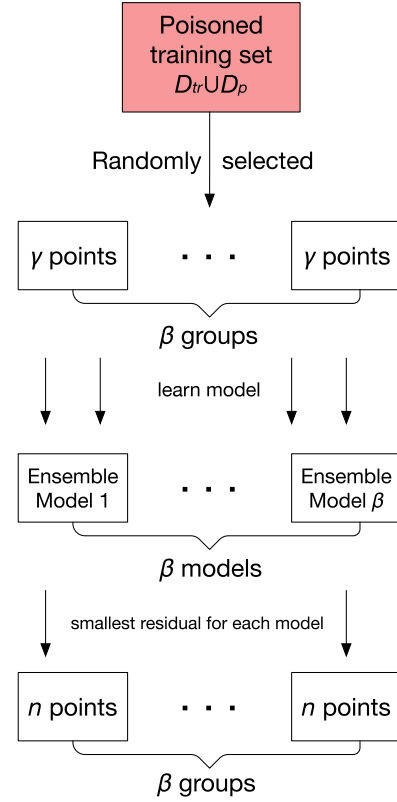

Fig. 4. The workflow of the defense algorithm.

Algorithm 2 Proda Algorithm
___
Input: Training data $D = D_{tr} \cup D_p$ of $|D| = N$; number of attack points $p = \alpha \cdot n$; γ; ε
1: $\beta = \varepsilon / (1 - a^\gamma)$
2: $i \leftarrow 1$ (iteration counter)
3: **for** $i \leq \beta$ **do**
4:    $J^{(i)} \leftarrow$ a random subset of size $\gamma \in \{1, \dots, a\}$;
5:    $L^{(i)} \leftarrow \arg\min_\theta L(J^{(i)}, \theta)$;
6:    $list^{(i)} \leftarrow$ distance of N points to $L^{(i)}$;
7:    $Q^{(i)} \leftarrow$ sorted$(list^{(i)})[: n]$;
8:    $S^{(i)} \leftarrow \arg\min_\theta L(Q^{(i)}, \theta)$;
9:    $M^{(i)} \leftarrow$ the MSE between $S^{(i)}$ and $Q^{(i)}$;
10:   $i \leftarrow i + 1$;
11: **end for**
12: $M^{(j)} \leftarrow \min(M^{(i)})$;
Output: The final optimizing sets $Q^{(j)}$.

---

they must conform to the original distribution and rules of the training dataset, and will have little impact on altering the regression model. Therefore, we argue that the Proda algorithm has good efficacy in mitigating the poisoning attack. An empirical evaluation on the efficiency of different defense algorithms will be demonstrated in Section V.

The goal is to select γ points that all belong to the original training set and that these points sufficiently reproduce the linear relationship of the original training set. Our algorithm can guarantee that the γ selected points are likely to all belong to the original training set, but whether these points can

represent the linear regression trend of the original training set depends on the value of γ.

It is known that for one-dimensional inputs, two points are required to define a linear relationship, between the input ($\mathbb{R}$) and the one dimensional output ($\mathbb{R}$), and for two dimensional inputs, three points, to define the linear relationship between the input ($\mathbb{R}^2$) and output ($\mathbb{R}$). Therefore, assuming the size of uncontaminated training sets is $m = n_o$, and the feature dimensionality of the training dataset is d, the minimum value of γ also needs to be at least one feature dimension greater than the input training set ($\mathbb{R}^d$), i.e., $\gamma \geqslant d + 1$.

When $\gamma = d + 1$, we can only guarantee that the line obtained by the defense algorithm is d-dimensional. Therefore, it is very difficult to make randomly selected γ points in training set of m points that conform to the original trends of the dataset.

Assuming that the points in the unpoisoned training set are evenly distributed, we can obtain the relationship between the MSE of the original training set D_{tr} on its corresponding line θ_{tr} and the MSE of the γ points D_γ on its corresponding line θ_γ, as shown in Equation (18):

$$\text{MSE}(D_{tr}, \theta_{tr}) \leq \frac{m}{\gamma} \cdot \text{MSE}(D_\gamma, \theta_\gamma). \tag{18}$$

As γ increases, the difference between $\text{MSE}(D_{tr}, \theta_{tr})$ and $\frac{m}{\gamma} \cdot \text{MSE}(D_\gamma, \theta_\gamma)$ becomes smaller, defense algorithms would also be more efficient, but the corresponding time complexity will become worse (We will analyze the time complexity of the defense algorithm in detail in Section IV-B.). While knowledge of α appears to be an essential parameter for Proda, the defense algorithm will still operate for an assumed value of α (0.2, a safe assumption due to practical bounds [1]). Recall that Proda creates groupings of points and selects the set of points least likely to contain poisoning points as a representative set for the dataset. Thus, if there are less poisoning points than assumed, the representative set should still only contain clean points representative of the data distribution. The largest consequence of assuming a worst case scenario for α is the increase in the time complexity, as we shall discuss in the next section; however, this still remains smaller than that of the competing defense of TRIM.

B. Time Complexity

We now compute the time complexity of our proposed defense. According to Equation (7), we know that the probability of at least one group of γ points all belonging to the unpolluted data set is:

$$P_u = 1 - (1 - (1 - \alpha)^\gamma)^\beta. \tag{19}$$

When $P \leq \varepsilon$, P_u approaches 1. The value of β satisfies:

$$\beta \geqslant \log_{1-(1-\alpha)^\gamma}(1 - P_u). \tag{20}$$

That is, at least β times of random point selection can be carried out to obtain a high probability of obtaining a group of γ points which are all uncontaminated data.

Thus, the time complexity of Proda algorithm is:

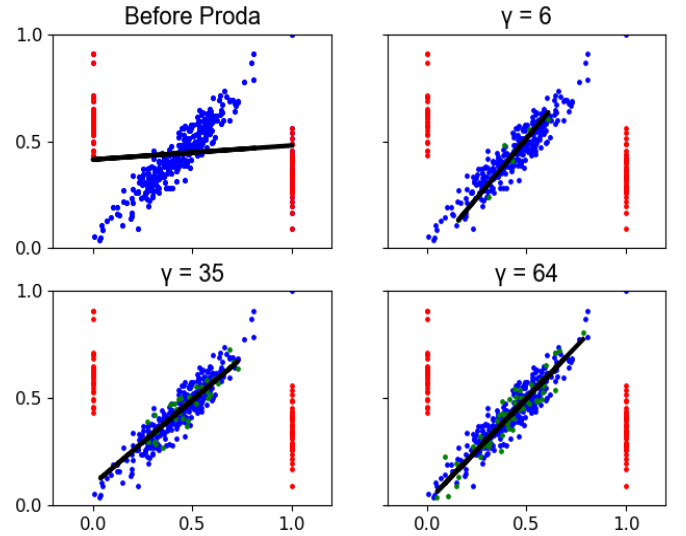$$T(n) = O(n) \times \log_{1-(1-\alpha)^\gamma}(1 - P_u), \tag{21}$$

Fig. 5. Different parameters γ used in the Proda algorithm. The original unpoisoned data-points are shown in blue, poisoning datapoints in red, and defensive data-points generated by the Proda algorithm in green, where green data-points are subsets of blue ones. The Proda algorithm screens out the subsets of uncontaminated datasets.

where $O(n)$ is the time complexity of the linear regression algorithm. We note that if the feature dimensionality of the training data set is d, and $\gamma \geqslant d + 1$, the minimum time complexity can be found as $T(n) = O(n) \times \log_{1-(1-\alpha)^{d+1}}(1 - P_u)$.

In contrast, for the time complexity of the TRIM algorithm, which we obtain by inspecting the TRIM algorithm, the worst case of TRIM involves the traversal of all n training subsets; thus, TRIM may iterate $\binom{M}{m}$ times.[4] Therefore, when the time complexity of TRIM is compared to our Proda algorithm, a huge improvement is expected.

V. EXPERIMENTAL ANALYSIS

Note that our poisoning attack algorithm in Section III is stated without any assumptions on the training data distribution. In practice, such information on training data is typically unavailable to attackers. Moreover, an adaptive attacker can also inject poisoning samples to modify the mean and covariance of training data. Thus, we argue that our attack algorithm results are stronger than prior works as we rely on fewer assumptions except for the work [1] (IEEE S&P 2018) of which the assumptions the same as ours. To evaluate the effectiveness of our attacks and defenses, we have selected two key metrics: MSE to measure the effects of poisoning as a result of the attacks and defenses (or the "success" of the attack), and the time complexity of each configuration; this time is computed by multiplying the number of iterations required for each configuration (we assume average of 1000 iterations per $1 \mu s$ on our computer hardware).

Our attack and defense algorithms were implemented in Python 3.7, leveraging the NumPy and scikit-learn libraries. We use a conventional cross-validation method to split the datasets into three equally-sized sets for training, validation,

---

[4]No time complexity analysis was provided in TRIM's proposal [1].

(a) House Prices Dataset   (b) Loans Dataset   (c) Pharm Dataset   (d) Bike Sharing Dataset
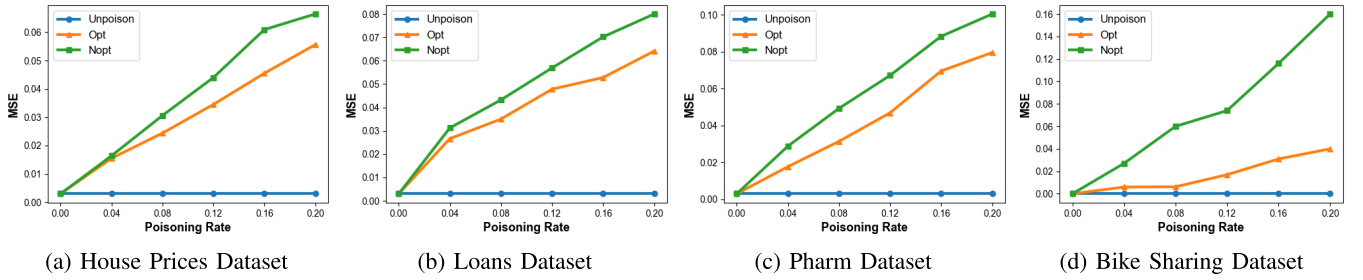
Fig. 6.   Mean Squared Error (MSE) of poisoning attacks on OLS regression on the four datasets.
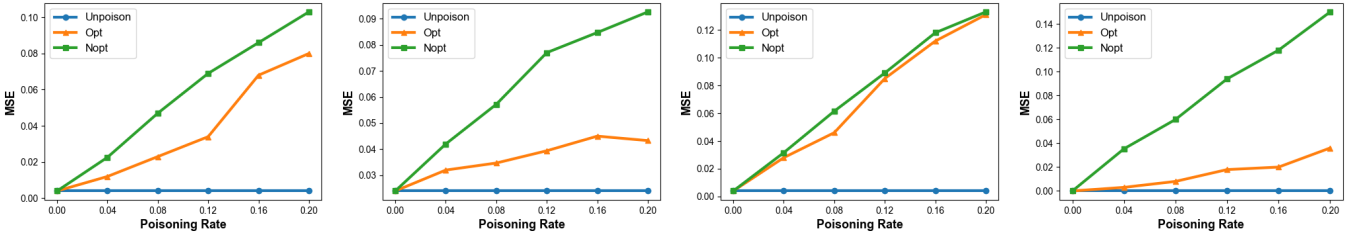

Fig. 7.   MSE of attacks on Ridge regression on the four datasets.

and testing. To ensure data splitting biases are not introduced, we repeat each experiment and average results over 5 independent runs.

The remainder of this section is laid out as follows. We first describe the datasets used in our experiments in Section V-A. Followed by Section V-B, we compare results obtained from our poisoning optimization algorithm with previous poisoning attacks on the datasets we have obtained, across four different types of regression models. Finally, we present the results of our Proda algorithm and compare it with previous defenses in Section V-C.

A. Datasets

We first introduce the four publicly available datasets used in our evaluation.

- **Housing Prices** [25], a dataset used for predicting the price of the house at the time of sale given attributes of the house structure, and location information.
- **Loans** [26], a lending dataset that seeks to estimate the appropriate interest rate of a loan given information about the total loan size, interest rate, amount of principal paid off, and the borrower's personal information such as credit status, and state of residence.
- **Pharm** [27], a pharmaceuticals dataset that estimates the dosage of Warfarin for a patient depending on physical attributes of said patient, such as age, height, and weight.
- **Bike Sharing** [28], the Capital Bikeshare system dataset estimates the number of vehicles within a certain time period, given hourly information of rental bikes and environmental variables like weather.

The aforementioned datasets, with the exception of the bike sharing dataset, have been used in previous evaluations of poisoning attacks on regression models [1].[5] All datasets

[5]We have used the datasets available at https://github.com/jagielski/manip-ml.

are pre-processed in the same manner, with categorical variables one-hot encoded, and numerical features normalized between 0 and 1. This produces 275, 89, 204, and 15 features for Housing, Loans, Pharm, and Bike Sharing, respectively. In total, each dataset contains 1460, 887383, 4683, and 17389 records, respectively. However, due to computational limitations, the only first 5, 4, 3, and 8 features of the Housing Prices, Loans, Pharm, Bike Sharing datasets, respectively, were used in the defense evaluation.

B. Nopt Poisoning Attack

We now perform experiments on the four selected regression datasets to evaluate our newly proposed attack. In addition, we compare our results to MSEs of the clean dataset and the optimization attack as proposed by Jagielski *et al.* [1]. We use MSE as the metric for assessing the effectiveness of an attack, and also compute the attacks' time complexity. We vary the poisoning rate between 4% and 20% at intervals of 4% with the goal of inferring the trend in attack success. Figures 6, 7, 8 and 9 show the MSE of each attack on OLS, Ridge, LASSO, and Elastic-net regression, respectively. We note that Jagielski *et al.* [1] had only evaluated Ridge and LASSO regression. We plot results for the clean dataset (called Unpoison), optimization attack, proposed by Jagielski *et al.* [1] (called Opt), in addition to our Nopt attack. The horizontal coordinate is the poisoning rate α, that is, the proportion of pollution data in the new training dataset, and the vertical coordinate is the MSE of the model trained on the poisoned dataset. Overall it can be observed from the diagrams that Nopt is able to achieve larger MSE (in comparison to Opt) for the same poisoning rate. While every configuration of the regression type, dataset and poisoning rate observes Nopt exceeding Opt, we note that Figures 7(c) and 8(c) show a relatively similar increase of MSE between the two attacks. We also observe that OPT is more suitable for Ridge
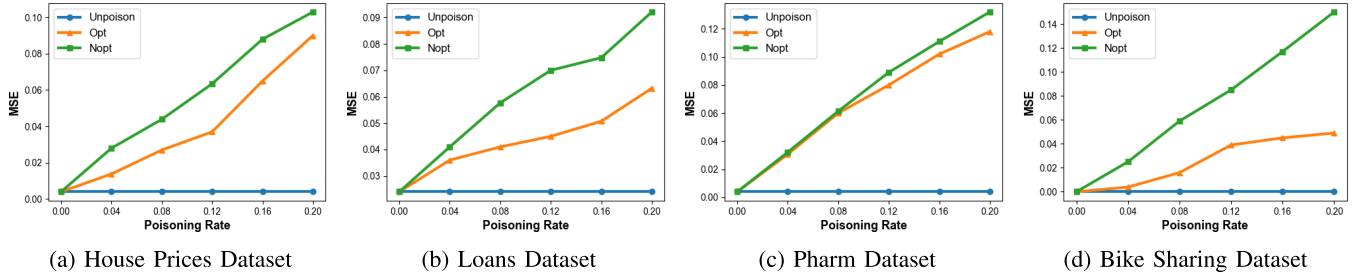
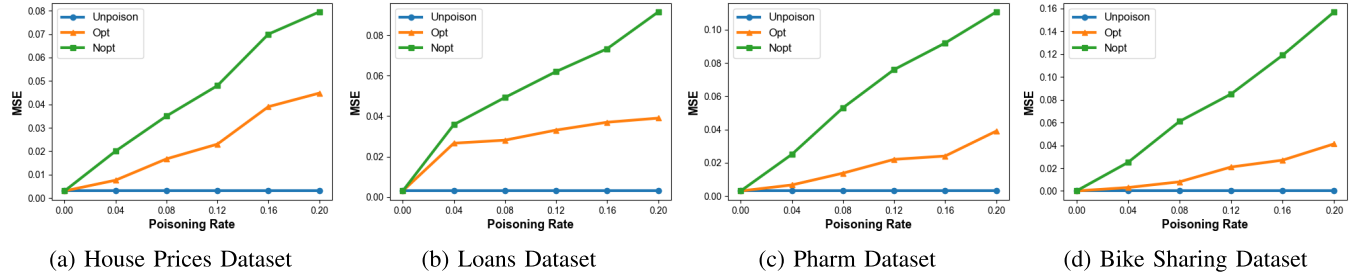Fig. 8. MSE of attacks on LASSO regression on the four datasets.

Fig. 9. MSE of attacks on Elastic-net regression on the four datasets.

Regression and Lasso Regression on PARM Dataset; However, NOPT is seen to produce more stable attack performance.

In Table I, we detail the specific MSEs of our new attack (Nopt) and the optimization attack proposed by Jagielski *et al.* [1] (Opt). For this table, we reproduce the numerical value of the MSE at a fixed poisoning rate of $\alpha = 0.2$. From Table I we can observe that when the poisoning rate is $\alpha = 0.2$, our attack (Nopt) is again consistently higher than that of the previous attack algorithm (Opt). Our results confirm that the optimization framework we design demonstrates increased effectiveness when poisoning both different linear regression models and across datasets. The Nopt attack can achieve MSEs with a factor of 1.3 higher than the Opt attack in the House dataset, a factor of 1.5 higher in the Loan dataset, a factor of 1.2 higher than Opt in the Pharm dataset, and a factor of 4.0 higher than Opt in the bike sharing dataset. We note that the primary difference between NOPT and OPT is the difference in the objective function. The objective function of the OPT algorithm is determined to make the model post-poisoning deviate a maximal amount from the original model. The objective function of NOPT, however, is to perturb the original training set to become more disordered, thereby increasing the degree of dispersion of the training data set, and thus affecting the MSE of the final model on the training data set. Therefore, it is expected that the NOPT algorithm will produce more effective poisoning than the OPT algorithm, a property we have demonstrated through experimentation.

C. Proda Defense Algorithm

In this section, we evaluate if our Proda algorithm can effectively defend against the optimized attack (Nopt), the attack that produced the highest MSE in Section V-B.

TABLE I

COMPARISON OF MSE BETWEEN OPT AND OUR NEWLY PROPOSED NOPT POISONING ALGORITHMS. IT CAN BE OBSERVED THAT IN ALL CONFIGURATIONS, OUR ATTACK ACHIEVES A LARGER MSE

Dataset	Regression	MSE after Poisoning ($\alpha = 0.2$)	
		Opt	Nopt
House Prices	OLS	0.055	0.07
	Ridge	0.07	0.10
	LASSO	0.085	0.10
	Elastic-net	0.04	0.08
Loans	OLS	0.061	0.081
	Ridge	0.043	0.095
	LASSO	0.062	0.094
	Elastic-net	0.032	0.091
Pharm	OLS	0.077	0.10
	Ridge	0.134	0.145
	LASSO	0.11	0.14
	Elastic-net	0.031	0.11
Bike Sharing	OLS	0.04	0.16
	Ridge	0.036	0.15
	LASSO	0.049	0.15
	Elastic-net	0.0413	0.157

We shall use two measures in the experiment: the difference of MSE and time complexity. We evaluated the MSE among the resulting dataset of the Proda algorithm, the resulting dataset of TRIM algorithm, and the clean dataset. We also evaluated the time complexity of our defense by measuring the running time of the algorithm. For the probability algorithm, different γ values will produce different operation results and time complexities; thus our experiments may also evaluate results for different values of γ, however unless otherwise stated, the default value of ε in our experiments is 10^{-5}.

1) Defense Efficiency: When the experiment was performed, γ and α are assumed known to be the defender (Recall

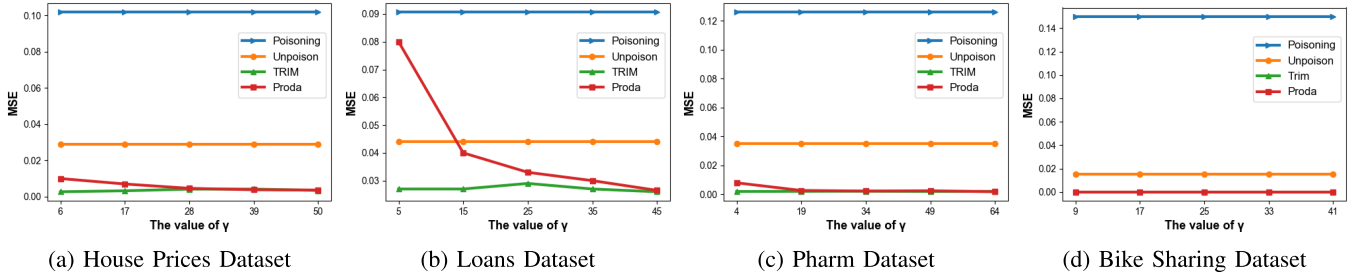| (a) House Prices Dataset | (b) Loans Dataset | (c) Pharm Dataset | (d) Bike Sharing Dataset |

Fig. 10. The effects of increasing γ on the efficiency of defense algorithms on our four datasets (The poisoning algorithm is Nopt on LASSO regression, poisoning rate $\alpha = 0.2$.). Both TRIM and Proda algorithms have smaller MSE results than clean data sets. When the γ value is large, the defensive effect of the two algorithms converge. In the bike sharing dataset, two lines of TRIM and Proda overlap. The Proda algorithm achieves a satisfying defense effect when γ is set to be the minimum value $d + 1$. There is no general value of γ, as can be seen from Fig. 10. For each dataset, as γ increases, the MSE of the training set when a defense is applied will plateau. Therefore, when increasing γ no longer significantly improving the MSE, we fix this value as the appropriate γ value.

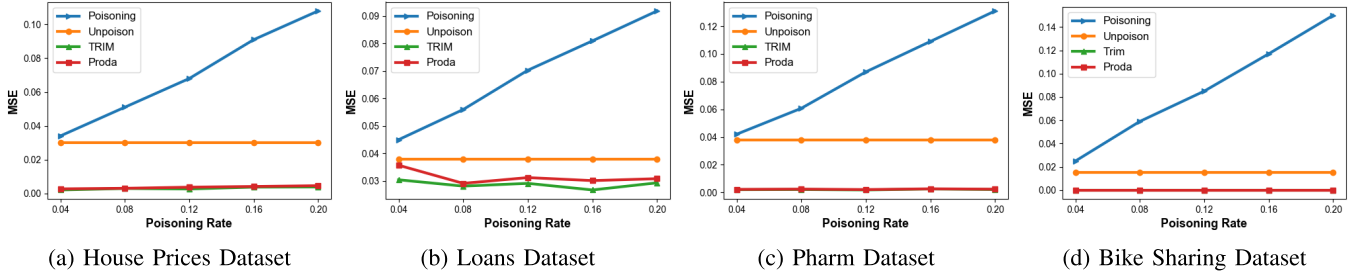| (a) House Prices Dataset | (b) Loans Dataset | (c) Pharm Dataset | (d) Bike Sharing Dataset |

Fig. 11. The effects of different poisoning rates α on the efficiency of defense algorithms on our four datasets (The poisoning algorithm is Nopt of LASSO regression, $\gamma = 50$.) Both Proda and TRIM algorithms display good defensive performances at various poisoning rates. In the house price dataset, pharm and bike dataset, the two trends overlap.

from Section II-B.2, α is known by the defender, as they can assume that every sample submitted to the learning process is likely to be malicious; in the event that the α is unknown and cannot be found by the defender, the defender can assume an upper bound of 0.2 poisoning rate.). Therefore in specific cases, β, the intermediate variable, would not change. So, if γ is given, MSE varies only with the poisoning rate (Figure 11, for a fixed $\gamma = 50$). If a poisoning rate α is given, MSE varies only with γ (Figure 10, for a fixed $\alpha = 0.2$).

We remark that the value of γ has been set to a specific value instead of a percentage of the training set. Empirically, we observed the selection of γ had no clear correlation with the size of the training sets, instead the value of γ is more strongly associated with the number of training set features. Specifically, we observed that the minimum value of γ should be greater than the number of training set features ($d + 1$). Consequently, we have analyzed in more detail the relationship between γ and the number of features in the training set. Figure 10 shows the resulting MSE of the poisoning attack when setting different values of γ for the defense algorithm. We can see that when γ is small, the defense efficacy of Proda is less than that of TRIM (with the exception of the bike sharing dataset, where the MSE is the same), but with an increase of γ, the effectiveness of Proda in removing the influence of poisoning samples increases, however at the cost of additional computational time (see Table III). As we have previously described in Section IV, Proda seeks to find a subset of data that contains only clean samples, as the presence of poisoning points will increase the MSE of the

resulting regression (in the overall model and in each subset evaluated by Proda), poisoning points that greatly increase the dispersion of the subset (and thus MSE) will be discarded by Proda. In Figure 10(b), when γ is equal to twice the number of features, the MSE of the probability algorithm result is lower than the MSE of the clean dataset, while the remaining datasets observe lower MSE compared to the clean dataset for all values of γ. This indicates that a proper γ can induce not only an effective defense result observed, but it also has assisted in the generalization of the model. We assert that the result is obtained as Proda also removes poor training points that may exist in the clean dataset. Interestingly, Proda on the Loan's dataset at very small values of γ demonstrates an MSE larger than that of the clean dataset, given that the Loans dataset contains the largest number of records. Selecting too small a set of points to act as the representative set will negatively impact the resulting regression. It can be seen from Figures 10 and 11 that the resulting MSE of TRIM algorithm is lower than the resulting MSE of the probability algorithm and lower than the MSE of the clean dataset until γ is much larger. *We note that we have not evaluated other defenses since Jagielski et al. [1] have shown that TRIM, their state of the art, is capable of outperforming prior defense mechanisms.* In the bike sharing dataset, we can see from Figure 10(d) that three lines of Unpoison, TRIM, and Proda are almost compressed to one line, indicating their performances resemble each other. Proda has achieved a satisfying defense effect when the parameter γ is set as a minimum threshold, i.e., $\gamma = d + 1$. Although the three lines overlap almost completely all the way

to $\gamma = 41$, if we further zoom in, the performance of Proda's defense will boost as the γ increases.

An Unknown Poisoning Rate α: Oftentimes the defender has zero knowledge of the poisoning rate α used by an attacker; however, due to the construction of the Proda algorithm, it can still protect against poisoning attacks of different poisoning rates, when the defender assumes a worst case scenario of $\alpha = 0.2$ (the argued largest realistic poisoning rate [1]). In Table II, we demonstrate results for the Proda algorithm run with a known α (as previously seen in Figure 11), and the Proda algorithm when executed with an assumed $\alpha = 0.2, 0.1$. From the table, we can observe that there is a small decrease in the MSE, when the assumed poisoning rate is larger than the true rate of α. Conversely, in the event that a defender underestimates the poisoning rate α used by an attacker (e.g., a conservative estimate was used), we can observe from Table II and Figure 12 that once the real poisoning rate (α) exceeds the defender's assumed poisoning rate, the MSE of the model exceeds that of the clean MSE. However, this increase still remains below that of the undefended MSE. This trend can be consistently observed across all datasets. Thus, it is recommended that the poisoning rate is set higher than any realistic poisoning rate for the strongest defense at the expense of time complexity (as we shall analyze in the following section); however, even a conservative estimate (for less time complexity) will still yield MSE reductions compared to no defense.

2) Time Complexity: In Section IV-B, we analyzed the time complexity of Proda and TRIM, and found that the worst case of Proda is superior to TRIM. In both Tables III and IV, we detail the respective computed time complexities of Proda and TRIM, for different values of γ (see Table III), and different values of α (see Table IV). We reiterate that the time complexity indicated in these tables is computed by the number of iterations established in Section IV-B with an assumed average processing speed of 1000 iterations per $1\mu s$ on our computing hardware.

In Table III, for a fixed $\alpha = 0.2$, the time complexity of Proda algorithm is directly related to γ and the number of features (recall that 5, 4, 3, and 8 features were used in House, Loan, Pharm, and Bike Sharing datasets, respectively), while the time complexity of the TRIM algorithm is related to the training data size (The size of our clean training set is 300.). From the average and worst-case time complexities shown in Table IV, it can be observed that TRIM is faster than the Proda on the average case; however, our Proda algorithm provides an upper bound on the time complexity. In Table IV, we observe time complexities for a fixed $\gamma = 50$, but a varying poisoning rate α, we can see that the worst-case time complexity of the two algorithms increases with the increase of the poisoning rate α; however, as we have shown earlier in Section IV-B, Proda is bounded by the worst-case scenario.

Note that TRIM algorithm uses an iterative search to find the smallest MSE subset as its defense. Suppose the size of the poisoned training set is k. The worst-case scenario for TRIM is to exhaust $\binom{k}{(1-\alpha)k}$ subsets to converge. In the case of a poisoning rate of $\alpha = 0.2$, TRIM will not terminate until it has compared MSEs of $\binom{k}{0.8k}$ subsets ($\geqslant 1.6^k$ subsets,

Dataset	Poisoning Rate α (Assumed)	Poisoning Rate α (Real)	MSE (Clean)	MSE (Proda)
House Prices	10%	4%	0.03	0.001
		12%	0.03	0.032
		20%	0.03	0.075
	20%	4%	0.03	0.002
		12%	0.03	0.013
		20%	0.03	0.005
	4%	4%	0.03	0.012
	12%	12%	0.03	0.003
	20%	20%	0.03	0.005
Loans	10%	4%	0.038	0.023
		12%	0.038	0.041
		20%	0.038	0.069
	20%	4%	0.038	0.014
		12%	0.038	0.018
		20%	0.038	0.028
	4%	4%	0.038	0.029
	12%	12%	0.038	0.029
	20%	20%	0.038	0.028
Pharm	10%	4%	0.036	0.001
		12%	0.036	0.037
		20%	0.036	0.104
	20%	4%	0.036	0.001
		12%	0.036	0.001
		20%	0.036	0.003
	4%	4%	0.036	0.003
	12%	12%	0.036	0.004
	20%	20%	0.036	0.003
Bike Sharing	10%	4%	0.015	1.37×10^{-19}
		12%	0.015	0.011
		20%	0.015	0.068
	20%	4%	0.015	8.94×10^{-20}
		12%	0.015	1.02×10^{-19}
		20%	0.015	1.78×10^{-19}
	4%	4%	0.015	1.96×10^{-19}
	12%	12%	0.015	1.69×10^{-19}
	20%	20%	0.015	1.78×10^{-19}

given Stirling's approximation). In contrast, Proda algorithm squarely selects the smallest MSEs of $1 - \alpha k$ clean data-points from our randomly chosen subsets; hence, the time complexity of Proda algorithm is dependent only on the value of the parameter γ we define, consequently offering a substantial reduction in the worst-case time complexity of such a defense.

3) Effectiveness of Proda Against Opt and Nopt: Both Opt and Nopt add previously poisoned data points during the optimization phase; thus it is desirable to compare whether the two different attack algorithms will perform differently against the same defense of Proda. For all four datasets and across three values of α (4%, 12%, 20%), it is observed in Table V that there is little to no difference between attack algorithms.

VI. DISCUSSION AND RELATED WORK

In this section, we discuss the limitations of this paper and survey the related work.
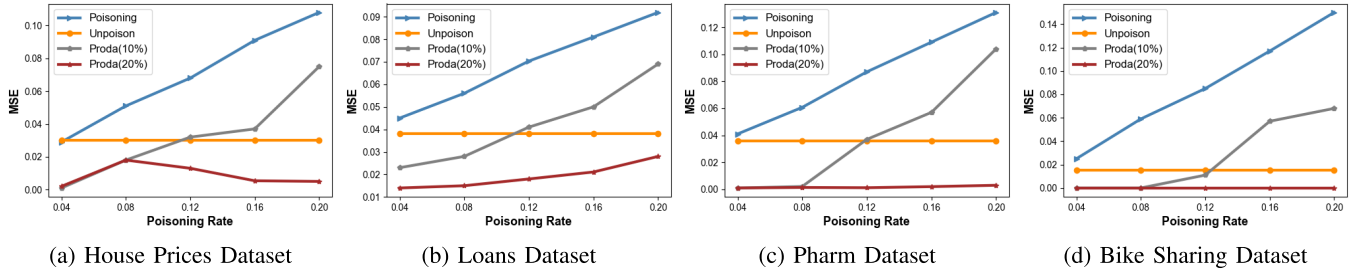
(a) House Prices Dataset   (b) Loans Dataset   (c) Pharm Dataset   (d) Bike Sharing Dataset

Fig. 12.   Comparison of defense performance when α is varied for Proda on four datasets (The poisoning attack used for this evaluation is Nopt on Ridge regression with $\gamma = 50$. The size of our clean training set is 300.). However, the defender does not possess knowledge about the attacker's poisoning rate, instead assuming a poisoning rate of 0.1 (10%) and 0.2 (20%).

TABLE III

COMPARISON OF DEFENSE ALGORITHM TIME PERFORMANCE WHEN γ IS VARIED FOR PRODA ON FOUR DATASETS (THE POISONING ATTACK USED FOR THIS EVALUATION IS NOPT ON RIDGE REGRESSION WITH THE POISONING RATE $\alpha = 0.2$. THE SIZE OF OUR CLEAN TRAINING SET IS 300)

Dataset	Algorithm	γ	Time Complexity (average, s)	Time Complexity (worst case, s)
House Prices	TRIM	-	0.021	$\geq 5^{80}$
	Proda	6	0.037	0.037
		28	5.946	5.946
		50	806.646	806.646
Loans	TRIM	-	0.021	$\geq 5^{80}$
	Proda	5	0.028	0.028
		25	3.041	9.294
		45	264.318	264.318
Pharm	TRIM	-	0.021	$\geq 5^{80}$
	Proda	4	0.021	0.021
		30	9.294	9.294
		60	7512.528	7512.528
Bike Sharing	TRIM	-	0.021	$\geq 5^{80}$
	Proda	9	0.079	0.079
		25	3.041	3.041
		41	108.261	108.261

TABLE IV

COMPARISON OF DEFENSE ALGORITHM TIME PERFORMANCE WHEN THE POISONING RATE α IS VARIED ON FOUR DATASETS (THE POISONING ATTACK IS NOPT ON RIDGE REGRESSION WITH A FIXED $\gamma = 50$. THE SIZE OF OUR CLEAN TRAINING SET IS 300)

Dataset	Algorithm	α	Time Complexity (average, s)	Time Complexity (worst case, s)
House Prices	TRIM	4%	0.021	$\geq 5^{18}$
		12%	0.021	$\geq 5^{43}$
		20%	0.021	$\geq 5^{80}$
	Proda	4%	0.143	0.143
		12%	12.162	12.162
		20%	806.646	806.646
Loans	TRIM	4%	0.021	$\geq 5^{18}$
		12%	0.021	$\geq 5^{43}$
		20%	0.021	$\geq 5^{80}$
	Proda	4%	0.143	0.143
		12%	12.162	12.162
		20%	806.646	806.646
Pharm	TRIM	4%	0.021	$\geq 5^{18}$
		12%	0.021	$\geq 5^{43}$
		20%	0.021	$\geq 5^{80}$
	Proda	4%	0.143	0.143
		12%	12.162	12.162
		20%	806.646	806.646
Bike Sharing	TRIM	4%	0.021	$\geq 5^{18}$
		12%	0.021	$\geq 5^{43}$
		20%	0.021	$\geq 5^{80}$
	Proda	4%	0.143	0.143
		12%	12.162	12.162
		20%	806.646	806.646

A. Limitations

Supervised machine learning algorithms can solve common regression and classification problems, but the training data of supervised machine learning may potentially be manipulated by attackers seeking to interfere with the training process for their own nefarious purposes. For example, an attacker may add poisoning data to interfere with a supervised machine learning algorithm and likewise, a defender can use data optimization to prevent data poisoning attacks. In data poisoning attacks, attackers may have a variety of goals to interfere the regression result. In this work we focus on maximizing the dispersion of the training set. As we have discussed earlier, an investigation into the possibility of a targeted attack is left for future work.

When defending against poisoning attacks, the Proda algorithm seeks to find a subset of points that minimize the linear regression loss function. In prior work the TRIM [1], a defense algorithm, has been proposed against linear regression poisoning attacks. Both TRIM [1] and our work are based on the premise that the loss function of linear regression

greatly increases with the addition of even a single poisoned point, a method that differs from other approaches which seek to classify or isolate poisoned points. What our approaches differ is that the worst-case time complexity of our defense algorithm Proda is better than TRIM [1]. While we have experimentally demonstrated the effectiveness in removing the effect of poisoning attacks, it is possible that Proda can be retooled to detect the subset of poisoning points.

Our Proda algorithm has a parameter γ, which controls the precision and time complexity of the defense algorithm. The setting of the parameter γ is data-dependent; however, we have established that γ should not be less than one more than the dimension of the dataset $d + 1$. Experimentally, we observed that a larger γ will produce lower MSE at the expense of a higher time complexity. It can be seen from Figure 10 that when γ approaches d^2, Proda algorithm can obtain basically

TABLE V

COMPARISON OF MSE AFTER DEFENSE BETWEEN OPT AND OUR NEWLY
PROPOSED NOPT POISONING ALGORITHMS WHEN THE DEFENSE
ALGORITHM IS THE SAME (THE POISONING ALGORITHM IS NOPT
OF LASSO REGRESSION. THE SIZE OF OUR CLEAN TRAINING
SET IS 300. THE DEFENSE USED FOR THIS EVALUATION IS
PRODA, $\gamma = 70$)

Dataset	α	MSE (Clean)	MSE Opt	MSE Nopt	MSE (Opt after Proda)	MSE (Nopt after Proda)
House Prices	4%	0.03	0.055	0.07	0.012	0.012
	12%	0.03	0.07	0.10	0.003	0.003
	20%	0.03	0.085	0.10	0.005	0.005
Loans	4%	0.038	0.061	0.081	0.029	0.028
	12%	0.038	0.043	0.095	0.029	0.029
	20%	0.038	0.062	0.094	0.028	0.028
Pharm	4%	0.036	0.077	0.10	0.004	0.003
	12%	0.036	0.134	0.145	0.004	0.004
	20%	0.036	0.11	0.14	0.004	0.003
Bike Sharing	4%	0.015	0.04	0.16	1.89×10^{-19}	1.96×10^{-19}
	12%	0.015	0.036	0.15	1.72×10^{-19}	1.69×10^{-19}
	20%	0.015	0.049	0.15	1.87×10^{-19}	1.78×10^{-19}

stable defense efficiency. The specific value of γ should be chosen based on a time complexity acceptable to the defender.

B. Related Work

1) Poisoning Attacks: Data poisoning attacks are a general class of attacks that manipulate the training data of a machine-learning system such that the learned model behaves in a way dictated by the attacker. Such poisoning techniques have been studied in various applications, such as anomaly detection [10] and email spam filtering [9]. It has also been shown that data poisoning attacks are indifferent to the underlying machine learning algorithms (SVMs [3], regression [1], [29], [30], graph-based approaches [6], neural networks [19], [31]–[34], and federated learning [35]). While the aforementioned works primarily compromise classification tasks, there has also been work on poisoning recommender systems [36]–[38]. Most relevant to ours are attacks against regression tasks [1], [5], whereas our attack outperforms the prior works.

2) Defenses Against Poisoning Attacks: Many defense mechanisms have been proposed to defend against poisoning attacks [9]–[11], [13]. One core approach to mitigating poisoning attacks is to recognize that the poisoned data is intentionally dissimilar to the clean training data (as to inflict the greatest change to the learning process). Due to the limitation on the proportion of poisoned data to clean data an attacker has control over, the poisoned data may be treated as outliers, and mitigated with data sanitization techniques [14]–[16]. The other means to defend against potentially poisoned data is to use robust learning algorithms [1], [10], [17], [18]. These algorithms are designed to limit the sensitivity to any single sample within the training data. Differential privacy has also been investigated as a means to mitigate data poisoning attacks [8]. Specifically for regression models, the closest to this work is TRIM [1], a defense capable of managing a large number of poisoning data points. Like Proda, TRIM finds a subset of training data that minimizes the model loss, and uses this subset as a representative set to train the non-poisoned model.

Presently, the defense methods against deep learning poisoning attacks can be divided into three stages: data and feature modification, model modification, and output defense.

Data and feature modification [9]–[11], [13] primarily refers to the processing of data or features before it is accepted as input into the model to achieve the defense objective.

A core tenet to mitigating poisoning attacks is to recognize that the poisoned data is intentionally dissimilar to the clean training data (as to inflict the greatest change to the learning process). Due to limitations to the proportion of poisoned data to clean data an attacker can control, the poisoned data is the minority class and may be treated as outliers, and mitigated with data sanitization techniques [14]–[16]. Another method to defend against potentially poisoned data is to use robust learning algorithms [1], [10], [17], [18]. These algorithms are designed to limit the model's sensitivity to any single sample within the training data. Differential privacy has also been investigated as a means to mitigate data poisoning attacks [8]. Specifically for regression models, TRIM [1] is a defense capable of managing a large number of poisoning data points. Like Proda, TRIM finds a subset of training data that minimizes the model loss, and uses this subset as a representative set to train the non-poisoned model. From the perspective of preprocessing features, Shen *et al.* [39] propose a method to automatically produce shielding features to identify and neutralize abnormal distributions.

Directly protecting the model involves the modification of the deep learning model. Liu *et al.* [40] propose a pruning algorithm to reduce the size of the backdoor networks by eliminating dormant neurons on pure inputs. Iandola *et al.* [41] propose a fine-tuning defense in which a potentially poisoned neural network is updated to disable backdoor triggers. DeepInspect [42], a detection framework, is proposed to leverage a conditional generation model to learn the probability distribution of potential triggers from the queried model to retrieve fingerprints left behind during the insertion of the backdoor.

An output defense mitigates an attack by analyzing the output behavior of the potentially poisoned deep learning model. Yang *et al.* [43] propose a loss-based method, which would trigger an accuracy check if the loss of the target model exceeded a threshold multiple times. Hitaj and Mancini [44] presents an integrated defense, by combining the prediction results of different models to judge the prediction categories of samples. Chandola et al's work detects poisoned inputs through Support Vector Machines (SVM) and decision trees [45]. Zhao *et al.* [46] propose the multi-task model defense and analyze the output results of the model through data cleaning to improve the robustness of multi-task joint learning.

VII. CONCLUSION

We systematically study the poisoning attack and its defense for regression models in MLaaS setting. We have proposed a modification to an attack optimization framework that requires no additional knowledge of the training process, yet produces better offensive results. Our attack allows attackers to carry out

both white-box and grey-box attacks and is capable of increasing the dispersion of the poisoned training set. However, as attackers may have many goals in the interference of the regression outcome, the possibility of targeted attacks remains in question. To respond to a more powerful poisoning attack developed in this paper, we designed a probabilistic defense algorithm, Proda, which can be tuned to effectively mitigate poisoning attacks on the regression model while significantly reducing the worst-case time complexity. We highlight that the time complexity of the state-of-the-art defense, TRIM, had not been estimated; however, we deduce from their work that TRIM can take exponential time complexity in the worst-case scenario, in excess of Proda's logarithmic time. Finally, we hope that our work will inspire future research to develop more robust learning algorithms that are not susceptible to poisoning attacks.

ACKNOWLEDGMENT

REFERENCES

[1] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 19–35, doi: 10.1109/SP.2018.00057.

[2] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Jose, CA, USA, May 2017, pp. 3–18, doi: 10.1109/SP.2017.41.

[3] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proc. 29th Int. Conf. Mach. Learn. (ICML)*, 2012, pp. 1467–1474. [Online]. Available: http://icml.cc/2012/papers/880.pdf

[4] L. Muñoz-González *et al.*, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, Nov. 2017, pp. 27–38.

[5] S. Mei and X. Zhu, "Using machine teaching to identify optimal training-set attacks on machine learners," in *Proc. 29th AAAI Conf. Artif. Intell.*, Austin, TX, USA, Jan. 2015, pp. 2871–2877. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9472

[6] B. Wang and N. Z. Gong, "Attacking graph-based classification via manipulating the graph structure," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 2023–2040, doi: 10.1145/3319535.3354206.

[7] M. Fang, N. Z. Gong, and J. Liu, "Influence function based data poisoning attacks to top-n recommender systems," in *Proc. Web Conf. (WWW)*, 2020, pp. 3019–3025, doi: 10.1145/3366423.3380072.

[8] Y. Ma, X. Zhu, and J. Hsu, "Data poisoning against differentially-private learners: Attacks and defenses," in *Proc. 28th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2019, pp. 4732–4738, doi: 10.24963/ijcai.2019/657.

[9] B. Nelson *et al.*, "Exploiting machine learning to subvert your spam filter," in *Proc. 1st USENIX Workshop Large-Scale Exploits Emergent Threats (LEET)*, 2008, pp. 1–9. [Online]. Available: http://www.usenix.org/events/leet08/tech/full_papers/nelson/nelson.pdf

[10] B. I. P. Rubinstein *et al.*, "ANTIDOTE: Understanding and defending against poisoning of anomaly detectors," in *Proc. 9th ACM SIGCOMM Internet Meas. Conf. (IMC)*, 2009, pp. 1–14, doi: 10.1145/1644893.1644895.

[11] B. Biggio, I. Corona, G. Fumera, G. Giacinto, and F. Roli, "Bagging classifiers for fighting poisoning attacks in adversarial classification tasks," in *Proc. 10th Int. Workshop Multiple Classifier Syst. (MCS)*, 2011, pp. 350–359, doi: 10.1007/978-3-642-21557-5_37.

[12] B. Nelson, B. Biggio, and P. Laskov, "Understanding the risk factors of learning in adversarial environments," in *Proc. 4th ACM Workshop Secur. Artif. Intell. (AISec)*, Chicago, IL, USA, Oct. 2011, pp. 87–92, doi: 10.1145/2046684.2046698.

[13] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognit.*, vol. 84, pp. 317–331, Dec. 2018, doi: 10.1016/j.patcog.2018.07.023.

[14] G. F. Cretu, A. Stavrou, M. E. Locasto, S. J. Stolfo, and A. D. Keromytis, "Casting out demons: Sanitizing training data for anomaly sensors," in *Proc. IEEE Symp. Secur. Privacy (S&P)*, Oakland, CA, USA, May 2008, pp. 81–95, doi: 10.1109/SP.2008.11.

[15] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artif. Intell. Rev.*, vol. 22, no. 2, pp. 85–126, Oct. 2004, doi: 10.1023/B:AIRE.0000045502.10941.a9.

[16] A. Paudice, L. Muñoz-González, A. Gyorgy, and E. C. Lupu, "Detection of adversarial training examples in poisoning attacks through anomaly detection," 2018, *arXiv:1802.03041*. [Online]. Available: http://arxiv.org/abs/1802.03041

[17] C. Liu, B. Li, Y. Vorobeychik, and A. Oprea, "Robust linear regression against training data poisoning," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, Nov. 2017, pp. 91–102, doi: 10.1145/3128572.3140447.

[18] G. Xu, Z. Cao, B.-G. Hu, and J. C. Principe, "Robust support vector machines based on the rescaled hinge loss function," *Pattern Recognit.*, vol. 63, pp. 139–148, Mar. 2017, doi: 10.1016/j.patcog.2016.09.045.

[19] T. Gu, B. Dolan-Gavitt, and S. Garg, "BadNets: Identifying vulnerabilities in the machine learning model supply chain," 2017, *arXiv:1708.06733*. [Online]. Available: http://arxiv.org/abs/1708.06733

[20] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017, *arXiv:1712.05526*. [Online]. Available: http://arxiv.org/abs/1712.05526

[21] S. Li, B. Z. H. Zhao, J. Yu, M. Xue, D. Kaafar, and H. Zhu, "Invisible backdoor attacks against deep neural networks," 2019, *arXiv:1909.02742*. [Online]. Available: http://arxiv.org/abs/1909.02742

[22] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," in *Proc. 34th AAAI Conf. Artif. Intell. (AAAI)*, 2020, pp. 11957–11965.

[23] A. Salem, R. Wen, M. Backes, S. Ma, and Y. Zhang, "Dynamic backdoor attacks against machine learning models," 2020, *arXiv:2003.03675*. [Online]. Available: http://arxiv.org/abs/2003.03675

[24] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur. (AsiaCCS)*, 2017, pp. 506–519, doi: 10.1145/3052973.3053009.

[25] Kaggle. (2016). *House Prices: Advanced Regression Techniques*. [Online]. Available: https://www.kaggle.com/c/house-prices-advanced-regression-techniques

[26] *Lending Club Loan Data*. (2013). [Online]. Available: https://www.kaggle.com/wendykan/lending-club-loan-data

[27] N. Limdi *et al.*, "International warfarin pharmacogenetics consortium: Warfarin pharmacogenetics: A single *V K O R C*1 polymorphism is predictive of dose across 3 racial groups," *Blood*, vol. 115, no. 18, pp. 3827–3834, 2010.

[28] D. Dua and C. Graff. (2017). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset

[29] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli, "Is feature selection secure against training data poisoning?" in *Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 1689–1698. [Online]. Available: http://proceedings.mlr.press/v37/xiao15.html

[30] J. Wen, B. Z. H. Zhao, M. Xue, and H. Qian, "PALOR: Poisoning attacks against logistic regression," in *Proc. Australas. Conf. Inf. Secur. Privacy*. Cham, Switzerland: Springer, 2020, pp. 447–460.

[31] S. Li *et al.*, "Hidden backdoors in human-centric language models," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, 2021.

[32] S. Li, M. Xue, B. Zhao, H. Zhu, and X. Zhang, "Invisible backdoor attacks on deep neural networks via steganography and regularization," *IEEE Trans. Dependable Secure Comput.*, early access, Sep. 3, 2020, doi: 10.1109/TDSC.2020.3021407.

[33] S. Li, S. Ma, M. Xue, and B. Z. H. Zhao, "Deep learning backdoors," 2020, *arXiv:2007.08273*. [Online]. Available: http://arxiv.org/abs/2007.08273

[34] L. Chen, H. Wang, B. Z. H. Zhao, M. Xue, and H. Qian, "Oriole: Thwarting privacy against trustworthy deep learning models," in *Proc. Australas. Conf. Inf. Secur. Privacy (ACISP)*, 2021.

[35] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local model poisoning attacks to Byzantine-robust federated learning," 2019, *arXiv:1911.11815*. [Online]. Available: http://arxiv.org/abs/1911.11815

[36] M. Fang, G. Yang, N. Z. Gong, and J. Liu, "Poisoning attacks to graph-based recommender systems," in *Proc. 34th Annu. Comput. Secur. Appl. Conf. (ACSAC)*, 2018, pp. 381–392, doi: 10.1145/3274694.3274706.

[37] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2016, pp. 1885–1893.

[38] G. Yang, N. Z. Gong, and Y. Cai, "Fake co-visitation injection attacks to recommender systems," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2017.

[39] S. Shen, S. Tople, and P. Saxena, "Auror: Defending against poisoning attacks in collaborative deep learning systems," in *Proc. Conf. Comput. Secur. Appl.*, 2016.

[40] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *Proc. Int. Symp. Res. Attacks, Intrusions, Defenses*, 2018, pp. 273–294.

[41] F. N. Iandola, K. Ashraf, M. W. Moskewicz, and K. Keutzer, "FireCaffe: Near-linear acceleration of deep neural network training on compute clusters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 2592–2600.

[42] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks," in *Proc. 28th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2019, pp. 4658–4664.

[43] C. Yang, Q. Wu, H. Li, and Y. Chen, "Generative poisoning attack method against neural networks," 2017, *arXiv:1703.01340*. [Online]. Available: https://arxiv.org/abs/1703.01340

[44] D. Hitaj and L. V. Mancini, "Have you stolen my model? Evasion attacks against deep neural network watermarking techniques," 2018, *arXiv:1809.00615*. [Online]. Available: https://arxiv.org/abs/1809.00615

[45] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009.

[46] M. Zhao, B. An, Y. Yu, S. Liu, and S. J. Pan, "Data poisoning attacks on multi-task relationship learning," in *Proc. 32nd AAAI Conf. Artif. Intell. (AAAI)*, 2018, pp. 2628–2635.

Benjamin Zi Hao Zhao is currently pursuing the Ph.D. degree with the School of Electrical Engineering and Telecommunications, University of New South Wales and CSIRO-Data61. His current research interests are authentication systems, and security and privacy with machine learning. His work has received the ACM AsiaCCS Best Paper Award.

Minhui Xue is currently a Lecturer (a.k.a. Assistant Professor) with the School of Computer Science, The University of Adelaide. He is also an Honorary Lecturer with Macquarie University. He was a recipient of the ACM SIGSOFT Distinguished Paper Award and the IEEE Best Paper Award. His work has been featured in the mainstream press, including The New York Times and Science Daily. He currently serves on the Program Committee of IEEE Symposium on Security and Privacy 2021, Oakland, ACM CCS 2021, USENIX Security 2021, NDSS 2021, ICSE 2021, ESORICS 2021, and PETS 2021 and 2020.

Alina Oprea is currently an Associate Professor with Northeastern University of Khoury College of Computer Sciences. She was a recipient of the Technology Review TR35 Award for Research in cloud security, in 2011 and the Google Security and Privacy Award 2019. She previously served as a Program Committee Co-Chair for the IEEE Symposium on Security and Privacy 2021, 2020, and NDSS 2019 and 2018.

Jialin Wen is currently pursuing the master's degree with the School of Computer Science of Technology, East China Normal University. She focuses primarily on the areas of machine learning and security, specifically exploring the robustness of machine learning models against various adversarial attacks.

Haifeng Qian received the B.S. and master's degrees in algebraic geometry from the Department of Mathematics, East China Normal University, Shanghai, China, in 2000 and 2003, respectively, and the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University in 2006. He is currently a Professor with the Software Engineering Institute, East China Normal University. His main research interests include network security, cryptography, and algebraic geometry.