Design of Poisoning Attacks on Linear Regression Using Bilevel Optimization

Zeynep Şuvak[a], Miguel F. Anjos[a,*], Luce Brotcorne[b], Diego Cattaruzza[c]

[a] *Operational Research and Optimization Group, School of Mathematics, University of Edinburgh, James Clerk Maxwell Building, Peter Guthrie Tait Road, Edinburgh, EH9 3FD, Scotland, United Kingdom*
[b] *Inria Lille-Nord Europe, Team Inocs, 40 Avenue Halley, 59650, Villeneuve D'Ascq, France*
[c] *CNRS, Centrale Lille, Inria, UMR 9189 - CRIStAL Lille, Univ. Lille, Lille, France*

Abstract

Poisoning attack is one of the attack types commonly studied in the field of adversarial machine learning. The adversary generating poison attacks is assumed to have access to the training process of a machine learning algorithm and aims to prevent the algorithm from functioning properly by injecting manipulative data while the algorithm is being trained. In this work, our focus is on poisoning attacks against linear regression models which target to weaken the prediction power of the attacked regression model. We propose a bilevel optimization problem to model this adversarial process between the attacker generating poisoning attacks and the learner which tries to learn the best predictive regression model. We give an alternative single level optimization problem by benefiting from the optimality conditions of the learner's problem. A commercial solver is used to solve the resulting single level optimization problem where we generate the whole set of poisoning attack samples at once. Besides, an iterative approach that allows to determine only a portion of poisoning attack samples at every iteration is introduced. The proposed attack strategies are shown to be superior than a benchmark algorithm from the literature by carrying out extensive experiments on two realistic datasets.

Keywords:
Poisoning attacks, bilevel optimization, regression, adversarial machine learning

1. Introduction

Today, many applications from different contexts rely on machine learning (ML) algorithms to generate automated decisions since ML based systems constantly self improve without an exogenous intervention and are capable of adjusting themselves to the changing data. The widespread usage of ML algorithms has raised questions about how vulnerable they are against malicious data coming from untrusted sources (Dalvi et al., 2004; Szegedy et al., 2013; Su et al., 2019). The input data used to train ML algorithms can be collected from various sources including humans, machines, internet and sensors. Therefore they can be erroneous as the data source is prone to make mistakes like

*Corresponding author
Email addresses: zsuvak@ed.ac.uk (Zeynep Şuvak), anjos@stanfordalumni.org (Miguel F. Anjos), luce.brotcorne@inria.fr (Luce Brotcorne), diego.cattaruzza@centralelille.fr (Diego Cattaruzza)

humans. Moreover, the data may be distorted by an adversary to deliberately manipulate the learning phase of predictive models by directly influencing the training data or to affect the accuracy of the model on new data during testing.

In this work, we consider a specific type of attack called *poisoning attack* where an adversary inserts a small number of *poisoned* data samples into the training set to mislead the targeted ML process in the course of fitting the best predictive model. Our focus is on determining poisoning attack strategies against *linear regression* which is an important and powerful method for prediction. *Linear regression* is an essential class of supervised learning which can be used for different purposes such as to predict the future demand of a product (Carbonneau et al., 2008), to optimize the value or price of a new service (Madhuri et al., 2019), to calculate the risk of a loan (Zhang and Thomas, 2012) or to personalize the dose of a medical treatment (Gage et al., 2008). Linear regression can be used for classification as well by converting the real valued outputs to class labels as done in logistic regression. Poisoning attacks against classification algorithms have been thoroughly studied in the literature (Biggio et al., 2012; Mozaffari-Kermani et al., 2015; Carnerero-Cano et al., 2020; Muñoz-González et al., 2019). However, there are few works considering poisoning attacks on linear regression learning although the latter is being used in a wide range of applications. This study aims to contribute by assessing how influential poisoning attacks can be on regression learning.

A poisoning attack is a good example of an adversarial game where the first player (the attacker) tries to confuse the second player (the learner) to a maximum extent by determining the poisoning attack samples whereas the second player minimizes its prediction error by learning from the poisoned training data and optimizes the regression parameters based on it. The described adversarial game can be played in such a manner that one of the players is the leader who makes a decision first and the other is the follower whose decision is optimized after observing the leader's action. Upon the follower's reaction, the leader may want to change its decision as it is no longer the best decision. In two player-games with one leader and one follower, the leader has the advantage of anticipating the follower's reaction and makes the optimal move taking into account the possible response of the follower. In the context of poisoning attacks, the attacker can be considered as the leader while the learner plays the role of the follower. It is typical to model these type of games as bilevel optimization problems (Dempe, 2002).

Although previous works point out that a *poisoning attack* can be modeled as a bilevel optimization problem (Jagielski et al., 2018), none of them attempts to solve it explicitly. For the purpose of filling this gap, a bilevel formulation for poisoning attacks on linear regression learning and a solution approach are introduced in this paper. As the solution method, an alternative single level nonlinear problem is obtained using the optimality conditions of the follower's problem and the resulting problem is solved using a commercial solver. The poisoning attack samples are generated both by solving the proposed model at once and through an iterative approach by optimizing only a fraction of poisoning attack samples at a time. The performance of these methods is measured with their running times and the Mean Squared Errors (MSE) of fitted regression model on new test data. Our method is compared to a gradient descent algorithm from the literature which is reported to provide the highest average MSE for benchmarking purposes. Based on the experiments with two different datasets, our approach improved the MSE of the benchmark algorithm by 20.4%, on average, and is competitive in terms of running time.

The remainder of this paper is organized as follows. We present the background of poisoning attacks and a brief review of the relevant work from the literature in Section 2. We start Section 3 with the problem definition, continue with the bilevel formulation of poisoning attacks which is followed by the alternative single level problem representation. Section 4 provides the details and implementation of the adopted solution approach. Section 5 includes the computational results obtained for assessing and comparing the performance of the solution approach. Finally, concluding remarks are given in Section 6.

2. Literature Review

Adversarial machine learning is a specific field of ML focusing on two main research topics. One of them is the attacks that try to deceive ML models by introducing corrupt input in order to prevent the targeted ML model from functioning properly, and the other is the defense mechanisms of the ML models against these attacks. The defense mechanisms include designing algorithms that detect the malicious input as well as the the exploration of robust learning techniques which function acceptably well despite the attacks. Defense strategies are beyond the scope of this paper but the interested reader is referred to Biggio and Roli (2018) for a recent survey.

Attacks against ML algorithms have been categorized in the early work of Barreno et al. (2006) considering how the attacker influences the ML algorithm, how specific the target of the attack is or what kind of malfunctioning is intended. An extended taxonomy of attacks regarding the attacker's goal and knowledge about the system and the extent to which the attacker can control or interfere with the targeted process can be found in the work by Xiao et al. (2015). One of the most common attack types is called *evasion* attacks in which the attacker has no control over the determination of the model parameters but manipulates the test data to get the desired output from the ML model. Evasion attacks are usually realized by generating adversarial examples to avoid being identified by the ML model (Ren et al., 2020) e.g. to bypass security checks like spam filters (Lowd, 2005), biometric recognition (Alegre et al., 2014; Qian et al., 2020) or banking fraud detection systems (Carminati et al., 2020).

The poisoning attacks that we study in this work differ from the evasion attacks as the attacker has access to the learning/training phase of the ML model. In the literature, various contributions can be found for poisoning attacks against classification algorithms. Among them we mention the work by Biggio et al. (2012) which targets support vector machines (SVM) and proposes a gradient ascent method. It is shown to considerably weaken the classification accuracy of SVM on the MNIST database (Lecun et al., 1995) used for handwriting digit recognition. Mozaffari-Kermani et al. (2015) consider poisoning attacks on a set of different classifiers including decision trees, naive Bayes classifiers and artificial neural networks. They develop a generic poisoning attack scheme which is suitable for any classification algorithm by benefiting from the properties of the attacked class. Koh et al. (2018) focus on generating stronger poisoning attacks against an SVM classifier assuming that the classifier implements various defense algorithms known as *data sanitization* techniques. An interesting recent study by Muñoz-González et al. (2019) introduces a generative adversarial network (GAN) to generate poisoning attacks against deep neural networks. The proposed GAN allows

incorporating detectability constraints which seek to prevent the poisoning attack samples from being detected by defense algorithms.

The studies on poisoning attacks against regression learning are relatively fewer and more recent compared to the ones against classification learning. The work by Xiao et al. (2015) is the first to introduce a gradient descent algorithm to generate poisoning attacks against *lasso* and *ridge* regression. The authors evaluate how regression coefficients and the importance of features change in the existence of poisoning attack samples. Jagielski et al. (2018) extend this work by developing a much faster but less effective attack scheme which generates attack samples from the estimated distribution of the feature space and a defense algorithm that discards the data points with high prediction errors. Wen et al. (2020) modify the gradient descent poisoning attack algorithm described by Xiao et al. (2015) through maximizing the variance of the poisoned training data set instead of the total loss due to the prediction error.

3. Notations and Problem Definition

In the setting of supervised learning, the parameters of a linear regression model are learned on *training data* which include both feature vectors and observed response values. The training data can be represented as $\mathcal{D}^{tr} = \{(\boldsymbol{x}_i^{tr}, y_i^{tr})\}_{i=1}^n$ where $\boldsymbol{x}_i^{tr} \in [0,1]^d$ is a d-dimensional feature vector and $y_i^{tr} \in [0,1]$ is the response variable. For the implementation of linear regression, the data must be standardized so that all the values are assumed to be between 0 and 1. After learning the parameters of the *regression model*, the obtained model is used to predict the value of the response variable on a completely new data set, namely *test data*, which must be standardized in exactly the same way as the training data.

The linear regression model is a linear function $f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{w}^\top \boldsymbol{x} + b = \sum_{j=1}^d w_j x_j + b$ that outputs a prediction for the response variable y, i.e. the predicted response, denoted by $\hat{y}$, is equal to $f(\boldsymbol{x}, \boldsymbol{\theta})$. The parameter set $\boldsymbol{\theta}$ is equivalent to $(\boldsymbol{w}, b) \in \mathbb{R}^{d+1}$ which consists of the feature weights $\boldsymbol{w} \in \mathbb{R}^d$ and the bias $b \in \mathbb{R}$. Given the training data $\mathcal{D}^{tr} = \{(\boldsymbol{x}_i^{tr}, y_i^{tr})\}_{i=1}^n$, determining $\boldsymbol{\theta} = (\boldsymbol{w}, b)$ is fundamentally an optimization problem which minimizes the following quadratic loss function:

$$\mathcal{L}(\mathcal{D}^{tr}, \boldsymbol{\theta}) = \frac{1}{n}\left(\sum_{i=1}^n (f(\boldsymbol{x}_i^{tr}, \boldsymbol{\theta}) - y_i^{tr})^2\right) + \lambda\Omega(\boldsymbol{w}), \tag{1}$$

where the first term is the MSE represented as

$$\text{MSE}(\mathcal{D}^{tr}, \boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^n (f(\boldsymbol{x}_i^{tr}, \boldsymbol{\theta}) - y_i^{tr})^2. \tag{2}$$

The MSE measures the prediction error of the regression model $f(., \boldsymbol{\theta})$ and is calculated as the average of squared residuals over training data samples. The second term of expression (1) is a regularization term to prevent the *overfitting* of the regression model to a specific set of training data. A good regression fit must have low MSE not only for training data but also for any unseen test data. Using a regularization term is an approach to avoid an overfitted regression model as it might lead to high MSE on test data samples. Furthermore, it can be regarded as a natural defense mechanism of the model against poisoning attacks. The optimization of the regularization parameter, λ, in the existence of poisoning

attacks is previously modeled as a multi-level optimization problem and shown to be a powerful defense method (Carnerero-Cano et al., 2020). In our model, we consider *ridge regression* which uses l_2-norm regularization, i.e. $\Omega(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2 = \frac{1}{2}\sum_{j=1}^{d} w_j^2$. Since the regularization term associated with *ridge regression* is differentiable over $\boldsymbol{w} \in \mathbb{R}^d$, l_2-norm regularization is preferred in our experiments.

A *poisoning attack* involves finding a set of data points denoted as $\mathcal{D}^{\boldsymbol{p}} = \{(\boldsymbol{x}_{\boldsymbol{k}}^{\boldsymbol{p}}, y_k^p)\}_{k=1}^q$ to be inserted into the original training data $\mathcal{D}^{\boldsymbol{tr}}$. Once the training data is poisoned, the regression model $f(\boldsymbol{x}, \boldsymbol{\theta})$ learns on $\mathcal{D}^{\boldsymbol{tr}} \cup \mathcal{D}^{\boldsymbol{p}}$; therefore the objective of the attacker is to generate $\mathcal{D}^{\boldsymbol{p}} = \{(\boldsymbol{x}_{\boldsymbol{k}}^{\boldsymbol{p}}, y_k^p)\}_{k=1}^q$ so that the prediction error, $\text{MSE}(\mathcal{D}^{\boldsymbol{tr}}, \boldsymbol{\theta})$, is maximized. The effectiveness of a poisoning attack strategy is measured by two performance metrics: the difference of MSEs on test data obtained with the regression models fitted to poisoned and unpoisoned data, and the time required to generate $\mathcal{D}^{\boldsymbol{p}}$.

3.1. Bilevel Optimization Formulation

In our setting, we assume that the attacker has full knowledge about the training data $\mathcal{D}^{\boldsymbol{tr}}$ which is classified as *white-box* attacks (Jagielski et al., 2018). In the alternative scenario which is known as *black-box* attacks, the adversary does not know the exact training data but uses a substitute training data $\mathcal{D}'$ which has the same distribution as $\mathcal{D}^{\boldsymbol{tr}}$. *White-box* poisoning attacks can be modeled as the following bilevel optimization problem which can also be used to formulate *black-box* attacks when $\mathcal{D}^{\boldsymbol{tr}}$ is replaced with $\mathcal{D}'$.

$$\max \quad \text{MSE}(\mathcal{D}^{\boldsymbol{tr}}, \boldsymbol{\theta}^*) = \frac{1}{n}\sum_{i=1}^{n}(f(\boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{tr}}, \boldsymbol{\theta}^*) - y_i^{tr})^2 \tag{3}$$

$$\text{s.t.} \quad \boldsymbol{x}_{\boldsymbol{k}}^{\boldsymbol{p}} \in [0,1]^d \qquad k = 1, ..., q \tag{4}$$

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmin}}\{\mathcal{L}(\mathcal{D}^{\boldsymbol{tr}} \cup \mathcal{D}^{\boldsymbol{p}}, \boldsymbol{\theta}) \ : \ \boldsymbol{\theta} = (\boldsymbol{w}, b) \in \mathbb{R}^{d+1}\}. \tag{5}$$

In the upper level of the bilevel formulation (3)–(5), the attacker, also referred to as the leader, determines the feature vectors of poisoning samples, i.e. $\mathcal{D}^{\boldsymbol{p}} = \{(\boldsymbol{x}_{\boldsymbol{k}}^{\boldsymbol{p}}, y_k^p)\}_{k=1}^q$. We assume that y_k^p are fixed prior to optimization and they are not treated as variables of the given problem (3)–(5). The objective function of the attacker to be maximized is the MSE on $\mathcal{D}^{\boldsymbol{tr}}$ instead of the loss function $\mathcal{L}(\mathcal{D}^{\boldsymbol{tr}}, \boldsymbol{\theta})$ which is preferred in earlier works (Xiao et al., 2015; Jagielski et al., 2018). This choice guarantees that high MSE is achieved on real training data instances rather than on the poisoning data samples. Moreover, it prevents the possibility that the regularization term drives the maximization process instead of the prediction error term which is not desired by the attacker. Also note that the objective function (3) of the attacker does not depend on its own decision variables which are $\boldsymbol{x}^{\boldsymbol{p}}$ of the set $\mathcal{D}^{\boldsymbol{p}}$. Instead, the decision variables of the lower level problem, also referred to as the follower, i.e. $\boldsymbol{\theta}$, are used in it. In fact, the constraint (5) that are equivalent to the follower's regression learning problem (or the lower level problem of the bilevel formulation) imply that only optimal values of $\boldsymbol{\theta}$, which is denoted as $\boldsymbol{\theta}^*$, are allowed to be used in the upper level objective function. The attacker's choice of $\mathcal{D}^{\boldsymbol{p}}$ affects $\boldsymbol{\theta}^*$ in the lower level problem represented by (5). Hence, the attacker has an indirect control over $\boldsymbol{\theta}^*$ and objective function (3). The feature values of poisoning attacks must comply with the standardized form as indicated by (4).

In the lower level problem given as (5), $\boldsymbol{\theta}$ is optimized assuming that $\mathcal{D}^{\boldsymbol{p}}$ is given, so that the quadratic loss function over the poisoned training data with a regularization term is minimized. The explicit form of the follower's objective function is written as

$$\mathcal{L}(\mathcal{D}^{\boldsymbol{tr}} \cup \mathcal{D}^{\boldsymbol{p}}, \boldsymbol{\theta}) = \frac{1}{n+q}\left(\sum_{i=1}^{n}(f(\boldsymbol{x_i^{tr}}, \boldsymbol{\theta}) - y_i^{tr})^2 + \sum_{k=1}^{q}(f(\boldsymbol{x_k^{p}}, \boldsymbol{\theta}) - y_k^{p})^2\right) + \lambda\Omega(\boldsymbol{w}). \tag{6}$$

3.2. The Single Level Nonlinear Optimization Formulation

Karush-Kuhn-Tucker (KKT) optimality conditions of the lower level problem can be retrieved easily if it belongs to the class of convex optimization problems like linear optimization problems or unconstrained convex quadratic optimization problems. In these cases, it is a common practice to replace the lower level problem with its KKT conditions (Dempe, 2002). Since KKT conditions correspond to a set of equations or inequalities, introducing them removes the bilevel nature of the formulation. This does not mean that the difficulty of the problem changes but the described alteration results in a single level formulation which allows using solution techniques suitable for single level problems.

For the purpose of obtaining a single level optimization problem, KKT conditions of the lower level problem (5) must be stated explicitly. This is an unconstrained convex quadratic problem with respect to the variable set $\boldsymbol{\theta}$. Then, every local minimum is a global one and the first order optimality conditions, i.e. the necessary optimality conditions, are sufficient for global optimality. In other words, the first order optimality conditions for the lower level problem (5) correspond to its KKT optimality conditions. The first order optimality conditions are the following:

$$\frac{\partial\mathcal{L}(\mathcal{D}^{\boldsymbol{tr}} \cup \mathcal{D}^{\boldsymbol{p}}, \boldsymbol{\theta})}{\partial w_j} = \frac{2}{n+q}\left(\sum_{i=1}^{n}(f(\boldsymbol{x_i^{tr}}, \boldsymbol{\theta}) - y_i^{tr})x_{ij}^{tr} + \sum_{k=1}^{q}(f(\boldsymbol{x_k^{p}}, \boldsymbol{\theta}) - y_k^{p})x_{kj}^{p}\right) + \lambda w_j = 0 \qquad j = 1, ..., d \tag{7}$$

$$\frac{\partial\mathcal{L}(\mathcal{D}^{\boldsymbol{tr}} \cup \mathcal{D}^{\boldsymbol{p}}, \boldsymbol{\theta})}{\partial b} = \frac{2}{n+q}\left(\sum_{i=1}^{n}(f(\boldsymbol{x_i^{tr}}, \boldsymbol{\theta}) - y_i^{tr}) + \sum_{k=1}^{q}(f(\boldsymbol{x_k^{p}}, \boldsymbol{\theta}) - y_k^{p})\right) = 0 \tag{8}$$

which state that the partial derivatives of (6) with respect to w_j, $j = 1, ..., d$, and b are equal to zero at optimality.

We substitute the lower level problem (5) with the conditions (7) and (8) in order to attain the following single level formulation:

$$\max \qquad \mathrm{MSE}(\mathcal{D}^{\boldsymbol{tr}}, \boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}(f(\boldsymbol{x_i^{tr}}, \boldsymbol{\theta}) - y_i^{tr})^2 \tag{9}$$

$$\text{s.t.} \qquad \frac{\partial\mathcal{L}(\mathcal{D}^{\boldsymbol{tr}} \cup \mathcal{D}^{\boldsymbol{p}}, \boldsymbol{\theta})}{\partial w_j} = 0 \qquad j = 1, ..., d \tag{10}$$

$$\frac{\partial\mathcal{L}(\mathcal{D}^{\boldsymbol{tr}} \cup \mathcal{D}^{\boldsymbol{p}}, \boldsymbol{\theta})}{\partial b} = 0 \tag{11}$$

$$\boldsymbol{x_k^{p}} \in [0, 1]^d \qquad k = 1, ..., q \tag{12}$$

$$\boldsymbol{\theta} = (\boldsymbol{w}, b) \in \mathbb{R}^{d+1}. \tag{13}$$

Notice that the objective function (9) of the single level formulation is convex with respect to the variables $\boldsymbol{w}$ and b while the variables $\boldsymbol{x}_{\boldsymbol{k}}^{\boldsymbol{p}}$, $k+1, ..., q$ appear only in the constraints (10) and (11). Therefore, the resulting problem is the maximization of a convex quadratic function with nonlinear constraints.

4. Solution Approach

This section describes the steps taken to solve the problem (9)–(13) presented in Section 3.

4.1. Data Preprocessing

Preprocessing of any given data set is essential to be able to fit a regression model on it. The first condition that must be satisfied is that all feature values must be numerical. The original data set might contain features which are numerically represented like the year a house was built or weight of a person whereas others are categorical like the neighborhood a house is situated in or the ethnicity of a person. The categorical features must be one-hot encoded before applying regression. The one-hot encoding deletes the column of the categorical feature and instead inserts as many feature columns as the number of categories for that feature. Only one of the newly inserted columns, namely that of the applicable feature, is assigned 1 and the others are left as 0. Also, all the numerical values must be standardized to obtain a reasonable regression model by using a simple transformation like *min-max scaling* so that the range of the values is [0,1].

4.2. Initialization of Poisoning Attacks

Since the problem represented by (9)–(13) is non-convex, the returned solution by the solver will be a local maximum that is not necessarily the global optimal solution. Nonlinear optimization solvers require initial values for the decision variables (which are set to zero if unspecified) and the initial choice of $\mathcal{D}^{\boldsymbol{p}}$ has a considerable impact on the quality of the returned local optimal solution. For initialization, the approach of Jagielski et al. (2018) with rounded inverse labels is adopted. We randomly select q data points from the training data $\mathcal{D}^{\boldsymbol{tr}}$ and set them as $\boldsymbol{x}_{\boldsymbol{k}}^{\boldsymbol{p}} \in \mathcal{D}^{\boldsymbol{p}}$, $k = 1, ..., q$. We replace the value of the response y of the randomly picked q data points by the rounded values of $(1 - y)$ to the nearest integer (either 0 or 1) and set them as $y_k^p \in \mathcal{D}^{\boldsymbol{p}}$, $k = 1, ..., q$.

4.3. Solving the Optimization Problem

The problem (9)–(13) can be solved by commercial nonlinear optimization solvers. One approach is determine values of $\boldsymbol{x}_{\boldsymbol{k}}^{\boldsymbol{p}}$ $k = 1, ..., q$, by calling the solver just once. This approach is called *single attack strategy* in the sequel.

In the alternative approach, we partition the initialized $\mathcal{D}^{\boldsymbol{p}}$ into N disjoint equal sized subsets so that $\mathcal{D}^{\boldsymbol{p}} = \mathcal{D}_{\boldsymbol{1}}^{\boldsymbol{p}} \cup ... \cup \mathcal{D}_{\boldsymbol{N}}^{\boldsymbol{p}}$, $\mathcal{D}_{\boldsymbol{u}}^{\boldsymbol{p}} \cap \mathcal{D}_{\boldsymbol{v}}^{\boldsymbol{p}} = \emptyset$ and $|\mathcal{D}_{\boldsymbol{u}}^{\boldsymbol{p}}| = |\mathcal{D}_{\boldsymbol{v}}^{\boldsymbol{p}}|$ for any $\{u, v\} \in \{1, ..., N\},\ u \neq v$. Note that every subset of $\mathcal{D}^{\boldsymbol{p}}$ contains exactly K poisoning samples implying that $K = q/N$. The poisoning attack samples in $\mathcal{D}^{\boldsymbol{p}}$ are optimized in exactly N iterations where only one subset of $\mathcal{D}^{\boldsymbol{p}}$ is determined at each iteration. The poisoned training data at iteration l for $l = 1, ..., N$ is denoted by $\mathcal{D}_{\boldsymbol{l}}^{\boldsymbol{tr}}$ and found by concatenating the training data associated with the previous iteration and

the determined poisoning attack subset at the current iteration , i.e. $\mathcal{D}_l^{tr} = \mathcal{D}_{l-1}^{tr} \cup \mathcal{D}_l^p$ where $\mathcal{D}_0^{tr} = \mathcal{D}^{tr}$. The nonlinear optimization problem to be solved in iteration $l \in \{1, ..., N\}$ is given as

$$\max \quad \text{MSE}(\mathcal{D}_0^{tr}, \boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} (f(\boldsymbol{x}_i^{tr}, \boldsymbol{\theta}) - y_i^{tr})^2 \tag{14}$$

$$\text{s.t.} \quad \frac{\partial \mathcal{L}(\mathcal{D}_{l-1}^{tr} \cup \mathcal{D}_l^p, \boldsymbol{\theta})}{\partial w_j} = 0 \qquad j = 1, ..., d \tag{15}$$

$$\frac{\partial \mathcal{L}(\mathcal{D}_{l-1}^{tr} \cup \mathcal{D}_l^p, \boldsymbol{\theta})}{\partial b} = 0 \tag{16}$$

$$\boldsymbol{x}_k^p \in [0, 1]^d \qquad k = 1, ..., K \tag{17}$$

$$\boldsymbol{\theta} = (\boldsymbol{w}, b) \in \mathbb{R}^{d+1}. \tag{18}$$

Notice that the objective function (14) does not depend on the iteration index l and remains the same throughout the whole process whereas the follower relearns the regression parameters on the continuously expanding training data as shown in (15) and (16). This method is called *iterative attack strategy* and allows fixing a fraction of poisoning attack samples earlier than the others as opposed to the *single attack strategy* in which all poisoning attack samples are determined at once. In fact, *single attack strategy* is a special case of *iterative attack strategy* where $N = 1$ and $K = q$. The overall *iterative attack strategy* (IAS) is listed as Algorithm 1.

Algorithm 1 Iterative Attack Strategy (IAS)

Input: $\mathcal{D}^{tr}$, initialized poisonining attack samples $\mathcal{D}^p$, subsets $\mathcal{D}_1^p, ..., \mathcal{D}_N^p$, K
Output: Final poisoning attack samples $\mathcal{D}^{p*}$

1: Set $l = 1$, $\mathcal{D}_0^{tr} = \mathcal{D}^{tr}$
2: **while** $l \leq N$ **do**
  Solve problem (14)–(18) and let $\mathcal{D}_l^{p*}$ be the returned solution.
  Set $\mathcal{D}_l^{tr} = \mathcal{D}_{l-1}^{tr} \cup \mathcal{D}_l^{p*}$, $l = l + 1$
3: **end while**
4: Return $\mathcal{D}^{p*} = \mathcal{D}_1^{p*} \cup ... \cup \mathcal{D}_N^{p*}$

4.4. Dealing With One-Hot Encoded Features

The input training data $\mathcal{D}^{tr}$ contains both continuous and binary valued features since the categorical features are transformed into binary valued feature columns during preprocessing. Therefore, the poisoning attack samples must obey the binary definition of one-hot encoded features. This allows to fully decode these features to the original representation of the dataset once the poisoning attack samples are determined. In other words, the corresponding feature values must be strictly binary valued and only one category must be equal to one for a categorical feature. We consider four strategies to deal with the one-hot encoded features. These strategies are discussed here in the following.

Proper formulation of one-hot encoded features. The first option is to formulate the binary valued features properly. Instead of giving the entire representation, we can describe how they are properly defined with a small example. Let us

assume that features $j \in 1, 2, 3$ are the one-hot encoded feature columns corresponding to the categories of a single categorical feature. Then, $x^p_{k1} + x^p_{k2} + x^p_{k3} = 1$ for any poisoning attack sample k to obtain a valid feature vector. This type of constraints are also known as *set-partitioning* constraints (Padberg, 1979). Moreover, these features must be defined as binary variables: $x^p_{k1}, x^p_{k2}, x^p_{k3} \in \{0, 1\},\ k \in \{1, ..., q\}$. Binary definition of one-hot encoded features and the set partitioning constraints give a proper representation of one-hot encoded features. Since the proper formulation together with the single level formulation represented by (9)–(13) result in a mixed integer nonlinear optimization formulation, it becomes much more difficult to solve and we will not consider it.

Postprocessing the relaxed binary one-hot encoded features. Another straightforward approach is to include the one-hot encoded features as continuous variables. They are postprocessed after optimization by setting the highest valued category to 1 and the others to 0. Since our initial experiments showed that this strategy does not result in good quality poisoning attacks samples, it is not considered further.

Relaxing the binary-encoded features but keeping set partitioning constraints. In this version, binary restrictions are relaxed but the set partitioning constraints are kept. The final values are postprocessed as described previously. However, keeping the set partitioning constraints does not have a significant impact and postprocessed poisoning attack points are observed to be poor again based on the preliminary experiments. Therefore, this approach is not adopted to handle the binary valued features.

Leaving binary-encoded features out. The one-hot encoded variables can be fixed to their initial values and only the remaining numerical feature values of the poisoning attacks are optimized. This option reduces the solution space of the problem considerably hence decreasing the time required to find the poisoning attack samples. Although this approach may leave out good solutions including the global optimal one, our experiment results indicate that fixing the binary valued features to their initial values and leaving them outside the optimization procedure leads to generating effective poisoning attack samples. Thus, this strategy is adopted to deal with one-hot encoded features as we carry out the experiments detailed in Section 5.

5. Experimental Study

In this section we describe the datasets, the testing environment and the test instances used to carry out the computational tests. The detailed and aggregated performance indicator values are reported to assess the efficiency of attack strategies. Furthermore, the impact of poisoning attacks on individual data samples is discussed. The results of an additional set of tests are provided to observe the performance of the proposed attack strategies in black-box scenario. The section is concluded by discussing how the running times can be improved.

5.1. Datasets and Test Instances

The experiments are carried out on two regression datasets which are also used by Jagielski et al. (2018) and Wen et al. (2020) and publicly available at `https://github.com/jagielski/manip-ml`. A brief description of these datasets is provided below.

House Price dataset. This dataset is used to predict the sale price of a house as a function of its features. The features include lot size of the house, type of the dwelling, types of utilities it has (Kaggle). The original dataset consists of 1460 houses and 81 features in total. Preprocessing by one-hot encoding increases the number of features to 275.

Healthcare dataset. Given the attributes of a patient, the therapeutic dose of a drug called Warfarin is estimated using this dataset. The attributes of a patient include physical features like age, weight and gender and other conditions like having diabetes or being a smoker. The preprocessed dataset contains 205 features and 4684 patients.

In order to evaluate the performance of the proposed attack methods, test instances are generated from the mentioned datasets. Each test instance consists of a randomly split training, validation and test data. They have 300, 250 and 500 data points, respectively. Initial poisoning attack sample set, which is a part of the input, is determined as previously described in Section 4.2. *Poisoning rate* is the ratio of the number of poisoning attack samples to the size of the training data. Five test instances are generated for the poisoning rates in $\{4\%, 8\%, 12\%, 16\%, 20\%\}$ which results in 25 test instances for each dataset. Poisoning rates higher than 20% are not considered as only small fractions of poisoning attacks can be injected in realistic settings.

5.2. Test Conditions

The training data and the initial poisoning attack samples are the inputs of the optimization problem. Prior to solving the problem, training and validation sets are used together to determine the value of the regularization parameter, λ, by applying 10-fold cross validation. The proposed optimization problems are modeled using GAMS 30.3.0 and solved by the commercial nonlinear solver KNITRO (version 11). KNITRO returns a local optimum due to the nature of the nonlinear optimization problem. The default settings of KNITRO to terminate the optimization process (e.g. time limit of 10^8 seconds, iteration limit of 10,000 iterations, optimality tolerance of 10^{-6}) are used and the solver is able to find local optimal solutions for all test instances. The output of the solver is a set of final poisoning attack samples.

In order to assess the quality of the solution methods proposed in this paper, we also run the gradient descent algorithm of Jagielski et al. (2018) to generate poisoning attack samples. The settings of the gradient descent (GD) algorithm are chosen so that the response variable is not optimized. The training data is used to calculate the gradients instead of the validation data and the poisoning attack samples are initialized in the same way as described in Section 4.2. These settings for GD are preferred since they correspond to exactly the same assumptions we make when we develop our solution approaches. Different settings might lead to draw biased conclusions. To exemplify, a different initial set of points can result in a better set of attack samples regardless of the attack strategy; any additional information coming

from the validation data could be advantageous; the ability to play with the value of the response variable gives more flexibility and a larger solution space.

After the optimization problem is solved by iterative attack strategy (IAS), (i) the poisoning attack samples are determined, (ii) a regression model is learned on the poisoned training data and, (iii) the regression model is tested on the test and validation sets. The same steps are repeated for the training data poisoned with the attack samples generated by GD. For the implementation of regression learning, sklearn package of Python is used. All experiments are carried out on a computer running with Microsoft Windows 10 Education operating system and having an Intel Core CPU 1.7 GHz processor with 16.0 GB RAM and it was ensured that the algorithms are run in a single thread environment to make a fair comparison.

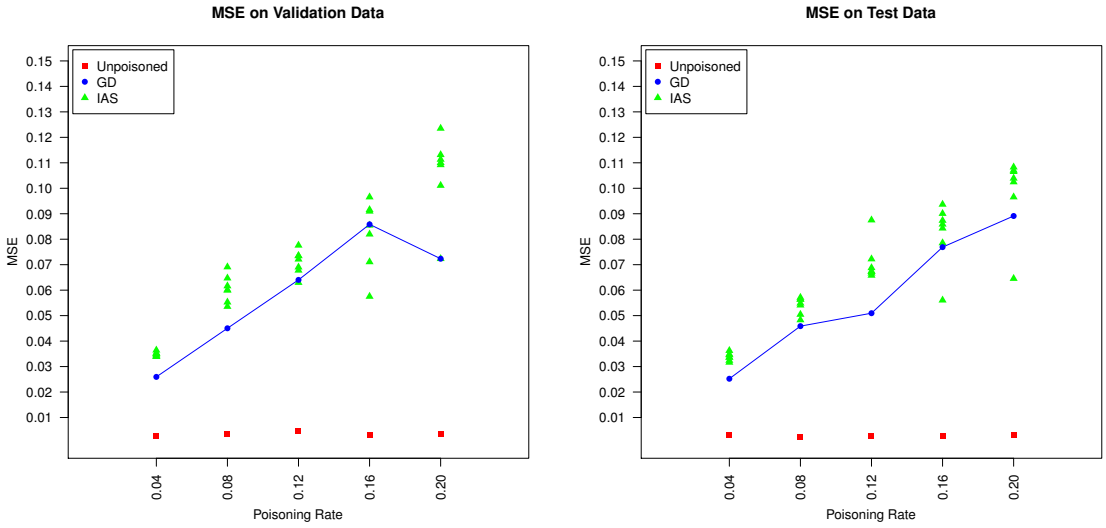
5.3. Efficiency of the Attack Strategies

For benchmarking purposes, we report the MSE values of the learned regression fits on test and validation data where higher MSE means better attacks. In Figure 1, we plot the MSE values on the different runs for each poisoning rate using House Price dataset. Each point on the plot is the average MSE value obtained over 5 instances for the same poisoning rate. The results show that any type of poisoning attack considerably increases the MSE of regression models. The MSEs with IAS are higher than the ones provided by GD for most of the test instances. We observe that there are cases where IAS is not as good as GD. The points below the GD line in Figure 1 correspond to the single attack strategy together with poisoning rates 16% and 20% indicating that single attack strategy is not better than GD for high poisoning rates. We also observe an exceptionally high MSE of GD for the poisoning rate of 16% in Figure 1(a) where IAS performs better than GD except for the cases $K = 1, 2$.

Figure 2 shows the plot of the average MSEs of 5 instances on Healthcare dataset for different poisoning rates. In this dataset, we make a similar observation so that the MSE performance of the single attack strategy is poorer than the baseline attack, GD, for poisoning rates 12%, 16% and 20%. The other versions of IAS outperform GD, in general. These plots reveal that solving the optimization problem in more than one iteration substantially improves the quality of the generated attack samples particularly for high poisoning rates. The tabulated results used to prepare the plots in Figures 1 and 2 are reported in the Appendix in Tables 4–7.

In order to assess the relative performance of IAS, we calculate the change in MSE with respect to the MSE obtained with GD poisoning for each test instance t and attack strategy a as follows: $\text{MSE}^{ta}_{change}(\%) = \frac{\text{MSE}^{ta} - \text{MSE}^{t\{GD\}}}{\text{MSE}^{t\{GD\}}} \times 100$. Positive MSE^{ta}_{change} values indicate that algorithm a provides higher MSE than the baseline strategy while negative values point out worse performance in terms of MSE.

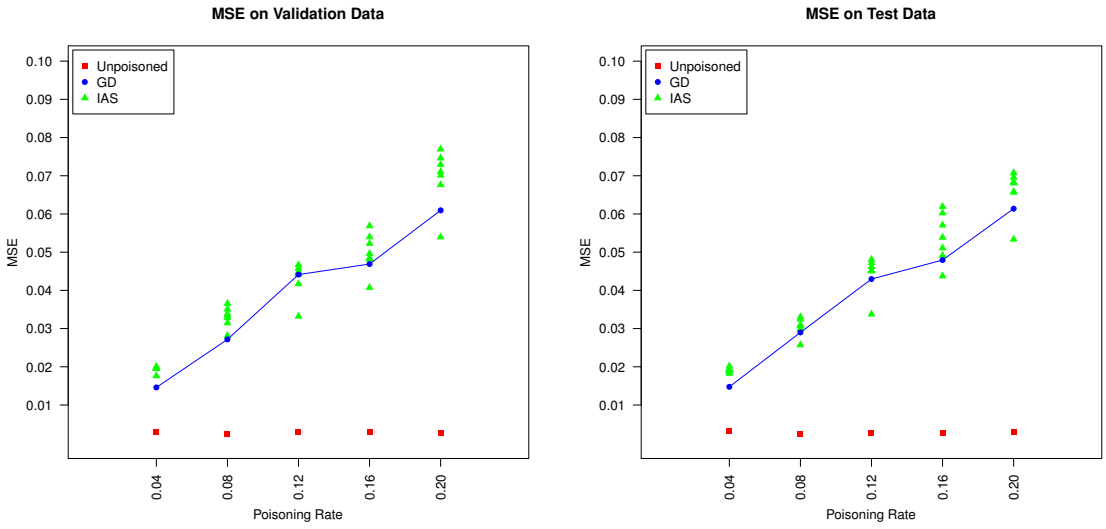
Table 1 summarizes the performance of the different attack strategies. The first column of Table 1 lists the attack strategies that we experiment with including the benchmark GD. Since the size of the training data is fixed and equal to 300, poisoning rates of 4%, 8%, 12%, 16% and 20% correspond to generating 12, 24, 36, 48 and 60 attack samples, respectively. Single attack strategy is the special case of IAS where all attack samples are generated at one iteration. Recall that K represents the number of attack samples generated during each iteration for the remaining strategies.



(a) Average MSE on validation data for House Price dataset

(b) Average MSE on test data for House Price dataset

Figure 1: The effect of poisoning attacks on MSE for House Price dataset



(a) Average MSE on validation data for Healthcare dataset

(b) Average MSE on test data for Healthcare dataset

Figure 2: The effect of poisoning attacks on MSE for Healthcare dataset

The last row contains the mean values of performance indicators over IAS and single attack strategies. In the second column, the average running time in seconds which is calculated over 50 instances is provided for each strategy. The running time of the methods is defined as the time elapsed to generate poisoning attacks. The time needed to calculate λ, initialization of attacks and learning the regression models are not taken into account since they are common for both IAS and GD and negligibly short. Notice that the single attack strategy is much faster than GD and the running time of IAS ($K = 12$) is quite competitive with GD. The running time of IAS increases as the value of K gets smaller, as expected. The third column of Table 1 lists MSE^a_{change} values representing the average of MSE^{ta}_{change} for each attack strategy a. In other words, MSE^a_{change} is the average improvement achieved in MSE by the attack strategy a with respect to GD. This value is calculated by averaging 100 MSE^{ta}_{change} values obtained over the validation and test sets of all instances taken from both datasets. From the third column, it can be seen that all strategies, except for the single attack strategy, achieve similar change of MSE with respect to GD poisoning. The MSE of single attack strategy is 5.7% better than GD while the others achieve at least 21.3% improvement on the MSE obtained with GD. IAS ($K = 12$) and IAS ($K = 4$) gives the highest MSE change with respect to GD poisoned regression. As we decrease K, the change in MSE value increases first and slightly decreases when $K < 4$ while the running times monotonically goes up with descending K values. The summary in Table 1 suggests that considering running time and MSE together, IAS ($K = 12$)

Attack Strategy	**Avg Running Time (s)**	MSE^a_{change} (%)
GD	105.2	0.0
Single Attack	46.6	5.7
IAS (K=12)	160.1	23.7
IAS (K=6)	311.3	21.3
IAS (K=4)	514.8	25.1
IAS (K=3)	676.7	22.5
IAS (K=2)	901.0	22.4
IAS (K=1)	1,794.5	21.9
IAS Average	629.3	20.4

Table 1: Summary of the performance of the attack strategies

is the best strategy. IAS ($K = 4$) can be considered as another efficient method when MSE is relatively more important than running times. Moreover, IAS with lower K values and the single attack strategy are not as efficient as them. All in all, MSE provided by GD is increased up to 25% with our proposed attack strategies most of which terminate in a reasonable amount of time.

Poisoning Rate	**GD Running Time (s)**	**Avg Running Time (s)**	**Avg** MSE_{change} (%)
0.04	74.7	163.3	38.4
0.08	68.4	372.6	23.6
0.12	74.7	630.5	17.6
0.16	108.6	974.8	7.2
0.2	199.8	1,005.3	14.9
Grand Avg	105.2	629.3	20.4

Table 2: Summary of the performance with respect to poisoning rates

Table 2 gives a summary by aggregating the previously stated performance measures for each poisoning rate. The first column contains the poisoning rate values used in the experiments. The second and third columns represent the running time of GD averaged over 10 instances and the average running time of IAS (including single attack strategy) in seconds which are computed over 70 runs, respectively. The running times are positively correlated with the value of poisoning rates, as expected. The last column gives the relative MSE change with respect to GD poisoned regression averaged over 140 MSE^{ta}_{change} values obtained with the same poisoning rates. The change in MSE with respect to GD regression reveals that our attack strategies output higher MSEs than GD for any poisoning rate and emphasizes that the highest MSE difference is obtained for lower poisoning rates, namely for 4% and 8%. The detailed results which we use to prepare the summary tables, namely Tables 1 and 2, can be found in Tables 4–9 of the Appendix .

5.4. Impact of Poisoning Attacks on Individual Predictions

In this section, we discuss the influence of poisoning attacks in real life using House Price and Healthcare datasets. For this purpose, we focus on the test instances with 4% poisoning rate where the attack strategy is assumed to be IAS with $K = 12$. We observe that for 54.2% of the houses the price is estimated higher than their real sale price. Hence, the poisoned regression model does not favor only low or high prices. One extreme example is a house with sale price of 67,000$; it is estimated to worth 409,655$. Similarly, one of the most expensive houses with 745,000$ sale price is predicted to worth only 34,900$. In 2.1% and 10.5% of the houses, the deviation between the real and predicted sale prices, which is calculated as $100 \times \frac{|\hat{y}-y|}{y}$, are more than 200% and 100%, respectively. Moreover, the percentage of houses whose predicted sale price is at least 75% and 50% away from the original price is 20.3% and 38.2%, respectively. On the other hand, 17.5% of the houses are estimated to have sale prices within 10% of the real value. One third of the houses fall in the category whose prediction error is within 20% despite the poisoning attacks. Therefore, the net worth of a considerable number of houses are estimated poorly even though the poisoning rate is as small as 4%.

The effect of 4% poisoning rate is even more significant on Healthcare dataset. We can state a patient whose drug dosage is changed from 3.5 to 43.5 mg per week and one whose weekly dosage is reduced from 98 mg to 2.1 mg as individual examples. For 9.3% of the patients, the estimated drug dosage is at least 200% away from the real dose. The drug dosage is predicted to be at least 100% different for 21.9% of the patients. Almost half of the patients (46.3%) and 62.8% of them have their dosage changed by at least 75% and 50%, respectively. Furthermore, the change in dosage is within 10% and 20% for only 7.9% and 14.9% of the patients, respectively. These numbers show that even a small-scale poisoning attack dramatically changes the dosage of a drug negatively affecting the well-being of patients.

5.5. Attacks in Shorter Times

The computational results of the experiments show that one disadvantage of IAS can be the running time of the method. The running time is directly related to the default termination settings of KNITRO solver. This section provides a comparison of running times when the optimality tolerance of KNITRO is changed from its default value (10^{-6}) to

10^{-3}. We re-ran the experiments for IAS with $K = 6$ and $K = 4$. These attacks are repeated only for poisoning rate of 20% for which the average running time is the highest. The experiments with the default setting of optimality tolerance equal to 10^{-6} take 745.3 seconds whereas the runs with increased optimality tolerance up to 10^{-3} terminates in 525.0 seconds, on average. The average MSE with default and altered settings are equal to 0.088 and 0.087, respectively. To sum up, experimenting with only one parameter of KNITRO results in 30% improvement in running times of IAS while not changing the MSE significantly. Therefore, IAS is a promising attack strategy for which shorter running times can be achieved by fine tuning the termination parameters of the solver.

5.6. Attacks Under Black-box Scenario

Attack Strategy	MSE^{a}_{change} (%)
Single Attack	-5.24
IAS (K=12)	4.24
IAS (K=6)	0.73
IAS (K=4)	0.37
IAS (K=3)	0.11
IAS (K=2)	2.45
IAS (K=1)	3.39
IAS Average	0.86

Table 3: Performance of strategies in black-box scenario

We also assess how IAS performs in black-box scenario described in Section 3.1. The experiments are carried out assuming that the training data used in the optimization problem is the substitute training dataset $\mathcal{D}'$, i.e. the guess of the attacker, since the attacker has limited knowledge about the real training data. We determine $\mathcal{D}'$ by random selection from the entire dataset. We adopted this approach because we know that both $\mathcal{D}'$ and $\mathcal{D}^{tr}$ have the same distribution. However, it is not known how similar they are. The poisoning attack samples are optimized with respect to $\mathcal{D}'$. Then, they are added to $\mathcal{D}^{tr}$ and a regression model is learned. Reported MSE is calculated on the real test and validation data of the learner. The performance of attack strategies in black-box scenario are summarized in Table 3. The MSE^{a}_{change} value is calculated by averaging over 100 MSE^{ta}_{change} values for every IAS and the benchmark GD. We observe that MSE obtained with IAS is 0.86% higher than of GD, on average. IAS provides slightly higher MSEs than GD except for the single attack strategy. Furthermore, IAS ($K = 12$) is the most successful attack strategy with black-box attacks with an average of 4.24% MSE increase. IAS ($K = 1$) and IAS ($K = 2$) are the other promising methods in black-box scenario. The summary in Table 3 is obtained from the detailed results provided in Tables 10–13 of the Appendix.

6. Conclusion and Future Research

This paper proposes a bilevel optimization formulation for poisoning attacks on ridge regression learning. To the best of our knowledge, it is the first study which explicitly solves a bilevel optimization problem to determine poisoning attacks on regression. To this end, a single level nonlinear formulation equivalent to the bilevel formulation is presented.

The resulting problem is solved using a commercial nonlinear optimization problem solver. An iterative solution approach is developed where a fraction of attacks is determined at every iteration in addition to the straightforward approach in which the optimization problem is solved at a single attempt. The performance of the newly proposed attack strategies is compared to a benchmark gradient descent algorithm from the literature. They are proven to perform significantly better than the benchmark in terms of MSE and they are competitively fast when the number of iterations are kept small. In the white-box scenario, our attack strategies achieve 20% higher MSE than the benchmark does. For small poisoning rates of 4% and 8%, the MSE obtained with the proposed attack strategies are significantly higher than MSE given by the benchmark attack algorithm. When the poisoning rate is 12%, 16% or 20%, the iterative attack strategy is far more successful than the single attack strategy and the benchmark. In the black-box scenario, the newly proposed attack strategies result in equally good attacks to the ones generated by the benchmark.

The attack strategies are promising as there is a room for improving the running times by tuning the parameters of the optimization solver. Future research will also consider applying the described strategies on lasso regression. The mixed integer modeling approach to handle categorical features would also be interesting to explore.

References

Alegre, F., Soldi, G., Evans, N., Fauve, B., Liu, J., 2014. Evasion and obfuscation in speaker recognition surveillance and forensics, in: 2nd International Workshop on Biometrics and Forensics, IEEE. doi:10.1109/iwbf.2014.6914244.

Barreno, M., Nelson, B., Sears, R., Joseph, A.D., Tygar, J.D., 2006. Can machine learning be secure?, in: Proceedings of the 2006 ACM Symposium on Information, computer and communications security - ASIACCS '06, ACM Press. doi:10.1145/1128817.1128824.

Biggio, B., Nelson, B., Laskov, P., 2012. Poisoning attacks against support vector machines arXiv:1206.6389v3.

Biggio, B., Roli, F., 2018. Wild patterns: Ten years after the rise of adversarial machine learning. Pattern Recognition 84, 317–331. doi:10.1016/j.patcog.2018.07.023.

Carbonneau, R., Laframboise, K., Vahidov, R., 2008. Application of machine learning techniques for supply chain demand forecasting. European Journal of Operational Research 184, 1140–1154. doi:10.1016/j.ejor.2006.12.004.

Carminati, M., Santini, L., Polino, M., Zanero, S., 2020. Evasion attacks against banking fraud detection systems, in: 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020), USENIX Association, San Sebastian. pp. 285–300.

Carnerero-Cano, J., Muñoz-González, L., Spencer, P., Lupu, E.C., 2020. Regularisation can mitigate poisoning attacks: A novel analysis based on multiobjective bilevel optimisation arXiv:2003.00040v2.

Dalvi, N., Domingos, P., Mausam, Sanghai, S., Verma, D., 2004. Adversarial classification, in: Proceedings of the 2004 ACM SIGKDD international conference on knowledge discovery and data mining - KDD '04, ACM Press. doi:10.1145/1014052.1014066.

Dempe, S., 2002. Foundations of Bilevel Programming. Springer US, Boston, MA.

Gage, B., Eby, C., Johnson, J., Deych, E., Rieder, M., Ridker, P., Milligan, P., Grice, G., Lenzini, P., Rettie, A., Aquilante, C., Grosso, L., Marsh, S., Langaee, T., Farnett, L., Voora, D., Veenstra, D., Glynn, R., Barrett, A., McLeod, H., 2008. Use of pharmacogenetic and clinical factors to predict the therapeutic dose of warfarin. Clinical Pharmacology & Therapeutics 84, 326–331. doi:10.1038/clpt.2008.10.

Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., Li, B., 2018. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning arXiv:1804.00308v1.

Kaggle, . House prices: Advanced regression techniques. "https://www.kaggle.com/c/house-prices-advanced-regression-techniques". Accessed 7 March 2021.

Koh, P.W., Steinhardt, J., Liang, P., 2018. Stronger data poisoning attacks break data sanitization defenses arXiv:1811.00741v1.

Lecun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Muller, U., Sackinger, E., Simard, P., Vapnik, V., 1995. Comparison of learning algorithms for handwritten digit recognition, in: Fogelman, F., Gallinari, P. (Eds.), International Conference on Artificial Neural Networks, Paris, EC2 Cie. pp. 53–60.

Lowd, D., 2005. Good word attacks on statistical spam filters, in: In Proceedings of the Second Conference on Email and Anti-Spam (CEAS).

Madhuri, C.R., Anuradha, G., Pujitha, M.V., 2019. House price prediction using regression techniques: A comparative study, in: 2019 International Conference on Smart Structures and Systems (ICSSS), IEEE. doi:10.1109/icsss.2019.8882834.

Mozaffari-Kermani, M., Sur-Kolay, S., Raghunathan, A., Jha, N.K., 2015. Systematic poisoning attacks on and defenses for machine learning in healthcare. IEEE Journal of Biomedical and Health Informatics 19, 1893–1905. doi:10.1109/jbhi.2014.2344095.

Muñoz-González, L., Pfitzner, B., Russo, M., Carnerero-Cano, J., Lupu, E.C., 2019. Poisoning attacks with generative adversarial nets arXiv:1906.07773v2.

Padberg, M.W., 1979. Covering, packing and knapsack problems, in: Discrete Optimization I, Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium. Elsevier, pp. 265–287. doi:10.1016/s0167-5060(08)70831-8.

Qian, Y., Ma, D., Wang, B., Pan, J., Wang, J., Gu, Z., Chen, J., Zhou, W., Lei, J., 2020. Spot evasion attacks: Adversarial examples for license plate recognition systems with convolutional neural networks. Computers & Security 95, 101826. doi:10.1016/j.cose.2020.101826.

Ren, K., Zheng, T., Qin, Z., Liu, X., 2020. Adversarial attacks and defenses in deep learning. Engineering 6, 346–360. doi:10.1016/j.eng.2019.12.012.

Su, J., Vargas, D.V., Sakurai, K., 2019. One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation 23, 828–841. doi:10.1109/tevc.2019.2890858.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R., 2013. Intriguing properties of neural networks arXiv:1312.6199v4.

Wen, J., Zhao, B.Z.H., Xue, M., Qian, H., 2020. With great dispersion comes greater resilience: Efficient poisoning attacks and defenses for online regression models arXiv:2006.11928v3.

Xiao, H., Biggio, B., Brown, G., Fumera, G., Eckert, C., Roli, F., 2015. Is feature selection secure against training data poisoning?, in: Bach, F., Blei, D. (Eds.), Proceedings of the 32nd International Conference on Machine Learning, PMLR, Lille, France. pp. 1689–1698.

Zhang, J., Thomas, L.C., 2012. Comparisons of linear regression and survival analysis using single and mixture distributions approaches in modelling LGD. International Journal of Forecasting 28, 204–215. doi:10.1016/j.ijforecast.2010.06.002.

7. APPENDIX: Detailed Experiment Results

Table 4–Table 13 include the computational results averaged over 5 instances. They are used to prepare the summary tables Table 1–Table 3 and the plots in Figures 1–2, in Section 5.

Poisoning rate	**Unpoisoned**	**GD**	**Single Attack**	**IAS K=12**	**IAS K=6**	**IAS K=4**	**IAS K=3**	**IAS K=2**	**IAS K=1**
0.04	0.003	0.026	0.035	0.035	0.034	0.034	0.035	0.034	0.036
0.08	0.003	0.045	0.054	0.069	0.055	0.065	0.062	0.060	0.060
0.12	0.005	0.064	0.078	0.072	0.073	0.063	0.073	0.069	0.068
0.16	0.003	0.086	0.058	0.092	0.097	0.091	0.085	0.082	0.071
0.2	0.003	0.072	0.072	0.111	0.101	0.124	0.109	0.113	0.110

Table 4: Average MSE values on validation data for House Price dataset

Poisoning rate	Unpoisoned	GD	Single Attack	IAS K=12	IAS K=6	IAS K=4	IAS K=3	IAS K=2	IAS K=1
0.04	0.003	0.025	0.034	0.034	0.032	0.032	0.035	0.035	0.036
0.08	0.003	0.046	0.048	0.057	0.050	0.054	0.055	0.056	0.057
0.12	0.003	0.051	0.088	0.066	0.072	0.067	0.069	0.067	0.067
0.16	0.003	0.077	0.056	0.084	0.094	0.090	0.086	0.087	0.079
0.2	0.003	0.089	0.065	0.102	0.097	0.108	0.107	0.107	0.104

Table 5: Average MSE values on test data for House Price dataset

Poisoning rate	Unpoisoned	GD	Single Attack	IAS K=12	IAS K=6	IAS K=4	IAS K=3	IAS K=2	IAS K=1
0.04	0.003	0.015	0.020	0.020	0.018	0.020	0.020	0.020	0.020
0.08	0.002	0.027	0.028	0.031	0.034	0.037	0.035	0.033	0.033
0.12	0.003	0.044	0.033	0.047	0.042	0.044	0.044	0.046	0.045
0.16	0.003	0.047	0.041	0.052	0.054	0.057	0.048	0.049	0.050
0.20	0.003	0.061	0.054	0.073	0.071	0.068	0.070	0.077	0.075

Table 6: Average MSE values on validation data for Healthcare dataset

Poisoning rate	Unpoisoned	GD	Single Attack	IAS K=12	IAS K=6	IAS K=4	IAS K=3	IAS K=2	IAS K=1
0.04	0.003	0.015	0.019	0.019	0.018	0.019	0.019	0.020	0.019
0.08	0.002	0.029	0.026	0.030	0.031	0.033	0.032	0.031	0.031
0.12	0.003	0.043	0.034	0.047	0.045	0.046	0.045	0.045	0.048
0.16	0.003	0.048	0.044	0.057	0.060	0.062	0.049	0.051	0.054
0.20	0.003	0.061	0.053	0.068	0.068	0.066	0.066	0.071	0.070

Table 7: Average MSE values on test data for Healthcare dataset

Poisoning rate	GD	Single Attack	IAS K=12	IAS K=6	IAS K=4	IAS K=3	IAS K=2	IAS K=1
0.04	127.1	18.9	22.2	41.0	114.0	106.6	279.1	273.1
0.08	96.2	32.0	49.4	80.3	94.8	184.5	288.0	867.2
0.12	91.2	43.6	72.5	111.3	419.1	525.2	614.2	1,058.9
0.16	148.7	75.4	294.6	222.7	198.0	598.6	1,037.4	1,590.9
0.2	192.0	118.0	258.9	505.3	337.0	632.2	593.7	2,007.2

Table 8: Average running times (in seconds) of different attack strategies for House Price dataset

Poisoning rate	GD	Single Attack	IAS K=12	IAS K=6	IAS K=4	IAS K=3	IAS K=2	IAS K=1
0.04	22.2	10.8	10.8	26.6	145.5	94.4	227.6	915.6
0.08	40.6	18.8	80.7	227.2	227.7	505.9	936.2	1,623.9
0.12	58.1	35.1	182.8	142.9	650.7	1,137.6	910.9	2,922.4
0.16	68.4	47.7	386.6	691.9	1,628.8	1,656.1	1,913.0	3,305.4
0.2	207.5	66.0	243.1	1,064.3	1,332.6	1,325.7	2,210.1	3,380.3

Table 9: Average running times (in seconds) of different attack strategies for Healthcare dataset

Poisoning rate	Unpoisoned	GD	Single Attack	IAS K=12	IAS K=6	IAS K=4	IAS K=3	IAS K=2	IAS K=1
0.04	0.003	0.038	0.040	0.040	0.034	0.036	0.033	0.036	0.036
0.08	0.003	0.061	0.059	0.055	0.056	0.060	0.057	0.063	0.062
0.12	0.005	0.087	0.072	0.086	0.081	0.076	0.081	0.080	0.082
0.16	0.003	0.105	0.063	0.098	0.091	0.088	0.092	0.091	0.111
0.2	0.003	0.112	0.081	0.131	0.123	0.111	0.121	0.125	0.115

Table 10: Black-box scenario: Average MSE values on validation data for House Price dataset

Poisoning rate	**Unpoisoned**	**GD**	**Single Attack**	**IAS K=12**	**IAS K=6**	**IAS K=4**	**IAS K=3**	**IAS K=2**	**IAS K=1**
0.04	0.003	0.038	0.038	0.038	0.035	0.033	0.037	0.035	0.037
0.08	0.003	0.058	0.052	0.052	0.053	0.054	0.051	0.056	0.055
0.12	0.003	0.082	0.074	0.085	0.082	0.072	0.078	0.077	0.083
0.16	0.003	0.098	0.056	0.087	0.093	0.086	0.088	0.096	0.100
0.2	0.003	0.119	0.074	0.134	0.127	0.123	0.127	0.127	0.112

Table 11: Black-box scenario: Average MSE values on test data for House Price dataset

Poisoning rate	**Unpoisoned**	**GD**	**Single Attack**	**IAS K=12**	**IAS K=6**	**IAS K=4**	**IAS K=3**	**IAS K=2**	**IAS K=1**
0.04	0.003	0.019	0.024	0.024	0.025	0.025	0.023	0.025	0.024
0.08	0.002	0.040	0.041	0.041	0.041	0.042	0.040	0.040	0.039
0.12	0.003	0.049	0.043	0.044	0.044	0.046	0.046	0.045	0.042
0.16	0.003	0.065	0.062	0.068	0.060	0.068	0.065	0.062	0.067
0.2	0.003	0.081	0.080	0.081	0.079	0.075	0.079	0.080	0.086

Table 12: Black-box scenario: Average MSE values on validation data for Healthcare dataset

Poisoning rate	**Unpoisoned**	**GD**	**Single Attack**	**IAS K=12**	**IAS K=6**	**IAS K=4**	**IAS K=3**	**IAS K=2**	**IAS K=1**
0.04	0.003	0.019	0.024	0.024	0.024	0.024	0.023	0.024	0.024
0.08	0.002	0.034	0.038	0.039	0.038	0.038	0.037	0.038	0.036
0.12	0.003	0.047	0.043	0.047	0.045	0.045	0.046	0.046	0.044
0.16	0.003	0.067	0.068	0.067	0.065	0.068	0.065	0.065	0.068
0.2	0.003	0.084	0.081	0.076	0.072	0.073	0.073	0.074	0.080

Table 13: Black-box scenario: Average MSE values on test data for Healthcare dataset