IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 20, 2019, accepted January 19, 2020, date of publication January 24, 2020, date of current version February 4, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2969276

Model Capacity Vulnerability in Hyper-Parameters Estimation

WENTAO ZHAO[1], XIAO LIU[1], QIANG LIU[1], (Member, IEEE), JIUREN CHEN[2], AND PAN LI[3]

[1]College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China
[2]Science and Technology on Test Physics and Numerical Mathematic Laboratory, Beijing 100073, China
[3]School of Electronic Engineering and Computer Science, Queen Mary University of London, London E1 4NS, U.K.

Corresponding author: Qiang Liu (qiangliu06@nudt.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61702539 and Grant U1811462, in part by the Hunan Provincial Natural Science Foundation of China under Grant 2018JJ3611, and in part by the NUDT Research Project under Grant ZK-18-03-47.

ABSTRACT Machine learning models are vulnerable to a variety of data perturbation. Recent research mainly focuses on the vulnerability of model training and proposes various model-oriented defense methods to achieve robust machine learning. However, most of the existing research overlooks the vulnerability of model capacity, which is more fundamental for model performance. In this paper, we study an adversarial vulnerability of model capacity caused by the poisoning on the estimation of model hyper-parameters. We further implement this vulnerability catering for the polynomial regression model, on which the evading of model-oriented detection is challenging, to illustrate the effectiveness of the adversarial vulnerability. Extensive experiments on one synthetic and three real-world data sets demonstrate that the vulnerability can effectively mislead the hyper-parameter estimation of the polynomial regression model by poisoning a few numbers of camouflage samples that cannot be detected by model-oriented defense methods.

INDEX TERMS Adversarial vulnerability, model capacity, hyper-parameter poisoning, gradient-based optimization.

I. INTRODUCTION

As artificial intelligence is becoming a national strategy in more and more countries, machine learning, which can be divided into supervised learning [1], Semi-supervised learning [2] and unsupervised learning [3], is concerned as the most important method in data science. But machine learning methods are vulnerable to a variety of disturbances [4], such as poisoning [5]–[7], evasion [8], and impersonation [9]. Most of these disturbances introduce adversarial samples, which leverage the vulnerability of machine learning models to achieve malicious goals. Similar to [10], we reiterate the definition of adversarial vulnerabilities here as the weakness points of a learning model where adversarial examples crafted by feeding ϵ-sized $\|\cdot\|$-perturbations into genuine ones can result in a significant performance difference. For example, some work takes advantage of information loss caused by feature extraction [11]–[13] and add noisy signals in facial images to fraud a well-trained face recognition system [14], [15]. For another example, adversarial malware examples generated by a generative adversarial network can bypass the detection models [16].

As an outcome of the vulnerability study, recent research proposes many robust machine learning methods, which introduce various defense methods to detect and filter adversarial samples. According to the defense mechanism, we can classify these defense methods into two categories: model-independent defense method and model-oriented defense method. The model-independent method filters outlier data (a.k.a., data sanitization [17]; it includes outlier detection, correction, and removing) before model training. Although the model-independent method works well, it requires a lot of artificial designs and work. As a result, the model-independent method may not fit the model that trains on a large amount of data. In contrast, the model-oriented method modifies a learning model to filter or tolerate outliers in model training [18]–[21]. This method wraps outlier filtering into model training to achieve robust machine learning efficiently. Accordingly, it becomes the most popular method against model vulnerability.

The associate editor coordinating the review of this manuscript and approving it for publication was Venkateshkumar. M.

The model-oriented defense method can against most of the existing vulnerabilities. The principle is that most vulnerabilities are caused by adversarial samples that disturb the parameters of a learning model [22], [23]; the model-oriented defense method enables the parameters of a learning model to tolerate adversarial samples. For example, defense method [21], [24] can reduce the impact of outliers in model training by changing the loss function of the learning model. For another example, some researches [25], [26] can address the feature manipulation activities in the testing stage by introducing the game theory in model training. By this means, the model-oriented defense method effectively induces robust learning models.

In this paper, we study a new vulnerability, namely adversarial vulnerability of model capacity, which cannot be defended by the existing model-oriented defense methods. Here, model capacity refers to the ability of a model to fit a wide variety of functions [27]. It is more fundamental for model performance. For a given data, a model can perform well if its capacity fits the data complexity. Otherwise, the model (with little capacity) cannot comprehensively learn the data distribution, or it (with large capacity) may get stuck during the learning process. The studied vulnerability is caused by the poisoning on the estimation of model hyper-parameters, which determine the capacity of a learning model. As shown in Fig.1, a machine learning process can be split into two stages: hyper-parameter estimation and model parameter training. Because the current model-oriented defense methods focus only on model parameter training, they will fail when facing the vulnerability introduced by adversarial hyper-parameter estimation.

To illustrate the effectiveness of the adversarial vulnerability, we further implement the vulnerability of model capacity catering for the polynomial regression model, on which evading model-oriented detection is challenging. Specifically, we propose a gradient-based degree confuse poisoning (DC for short) method to implement the vulnerability. The DC method generates poisoning points that can perform very well at a specified degree but perform poorly at the other degrees. Accordingly, the generated points will mislead a wrong selection of the degree of polynomial regression. Also, these points can escape from the detection of model-oriented defense methods because these methods conduct only on the model parameter training stage.

In summary, this paper makes the following contributions:

- As far as we know, it is the first work to study the vulnerability of model capacity.
- This paper proposes an adversarial vulnerability of model capacity caused by the poisoning on the estimation of model hyper-parameters.
- This paper instantiates the adversarial vulnerability of model capacity on polynomial regression model by a degree confuse poisoning method.

We evaluate the adversarial vulnerability on one synthetic data set and three public data sets from different application fields. Extensive experiments demonstrate that the vulnerability can effectively mislead the hyper-parameter estimation of the polynomial regression model by poisoning a few number of camouflage samples (no more than 10% of the population) that cannot be detected by model-oriented defense methods.

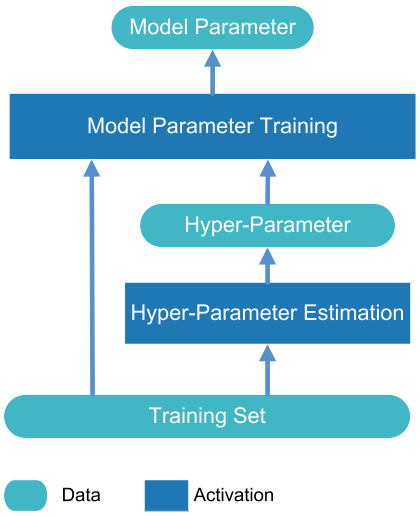


FIGURE 1. Model training process.

II. RELATE WORK

A. ADVERSARIAL VULNERABILITY OF MODEL

A lot of vulnerabilities have been studied target the model parameter. These vulnerabilities are mainly caused by *poisoning* and *spoofing*.

The poisoning method can change the parameters of a model in the training stage by injecting adversarial points to the training data set and reduce the performance of the classification and regression model. This method is mainly conducted on classification models. For example, Shen *et al.* [28] apply traditional poisoning points against NNs to prevent users' privacy images from being stolen and analyzed. Mozaffari-Kermani *et al.* [29] conduct poisoning against healthcare systems, resulting in severe and life-threatening consequences. Laishram and Phoha [30] examine the poisoning method against SVM. Recently, limited studies have been made on poisoning against regression models. For instance, Mei and Zhu [31] use gradient methods on an implicit function to solve the bi-level optimization problem. The machine with certain Karush-Kuhn-Tucker (KKT) conditions such as SVM, logistic regression and linear regression will perform worse with the generated poisoning points.

To exploit the vulnerabilities in model parameters, the spoofing method input elaborate adversarial samples into machine learning models, enabling these samples to avoid detection (evasion) or disguise himself as a specific target in classification methods (impersonate).

As the machine learning method has become an important method in the instruction detection system (IDS), the evasion method mainly used against IDS: Biggio *et al.* [32] present a gradient-based approach that can evade the malware detection

in PDF files. Zhang *et al.* [33] use gradient methods to optimize malicious samples so they can evade the detection of spam filtering systems.

The impersonate method mainly targets and classifies (especially multi-class) models: Kurakin *et al.* [9] demonstrated an impersonate method in the physical world, they print out the adversarial images as the inputs of the camera and successfully deceived the image classifier. Sharif *et al.* [14] achieve a spoofing attack via a pair of eyeglass frame. The adversary can evade being recognized or impersonate another individual by wearing the special glasses.

In general, the existing adversarial method mainly targets the vulnerability of model parameters. In this paper, we study the vulnerability of model capacity and prove that this vulnerability also has a significant impact on model performance.

B. VULNERABILITY DEFENSE METHOD

Aiming at protecting the vulnerability of model parameters, many researchers have proposed two kinds of methods: model-oriented defense method and model-independent method.

The model-independent method focus on the training set itself and regardless of subsequent training models. This method is mainly used to solve the problem of missing data or malformed data in the data set. Some researchers also use it to detect adversarial samples. Steinhardt *et al.* [34] remove the outliers by constructing approximate upper bounds on the loss across a broad family of attacks.

As manual data sanitization methods require a lot of manpower and the data sanitization programs have poor performance in detecting outliers, the current mainstream defense methods are model-oriented defense methods.

Some model-oriented methods aimed to remove or reduce the impact of the outliers in the training set. For example, the RONI method [35] measure the incremental effect of a sample by testing the performance difference with and without that sample. Then this method deletes the samples that have a great impact on the performance of the model. The Quantile regression [24] aims at estimating either the conditional median or other quantiles of the response variable, which makes it robust against outliers in the response measurements.

There are also some model-oriented methods that improve the robustness of the model during the model training stage to defend against possible security threats during the test stage. Globerson and Roweis [36] construct a classifier that is optimal in the worst case using the thought of game theory. Teo *et al.* [26] introduce adversarial samples with labels to training data to get a more robust model in the training stage.

However, the existing model-oriented defense methods only consider the vulnerabilities of model parameters. In this paper, we propose a vulnerability of model capacity, which is caused by poisoning on model hyper-parameters. Accordingly, this new vulnerability cannot be effectively defended by the existing model-oriented methods.

III. THE PROPOSED VULNERABILITY

In this section, we introduce a novel vulnerability that targets on model capacity. The introduced model capacity vulnerability is caused by poisoning data injected in a training data set, which is similar to the existing vulnerabilities caused by poisoning methods. However, the poisoning data lead to an inappropriate hyper-parameter selection in model capacity vulnerability instead of an improper model parameter training in the existing vulnerabilities.

The proposed model capacity vulnerability can be constructed by iteratively generating poisoning data to shift the target model hyper-parameters to an expected direction. As is shown in Fig.2, this construction repeats a two-steps procedure (i.e., *poisoning data optimization* and *poisoning data injection*) in the hyper-parameter estimation stage. The first step optimizes poisoning data according to the best hyper-parameters obtained in the current pre-training process. The second step injects the optimized poisoning data into the training data set for pre-training. The construction procedure iteratively repeats these two steps until the hyper-parameters estimated by the pre-training equal to expected values.

The proposed vulnerability could be very effective on most machine learning algorithms against the existing model-oriented defense methods. The rationale is as follows. In most of the machine learning algorithms, the performance of the model is directly determined by the model parameters. But most of the time, the hyper-parameter is only regarded as an optional factor of model performance without concerning its decisive role in model capacity. Therefore, there is not much attention to the stage of hyper-parameter selection. For example, many researchers leave the task of hyper-parameter selection to their students and thus get one of the most popular methods of hyper-parameter selection: babysitting method (which is also named as grad student descent method).

It is significant to study model capacity vulnerability since most machine learning methods have a hyper-parameter selection stage. For example, the support vector machine (SVM) needs to decide the kernel function as a hyper-parameter, such as linear function, polynomial function, RBF function and so on. Different kernel functions have their own hyper-parameters and in actual training, and on summary, the SVM function in the sklearn python library has 14 hyper-parameters that need to be input. The hyper-parameter adjustment is also a difficult job in the convolutional neural network (CNN): learning rate, iteration number, batch size, activation function, dropout, number of hidden layers, the units and so on. Researches on the model capacity vulnerability can help models to combat potential threats to model capability and enhance robustness.

IV. UTILIZATION OF MODEL CAPACITY VULNERABILITY ON POLYNOMIAL REGRESSION

In this section, we explore the utilization of the model capacity vulnerability and proposed a gradient-based degree

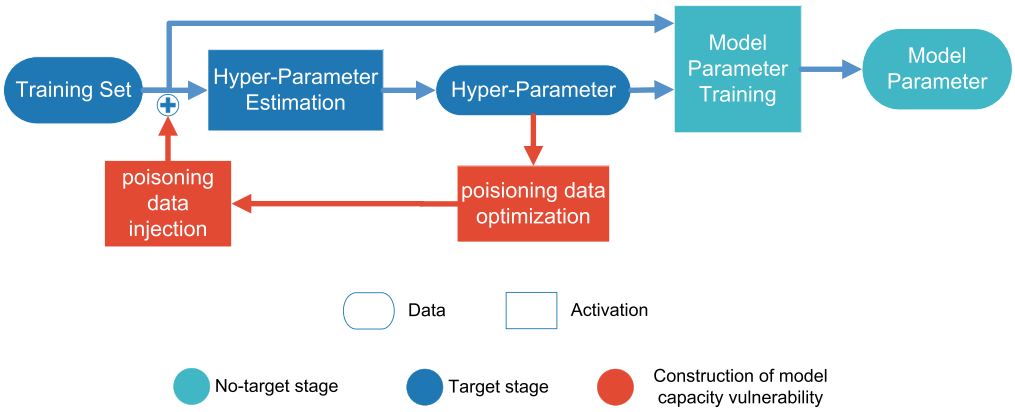


FIGURE 2. Utilization of model capacity vulnerability.

confusion poisoning method targeting the degree selection stage in a polynomial regression model.

A. BACKGROUND AND PRELIMINARY

Polynomial regression can be regarded as the extension of the linear regression problem to a high degree. For the polynomial regression model in degree k, the formula is:

$$\begin{aligned}\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \\ + \beta_j x_j + \beta_{j+1} x_1^2 + \beta_{j+2} x_1 x_2 + \cdots \\ + \beta_{C_{j+2}^2 - 1} x_j^2 + \beta_{C_{j+2}^2} x_1^3 + \cdots + \beta_{C_{j+k}^k - 1} x_j^k, \end{aligned} \quad (1)$$

where $\boldsymbol{x} = (x_1, x_2, .., x_j)$ and $\hat{y}$ is the predicted value of $\boldsymbol{x}$. We simplify its expression by defining $\boldsymbol{x}^{[k]}$ as:

$$\boldsymbol{x}^{[k]} = (1, x_1, \cdots, x_j, x_1^2, x_1 x_2, \cdots, x_j^2, x_1^3, \cdots, x_j^k)^{\mathrm{T}}, \quad (2)$$

then we get:

$$\hat{y}_i = \boldsymbol{x}_i^{[k]} \cdot \boldsymbol{\beta}, \quad (3)$$

where $\boldsymbol{\beta} = (\beta_0, \beta_1, \cdots, \beta_{C_{j+k}^k - 1})$. The dimension of parameter $\boldsymbol{\beta}$ is determined by both $\boldsymbol{x}$ and k, it can be divided into $k+1$ parts and every part contains all possible combinations of features with a total power equals to r ($r \in (0, 1, 2, \ldots, k)$). So the r^{th} part has C_{j+r-1}^r items and the total dimension of $\boldsymbol{\beta}$ is C_{j+k}^k.

It is clear that the degree k determines the dimension of the polynomial regression model parameters $\boldsymbol{\beta}$. A wrong degree will completely change the structure of a polynomial regression model, and then affect the performance of the model.

B. THE DEGREE CONFUSION POISONING METHOD

The hyper-parameter that needs to be estimated in polynomial regression is the polynomial degree. The degree is usually decided according to the algorithm as illustrated in Alg.1. The algorithm selects hyper-parameters by several random validation epochs. In each epoch, the algorithm randomly selects a small part (e.g., 30%) of training data as validation data and uses the remaining part to train the model. Then the algorithm traverses all possible degree values (for example, the degrees that are not too big to cause overfitting) and selects the degree with the smallest validation error as the best degree of this cycle. After r epochs, the algorithm sets the most frequent degree to the best degree k_{best}.

Algorithm 1 Degree Selection

Require: $\mathbb{D}$, repeat times r

$dict\_k \leftarrow [1:0; 2:0; 3:0; 4:0; 5:0; 6:0]$
$k_{best} \leftarrow 0$
for $i = 1$ to r **do**
  $tmp\_err \leftarrow \inf$
  $tmp\_k \leftarrow 0$
  for $k = 1$ to 6 **do**
    $\mathbb{D}_{trn}, \mathbb{D}_{val} \leftarrow division(\mathbb{D})$
    $\beta \leftarrow \arg\min_\beta L(\mathbb{D}_{train}^{[k]}, \beta)$
    if $MSE(\mathbb{D}_{val}^{[k]}, \beta) < tmp\_err$ **then**
      $tmp\_err \leftarrow MSE(\mathbb{D}_{val}^{[k]}, \beta)$
      $tmp\_k \leftarrow k$
    end if
  end for
  $dict\_k[tmp\_k] += 1$
end for
$k_{best} \leftarrow \max(dict\_k)$
for $j = 1$ to 6 **do**
  $dict\_k[j] \leftarrow dict\_k[j]/(r - dict\_k[k_{best}])$
end for
return Best degree k_{best}, $dict\_k$

Target on the vulnerability of the model capacity, a malicious adversary can inject poisoning points that fit a target degree k_{targ} but perform badly in other degrees (especially in the most suitable degree k_{best}). In degree selection stage, the poisoning points which used in training should make the model deviate greatly in non-target degree and that used in validation should perform bad in no-target degree. In order to distinguish models under different degrees, we make $\boldsymbol{\beta}^{[k]}$ as the training model when the degree is k, $\hat{y}^{[k]}$ be the prediction value of $\boldsymbol{x}$ in model $\boldsymbol{\beta}^{[k]}$ (s.t. $\hat{y}^{[k]} = \boldsymbol{x}^{[k]}\boldsymbol{\beta}^{[k]}$), and $\mathbb{D}^{[k]}$ be

the transformed data set when degree is k. The DC poisoning algorithm affect the model by maximizing the difference of $\hat{y}_i^{[k_{best}]}$ and $\hat{y}_i^{[k_{targ}]}$:

$$\begin{aligned} &\arg\max_{\boldsymbol{x}_c} \hat{y}_c^{[k_{best}]} - \hat{y}_c^{[k_{targ}]} \\ &s.t.\ \beta^{[k]} = \arg\min_{\beta} L(\{(\boldsymbol{x}_i^{[k]}, y_i)\}, \beta), \\ &\quad (\boldsymbol{x}_i, y_i) \in \mathbb{D}_{train} \\ &\quad (\boldsymbol{x}_c, y_c) \in \mathbb{D}_{poison} \end{aligned} \tag{4}$$

where $\mathbb{D}_{train}$ is the clean training set and the $\mathbb{D}_{poison}$ is a set of poisoning points. As is mentioned before, the degree selection algorithm randomly selection points to build a validation set, so it is not sure that if the poisoning points are used in training or validation. In this case, the influence of the poisoning points on the model parameter $\boldsymbol{\beta}$ is not considered in the optimizing process, i.e., $\boldsymbol{\beta}$ will not be updated in iteration:

$$\begin{aligned} &\nabla_{\boldsymbol{x}_c}(\hat{y}_c^{[k_{best}]} - \hat{y}_c^{[k_{targ}]}) \\ &\quad = \nabla_{\boldsymbol{x}_c}(\beta^{[k_{best}]} x_c^{[k_{best}]} - \beta^{[k_{targ}]} x_c^{[k_{targ}]}) \\ &\quad = \beta^{[k_{best}]} \nabla_{x_c} x_c^{[k_{best}]} - \beta^{[k_{targ}]} \nabla_{\boldsymbol{x}_c} x_c^{[k_{targ}]}. \end{aligned} \tag{5}$$

The computing method of $\nabla_{x_c} x_c^{[k]}$ is:

$$\nabla_{\boldsymbol{x}} \boldsymbol{x}^{[k]} = \begin{bmatrix} \frac{\partial 1}{\partial x_1} & \frac{\partial 1}{\partial x_2} & \cdots & \frac{\partial 1}{\partial x_j} \\ \frac{\partial x_1}{\partial x_1} & \frac{\partial x_1}{\partial x_2} & \cdots & \frac{\partial x_1}{\partial x_j} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial x_j^k}{\partial x_1} & \frac{\partial x_j^k}{\partial x_2} & \cdots & \frac{\partial x_j^k}{\partial x_j} \end{bmatrix}. \tag{6}$$

It is clear that the function $\hat{y}_c^{[k_{best}]} - \hat{y}_c^{[k_{targ}]}$ is not convex, which means that the poisoning point may be optimized to the extreme point during the optimization process. In this paper we add additional 5% points at the initial poisoning points selection stage, and optimize these poisoning points at the same time. Poor poison points will be removed at the end of the algorithm. As is shown in Alg.2, the algorithm first tests the most suitable degree of the data set. Then, given a poisoning rate rt, an initial poisoning points set $\mathbb{D}_{poison}$ contains $(rt + 0.05) \cdot |\mathbb{D}_{train}|$ points is randomly selected from training set $\mathbb{D}_{train}$. In every iteration, the algorithm calculates the probability that each degree value is selected as the best degree. For every possible degree k, the algorithm compute the gradient $\nabla_{x_c}(\hat{y}_c^{[k]} - \hat{y}_c^{[k_{targ}]})$ and weighted summation them based on their probability of occurrence as an optimization gradient. Using this gradient, the poisoning points are optimized to increase the difference between the validation error of the target degree and other possible degrees. At the end of each iteration, the label of the poisoning point $\boldsymbol{x}_c$, y_c is assigned the predicted value $\hat{y}_c^{k_{targ}}$ of $\boldsymbol{x}_c$. This allows the poisoned points to confuse the best degree in the degree selection stage, and also to evade detection of the model-oriented defense approach during the training stage. The algorithm ended if the most suitable degree k_{best} equals to the target degree k_{targ}.

Algorithm 2 DC Poisoning Method

Require: training set $\mathbb{D}_{train}$, k_{targ}, poisoning rate rt, step size λ, maximum degree k_{max}

```
k_best, dict_k ← DegreeSelection(D_train)
D_poison ← random(D_train, rt + 0.5)
while k_targ ≠ k_best do
    for x_c, y_c ∈ D_poison do
        x_c ← x_c + λ Σ_{k=1}^{k_max} dict_k[k] · ∇_{x_c}(ŷ_c^k − ŷ_c^{k_targ})
        y_c ← ŷ_c^{k_targ}
    end for
    D'_poison =remove poor poisoning point from D_poison
    D' ← D_train ∪ D'_poison
    k_best, dict_k ← DegreeSelection(D')
end while
return D'_poison
```

C. DISCUSSION

1) LINKS TO THE EXISTING POISONING METHODS

Although the proposed DC method is similar to the traditional poisoning method in injecting poisoning samples to the training set, there are some differences between the two kinds of poisoning points.

a: DIFFERENT OPTIMIZATION OBJECTIVE

According to the adversary intentions, the traditional poisoning method can be divided into two categories: one intends to invariably increase the error of the model during the testing stage while another only increases the testing error of a certain type of sample. Different from that, the hyper-parameter poisoning method intends to change the model capacity in the hyper-parameter selection stage.

b: DIFFERENT TRAINING ERROR

In order to change the model parameters during the training stage, traditional machine learning methods often inject poisoning points that are contrary to normal samples, forcing machine learning algorithms to deviate from normal samples. However, as the number of poisoning samples is much smaller than normal samples, the machine learning algorithms will preferentially fit normal samples, which results in a very large training error for poisoning samples. The hyper-parameter poisoning samples aimed at changing the result of the hyper-parameter selection stage, so it is not necessary for them to behave abnormally during the training phase.

2) OPEN PROBLEMS IN MODEL CAPACITY VULNERABILITY

Although the DC method demonstrates an example of exploiting the vulnerability of model capacity, there still are many open problems to be studied in about vulnerability:

a: SENSITIVITY OF THE DATA SET TO HYPER-PARAMETERS

The hyper-parameter poisoning method manipulates a model through change the hyper-parameter of the model. But if

the data set is insensitive to the hyper-parameter, changing the hyper-parameter will not lead to a great increased in the testing error.

b: CURSE OF DIMENSIONALITY

The model complexity is usually determined by data dimensions and hyper-parameters. In the majority of cases, a set of normal hyper-parameters will not makes the model too complicated, but the hyper-parameter poisoning points will manipulate the machine learning model to select a set of abnormal hyper-parameter which may lead to high-dimensional parameters. Even if there is no detection mechanism for anomalies, abnormal models may cause alertness to trainers.

V. EXPERIMENTS

In this section, we evaluate the effectiveness of model capacity vulnerability. In order to do that, we verified three properties of the DC poisoning: spatial distribution of poisoning points, ability to change degree, and impact on model performance.

A. COMPARISON METHODS

In this paper, we compare the proposed poisoning method with two typical methods: (1) *Baseline*: The baseline method randomly select some points from the training set and assigns new predictive values of these examples to $y_i' = 1 - y_i$ $(i = 1, \cdots, m)$, resulting in a baseline poisoning set $\{(x_i, y_i')\}_{i=1}^m$. (2) *Maximum Error (ME) poisoning*: For comparison with the traditional poisoning methods, we proposed a gradient-based poisoning against polynomial regression model. Aimed at maximizing the training error, this method iteratively calculates the gradient $\nabla_{\boldsymbol{x}_c} MSE(\mathbb{D}_{val}, \beta)$ and optimizes poisoning points $\boldsymbol{x}_c$. Detailed method analysis is showed in the appendix.

Meanwhile, we apply the poisoning methods on several model-oriented robust regression algorithms: (1) RANSAC [19] method first randomly select a subset from the training data set called consensus set and trains a model using this data set, then the algorithm adds the points which are consistent with the model to the consensus. The RANSAC algorithm iteratively repeats the above two steps until the obtained consensus set in certain iteration has enough points and this consensus set is considered as the clean training set. (2) Theil-Sen regression computes all the slope between pairs of points and chooses the median as the estimate of the regression slope. Then pass a line through each pair of $(\boldsymbol{x}, y)$ using the slope before, choose the median of the intercepts as the estimate of the regression intercept. (3) TRIM is proposed by Jagielski in [18]. This method iteratively estimates the regression parameters, while at the same time training on a subset of points with the lowest residuals in each iteration. Finally, it can get a model trained by the points with low residuals.

B. DATA SETS

In order to evaluate the vulnerability of the polynomial regression model capacity, one synthetic data set and three public data sets are used in our experiment. Among them, the synthetic data set is used to show the sample distribution, and the public data set is used to verify the poisoning effect. Each data set is divided into three parts: 60% of points in the data set are for training, 20% for validation, 20% for testing.

1) SYNTHETIC DATA SETS

The synthetic data is a randomly generated data set that fits a certain distribution function. This kind of data can meet the need for visualizing the distribution of adversarial points. In this paper, we generate the synthetic data set which fits the sine function:

$$y = sin(x) + e, \tag{7}$$

where e is a random error. The sine function is a typical function that can be fitted by polynomial function:

$$sin(x) = \frac{x^1}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots. \tag{8}$$

Generally speaking, a polynomial regression with a higher degree will fit this data set better. But the generated synthetic data set does not conform to the distribution of the sine function strictly, a polynomial regression model with a high degree may be overfitting on the synthetic data set. As the degree selection result shows (Tab.1), the validation error of the synthetic data set reaches a minimum at degree 8 and then begins to increase. Therefore, for the synthetic data set, we only consider its performance in the polynomial regression model with a degree of 10 or less.

2) PUBLIC DATA SETS

The public data sets are collected from UCI machine learning repository [37]. The data sets are both multiple-input and a single output, the degree selection program is executed on the data sets to get the most suitable degrees (in Tab.2).

a: AIRFOIL SELF-NOISE DATA SET ($\mathbb{D}_{air}$) [38]

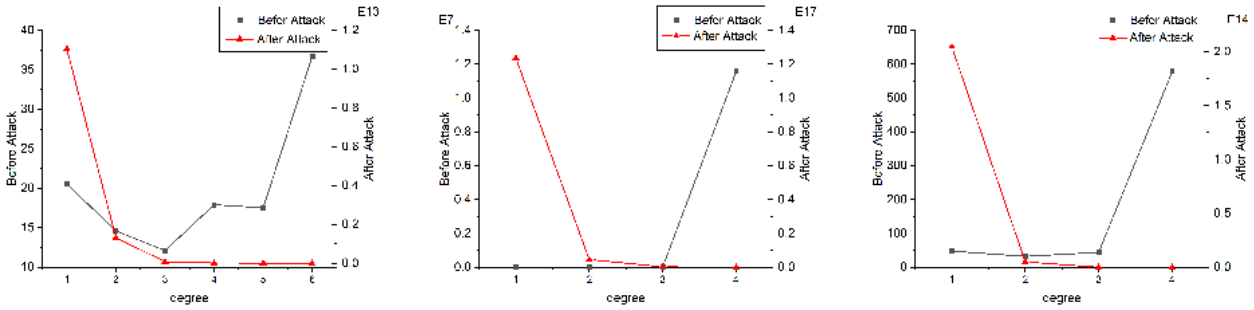
As described in [38], There are five features in the data set, namely frequency, the angle of attack, chord length, free-stream velocity, and suction side displacement thickness. The data set has only one output called scaled sound pressure level.

b: CONCRETE COMPRESSIVE STRENGTH DATA SET ($\mathbb{D}_{con}$) [39]

This data set contains 1030 data examples, which have 8 quantitative input variables and 1 quantitative output variable.

c: REAL ESTATE VALUATION DATA SET ($\mathbb{D}_{est}$) [40]

This market historical data set of real estate valuation is collected from Sindian Dist., New Taipei City, Taiwan. It contains 6 input variables and 1 output variable.



(a) Airfoil self-Noise data set (b) Concrete compressive strength data set (c) Real estate valuation data set

FIGURE 3. The validation error of public data sets before and after poisoning.

TABLE 1. Degree selection result for synthetic data set, where the frequency(F) represents the frequency of this degree as the best degree in repeated training, and VE is the validation error.

degree	1	2	3	4	5	6	7	8	9	10
F	0.0%	0.0%	0.0%	0.0%	0.0%	26.2%	15.0%	49.1%	6.9%	2.8%
VE	3.9e-01	4.8e-01	5.1e-01	5.9e-02	5.4e-02	9.7e-03	9.2e-03	7.7e-03	7.8e-03	7.8e-03

TABLE 2. The result of degree selection for public data sets. The most frequency degrees are concerned as the best degree.

Degree	**1**	**2**	**3**	**4**	**5**	**6**	**7**
$\mathbb{D}_{air}$	0	43	562	382	13	0	0
$\mathbb{D}_{con}$	7	187	806	0	0	0	0
$\mathbb{D}_{est}$	209	781	10	0	0	0	0

C. VERIFICATION OF THE MODEL CAPACITY VULNERABILITY

Since the model capacity is greatly affected by the change of hyper-parameters, a key question is whether the hyper-parameters estimation stage will be manipulated by the adversary. To verify the vulnerability of model capacity, we perform the proposed DC poisoning method target the polynomial regression model on the three public data sets and set the poisoning rate to 10% for all of the three public data sets.

we consider the validation error of the polynomial regression model at different degrees to determine the target degree k_{targ}. However, as the capacity of the polynomial model will increase greatly with the increase of the degree, an high degree may cause the dimension of the model parameters larger than the number of data, and eventually alert the trainer. In this case, we use both data dimension and size of data set to limit the target degree (s.t. $C_{j+k}^{k} < |\mathbb{D}_{train}|$).

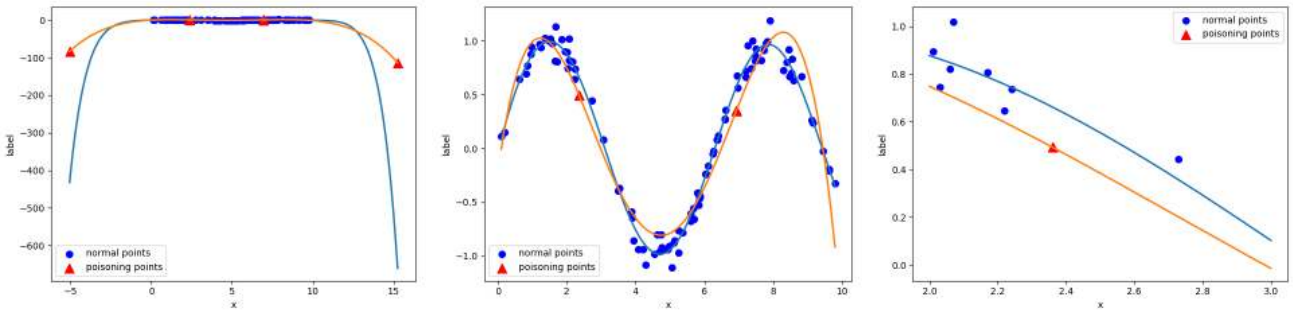
According to the data dimension and the size of data set, degrees of the three public data sets are limited to below 6, 4, 4. Then we compute the validation errors in every possible degree of three public data set and get the target degree which can cause maximum validation error: 6 for airfoil self-noise data set, 4 for concrete compressive strength data set and 4 for real estate validation data set. The result is shown in Tab.6.

Since the goal of DC poisoning method is changing the degree of the polynomial regression model, this poisoning method is concerned to be successful as long as the polynomial regression model selects the target degree in the degree selection stage. As is showed in Tab.2, the best degrees of the three public data sets are 3, 3, 2 before poisoning. Note that the degree selection program (in Alg.1) chooses the degree with the highest frequency of occurrence as the best degree. We mix the poisoning points with the training points and get the validation error. The comparative results are shown in Fig.3, we can find that the DC poisoning method leads to a sharp rise in validation error for all degrees except the target degree. The target degree has become the most suitable degree after poisoning.

The experiments show that the hyper-parameter estimation stage can be easily affected by the poisoning method with a small poisoning rate which is a key reason for the vulnerability of model capacity. By injecting elaborate poisoning points, the adversary can change the degree of the polynomial regression model to the specified value, thereby making the model overfitting or underfitting.

D. DISTRIBUTION OF POISONING POINTS

As most of the public data sets have high data dimensions, experimental results are difficult to visualize. In this paper, the proposed DC poisoning method is applied to the low dimensional synthetic data set. Through the scatter plots, we can intuitively see the spatial position relationship of poisoning points, normal points, normal model and poisoned model, which can help us understand the spatial distribution of the poisoning points vulnerability.



(a) The global view **(b)** The enlarged view **(c)** A poisoning point in extreme value

FIGURE 4. Distribution of normal points, poisoning points, normal model and poisoned model, the blue line is the clean polynomial model and the red line is the poisoned model.

TABLE 3. Testing error of different defense methods over airfoil self-noise data set, the proposed DC method perform worse when there is no defense method as the data set is hyper-parameter insensitive.

	baseline	ME	DC
no-defense	5.41e+02	**1.02e+03**	5.54e+01
RANSAC	2.77e+01	1.14e+02	**1.56e+02**
TRIM	2.13e+01	1.82e+01	**4.90e+01**
Theil-Sen	2.78e+01	1.71e+03	**6.07e+03**

TABLE 4. Testing error of different defense methods over concrete compressive strength data set.

	baseline	ME	DC
no-defense	1.40e+02	2.20e+02	**7.67e+06**
RANSAC	1.86e+02	3.24e+02	**2.82e+08**
TRIM	1.05e+02	1.05e+02	**9.24e+06**
Theil-Sen	1.78e+02	1.14e+02	**1.44e+07**

TABLE 5. Testing error of different defense methods over real estate valuation data set.

	baseline	ME	DC
no-defense	1.20e+02	1.96e+02	**4.59e+02**
RANSAC	6.90e+01	3.53e+03	**1.18e+04**
TRIM	7.13E+01	6.12e+01	**5.78e+02**
Theil-Sen	8.53e+01	**5.22e+02**	3.44e+02

The DC poisoning method effects the polynomial regression model choosing a pre-specified degree k_{targ}. For the synthetic data set, we set the target degree k_{targ} to 4. The result is shown in Fig.4. The global view shows that the poisoning points which distribute far away from normal points can be also far away from the model with k_{best} but fits the model with k_{targ}.

The experimental results show that in the poisoning-based model capacity vulnerability utilization, it is difficult for the model with normal hyper-parameters to fit the poisoning sample. In this case, the model will select a set of wrong hyper-parameters to get an overall best match (in the polynomial regression model, a wrong degree will be selected to get a lower validation error).

We can also find from the enlarged view that some poisoning points are very similar to normal points. That is because the gradient-descent method can only optimize the poisoning points to a locally optimal position when the function is not convex, so these poisoning points which move to extreme positions can not continue to be optimized.

E. AFFECTION OF THE MODEL CAPACITY VULNERABILITY

As the changes to model capacity will have an impact on model performance, in this section we will verify the effect of model capacity on the model performance and compare the hyper-parameter poisoning method with traditional poisoning method. The model performance is measured by testing error, which is measured using mean square error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2 \tag{9}$$

We first mix the poisoning points with the training points and select a suitable degree for the mixed data set. Then, several defense methods mentioned before are used on the data set. Finally, we can get four training models: model without defense method, model with RANSAC method, model with Theil-Sen method and model with TRIM method. The comparative results are shown in Tab.3, Tab.4 and Tab.5.

When there are no defense method in the polynomial regression model, we see that the ME method perform better than DC method in airfoil self-noise data set. That is because the airfoil self-noise data set has no major improvement when changing the degree (in Fig.3a), in other words, this dataset is not sensitive to model capacity. But in other two data sets, the change of hyper-parameter have a greater impact on model performance than change in parameters.

As said before, the traditional poisoning method is easy to be found by the model-oriented. The experiment result shows that the RANSAC method and the TRIM method can greatly reduce the impact of ME attacks and the Theil-Sen

TABLE 6. Choosing the target degree of the three public data sets by validation error: the degree with minimum validation error is best degree while that with maximum validation error is the target degree.

	dimension	size	max degree	min error	best degree	max error	target degree
airfoil self noise	5	632	6	14.09	3	404.74	6
concrete data	8	618	4	50.28	3	76358.02	4
real estate valuation	6	248	4	32.03	2	577.83	4

method works bad because it does not delete the poisoned points, but only reduces their weights. Different from the traditional poisoning method, the DC method affects model performance by changing model capacity, so when faced with three defense methods, its performance is almost not degraded. In most of times, as the DC poisoning points have lower training error than normal points, the defense method may treat the normal points as outliers and make the model perform worse.

In general, the impact of changing model capacity is greater than directly changing model parameters, and existing defense methods not only fail to correct changes in model capacity, but also worsen model performance.

VI. CONCLUSION

In this paper, we examine the vulnerability of the machine learning model capacity. Different from the model parameter vulnerability which happens in the training stage and testing stage, the model capacity vulnerability happens in the hyper-parameter estimation stage. We find that a minor change to the model hyper-parameters can cause huge changes in the model capacity, which may cause the model to overfit or underfit. Accordingly, it is significant to learn the vulnerability of model capacity, which has not received much attention.

We also illustrate the vulnerability by a gradient-descent based hyper-parameter poisoning method, which targets on the degree selection stage of a polynomial regression model. We show that the poisoning points generated by this method can induce an inappropriate model capacity and cannot be detected by the existing model-oriented defense methods.

APPENDIX
TRADITIONAL POISONING METHOD TARGET POLYNOMIAL REGRESSION MODEL

The traditional poisoning methods inject poisoning points into the training set to affect the model training stage and get a maximized testing error. The optimization function is formulated as:

$$\begin{aligned} &\arg\max_{\mathbb{D}_{poison}} \frac{1}{m}\sum_{i=1}^{m}(y_i-\hat{y}_i(\boldsymbol{x}_i,\beta^*))^2 \\ &s.t.\ \beta^*=\arg\min_{\beta} L(\mathbb{D}_{train}\cup\mathbb{D}_{poison},\beta), \\ &\quad (\boldsymbol{x}_i,y_i)\in\mathbb{D}_{val}, \end{aligned} \tag{10}$$

Algorithm 3 ME Poisoning Method

Require: $\mathbb{D}_{train}$, $\mathbb{D}_{val}$, a poisoning rate rt

$i \leftarrow 0$
$\mathbb{D}_p^{(i)} \leftarrow baseline(\mathbb{D}_{train}, rt)$
$\boldsymbol{\beta}^{(i)} \leftarrow polyRegTrain(\mathbb{D}_p^{(i)} \cup \mathbb{D}_{train})$
$valMSE^{(i)} \leftarrow MSE(\mathbb{D}_{val}, \boldsymbol{\beta}^{(i)})$
repeat
  for $k = 1, 2, \cdots, m$ **do**
    Retrieve a poisoning example $(\boldsymbol{x}_k^{(i)}, y'_k)$ from $\mathbb{D}_p^{(i)}$
    $\boldsymbol{x}_k^{(i+1)} \leftarrow \boldsymbol{x}_k^{(i)} + \lambda_k^{(i)} \nabla_{\boldsymbol{x}_k^{(i)}} MSE(\mathbb{D}_{val}, \boldsymbol{\beta}^{(i)})$
  end for
  $\mathbb{D}_p^{(i+1)} \leftarrow \bigcup_{i=1}^{m} \{(\boldsymbol{x}_i^{(i+1)}, y'_i)\}$
  $\boldsymbol{\beta}^{(i+1)} \leftarrow polyRegTrain(\mathbb{D}_p^{(i+1)} \cup \mathbb{D}_{train})$
  $valMSE^{(i+1)} \leftarrow MSE(\mathbb{D}_{val}, \boldsymbol{\beta}^{(i+1)})$
  $i \leftarrow i + 1$
until $\left|valMSE^{(i)} - valMSE^{(i+1)}\right| < \varepsilon$
$\mathbb{D}_{poison} \leftarrow \mathbb{D}_p^{(i)}$
return Resulting adversarial examples $\mathbb{D}_{poison}$

where L denotes the loss function of the target polynomial regression model. The bi-level optimization problem is usually solved by gradient descent method.

In this paper, we propose a gradient-based poisoning method named ME method. This method start from the baseline method. Specifically, n represents the size of a clean training data $\mathbb{D}_{train}$, rt refers to the poisoning rate. The baseline method select m ($m = rt * n$) training points to make an initial adversarial point set $\mathbb{D}_p^{(0)} = \{(x_i, y'_i)_{i=1}\}^m$. After that, the algorithm optimizes the features of the initial poisoning points through the gradient descent method. In every iteration, a gradient $\nabla_{\boldsymbol{x}} MSE$ is calculated for every adversarial examples in $\mathbb{D}_p^{(0)}$. The algorithm converges and outputs resulting in adversarial examples when the MSE difference between two continuous iterations is small enough. The pseudo code is showed in Alg.3.

The key parameter in optimization progress is the gradient $\nabla_{\boldsymbol{x}} MSE(\mathbb{D}_{val}, \boldsymbol{\beta})$.

$$\nabla_{\boldsymbol{x}} MSE = \nabla_{\boldsymbol{x}}\boldsymbol{\beta} \cdot \nabla_{\boldsymbol{\beta}} MSE. \tag{11}$$

The second part can be calculated directly:

$$\nabla_{\boldsymbol{\beta}} MSE(\mathbb{D}_{val}, \boldsymbol{\beta}) = \frac{2}{n}\sum_{t=1}^{n}(y_t - (\boldsymbol{x}_t\boldsymbol{\beta} + e_t))\boldsymbol{x}_t. \tag{12}$$

Now we consider the former part in the equation (11). As is mentioned before, polynomial regression models are usually solved by transforming it into linear regression. So we express $\nabla_{\boldsymbol{x}}\boldsymbol{\beta}$ as:

$$\nabla_{\boldsymbol{x}}\boldsymbol{\beta} = \nabla_{\boldsymbol{x}}\boldsymbol{x}^{[k]} \cdot \nabla_{\boldsymbol{x}^{[k]}}\boldsymbol{\beta}. \tag{13}$$

According to vector derivation rules, the calculation method of $\nabla_{\boldsymbol{x}}\boldsymbol{x}^{[k]}$ has been given in Eq.6

We have shown in line 3 of Algorithm 3 that $\boldsymbol{\beta}$ is the best solution of polynomial regression model whose training data is $\mathbb{D}_{train} \cup \mathbb{D}_p^{(i)}$. Then, it satisfies:

$$\nabla_{\boldsymbol{\beta}} L(\mathbb{D}_{train} \cup \mathbb{D}_{poison}, \boldsymbol{\beta}) = 0 \tag{14}$$

Meanwhile, this condition remains valid when updating $\boldsymbol{x}$. Hence, we also have:

$$\nabla_{\boldsymbol{x}^{[k]}}(\nabla_{\boldsymbol{\beta}} L(\mathbb{D}_{train} \cup \mathbb{D}_{poison}, \boldsymbol{\beta})) = 0. \tag{15}$$

Then,

$$\nabla_{\boldsymbol{x}^{[k]}}\nabla_{\boldsymbol{\beta}} L + \nabla_{\boldsymbol{x}^{[k]}}\boldsymbol{\beta}^T \cdot \nabla^2_{\boldsymbol{\beta}} L = 0. \tag{16}$$

Finally,

$$\nabla_{\boldsymbol{x}^{[k]}}\boldsymbol{\beta}^T = -\nabla_{\boldsymbol{x}^{[k]}}\nabla_{\boldsymbol{\beta}} L(\nabla^2_{\boldsymbol{\beta}} L)^{-1}. \tag{17}$$

REFERENCES

[1] L. Xiang, G. Guo, J. Yu, V. S. Sheng, and P. Yang, "A convolutional neural network-based linguistic steganalysis for synonym substitution steganography," *Math. Biosci. Eng.*, vol. 17, no. 2, pp. 1041–1058, 2019.

[2] X. J. Zhu, "Semi-supervised learning literature survey," Dept. Comput. Sci., Univ. Wisconsin-Madison, Madison, WI, USA, Tech. Rep., 2005.

[3] L. Xiang, G. Zhao, Q. Li, W. Hao, and F. Li, "TUMK-ELM: A fast unsupervised heterogeneous data learning approach," *IEEE Access*, vol. 6, pp. 35305–35315, 2018.

[4] Q. Liu, P. Li, W. Zhao, W. Cai, S. Yu, and V. C. M. Leung, "A survey on security threats and defensive techniques of machine learning: A data driven view," *IEEE Access*, vol. 6, pp. 12103–12117, 2018.

[5] P. Li, Q. Liu, W. Zhao, D. Wang, and S. Wang, "Chronic poisoning against machine learning based IDSs using edge pattern detection," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–7.

[6] P. Li, W. Zhao, Q. Liu, X. Liu, and L. Yu, "Poisoning machine learning based wireless IDSs via stealing learning model," in *Proc. Int. Conf. Wireless Algorithms, Syst., Appl.* Cham, Switzerland: Springer, 2018, pp. 261–273.

[7] Y. Dong, P. Zhu, Q. Liu, Y. Chen, and P. Xun, "Degrading detection performance of wireless IDSs through poisoning feature selection," in *Proc. Int. Conf. Wireless Algorithms, Syst., Appl.* Cham, Switzerland: Springer, 2018, pp. 90–102.

[8] H. Kwon, Y. Kim, K.-W. Park, H. Yoon, and D. Choi, "Multi-targeted adversarial example in evasion attack on deep neural network," *IEEE Access*, vol. 6, pp. 46084–46096, 2018.

[9] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2016, *arXiv:1607.02533*. [Online]. Available: https://arxiv.org/abs/1607.02533

[10] C.-J. Simon-Gabriel, Y. Ollivier, L. Bottou, B. Schölkopf, and D. Lopez-Paz, "First-order adversarial vulnerability of neural networks and input dimension," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5809–5817.

[11] W. Ding, C.-T. Lin, and W. Pedrycz, "Multiple relevant feature ensemble selection based on multilayer co-evolutionary consensus MapReduce," *IEEE Trans. Cybern.*, vol. 50, no. 2, pp. 425–439, Feb. 2020.

[12] W. Ding, C.-T. Lin, and M. Prasad, "Hierarchical co-evolutionary clustering tree-based rough feature game equilibrium selection and its application in neonatal cerebral cortex MRI," *Expert Syst. Appl.*, vol. 101, pp. 243–257, Jul. 2018.

[13] L. Xiang, X. Shen, J. Qin, and W. Hao, "Discrete multi-graph hashing for large-scale visual search," *Neural Process. Lett.*, vol. 49, no. 3, pp. 1055–1069, Jun. 2019.

[14] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2016, pp. 1528–1540.

[15] B. Biggio, L. Didaci, G. Fumera, and F. Roli, "Poisoning attacks to compromise face templates," in *Proc. Int. Conf. Biometrics (ICB)*, Jun. 2013, pp. 1–7.

[16] Z. Wang, "Deep learning-based intrusion detection with adversaries," *IEEE Access*, vol. 6, pp. 38367–38384, 2018.

[17] S. Wu, "A review on coarse warranty data and analysis," *Rel. Eng. Syst. Saf.*, vol. 114, pp. 1–11, Jun. 2013.

[18] 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.

[19] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[20] L. Y. Xiang, Y. Li, W. Hao, P. Yang, and X. B. Shen, "Reversible natural language watermarking using synonym substitution and arithmetic coding," *Comput. Mater. Continua*, vol. 55, no. 3, pp. 541–559, 2018.

[21] X. Dang, H. Peng, X. Wang, and H. Zhang, "Theil-sen estimators in a multiple linear regression model," Tech. Rep., 2008.

[22] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," in *Proc. Adv. Neural Inf. Process. Syst. 29 (NIPS)*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, 2016, pp. 1885–1893. [Online]. Available: http://papers.nips.cc/paper/6142-data-poisoning-attacks-on-factorization-based-collaborative-filtering.pdf

[23] S. Alfeld, X. Zhu, and P. Barford, "Data poisoning attacks against autoregressive models," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 1452–1458.

[24] R. Koenker and G. Bassett, "Regression quantiles," *Econometrica*, vol. 46, no. 1, pp. 33–50, Jan. 1978.

[25] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on GAN," 2017, *arXiv:1702.05983*. [Online]. Available: https://arxiv.org/abs/1702.05983

[26] C. H. Teo, A. Globerson, S. T. Roweis, and A. J. Smola, "Convex learning with invariances," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1489–1496.

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[28] J. Shen, X. Zhu, and D. Ma, "TensorClog: An imperceptible poisoning attack on deep neural network applications," *IEEE Access*, vol. 7, pp. 41498–41506, 2019.

[29] M. Mozaffari-Kermani, S. Sur-Kolay, A. Raghunathan, and N. K. Jha, "Systematic poisoning attacks on and defenses for machine learning in healthcare," *IEEE J. Biomed. Health Inform.*, vol. 19, no. 6, pp. 1893–1905, Nov. 2015.

[30] R. Laishram and V. V. Phoha, "Curie: A method for protecting SVM classifier from poisoning attack," 2016, *arXiv:1606.01584*. [Online]. Available: https://arxiv.org/abs/1606.01584

[31] S. Mei and X. Zhu, "Using machine teaching to identify optimal training-set attacks on machine learners," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 1–7.

[32] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Berlin, Germany: Springer, 2013, pp. 387–402.

[33] F. Zhang, P. P. K. Chan, B. Biggio, D. S. Yeung, and F. Roli, "Adversarial feature selection against evasion attacks," *IEEE Trans. Cybern.*, vol. 46, no. 3, pp. 766–777, Mar. 2016.

[34] J. Steinhardt, P. W. W. Koh, and P. S. Liang, "Certified defenses for data poisoning attacks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3517–3529.

[35] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. Rubinstein, U. Saini, C. Sutton, J. Tygar, and K. Xia, "Misleading learners: Co-opting your spam filter," in *Machine Learning in Cyber Trust*. Boston, MA, USA: Springer, 2009, pp. 17–51.

[36] A. Globerson and S. Roweis, "Nightmare at test time: Robust learning by feature deletion," in *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*, 2006, pp. 353–360.

[37] D. Dua and C. Graff. (2017). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml

[38] A. Asuncion and D. Newman. (2007). *UCI Machine Learning Repository*. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/airfoil+self-noise
[39] I.-C. Yeh, "Modeling of strength of high-performance concrete using artificial neural networks," *Cement Concrete Res.*, vol. 28, no. 12, pp. 1797–1808, Dec. 1998.
[40] I.-C. Yeh and T.-K. Hsu, "Building real estate valuation models with comparative approach through case-based reasoning," *Appl. Soft Comput.*, vol. 65, pp. 260–271, Apr. 2018.

WENTAO ZHAO received the Ph.D. degree from the National University of Defense Technology (NUDT), in 2009. He is currently a Professor with NUDT. His research interests include network performance optimization, information processing, and machine learning. He has edited one book *Database Principle and Technology* and several technical articles, such as Communications of CCF, WCNC'17, ICANN'17, WF-IoT, MDAI, and FAW. Since 2011, he has been serving as a member of the Council Committee of Postgraduate Entrance Examination of computer science and technology, NUDT.

XIAO LIU received the B.Eng. degree from the National University of Defense Technology, in 2017, where he is currently pursuing the M.S. degree. His research interests include machine learning and cyber security.

QIANG LIU (Member, IEEE) received the Ph.D. degree in computer science and technology from the National University of Defense Technology (NUDT), in 2014. From 2011 to 2013, he was a Visiting Scholar with the Department of Electrical and Computer Engineering, The University of British Columbia (UBC), Canada. He is currently an Assistant Professor with NUDT. He has contributed over 50 archived journal and international conference papers, such as the *IEEE Network Magazine*, the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the IEEE TRANSACTIONS ON CYBERNETICS, the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, *Pattern Recognition*, the IEEE COMMUNICATIONS LETTERS, *Neurocomputing*, *Neural Computing and Applications*, *Mobile Information Systems*, EDBT'17, WCNC'17, ICANN'17, and SmartMM'17. His research interests include 5G networks, the Internet of Things, wireless network security, and machine learning. He is a member of China Computer Federation (CCF). He serves on the Editorial Review Board of *Artificial Intelligence Research Journal*.

JIUREN CHEN received the B.E. and M.S. degrees in electronic science and technology from the University of Science and Technology Beijing, China, in 2012 and 2017, respectively. He is currently a Research Assistant with the Science and Technology on Test Physics and Numerical Mathematic Laboratory. His research interests include sentiment analysis, machine learning, and information security.

PAN LI received the B.Eng. degree from the University of Science and Technology Beijing, in 2016, and the M.S. degree from the National University of Defense Technology. He is currently pursuing the Ph.D. degree with the Queen Mary University of London. His research interests include machine learning and cyber security.