Closing the BIG-LID: An Effective Local Intrinsic Dimensionality Defense for Nonlinear Regression Poisoning

Sandamal Weerasinghe[1*], **Tamas Abraham**[2], **Tansu Alpcan**[1], **Sarah M. Erfani**[1], **Christopher Leckie**[1] and **Benjamin I. P. Rubinstein**[1]
[1]University of Melbourne, Australia
[2]Defence Science and Technology Group, Australia
prameesha.weerasinghe@unimelb.edu.au, tamas.abraham@dst.defence.gov.au,
{tansu.alpcan, sarah.erfani, caleckie, benjamin.rubinstein}@unimelb.edu.au

Abstract

Nonlinear regression, although widely used in engineering, financial and security applications for automated decision making, is known to be vulnerable to training data poisoning. Targeted poisoning attacks may cause learning algorithms to fit decision functions with poor predictive performance. This paper presents a new analysis of local intrinsic dimensionality (LID) of nonlinear regression under such poisoning attacks within a Stackelberg game, leading to a practical defense. After adapting a gradient-based attack on linear regression that significantly impairs prediction capabilities to nonlinear settings, we consider a multi-step unsupervised black-box defense. The first step identifies samples that have the greatest influence on the learner's validation error; we then use the theory of local intrinsic dimensionality, which reveals the degree of being an outlier of data samples, to iteratively identify poisoned samples via a generative probabilistic model, and suppress their influence on the prediction function. Empirical validation demonstrates superior performance compared to a range of recent defenses.

1 Introduction

Regression analysis, being a fundamental task in supervised machine learning, has applications in finance, cybersecurity, and engineering. However, compelling evidence suggests that the prediction performance of regression models degrades in the presence of poisoned training data [Liu *et al.*, 2017; Jagielski *et al.*, 2018]. By poisoning a subset of training data, adversaries force the learner to compute a compromised decision function that differs from the one it would have obtained under uncorrupted training data. As a consequence, applications that rely on regression analysis for high-stakes automated decision making may take incorrect actions with severe consequences. Therefore, investigating defense mechanisms against poisoning attacks is a continuing concern within the adversarial machine learning community.

*Corresponding Author

In robust statistics, estimation procedures are available for robust regression that are not strongly affected by the presence of stochastic noise and outliers [Huber, 1992]. However, these methods fail in high dimensions and struggle to identify carefully crafted adversarial samples [Xu *et al.*, 2012]. This has led to the development of defense algorithms against poisoning attacks on regression models. Liu *et al.* [2017] and Jagielski *et al.* [2018] present two such defenses that iteratively exclude data samples with the largest residuals from the training process. The proposed defenses require access to the number of benign samples in the training dataset (or an upper bound) *a priori*, thereby overestimating the learner's knowledge and limiting their practicality. While several other works have developed robust estimators for regression problems [Bhatia *et al.*, 2015; Balakrishnan *et al.*, 2017], they either specialize on specific problems or use complex convex optimization algorithms that scale poorly on large datasets [Diakonikolas *et al.*, 2019].

We propose a novel black-box defense called BIG-LID for *Balanced Information Gaussian - Local Intrinsic Dimensionality* that reduces the influence of poisoned samples on the cost function of the learner. At its core, BIG-LID relies upon the theory of Local Intrinsic Dimensionality (LID) [Houle, 2017], a local metric, to identify poisoned samples. It has previously been observed that LID can detect adversarial examples in classification settings in Deep Neural Networks (DNNs) and can serve as a mechanism to identify mislabeled samples [Ma *et al.*, 2018]. More importantly, recent work by Amsaleg *et al.* [2020] established that data points with relatively large LID values are vulnerable to adversarial perturbations compared to data points with smaller LID values.

BIG-LID is a multi-step iterative algorithm that brings together several previously established defense components together (i.e., influence, LID and weighting). In the first step, we filter the most influential samples (on the validation error) from the training data. In step two, we identify potentially poisoned samples from the above set using an approach based on LID. Finally, we suppress the samples that are suspected to be poisoned using a sample weighted optimization process. Due to the plug-in nature of BIG-LID, it can be used across different learning algorithms such as Support Vector Regression (SVR) and Neural Network Regression (NNR). We compare BIG-LID against other state-of-the-art alterna-

tives to show its superior performance in practice. Finally, we create an adaptive attacker that is aware of BIG-LID and demonstrate that such an attacker fails to achieve its objectives.

2 Related Work

The problem of learning under adversarial conditions has inspired a wide range of research from the machine learning community; see the work of Vorobeychik and Kantarcioglu [2018] for a survey. Although adversarial learning in classification and anomaly detection has received a lot of attention [Biggio and Roli, 2018; Carlini and Wagner, 2017], adversarial regression remains relatively unexplored.

Much of the current literature on adversarial regression learning is focused on linear models. Xiao *et al.* [2015] examined how the parameters of the linear regression model change under adversarial perturbations by introducing a novel threat model. Liu *et al.* [2017] use robust PCA to transform the training data to a lower-dimensional subspace and follow an iterative trimmed optimization procedure where only the n samples with the lowest residuals are used to train the model (n is the number of normal samples in the training data set). Jagielski *et al.* [2018] use a similar approach for the algorithm "TRIM", where trimmed optimization is performed in the input space itself. Therefore, they do not assume a low-rank feature matrix as done by Liu *et al.* [2017].

Koh and Liang [2017] introduced the notion of influence functions to construct adversarial training examples. In our work, we use the influence of training samples on the validation error in the first step of the defense. Similarly, the algorithm introduced by Zhang *et al.* [2018] identifies samples that have the biggest influence on the validation error of a trusted validation set. The method by Diakonikolas *et al.* [2019] constructs outlier scores via singular value decomposition of gradients. They offer theoretical guarantees under certain assumptions about the dataset. However, since the influence of samples alone was not sufficient to construct a capable defense, we used a LID based second step in BIG-LID.

Tong *et al.* [2018] consider an attacker that perturbs data samples during the test phase to induce incorrect predictions. The authors use a single attacker, multiple learner framework modeled as a multi-learner Stackelberg game. From the attack point of view, the most relevant studies to this paper are the works by Jagielski *et al.* [2018] and Alfeld *et al.* [2019]. We adapt the former attack to nonlinear settings for the experiments of this paper.

In summary, to the best of our knowledge, no defense has been proposed for nonlinear regression that reduces the effects of adversarial perturbations in training data using LID. More importantly, existing works on LID based defenses are not black-box. Most existing work on adversarial regression relies on assumptions about the attacker/attack or the dataset, making them impractical, while we propose a novel black-box defense that makes no such assumptions.

3 Problem Definition

The primary aim of this paper is to introduce a novel unsupervised black-box defense to counter poisoning attacks. The following is a brief description of the problem being addressed.

Define the set of benign training samples as $Z_{\text{tr}} = (X_{\text{tr}}, y_{\text{tr}})$, where $X_{\text{tr}} \in \mathbb{R}^{n\times d}$ is the feature matrix and $y_{\text{tr}} \in \mathbb{R}^{n\times 1}$ the corresponding response variable vector. We let x_i denote the i^{th} training sample, associated with a corresponding response variable $y_i \in \mathbb{R}$ from y. Note that each $y_i \in [0, 1]$ for $i \in \{1, \ldots, n\}$. The attacker has access to the benign data Z_{tr}, as well as perfect knowledge of the learner's training algorithm and its hyper-parameters (i.e., white-box attack). While these assumptions may lead to an overpowered attacker, our aim is to create a worst-case scenario for the learner (with the expectation that a defense that can withstand a strong attacker would stand against a less powerful real-world attacker). Moreover, security through obscurity is considered as poor practice when designing for security applications.

Armed with this knowledge, the attacker injects a subset of poisoned data samples $\bar{Z}_{\text{tr}} = (\bar{X}_{\text{tr}} \in \mathbb{R}^{p\times d}, \bar{y}_{\text{tr}} \in \mathbb{R}^{p\times 1})$ into the training set. The attacker aims to maximize the learner's prediction error for unseen data samples (i.e., increase test error). In real-world applications, such attacks can occur through unauthorized access (using malware), or when data is collected from multiple sources such as crowd-sourcing marketplaces where the authenticity of data contributors cannot be guaranteed. Due to the size and complexity of modern datasets, the sponsoring organization may not be able to extensively validate the quality of all data/labels.

The learner is presented with the poisoned training data $\tilde{Z}_{\text{tr}} = (\tilde{X}_{\text{tr}}, \tilde{y}_{\text{tr}})$, where $\tilde{X}_{\text{tr}} = X_{\text{tr}} \cup \bar{X}_{\text{tr}}$ and $\tilde{y}_{\text{tr}} = y_{\text{tr}} \cup \bar{y}_{\text{tr}}$. The learner's objective is to recover the nonlinear prediction model it would have obtained had the entire training data set been benign.

The learner is free to choose any nonlinear regression algorithm that supports a weighted loss function (e.g., NNR, SVR and basis function regression). In basis function regression, the data is first transformed to a higher ($q \gg d$) dimensional space using a nonlinear function (i.e., $\varphi : \mathbb{R}^d \to \mathbb{R}^q$), prior to fitting a hyperplane in the transformed space. The type of basis functions (common choices being polynomials and the Radial Basis Function, RBF) is chosen to appropriately model the nonlinearity in the relationship between the input features and the response variables. The resulting regression model can be described by a linear function of the form $h_\theta(x_i) = \omega^T\varphi(x) + b$, in which h is parameterized by the vector $\theta = (\omega, b) \in \mathbb{R}^{q+1}$, where $\omega \in \mathbb{R}^q$ are the feature weights and $b \in \mathbb{R}$ is the bias of the hyperplane.

The parameter vector θ is obtained by applying a learning algorithm, such as ridge regression (which uses ℓ_2-norm regularization) as follows:

$$\arg\min_{\theta} J(X_{\text{tr}}, y_{\text{tr}}, \theta) = \frac{1}{n}\sum_{i=1}^{n}\left(h_\theta(x_i) - y_i\right)^2 + \lambda\|\omega\|_2^2 . \quad (1)$$

Note that $\lambda > 0$ is a regularization hyperparameter. In our attack analysis (Section 4), we chose ridge regression as the learner's algorithm due to its objective function being differentiable. More generally, the attack could use subdifferentials.

3.1 Game-Theoretic Model

The interaction between the adversary and the learner can be simulated by a zero-sum continuous kernel leader-follower game. The attacker, being the leader, first chooses the attack strategy (i.e., adversarial perturbations $\bar{Z}_{\text{tr}}$ using Program 2), and then the learner chooses the best response (i.e., attack-resistant model θ^*) to the adversary's choice. We discuss the approaches used to obtain these in detail in the sections that follow (Algorithms 1 and 2). The adversary's goal is to maximize the learner's testing error (MSE on the test set) while adhering to constraints that limit the number of samples that can be perturbed and the size of the perturbations. The learner's goal is to minimize the testing error by suppressing the influence of poisoned data points on the trained model.

Having defined the problem being addressed in this paper, we now move on to discuss the formal attack model.

4 Attack Model

The section below formalizes the attack model for poisoning nonlinear regression models, which is inspired by previous works of Xiao *et al.* [2015] and Alfeld *et al.* [2019]. We start by outlining the adversary's goal, followed by its knowledge of the learner, its limitations.

4.1 Adversary's Goal

The adversary's goal is to force the learning algorithm to learn a compromised prediction function during the training phase, such that its prediction error for unseen data samples (i.e., testing phase) would be maximized. This poisoning attack can be categorized as an attack on availability, where the goal is to affect the prediction results indiscriminately. However, by changing the attacker's objective function it can be converted to an attack on integrity, where the attacker intends to cause specific mispredictions [Alfeld *et al.*, 2019].

4.2 Adversary's Knowledge

As previously stated, we consider a white-box attack where the attacker knows the benign training data Z_{tr} and the learning algorithm J (including λ and φ). Many researchers have utilized similar attacks to evaluate the resistance of defense algorithms against adversaries [Kurakin *et al.*, 2018].

4.3 Adversary's Limitations

While it is possible to consider an adversary with unbounded capabilities, it is unrealistic to encounter such an attacker in the real-world e.g., due to considerations of stealth. Therefore, we impose the following constraint on the attacker. The attacker is only allowed to inject a maximum of p poisoned samples into the training set. The size p bounds the attacker's capability and is fixed *a priori*. The number of samples in the training set thus becomes $n + p$, with n the number of benign samples. We define the poison percentage as $p/(n + p)$, which is the fraction of poisoned samples in the training set. Although previous work in adversarial linear regression has considered poison percentages up to 20%, we observed that in nonlinear conditions, the attacker could exert considerable damage with far fewer poisoned points. The attacker randomly selects a subset of p samples from the benign training set Z_{tr} as the starting points for the attack and iteratively perturbs their features to obtain the poisoned subset $\bar{Z}_{\text{tr}}$.

4.4 Poisoning Algorithm

We formalize the attacker's objective function as the following bi-level optimization problem:

$$\begin{aligned} \arg\max_{\bar{Z}_{\text{tr}}} \quad & E\big(X_{val}, y_{val}, \tilde{\theta}(\bar{Z}_{\text{tr}})\big) \ , \\ \text{s.t.} \quad & \tilde{\theta} \in \arg\min_{\theta} J(\tilde{X}_{\text{tr}}, \tilde{y}_{\text{tr}}, \theta) \ , \\ & |\bar{z}_{i,j} - z_{i,j}| \leq \epsilon, \ \forall \bar{z}_{i,j} \in \bar{Z}_{\text{tr}} \ , \end{aligned} \tag{2}$$

where E measures the squared loss $\frac{1}{m}\sum_{i=1}^{m}\big(h_{\tilde{\theta}}(x_i) - y_i\big)^2$ on a separate validation set, and is reflective of the learner's test error. Although large perturbations would have a bigger impact on the learner's prediction accuracy, they would also make the poisoned points detectable. Thus, it would be beneficial for the adversary to place poisoned data points closer to the benign data distribution. Which is why we limit the attacker's capability by constraining the perturbations to be within some threshold ϵ from the starting (benign) points using the second constraint in (2). However, it should be noted that this constraint can be changed according to the application being attacked.

In the gradient-based attack, the attacker updates $\bar{Z}_{\text{tr}}$ in the direction that maximizes E. However, as shown in (2), E depends implicitly on $\bar{Z}_{\text{tr}}$ through θ. Thus, using the chain rule we obtain

$$\nabla_{\bar{Z}_{\text{tr}}} E = \nabla_{\bar{Z}_{\text{tr}}} \theta^T . \nabla_{\theta} E \ . \tag{3}$$

If the inner optimization problem is convex (as is the case for ridge regression) and regular, we can calculate $\nabla_{\bar{Z}_{\text{tr}}}\theta$ in closed form. Similar to Xiao *et al.* [2015], we first replace the inner optimization problem of (2) with its Karush-Kuhn-Tucker (KKT) equilibrium conditions as:

$$\nabla_{\theta} J(\tilde{X}_{\text{tr}}, \tilde{y}_{\text{tr}}, \theta) = \begin{bmatrix} \frac{\partial J}{\partial \omega}^T \\ \frac{\partial J}{\partial b} \end{bmatrix} = 0 \ , \tag{4}$$

where

$$\frac{\partial J}{\partial \omega}^T = \frac{2}{n}\sum_{i=1}^{n+p}\big(h_{\theta}(\tilde{x}_i) - \tilde{y}_i\big)\varphi(\tilde{x}_i) + \lambda\omega^T \tag{5}$$

and

$$\frac{\partial J}{\partial b} = \frac{2}{n}\sum_{i=1}^{n+p}\big(h_{\theta}(\tilde{x}_i) - \tilde{y}_i\big) \ . \tag{6}$$

In order to ensure that the KKT conditions remain valid while updating $\bar{Z}_{\text{tr}}$, we set its derivative w.r.t. $\bar{Z}_{\text{tr}}$ to be equal to zero as follows,

$$\nabla_{\bar{Z}_{\text{tr}}}\big(\nabla_{\theta} J(\tilde{X}_{\text{tr}}, \tilde{y}_{\text{tr}}, \theta)\big) = 0 \ . \tag{7}$$

We need to obtain $\nabla_{\bar{Z}_{\text{tr}}}\theta^T$ to solve (3). The function $\nabla_{\theta} J$ is composed of both θ and $\bar{Z}_{\text{tr}}$. Since $\nabla_{\theta} J$ is continuously differentiable and $\nabla^2_{\theta} J$ is full rank, using the implicit function theorem results in,

$$\nabla_{\bar{Z}_{\text{tr}}}\theta^T = -\nabla_{\bar{Z}_{\text{tr}}}\nabla_{\theta} J(\nabla^2_{\theta} J)^{-1} \ . \tag{8}$$

Algorithm 1 Poisoning nonlinear regression

Input: benign data Z_{tr}, validation set Z_{val}, poison size p
Output: poisoned data $\bar{Z}_{\text{tr}}$

$t \leftarrow 0$
$\bar{Z}_{\text{tr}}^{(t)} \leftarrow$ select_initial_points(Z_{tr}, p) {randomly select the initial points}
$\tilde{Z}_{\text{tr}}^{(t)} \leftarrow Z_{\text{tr}} \cup \bar{Z}_{\text{tr}}^{(t)}$
$\tilde{\theta}^{(t)} \leftarrow$ train_model$(\tilde{Z}_{\text{tr}}^{(t)})$ {initial model}
$e^{(t)} = E(\tilde{\theta}^{(t)}, Z_{\text{val}})$ {initial validation error}
repeat
  $\nabla_{\bar{Z}_{\text{tr}}} E \leftarrow$ calculate_derivative$(\bar{Z}_{\text{tr}}^{(t)}, \tilde{\theta}^{(t)})$
  $\alpha \leftarrow$ select the step size through line search
  $\bar{Z}_{\text{tr}}^{(t+1)} \leftarrow \mathcal{P}(\bar{Z}_{\text{tr}}^{(t)} - \alpha \nabla_{\bar{Z}_{\text{tr}}} E)$ {project onto the feasible region}
  $t \leftarrow t + 1$ {increment the counter}
  $\tilde{Z}_{\text{tr}}^{(t)} \leftarrow Z_{\text{tr}} \cup \bar{Z}_{\text{tr}}^{(t)}$
  $\tilde{\theta}^{(t)} \leftarrow$ train_model$(\tilde{Z}_{\text{tr}}^{(t)})$ {update prediction model}
  $e^{(t)} = E(\tilde{\theta}^{(t)}, Z_{\text{val}})$ {calculate the validation error}
until $|e^{(t)} - e^{(t-1)}| <$ required precision

By solving the r.h.s. of (8) for the tensor $\nabla_{\bar{Z}}\theta$, we obtain,

$$\begin{aligned}\nabla_{\bar{Z}}\theta &= \begin{bmatrix} \frac{\partial\omega}{\partial\bar{x}} & \frac{\partial\omega}{\partial\bar{y}} \\ \frac{\partial b}{\partial\bar{x}} & \frac{\partial b}{\partial\bar{y}} \end{bmatrix} \\ &= -\frac{2}{n}\begin{bmatrix}\Lambda & \Gamma \\ \Omega & -1\end{bmatrix} \\ &\quad \begin{bmatrix} \frac{2}{n}\sum_{i=1}^{n+p}\varphi(x_i)^T\varphi(x_i) + \lambda\mathbb{I}_q & \frac{2}{n}\sum_{i=1}^{n+p}\varphi(x_i) \\ \frac{2}{n}\sum_{i=1}^{n+p}\varphi(x_i) & 1 \end{bmatrix}^{-1}.\end{aligned} \tag{9}$$

The first component of the r.h.s. of (8) is a tensor of size $p \times (d+1) \times (q+1)$. For each $i = 1, \ldots, p$,

$$\Lambda \in \mathbb{R}^{d\times q} = \Big(\omega^T\varphi(x_i) + \big(h(x_i) - y_i\big)\Big)\frac{\partial\varphi(x_i)}{\partial x_i},$$

$$\Omega \in \mathbb{R}^d = \omega^T\frac{\partial\varphi(x_i)}{\partial x_i}, \text{ and}$$

$$\Gamma \in \mathbb{R}^q = -\varphi(x_i).$$

Differentiating E w.r.t. θ gives,

$$\begin{aligned}\nabla_\theta E &= \begin{bmatrix}\nabla_\omega^T E \\ \nabla_b E\end{bmatrix} \\ &= \begin{bmatrix}\frac{2}{m}\sum_{j=1}^m \big(h_\theta(x_j) - y_j\big)\varphi(x_j) \\ \frac{2}{m}\sum_{j=1}^m \big(h_\theta(x_j) - y_j\big).\end{bmatrix}\end{aligned} \tag{10}$$

Finally, by substituting (10) and (9) in (3) we obtain $\nabla_{\bar{Z}_{\text{tr}}} E$. The high level procedure used by the attacker is formalized in Algorithm 1. The function $\mathcal{P}$ projects the perturbed data onto the feasible domain (where the feature values are bounded) and ensures that the constraint in (2) is satisfied.

5 Defense Model

Having discussed the attack model, we now turn to the defense algorithm. First, we briefly introduce the theory of LID for assessing the dimensionality of data subspaces, which is at the heart of our defense, and then present our novel LID-based mechanism to reduce the influence of poisoned data points on the learned model.

The Intrinsic Dimension (ID) of a data set can be considered as the minimum number of latent variables required to represent that data. LID measures the intrinsic dimensionality of data points based on their distribution of distances to the neighboring samples [Houle, 2017]. Recent evidence suggests that LID can be used to capture the degree of being an outlier of data samples [Houle *et al.*, 2018]. In this paper, we characterize the degree of being an outlier of poisoned samples based on their intrinsic dimensionality and reduce their influence on the learning process. Although several estimators of LID exist [Amsaleg *et al.*, 2015], due to having a better trade-off between convergence and bias, we choose the maximum likelihood estimator (MLE), defined below, in all of our experiments.

Definition 1. *(Estimation of LID)*
Given a reference sample $x \sim \mathcal{P}$, where $\mathcal{P}$ represents the data distribution, the maximum likelihood estimator of the LID at x is defined as follows [Amsaleg et al.*, 2015]:*

$$\widehat{LID}(x) = -\left(\frac{1}{k}\sum_{i=1}^{k} log\frac{r_i(x)}{r_{max}(x)}\right)^{-1}. \tag{11}$$

Here, $r_i(x)$ denotes the distance between x and its i-th nearest neighbor within a sample of k points drawn from $\mathcal{P}$, and $r_{max}(x)$ is the maximum of the neighbor distances.

The above estimation assumes that samples are drawn from a tight neighborhood, in line with its development from Extreme Value Theory. In practice, the sample set is drawn uniformly from the available training data (omitting x itself), which itself is presumed to have been randomly drawn from $\mathcal{P}$.

5.1 Sensitivity of LID to Adversarial Perturbations

Amsaleg *et al.* [2020] showed that data points with relatively large LID values are vulnerable to adversarial perturbations. In this work, we focus on how adversarial distortions affect the LID estimator (11). Take $z_c^0 = (x_c^0, y_c^0)$ as the starting point of the sample being poisoned. By adding a perturbation vector $\delta \in \mathbb{R}^{1\times d+1}$ to z_c^0, the poisoned point becomes $z_c = (x_c, y_c)$. Thus, the sensitivity of LID with respect to adversarial perturbations would be $\nabla_\delta \widehat{\text{LID}}(z_c)$.

The sensitivity of LID to adversarial perturbations gives more insights to the attacker than the learner, as the learner only sees the final product after perturbations z_c. The attacker can add perturbations in directions that hardly change the LID value (i.e., $\nabla_\delta \widehat{\text{LID}}(z_c) \approx 0$) or in directions that maximally decrease $\widehat{\text{LID}}(z_c)$. In Section 6.4 we show how an adversary may use this information to create a more sophisticated attack and how it impacts the overall objective of the attacker.

5.2 BIG-LID Algorithm

The learner's goal is to train on the poisoned training data set, yet be unaffected by the poisoned samples. It aims to obtain

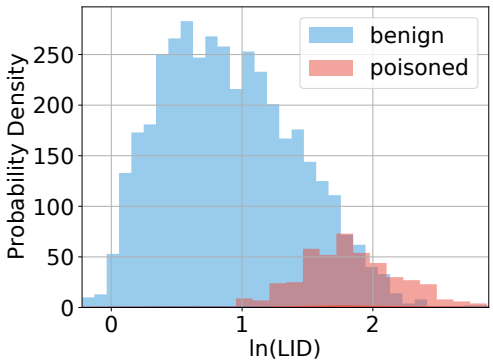


Figure 1: The LID PDFs of benign and poisoned samples when 10% of the data is poisoned. Note the significant imbalance.

a prediction model that is similar to the model it would have obtained in the absence of an attack. The learner achieves this goal in a two-step iterative process: (i) obtain a subset of the training data in which the proportion of poisoned points is similar to the proportion of benign points, (ii) identify the poisoned samples and reduce their influence on the learned model. We describe the two steps in detail below.

Balancing the Benign and Poisoned Data

Since the learner's goal is to have good prediction capabilities on unseen data (i.e., testing phase), we first attempt to identify the training samples that have the greatest influence on prediction accuracy. In certain machine learning algorithms, such as SVMs, only training data points that are on or beyond the separating margin contribute to the decision function. However, a similar property is not present in regression models. Although prior work has used influence of data samples for identifying adversarial samples [Koh and Liang, 2017; Zhang *et al.*, 2018; Diakonikolas *et al.*, 2019], we find that influence of samples alone is not sufficient to distinguish adversarial samples from benign samples. Therefore, it is used in BIG-LID as a filtering step.

Take $Z^L_{val} = (X^L_{val}, y^L_{val})$ as the learner's validation set. Similar to the adversary, the learner uses the chain rule to calculate the gradient of its validation error $E(X^L_{val}, y^L_{val}, \theta)$, w.r.t. the training samples ($\tilde{Z}_{\text{tr}}$). We then calculate the norm of the gradient vector $|\nabla_{z_i} E|$ for each $z_i \in \tilde{Z}_{\text{tr}}$. This norm indicates the sensitivity of the validation error to small perturbations made to training samples. For example, take two training samples z_u and z_v with $|\nabla_{z_u} E| > |\nabla_{z_v} E|$. A small perturbation to z_u would result in a larger change in E compared to a small perturbation made to z_v: z_u is an influential sample. Note that having a large gradient does not necessarily mean that the sample is poisoned, we merely select these for further processing, calling such points *influential samples*. Our experimental evaluations revealed that for each training dataset, the influential samples had a substantial portion of poisoned samples, thereby bringing the proportion of poisoned samples closer to the proportion of benign samples within that subset.

Identifying Poisoned Data

Having found a subset of training samples in which the poisoned samples and benign samples are similar in number, we move to step two, where we attempt to identify the poisoned samples.

First, we discuss the intuition behind using LID to identify adversarial samples during training. In a gradient-based poisoning attack, the attacker selects a set of benign samples as the starting points and iteratively perturbs them in the direction that maximizes its objective function. Due to the added perturbations, the poisoned samples would not be embedded within the benign data distribution. Thus, each poisoned sample would have an irregular distribution of the local distance to its neighbors, which would be reflected by its LID value. Although, as shown in Figure 1, LID is a powerful metric that can capture the degree of being an outlier of poisoned samples compared to benign samples, due to **the proportion of poisoned points being significantly smaller than benign points, we could not use the LID distribution of the entire training set directly.** This is why the data balancing in step one was necessary.

As the maximum likelihood estimator of LID (11) ensures asymptotic normality [Amsaleg *et al.*, 2015], as shown in Figure 2, the LID estimates ($\ln(\text{LID})$) of the influential samples (benign data and poisoned data) resemble a Gaussian distribution. Therefore, assuming the proportion of one distribution is not going to zero, we have a mixture of Gaussians. A Gaussian mixture model (GMM) is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions. The parameters of the Gaussian distributions are found using expectation maximization (EM). Since the distribution of LID values of the influential samples is a mixture of two Gaussian distributions, we use a GMM to identify the two components.

Figure 2 compares the output of the GMM with the two PDFs of the LID values of benign and poisoned samples. It also provides the full PDF of the LID values and the posterior probability of each Gaussian component given the data (responsibilities). It can be seen from the figure that **poisoned samples tend to have a high responsibility for the second Gaussian component compared to the first component**. Thus, at each iteration, the learner selects the data samples for which the probability of the poisoned Gaussian distribution is greater than some threshold δ and adds them to a list of suspected samples.

Reducing the Impact of Poisoned Data

So far, we have found the samples that have the biggest influence on the validation error and selected data points that are suspected to be poisoned based on the GMM output (*suspected list*). We now look at how their influence on the prediction function can be reduced using two commonly used approaches: (i) removing them from the training set [Jagielski *et al.*, 2018; Liu *et al.*, 2017], and (ii) using a weighted cost function for the learner and reducing the weights of poisoned samples.

Figure 3 shows the composition of the *suspected list* as the learner moves through the iterative defense. As seen from the figure, the *suspected list* comprises over 50% of poisoned

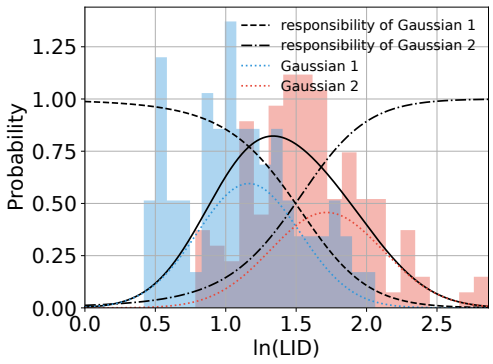


Figure 2: The GMM output on the LID distribution.

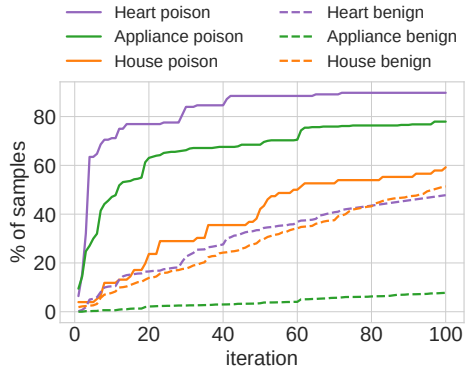


Figure 3: The composition of the suspected list.

data by the eightieth iteration, for all three datasets. Although ideally, we would like the *suspected list* to be entirely populated with poisoned samples, we also observe that a relatively small percentage of benign samples are detected as suspected samples. Reducing the influence of poisoned samples improves the prediction performance of the learner, and unsurprisingly, reducing the influence of benign samples impairs it. Of the two approaches considered, we experimentally found that removing samples (benign and poisoned from the *suspected list*) from the training set results in an overall increase in the MSE. Thus, we opted to instead use a weighting mechanism where we assign a uniform low weight to all the suspected samples at each iteration.

In *weighted least squares (WLS)*, a weight $\beta_i \in [0, 1]$ is assigned for each sample in order to discriminate and vary their influence on the optimization problem. A small β_i value would allow for a large residual value, and the effect of the sample would be de-emphasized. Conversely, a large β_i value would emphasize that sample's effect. Therefore, by carefully selecting a weight vector β, the learner reduces the influence of poisoned samples on the nonlinear prediction model. Algorithm 2 summarizes the defense procedure of the learner.

Algorithm 2 Balanced Information GMM LID (BIG-LID)

Input: poisoned data: $\tilde{Z}_{\text{tr}} = (\tilde{X}_{\text{tr}}, \tilde{y}_{\text{tr}})$, validation set: Z^L_{val}, influential_samples size: s, LID values of $\tilde{Z}_{\text{tr}}$: all_lid_values, GMM threshold: δ, num_of_iterations
Output: regression model: θ^*

$t \leftarrow 0$
best_validation_error $= 1000$
suspected_samples=[ ]
$\tilde{\theta}^{(t)} \leftarrow$ train_model$(\tilde{Z}_{\text{tr}})$ {initial model}
$e^{(t)} = E(\tilde{\theta}^{(t)}, Z^L_{\text{val}})$ {initial validation error}
repeat
  $\nabla_{\tilde{Z}_{\text{tr}}} E \leftarrow$ calculate_derivative$(\tilde{Z}_{\text{tr}}, \tilde{\theta}^{(t)})$
  $\nabla_{\tilde{Z}_{\text{tr}}} E$[suspected_samples, :] $\leftarrow 0$ {in order to avoid re-detecting already suspected samples}
  influential_samples $\leftarrow$ get_top_s$(|\nabla_{\tilde{Z}_{\text{tr}}} E|, s)$ {indices of the largest gradients}
  lid_values $\leftarrow$ all_lid_values(influential_samples) {get LID values of influential_samples}
  gmm $\leftarrow$ train_gmm(lid_values) {train a GMM on the LID values}
  suspected_poison $\leftarrow$ find_poisoned_gmm(gmm, δ)
  suspected_samples $\leftarrow$ append(suspected_poison)
  $t \leftarrow t + 1$ {increment the counter}
  $\beta^{(t)} \leftarrow$ get_weights_for_samples(suspected_samples) {get the weight vector where suspected_samples have lower weights}
  $\tilde{\theta}^{(t)} \leftarrow$ train_model$(\tilde{Z}_{\text{tr}}, \beta^{(t)})$ {train weighted model}
  $e^{(t)} = E(\tilde{\theta}^{(t)}, Z^L_{\text{val}})$ {calculate the validation error}
  if $e^{(t)} <$ best_validation_error **then**
    best_validation_error $\leftarrow e^{(t)}$
    $\theta^* \leftarrow \tilde{\theta}^{(t)}$
  end if
until $t + 1 =$ num_of_iterations

6 Experimental Results and Discussion

In this section, we evaluate the performance of the proposed novel defense against three real-world regression datasets: House, Heart disease and Appliances. Our objective is to investigate the impact of the gradient-based attack against several types of defenses and to evaluate the effectiveness of the BIG-LID defense. For each algorithm, we present the percentage increase in MSE compared to the MSE of a ridge model trained on benign data. Our code is available at https://github.com/sandamal/big-lid.

6.1 Iterative Detection in BIG-LID

Turning back to Figure 3, one interesting finding is that, although the number of poisoned samples discovered increases with each iteration, the rate of discovery peaks during the initial iterations and stagnates afterward. However, the rate at which benign samples are identified remains steady. As previously mentioned, reducing the weights, and thus the influence of benign samples results in a decrease in prediction accuracy, and reducing the weights of poisoned samples results in an increase of the same. Therefore, it appears that it would

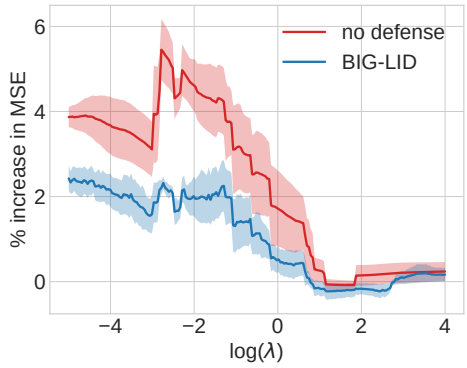


Figure 4: The effect of poisoning attacks when λ varies.

be beneficial for the learner to stop at an earlier iteration to obtain the optimal defense model. However, since the BIG-LID defense reverts to the model with the lowest validation error after all the iterations, this phenomenon does not affect the final outcome of the defense. Further research should be undertaken to reduce the discovery of benign samples as suspected samples in order to further improve the performance of the defense.

6.2 Effect of Regularization

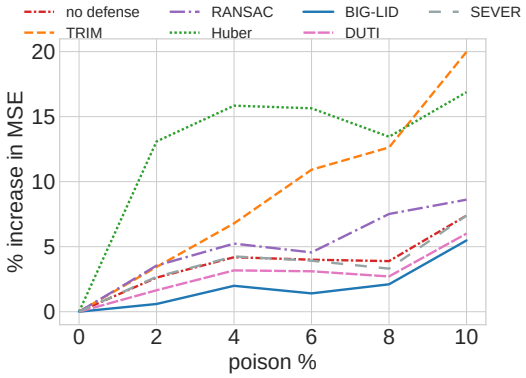
We now look at the influence of regularization on the poisoning attack. We compare the percentage increase in MSE for a ridge regression model with no defense and with BIG-LID when the regularization parameter λ is varied from 10^{-5} to 10^{4}. A large λ value results in a linear model, whereas a small λ value results in a nonlinear model. Thus, Figure 4 shows how the effectiveness of the attack changes when moving from a nonlinear model to a linear model. The results indicate that the effectiveness of the attack diminishes when the prediction model becomes linear. This result is somewhat expected, as there is relatively high variance in nonlinear models compared to linear models, therefore the learned model has a higher probability of being influenced by poisoned data.

This finding supports the connection between robustness and regularization that has been established in prior work [Chen and Paschalidis, 2018]. Furthermore, it suggests that attacking nonlinear estimators is easier compared to attacking linear estimators due to the high variance of the former.

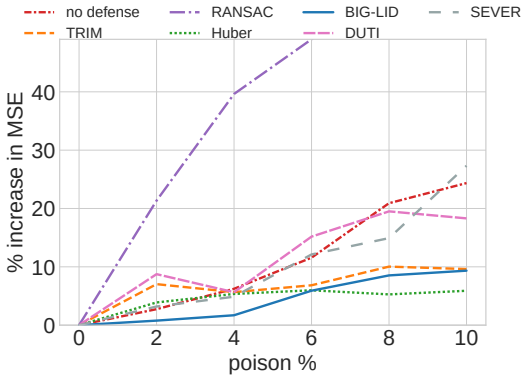
6.3 Comparison of Defenses

Turning now to the performance of an undefended ridge learner against the proposed poisoning attack (Figure 5), we observe that the percentage increase in MSE is 7.4 on Appliances, 28.8 on Heart disease and 24.4 on the House dataset when 10% of the training data is poisoned. This suggests that an undefended nonlinear regression model can be severely affected by the proposed attack.

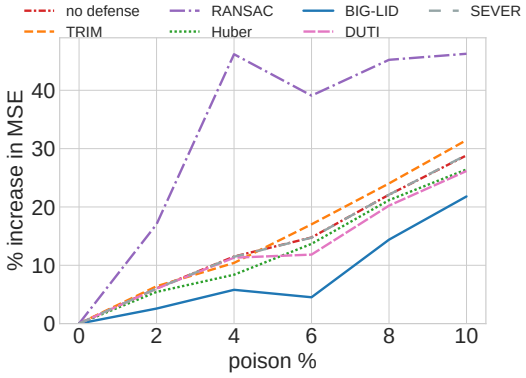
Finally, we compare the prediction performance of BIG-LID defense against TRIM [Jagielski *et al.*, 2018], RANSAC [Fischler and Bolles, 1981], Huber [Huber, 1992], DUTI [Zhang *et al.*, 2018] and SEVER [Diakonikolas *et al.*, 2019] under adversarial conditions. On majority of the test cases, we observed that the BIG-LID defense outperformed the



(a) Appliances dataset.



(b) House dataset.



(c) Heart Disease dataset.

Figure 5: The percentage increase in MSE for the defenses considered on the Appliances, House and Heart Disease datasets when the poison percentage increases from 0% to 10%.

other defenses demonstrating its effectiveness in identifying and reducing the influence of poisoned samples. The percentage increase in MSE remained less than 5% on House and Appliances datasets even when 6% of the training data was poisoned.

The most striking result to emerge from the experiments is that TRIM, RANSAC and Huber performed worse than an undefended ridge model in most test cases. This behavior of Huber and RANSAC may be explained by the fact

that these algorithms were designed to withstand stochastic noise/outliers, not adversarially poisoned data. In a nonlinear regression setting where the data is already complex, these defenses may lack the power to distinguish poisoned data from benign data, especially when the adversary is only making small perturbations. However, TRIM, which follows an iterative optimization process where only the samples with the lowest residuals are used for training, was introduced as a defense against poisoning attacks on linear regression models. We believe that the lack of performance of TRIM is due to the removal of benign samples from the training process which leads to an increase in MSE. DUTI was originally proposed to fix errors in labels using a trusted data set. The DUTI algorithm outputs a list of indices (of the training dataset) ordered according to the likelihood of having incorrect labels. Therefore, the data point indices that are at the head of the list are most likely to be incorrect compared to data point indices that are at the tail end. In our experiments, we choose the top n indices from the tail end of the list to train the prediction model.

6.4 Adaptive Attack Against BIG-LID

To further evaluate the robustness of BIG-LID, we modify the attacker's objective function (2) by introducing a LID cost as follows:

$$\arg\max_{\bar{Z}_{\text{tr}}} E\big(X_{val}, y_{val}, \tilde{\theta}(\bar{Z}_{\text{tr}})\big) - a \sum_{x_i \in \bar{Z}_{\text{tr}}} \widehat{\text{LID}}(x_i) \ . \quad (12)$$

The rationale for the minimization of the LID component in (12) is that BIG-LID identifies samples with large LID values as suspected samples. By introducing the LID cost, the adversary is forced to minimize the LID values of the samples being poisoned to avoid being detected. The scalar a controls the contribution of the two components to the objective function. In our experiments we found that the increase in MSE was less than 1% on all three datasets, when 10% of the data was poisoned with 0.1 severity (in contrast, the vanilla attack achieved a 24% increase on the House dataset, 7% on the Appliances dataset and 29% on the Heart dataset). These results support previous findings by Athalye *et al.* [2018] and Ma *et al.* [2018].

Finding the k nearest neighbors of each sample, which is an integral part of the LID calculation, is a non-differentiable operation. However, the second term of (12) is differentiable at any given time step of the iterative optimization process. The problem LID imposes on the attacker is that gradient ascent does not succeed in optimizing the true LID cost [Athalye *et al.*, 2018]. At each time step of an iterative gradient ascent algorithm, the gradient is calculated with respect to the k nearest neighbors at that particular time, which is not representative of the true direction to perturb (i.e., gradient with respect to the optimal set of k nearest neighbors). Thus, we find that the adaptive attacker fails to achieve its objective.

In summary, the results of this paper indicate that: (i) nonlinear regression algorithms such as ridge regression are vulnerable to poisoning attacks, (ii) unlike linear data, nonlinear data is complex, and it is difficult to distinguish poisoned samples from benign samples especially when the attacker deliberately limits its perturbations, and (iii) unintentionally removing benign data from the training process can have severe consequences (where data redundancy is low), and the effects of removing benign samples may outweigh the effects of removing poisoned samples resulting in a net increase in MSE. Taken together, these findings suggest that a defense such as BIG-LID, which brings together several components previously explored in adversarial learning (i.e, influence, LID and weighting), is useful against sophisticated adversaries as it can withstand sophisticated attacks and makes adaptive attacks impractical.

7 Conclusions

This paper addresses the problem of defending nonlinear regression models against poisoning attacks. We observed that gradient-based poisoning attacks significantly degrade the prediction performance of nonlinear regression models. We introduced a gradient-based approach to filter the most influential training data points and constructed a LID based black-box defense (BIG-LID) that reduces the influence of poisoned samples on the learned model. Exploring the possibility of extending this defense to reinforcement learning settings would be a fruitful area for further work.

Acknowledgements

This research was supported in part by the Defence Science and Technology Group's Next Generation Technologies Program and the Australian Research Council Linkage Project under the grant LP190101287.

References

[Alfeld *et al.*, 2019] Scott Alfeld, Ara Vartanian, Lucas Newman-Johnson, and Benjamin I. P. Rubinstein. Attacking data transforming learners at training time. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 3167–3174, 2019.

[Amsaleg *et al.*, 2015] Laurent Amsaleg, Oussama Chelly, Teddy Furon, Stephane Girard, Michael E. Houle, Ken-Ichi Kawarabayashi, and Michael Nett. Estimating local intrinsic dimensionality. In *21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 29–38. ACM, 2015.

[Amsaleg *et al.*, 2020] Laurent Amsaleg, James Bailey, Amelie Barbe, Sarah M. Erfani, Teddy Furon, Michael E Houle, Miloš Radovanović, and Xuan Vinh Nguyen. High intrinsic dimensionality facilitates adversarial attack: Theoretical evidence. *IEEE Transactions on Information Forensics and Security*, 16:854–865, 2020.

[Athalye *et al.*, 2018] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning*, pages 274–283, 2018.

[Balakrishnan *et al.*, 2017] Sivaraman Balakrishnan, Simon S. Du, Jerry Li, and Aarti Singh. Computationally efficient robust sparse estimation in high dimensions. In *Conference on Learning Theory*, pages 169–212, 2017.

[Bhatia *et al.*, 2015] Kush Bhatia, Prateek Jain, and Purushottam Kar. Robust regression via hard thresholding. *Advances in neural information processing systems*, 28:721–729, 2015.

[Biggio and Roli, 2018] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[Carlini and Wagner, 2017] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14, 2017.

[Chen and Paschalidis, 2018] Ruidi Chen and Ioannis Ch Paschalidis. A robust learning approach for regression models based on distributionally robust optimization. *The Journal of Machine Learning Research*, 19(1):517–564, 2018.

[Diakonikolas *et al.*, 2019] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. In *International Conference on Machine Learning*, pages 1596–1606, 2019.

[Fischler and Bolles, 1981] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[Houle *et al.*, 2018] Michael E. Houle, Erich Schubert, and Arthur Zimek. On the correlation between local intrinsic dimensionality and outlierness. In *International Conference on Similarity Search and Applications*, pages 177–191. Springer, 2018.

[Houle, 2017] Michael E. Houle. Local intrinsic dimensionality I: an extreme-value-theoretic foundation for similarity applications. In Christian Beecks, Felix Borutta, Peer Kröger, and Thomas Seidl, editors, *Similarity Search and Applications*, pages 64–79, 2017.

[Huber, 1992] Peter J. Huber. Robust estimation of a location parameter. In *Breakthroughs in Statistics*, pages 492–518. Springer, 1992.

[Jagielski *et al.*, 2018] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 19–35. IEEE, 2018.

[Koh and Liang, 2017] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *34th International Conference on Machine Learning*, volume 70, pages 1885–1894, 2017.

[Kurakin *et al.*, 2018] Alex Kurakin, Dan Boneh, Florian Tramèr, Ian Goodfellow, Nicolas Papernot, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *ICLR*, 2018.

[Liu *et al.*, 2017] Chang Liu, Bo Li, Yevgeniy Vorobeychik, and Alina Oprea. Robust linear regression against training data poisoning. In *10th ACM Workshop on Artificial Intelligence and Security*, pages 91–102. ACM, 2017.

[Ma *et al.*, 2018] Xingjun Ma, Bo Li, Yisen Wang, Sarah M. Erfani, Sudanthi N. R. Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E. Houle, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.

[Tong *et al.*, 2018] Liang Tong, Sixie Yu, Scott Alfeld, and Yevgeniy Vorobeychik. Adversarial regression with multiple learners. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4946–4954, 2018.

[Vorobeychik and Kantarcioglu, 2018] Yevgeniy Vorobeychik and Murat Kantarcioglu. Adversarial machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–169, 2018.

[Xiao *et al.*, 2015] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. Is feature selection secure against training data poisoning? In *International Conference on Machine Learning*, pages 1689–1698, 2015.

[Xu *et al.*, 2012] Huan Xu, Constantine Caramanis, and Shie Mannor. Outlier-robust PCA: The high-dimensional case. *IEEE Transactions on Information Theory*, 59(1):546–572, 2012.

[Zhang *et al.*, 2018] Xuezhou Zhang, Xiaojin Zhu, and Stephen Wright. Training set debugging using trusted items. In *AAAI Conference on Artificial Intelligence*, 2018.