On Robustness of Linear Classifiers to Targeted Data Poisoning

Nakshatra Gupta[1], **Sumanth Prabhu S**[1*], **Supratik Chakraborty**[2], **Venkatesh R**[1]

[1]Tata Consultancy Services, Pune, India
[2]IIT Bombay, Mumbai, India
nakshatra.g@tcs.com, sumanthsprabhu@gmail.com, supratik@cse.iitb.ac.in, r.venky@tcs.com

Abstract

Data poisoning is a training-time attack that undermines the trustworthiness of learned models. In a targeted data poisoning attack, an adversary manipulates the training dataset to alter the classification of a targeted test point. Given the typically large size of training dataset, manual detection of poisoning is difficult. An alternative is to automatically measure a dataset's robustness against such an attack, which is the focus of this paper. We consider a threat model wherein an adversary can only perturb the labels of the training dataset, with knowledge limited to the hypothesis space of the victim's model. In this setting, we prove that finding the robustness is an NP-Complete problem, even when hypotheses are linear classifiers. To overcome this, we present a technique that finds lower and upper bounds of robustness. Our implementation of the technique computes these bounds efficiently in practice for many publicly available datasets. We experimentally demonstrate the effectiveness of our approach. Specifically, a poisoning exceeding the identified robustness bounds significantly impacts test point classification. We are also able to compute these bounds in many more cases where state-of-the-art techniques fail.

Extended Version —
https://doi.org/10.5281/zenodo.17627646

Introduction

Data poisoning is a training-time attack wherein an adversary perturbs the training dataset to alter the predictions of the trained models (Biggio, Nelson, and Laskov 2012). Here, the perturbation is called poisoning. Manually determining whether a training dataset is poisoned or not is challenging due its size. Consequently, this attack affects the trustworthiness of machine learning models and hinders their deployment in industries (Kumar et al. 2020).

Targeted data poisoning refers to an adversary manipulating the training data to alter the prediction of a single (or a small fixed subset of) test data. While targeted data poisoning can be sophisticated, such as backdoor attacks that use patterns called triggers, we focus on trigger-less attack. In such an attack, an adversary targets a specific test point and aims for the victim's classifier to classify this point as a desired class. An example is a loan applicant adversary who poisons the training data so that the model trained on the poisoned data approves the adversary's loan.

We consider a practical adversary who can poison the training dataset only by contaminating labels. While contaminating the features of a dataset can be difficult, labels can be easily manipulated as label errors are common, especially when they are collected from external sources (Adebayo et al. 2023). Furthermore, we assume that the adversary has knowledge of only the hypothesis space of the victim's classifier, but not the training procedure or the actual classifier. In other words, the victim's training procedure is treated as a black-box. In this setting, we define robustness as:

> *The minimum number of label perturbations in the training dataset required for a classifier from the hypothesis space to classify the test point as desired.*

For example, consider the dataset Pen-based Handwritten Digit Recognition (Alpaydin and Alimoglu 1996). In the dataset, a test point shown in Figure 1 represents the handwritten digit '0'. However, our tool finds that it can be labeled as the digit '4' by changing the labels of just two points in the training dataset, out of thousands.

The above notion of robustness is not only useful to measure confidence in the training data, but also in contesting a model's predictions. When a model produces an undesirable decision, the user may want to identify the minimal set of training point labels that potentially influenced the outcome. If any of these labels are incorrect, the user can flag and contest the decision. This application is directly addressed by a solution to the robustness problem defined above.

In our setting of targeted black-box poisoning by label perturbation, we focus on linear classifiers in this work. While existing data poisoning work typically considers linear classifiers (Biggio, Nelson, and Laskov 2011, 2012; Xiao, Xiao, and Eckert 2012; Xiao et al. 2015; Suya et al. 2024; Zhao et al. 2017; Paudice, Muñoz-González, and Lupu 2019; Yang, Xu, and Yu 2023), our setting considers a weaker threat model. Moreover, linear models are still relevant as they perform well on many tasks (Ferrari Dacrema, Cremonesi, and Jannach 2019; Tramer and Boneh 2020; Chen et al. 2021; Chen, Ding, and Wagner 2023) and are subject of several recent related works (Yang, Xu, and Yu 2023; Yang, Jain, and Wallace 2023; Suya et al. 2024).

[*]Now at Relyance AI, Bangalore, India

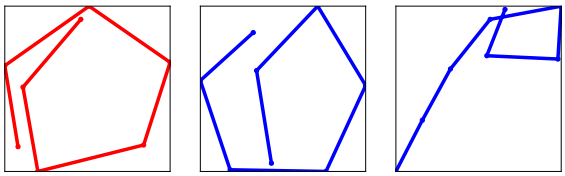

Figure 1: A test point (leftmost plot) and two training points all originally labeled as digit '0'. Our tools finds that modifying only these two training points as digit '4' can change the test point's label to digit '4'.

We make five contributions in this paper. We first establish a theoretical result showing that the problem of finding robustness is NP-Complete. To address this challenge, we propose techniques that approximate robustness. In this direction, our second contribution is a partition-based method that computes a lower bound for robustness. Third, we introduce an augmented learning procedure designed to find an upper bound for robustness. Our fourth contribution is a prototype tool ROBUSTRANGE, which implements our novel techniques. Finally, we evaluate ROBUSTRANGE on 15 publicly available datasets of different sizes. Our experiments reveal several interesting facts: the average robustness can be as low as 3%, ROBUSTRANGE performs better than a SOTA tool (Yang, Xu, and Yu 2023), even when the SOTA tool knows the victim's classifier and learning procedure.

Related Work

Data poisoning attacks have received considerable attention as a primary security threat during the training stage of machine learning (Barreno et al. 2010; Tian et al. 2022). Depending on the adversary's goals, these attacks can be either indiscriminate or targeted. In an indiscriminate attack, the adversary aims to maximize the overall loss of victim's classifier, which was the focus of many works (Biggio, Nelson, and Laskov 2011, 2012; Xiao, Xiao, and Eckert 2012; Xiao et al. 2015; Suya et al. 2024). In contrast, a targeted attack aims to alter the prediction of a specific test point while minimally affecting overall loss, which is the focus of this paper.

A targeted attack can be achieved by assuming different capabilities of the adversary. For instance, an attacker can add/modify/remove training points without changing labels (Shafahi et al. 2018; Suciu et al. 2018; Zhu et al. 2019; Yang, Jain, and Wallace 2023), posses full or partial knowledge of the victim's learning process (Cinà et al. 2021; Şuvak et al. 2022; Paudice, Muñoz-González, and Lupu 2019), or trigger poisoning when the test point is embedded by certain patterns (Chen et al. 2017; Saha, Subramanya, and Pirsiavash 2020; Gu et al. 2019). We assume a more constrained attacker who can only perturb labels, lacks knowledge of the victim's learning process (black-box), and targets a specific test point.

There are techniques that construct a poisoned dataset with stronger threat model. In particular, (Zhao et al. 2017) also considers label perturbations and a black-box victim model, but requires a bound on robustness to compute the poisoned dataset, along with an objective model. The work in (Paudice, Muñoz-González, and Lupu 2019) also computes a perturbation set, but it similarly requires a bound and the victim's loss function. The work closest to ours is (Yang, Xu, and Yu 2023), which aims to find the minimal label-perturbed set. However, it only computes an upper bound, and, more importantly, requires knowledge of the victim's model and loss function to determine the *influence function* (Koh and Liang 2017).

(Gao, Karbasi, and Mahmoody 2021) gives a relationship between the dataset size and maximum allowed perturbations, whose accuracy has been subsequently improved in (Wang, Levine, and Feizi 2022a). The work in (Hanneke et al. 2022) characterizes the optimal error rate when the dataset is poisoned. In these works, data points are either removed or inserted. Furthermore, these works characterizes bounds on targeted data poison, while we compute upper and lower bounds. An orthogonal line of work is that of certified robustness (Rosenfeld et al. 2020; Levine and Feizi 2020; Wang, Levine, and Feizi 2022b; Jia et al. 2022), which gives a model that predicts with guaranteed robustness.

Problem Setting

Preliminaries

Consider the task of binary classification from d dimensional input features $\mathcal{X} \subseteq \mathbb{R}^d$ to binary output labels $\mathcal{Y} = \{+1, -1\}$. The goal of the classification task is to find a function (called classifier) $f : \mathcal{X} \to \mathcal{Y}$ that has a small generalization error. In machine learning, f is derived from a given set of hypotheses (called hypothesis space) $\mathcal{H}$ and a set of labeled training data $\mathcal{D}$. We assume $\mathcal{H}$ is a set of linear functions $f_{\mathbf{w},b}$ of the form $f_{\mathbf{w},b}(x) = \text{sign}(\mathbf{w}^T x + b)$, where $\mathbf{w} \in \mathbb{R}^d$ is the weight parameter, and $b \in \mathbb{R}$ is the bias parameter. The labeled training data $\mathcal{D}$ is generated from an i.i.d over $\mathcal{X} \times \mathcal{Y}$. Given $\mathcal{H}$ and $\mathcal{D}$, the classification task is to find a model $f_{\mathbf{w},b} \in \mathcal{H}$ that minimizes a loss function (e.g., hinge loss) over $\mathcal{D}$.

Threat Model

We model the overall workflow via *causative targeted attack* (Barreno et al. 2010), which is a game between a *victim* and an *attacker*, that proceeds as follows:

1. The victim generates a clean training data $\mathcal{D}_c = \{(x_i, y_i) \mid i \in [m]\}$ from an i.i.d over $\mathcal{X} \times \mathcal{Y}$.
2. The attacker *poisons* $\mathcal{D}_c$ by perturbing (a subset of) labels to get $\mathcal{D}_p = \{(x_i, y_i') \mid i \in [m],\ y_i' = y_i \text{ or } y_i' = -y_i\}$.
3. The victim finds a model $f_{\mathbf{w},b} \in \mathcal{H}$ that minimizes a loss function over $\mathcal{D}_p$.
4. The attacker has a target test point with label $(x_t, y_t) \in \mathcal{X} \times \mathcal{Y}$, where (x_t, y) is not in $\mathcal{D}_c$ for any $y \in \mathcal{Y}$. The attacker wins the game if the victim learns a model $f_{\mathbf{w},b}$ from $\mathcal{D}_p$ such that $f_{\mathbf{w},b}(x_t) = y_t$. Otherwise, victim wins the game.

We assume an attacker who can only perturb labels in $\mathcal{D}_c$, and is aware of the victim's hypothesis space, but not the training procedure, the loss function, or the model's weights.

Robustness

The robustness r of a dataset $\mathcal{D}_c = \{(x_i, y_i) \mid i \in [m]\}$ with respect to a target test point (x_t, y_t) is the minimum number of label perturbations required for the attacker to win the above game. Sans knowledge of the victim's loss function, we define robustness as follows. Let $\mathfrak{P}$ be the set of all (possibly) poisoned datasets $\mathcal{D}_p = \{(x_i, y_i') \mid i \in [m],\ y_i' = y_i \text{ or } y_i' = -y_i\}$ s.t. there exists $f_{\mathbf{w},b} \in \mathcal{H}$ with:

1. $f_{\mathbf{w},b}(x_i) = y_i', \quad \forall i \in [m]$
2. $f_{\mathbf{w},b}(x_t) = y_t$

Then $r = \min_{\mathcal{D}_p \in \mathfrak{P}} \sum_{i=1}^{m} \mathbb{I}(y_i' \neq y_i)$, where $\mathbb{I}$ is the indicator function.

Note that some authors prefer to define robustness as the *maximum* number of label perturbations in $\mathcal{D}_c$ such that no classifier in $\mathcal{H}$ learned from the poisoned data classifies the test point x_t as y_t. Clearly, this is $r-1$, where r is as defined above. For computational simplicity, we use the definition provided above.

Computing Robustness

A naive way to compute robustness is by perturbing labels of subsets of the training dataset $\mathcal{D}_c$ in increasing order of subset size, and finding whether there exists a classifier that satisfies all required conditions for the poisoned dataset. The time complexity of such an algorithm is clearly exponential in $|\mathcal{D}_c|$. Hence, a natural question is whether this algorithm can be improved. Theorem 1 shows that this is unlikely.

Theorem 1. *Given a dataset $\mathcal{D}_c = \{(x_i, y_i) \mid i \in [m]\}$ and a target test point (x_t, y_t), deciding whether its robustness r is less than a threshold κ is NP-Complete, when $\mathcal{H}$ is a set of linear binary classifiers.*

Proof Sketch. **In NP**: Given a witness classifier $f_{\mathbf{w},b}$, we can check whether (a) $f_{\mathbf{w},b}(x_t) = y_t$, and (b) $\sum_{i=1}^{m} \mathbb{I}(f_{\mathbf{w},b}(x_i) \neq y_i) < \kappa$ in polynomial time. Alternatively, given a witness poisoned dataset $\mathcal{D}_p = \{(x_i, y_i') \mid i \in [m]\}$, we can check in polynomial time if (a) $\sum_{i=1}^{m} \mathbb{I}(y_i' \neq y_i) < \kappa$, and (b) there exists a linear classifier $f_{\mathbf{w},b}$ that satisfies $f_{\mathbf{w},b}(x_t) = y_t$ and $f_{\mathbf{w},b}(x_i) = y_i'$ for all $i \in [m]$. The latter can be done by standard techniques for solving linear systems of equations, viz. Gaussian elimination.

NP-hardness: We show this by reduction from the vertex cover problem. Let $G = (V, E)$ be an undirected graph, where V is the set of vertices, and $E \subseteq V \times V$ is the set of edges. A vertex cover of G is a subset C of V such that for every edge $(u, v) \in V$, at least one of u, v is in C. Given G and a threshold κ, the vertex cover problem asks whether G has a vertex cover C s.t. $|C| < \kappa$. It is well known (Karp 2009) that the vertex cover problem is NP-complete. We give below a polynomial-time reduction from the vertex cover problem to the problem of deciding if the robustness of a dataset w.r.t. a target test point is less than a threshold. This proves NP-hardness of deciding if robustness is less than a threshold.

Let $V = \{v_1, \ldots v_n\}$ be the set of vertices in G. We create a training dataset $\mathcal{D}_c \subseteq \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} = \{0,1\}^{n+1}$ and $\mathcal{Y} = \{+1, -1\}$. Thus, every $x \in \mathcal{X}$ can be thought of as a 0-1 vector of $n+1$ dimensions. Below, we use $x[j]$ to denote the j^{th} component of vector x, for $j \in \{1, \ldots n+1\}$.

For every $v_i \in V$, we add a training datapoint (x_i, y_i) to $\mathcal{D}_c$, where (a) $x_i[j] = 1$ iff $i = j$, and (b) $y_i = -1$. We say that these datapoints encode vertices in V. Similarly, for every edge $(v_i, v_j) \in E$, we add a training datapoint $(x_{i,j}, y_{i,j})$, where (a) $x_{i,j}[k] = 1$ iff either $i = k$ or $j = k$ or $k = n+1$, and (b) $y_{i,j} = +1$. We say that these datapoints encode edges in E. As an example, if $V = \{v_1, v_2, v_3\}$ and $E = \{(v_1, v_2), (v_2, v_3), (v_3, v_1)\}$, then $\mathcal{D}_c = \mathcal{D}_c^V \cup \mathcal{D}_c^E$, where $\mathcal{D}_c^V = \{\big((1,0,0,0), -1\big), \big((0,1,0,0), -1\big), \big((0,0,1,0), -1\big)\}$ encodes the three vertices, and $\mathcal{D}_c^E = \{\big((1,1,0,1), +1\big), \big((0,1,1,1), +1\big), \big((1,0,1,1), +1\big)\}$ encodes the three edges. Notice that for $\mathcal{D}_c$ defined above, there exists a linear classifier $f_{\mathbf{w},b}$ such that $f_{\mathbf{w},b}(x_i) = y_i$ for all $(x_i, y_i) \in \mathcal{D}_c$. Indeed, if $w[j] = -1$ for $j \in \{1, \ldots n\}$, $w[n+1] = 3$, and if $b = 0$, then $\text{sign}(\mathbf{w}^T x + b) = y$ for all $(x, y) \in \mathcal{D}_c$.

Finally, we construct the test datapoint (x_t, y_t), where $x_t[j] = 0$ for all $j \in \{1, \ldots n\}$, $x_t[n+1] = 1$ and $y_t = -1$. For the abve example, $(x_t, y_t) = \big((0,0,0,1), -1\big)$. Notice that the linear classifier discussed above that correctly classifies all datapoints in $\mathcal{D}_c$ no longer gives $f_{\mathbf{w},b}(x_t) = y_t$.

With $\mathcal{D}_c$ and (x_t, y_t) defined as above, we claim that the graph G has a vertex cover of size less than κ iff the robustness of $\mathcal{D}_c$ w.r.t. (x_t, y_t) is less than κ. The only if part is easy to prove. Suppose C is a vertex cover of G, and $|C| < \kappa$. We construct the poisoned dataset $\mathcal{D}_p$ by changing the label (to $+1$) of only those datapoints in $\mathcal{D}_c$ that encode vertices in C. A linear classifier $f_{\mathbf{w},b}(x) = \text{sign}(\mathbf{w}^T x + b)$ for $\mathcal{D}_p$ that also satisfies $f_{\mathbf{w},b}(x_t) = y_t$ can now be obtained as follows: $w[i] = 3$ for all $v_i \in C$, $w[i] = -1$ for all $v_i \notin C$, $w[n+1] = -1$ and $b = 0$. In our running example with 3 vertices, considering the vertex cover $C = \{v_1, v_2\}$, we get $\mathcal{D}_p = \{\big((1,0,0,0), +1\big), \big((0,1,0,0), +1\big), \quad \big((0,0,1,0), -1\big), \quad \big((1,1,0,1), +1\big), \big((0,1,1,1), +1\big), \big((1,0,1,1), +1\big)\}$.

To prove the if part, let S be the subset of datapoints in $\mathcal{D}_c$ whose labels are flipped to obtain $\mathcal{D}_p$, and suppose $|S| < \kappa$. Let $f_{\mathbf{w},b}$ be a linear classifier corresponding to $\mathcal{D}_p$ that satisfies $f_{\mathbf{w},b}(x_t) = y_t$. From the definition of (x_t, y_t), it follows that $w[n+1] + b < 0$. Note that for every non-poisoned datapoint $(x_{i,j}, +1)$ corresponding to an edge $(v_i, v_j) \in E$, we must also have $f_{\mathbf{w},b}(x_{i,j}) = 1$, or $w[i] + w[j] + w[n+1] + b \geq 0$. This requires at least one of the datapoints corresponding to v_i or v_j to be poisoned, as otherwise, we would have $w[i] < 0$ and $w[j] < 0$, which is inconsistent with $w[i] + w[j] + w[n+1] + b \geq 0$. In general, the set S may contain datapoints encoding both vertices and edges in G. We describe below a process for successively transforming S, such that it eventually contains only datapoints encoding vertices in V. Specifically, for every datapoint $(x_{i,j}, +1)$ in S that encodes an edge (v_i, v_j) in graph G, we check if the datapoint encoding v_i (or v_j) is also in S. If so, we simply remove $(x_{i,j}, +1)$ from S; otherwise, we add the datapoint corresponding to v_i (resp. v_j) to S and remove all datapoints corresponding to edges incident on v_i (resp. v_j) from S. By repeating this process, we obtain a

poisoned dataset $\mathcal{D}'_p$ in which every poisoned datapoint corresponds to a vertex in V. Let the corresponding set of datapoints whose labels have been flipped in $\mathcal{D}'_p$ be called S'. Following the same reasoning as in the proof of the "only if" part above, the linear classifier $f_{\mathbf{w},b}$ corresponding to $\mathcal{D}_p$ can be modified to yield a linear classifier $f_{\mathbf{w},b}{}'$ for $\mathcal{D}'_p$ that also satisfies $f_{\mathbf{w},b}{}'(x_t) = y_t$. Since the datapoint corresponding to every edge $(v_i, v_j) \in E$ has its label unchanged in $\mathcal{D}'_p$, the datapoint corresponding to either v_i or v_j must have been poisoned in $\mathcal{D}'_p$. Hence, the set of poisoned datapoints must correspond to a subset of vertices that forms a vertex cover of G. Since we added at most one datapoint corresponding to a vertex in S when removing a datapoint corresponding to an edge from S, we have $|S'| \leq |S| < \kappa$. □

Since perturbing subsets results in an exponential algorithm, it does not scale even for a small dataset. To overcome this, we propose to compute an approximation of robustness r. Notice that the victim is interested in how low the robustness of their dataset is, to ensure that it is not susceptible to an attack, while the attacker's interest lies in how high the robustness is, so that the attack is likely to succeed without detection. Considering these perspectives, we compute both a lower bound $\check{r}$ and an upper bound $\hat{r}$ of robustness r, such that it is guaranteed that $\check{r} \leq r \leq \hat{r}$.

Lower Bound Robustness $\check{r}$ via Partitioning

We observe that the problem of finding robustness for linear classifiers can be formulated as an optimization problem. In particular, it can be encoded as a mixed-integer linear program (MILP) using the big M method (Bazaraa, Jarvis, and Sherali 2011), assuming a known bound on the range of the linear classifier function. A solution to this optimization problem determines the weights of the function that classifies the test point as required, while minimizing misclassifications on the training dataset. More formally, for a dataset $\mathcal{D}_c$ and a test point (x_t, y_t), we encode the problem as the following optimization problem:

Variables:

- $w_1, \ldots, w_d, b \in \mathbb{R}$ as weights and bias, and
- $\delta_1, \ldots, \delta_m \in \{0, 1\}$ as indicators for label perturbations.

Objective: Minimize $\sum_{i=1}^{m} \delta_i$

Constraints:

$$y_t \left(\mathbf{w} \cdot x_t + b\right) \geq \epsilon$$
$$y_i \left(\mathbf{w} \cdot x_i + b\right) + M\delta_i \geq \epsilon, \quad \forall i \in [m]$$
$$y_i \left(\mathbf{w} \cdot x_i + b\right) - M(1 - \delta_i) \leq -\epsilon, \quad \forall i \in [m]$$

Parameters: $M \gg 0$ is a large integer constant, $\epsilon \approx 0$, and $(x_i, y_i) \in \mathcal{D}_c$.

A solution to the problem provides weights and bias $(w_1, \ldots, w_d, b)$ that classifies the test point x_t as y_t and minimizes the sum of perturbed labels in $\mathcal{D}_c$, denoted by $\sum_{i=1}^{m} \delta_i$. The first constraint ensures that $y_t = \text{sign}(\mathbf{w} \cdot x_i + b)$. For each point $(x_i, y_i) \in \mathcal{D}_c$, the next two constraints ensure the following:

$$\delta_i = \begin{cases} 0 & y_i = \text{sign}(\mathbf{w} \cdot x_i + b) \\ 1 & y_i \neq \text{sign}(\mathbf{w} \cdot x_i + b) \end{cases}$$

For instance, $\delta_i = 0$ makes the second constraint $y_i \left(\mathbf{w} \cdot x_i + b\right) \geq \epsilon$, which in turn makes $y_i = \text{sign}(\mathbf{w} \cdot x_i + b)$. Similarly, $\delta_i = 1$ makes the third constraint $y_i \left(\mathbf{w} \cdot x_i + b\right) \leq \epsilon$, which makes $y_i \neq \text{sign}(\mathbf{w} \cdot x_i + b)$. These constraints additionally adds the conditions $y_i \left(\mathbf{w} \cdot x_i + b\right) < M$ and $y_i \left(\mathbf{w} \cdot x_i + b\right) > -M$, resp..

Hence, a solution gives the robustness r (where $r = \sum_{i=1}^{m} \delta_i$), along with the labels that were perturbed (y_i for which δ_i is 1). A solution exists only when $\mathbf{w} \cdot x_i + b$ is in the interval $(-M, M)$.

Since the complexity of solving such optimization problems is exponential, we can get robustness in this way for only small datasets. To scale this, we propose an optimization approach that finds a lower bound of robustness. In this approach, the dataset is partitioned into smaller size, then robustness is computed individually for each partition, and finally the results are summed. Although this sum provides a lower bound, it significantly improves the scalability as the optimization problem is on small sized sets.

Consider the partition of $\mathcal{D}_c$ into $k \in \mathbb{N}$ disjoint subsets $\mathcal{D}_c^1 \ldots \mathcal{D}_c^k$, where each subset $\mathcal{D}_c^j$ contains m/k points, except for the last subset, which contains the remaining points.

For each partition $\mathcal{D}_c^j$ we compute robustness r^j by encoding it as the optimization problem(as before). Finally, we compute the sum of these robustness $\check{r} = \sum_{j=1}^{k} r^j$.

Theorem 2. *For dataset $\mathcal{D}_c$ and a test point (x_t, y_t), $\check{r} \leq r$.*[1]

Upper Bound Robustness $\hat{r}$ via Augmentation

In addition to the lower bound, we also compute an upper bound of robustness. For this purpose, we train a linear classifier on an augmented version of the training dataset $\mathcal{D}_c$. The augmentation biases the learning to train a classifier that classifies the test point x_t as y_t. While theoretically this classifier may not minimize the label perturbations of $\mathcal{D}_c$, in practice we observe that it will give a tighter upper bound. Moreover, this procedure will be quick, as it requires only learning a classifier, hence scales for large datasets.

In order to find such a classifier, we restrict our objective to learn a classifier $f_{\mathbf{w},b} \in \mathcal{H}$ such that $f_{\mathbf{w},b}(x_t) = y_t$. Since not all classifiers can achieve this objective, we first introduce a targeted augmentation scheme, wherein $k' \in \mathbb{N}$ identical copies of the test point (x_t, y_t) are added to $\mathcal{D}_c$. Let the augmented set be $\mathcal{D}'_c = \mathcal{D}_c \bigcup \{(x_t, y_t)\}^{k'}$, where $\{(x_t, y_t)\}^{k'}$ is a multi-set of k' copies of (x_t, y_t).

We then train a classifier on $\mathcal{D}'_c$ by minimizing the empirical loss with respect to a loss function l:

$$f_{\mathbf{w},b} = \arg \min_{f_{\mathbf{w},b} \in \mathcal{H}} \frac{1}{m + k'} \left(\sum_{i=1}^{m} l(f_{\mathbf{w},b}(x_i), y_i) + \sum_{j=1}^{k'} l(f_{\mathbf{w},b}(x_t), y_t) \right)$$

Once a classifier $f_{\mathbf{w},b}$ is learned on the augmented dataset, it is checked whether the classifier correctly classifies the

[1] Remaining proofs are in the extended version.

Algorithm 1 ROBUSTNESSINTERVAL($\mathcal{D}_c$, (x_t, y_t))

Input: $\mathcal{D}_c$ – training dataset, (x_t, y_t) – test point
Parameter: M, k, k', l, ϵ – hyperparameters
Output: lower bound $\check{r}$ and upper bound $\hat{r}$ robustness

$\mathcal{D}_c^1, \ldots, \mathcal{D}_c^k \leftarrow$ RANDOMPARTITION$(\mathcal{D}_c, k)$
for $j = 1$ **to** k **do**
  $r^j \leftarrow$ MILPSOLVE$(\mathcal{D}_c^j, (x_t, y_t), M)$
end for
$\check{r} \leftarrow \sum_{j=1}^{k} r^j$
$\mathcal{D}_c' \leftarrow$ AUGMENT$(\mathcal{D}_c, (x_t, y_t), k')$
$f_{\mathbf{w},b} \leftarrow$ LEARNCLASSIFIER$(\mathcal{D}_c', l)$
$\hat{r} \leftarrow \sum_{i=1}^{m} \mathbb{I}(f_{\mathbf{w},b}(x_i) \neq y_i)$
return $(\check{r}, \hat{r})$

test point x_t as y_t. If it does, then the number of misclassified points from the original dataset $\mathcal{D}_c$ will be the the upper bound. In other words, $\hat{r} = \sum_{i=1}^{m} \mathbb{I}(f_{\mathbf{w},b}(x_i) \neq y_i)$.

Theorem 3. *For dataset $\mathcal{D}_c$ and a test point (x_t, y_t), $r \leq \hat{r}$.*

Algorithm

Algorithm 1 gives a formal description of our technique discussed in previous sections. It takes as input a training dataset $\mathcal{D}_c$ and a test point x_t with target label y_t. It also assumes hyperparameter M, k, k', l are set. It starts with computing random k partitions of $\mathcal{D}_c$ into $\mathcal{D}_c^1, \ldots, \mathcal{D}_c^k$. Then, for each partition $\mathcal{D}_c^j$ it computes its robustness r^j by encoding the problem as a MILP using M and then using a solver to find the optimal solution. These robustness are then summed up to get $\check{r}$. In order to computer $\hat{r}$, the algorithm starts with augmenting $\mathcal{D}_c$ with k' copies of the test point and target label. This augmented data $\mathcal{D}_c$ is then passed to a learning algorithm along with a loss function (l). When the learning algorithm returns a classifier $f_{\mathbf{w},b}$, $\hat{r}$ is computed by calculating the number of misclassifications $f_{\mathbf{w},b}$ makes with respect to $\mathcal{D}_c$. Finally, the pair $(\check{r}, \hat{r})$ is returned to the user.

Implementation Details

We implemented Algorithm 1 as a Python tool ROBUSTRANGE. It uses SCIP (Bolusani et al. 2024) as the MILP solver (within Google OR-Tools v9.12) to compute lower bound robustness, and `SGDClassifier` from scikit-learn (Pedregosa et al. 2011) (v1.3.2) to learn a classifier from the augmented dataset. As an optimization, 10 classifiers were learned from the augmented dataset and the minimum upper bound robustness $\hat{r}$ among them was chosen. The hyperparameters used had the following values: M and ϵ in the MILP encoding are set to 1000 and 10^{-10}, resp., the number of partitions k had multiple values such as 20, 1000, 250, and 100 depending on the dataset, the number of augmented test point k' was set to $m + 1$, and the loss function l used were hinge loss, log loss, and modified hueber.

Evaluation

In this section, we present evaluation of our tool ROBUSTRANGE on several publicly available datasets with different sizes. We first describe the datasets used.

Datasets

We use datasets that represent different classification tasks where linear classifiers are used. Specifically, we use: Census Income (Kohavi 1996), Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017), Pen-Based Handwritten Digits Recognition (Alpaydin and Alimoglu 1996), Letter Recognition (Slate 1991), Emo (Yang, Xu, and Yu 2023), Speech (de Gibert et al. 2018), SST (Socher et al. 2013), and Tweet (Go, Bhayani, and Huang 2009), as well as the Essays Scoring (Hamner et al. 2012) and the Loan Prediction (Surana 2021) dataset from Kaggle. For the text datasets, we use both bag-of-words (BOW) and BERT-based (BERT) representations where indicated. Multi-class classification datasets were converted to binary classification datasets by only two classes. This was done for Fashion-MNIST(Pullover vs. Shirt), Pen-Based Handwritten Digits Recognition(4 vs. 0) and Letter Recognition(D vs. O).

ROBUSTRANGE was used to compute both robustness bounds for all test points in each dataset. For datasets without predefined test-train splits (e.g., Census Income, Fashion MNIST, Letter Recognition, Pen-Based Handwritten Digits Recognition), 10% of points were randomly sampled as test data, with the remainder constituted the train data. Dataset summaries are in Table 1.

In order to evaluate ROBUSTRANGE, we address the following research questions (RQs):

RQ1: How effective is our technique in finding upper bound robustness $\hat{r}$?

RQ2: How does upper bound robustness $\hat{r}$ impact victim's training process?

RQ3: How does our technique for finding $\hat{r}$ compare against SOTA?

RQ4: How effective is our technique in finding lower bound robustness $\check{r}$?

In the remaining section, we address our research questions through experiments conducted on an 8-core CPU with 30 GB RAM running Ubuntu 20.04.

RQ1: How effective is our technique in finding upper bound robustness $\hat{r}$?

Table 1 reports the average upper bound robustness $\hat{r}$ (column $\hat{r}$) computed using the hinge loss function (denoted as l in Algorithm 1), while Figure 2 shows the average computation time. The average $\hat{r}$ value ranged from 1% (Digits Recognition) to 31% (Loan (BOW)) of the training points. For five datasets (specifically, Census Income, Essays BOW and BERT, and Letter and Digits Recognition), ROBUSTRANGE identified $\hat{r}$ values as low as 4% (or even less) of the training points. The average time required to compute $\hat{r}$ for a test point was under two minutes.

Figure 3 shows the $\hat{r}$ histogram for the Census Income dataset, where nearly 60% of test points have robustness values around 3% of the training points. Similar results for other datasets are provided in the extended version.

Overall, ROBUSTRANGE was able to compute low $\hat{r}$ values for most datasets within a short duration.

#	Dataset	d	m	$\#\mathbf{x_t}$	$\hat{r}$	$\hat{r}_{IPr}$(normalized)	$\hat{r}_{IPr}$	%Found $\hat{r}_{IPr}$	ρ	ρ_{IPr}	$\check{r}$
1	Letter Recognition	16	1335	149	**38.16**	569.62	168.18	66	**0.55**	0.34	0.68
2	Digits Recognition	16	1404	156	**10.79**	1305.88	226.54	8	**0.17**	0.04	0.03
3	SST (BOW)	300	6920	872	1758.08	**93.76**	66.99	100	**0.99**	0.61	0.96
4	SST (BERT)	768	6920	872	1141.54	**439.64**	247.90	97	**0.97**	0.52	0.00
5	Emo (BOW)	300	9025	1003	1558.92	**237.95**	167.28	99	**0.99**	0.52	0.00
6	Speech (BOW)	300	9632	1071	1071.90	**624.09**	373.20	97	**0.61**	0.58	0.49
7	Speech (BERT)	768	9632	1071	**891.28**	931.04	593.20	96	**0.54**	0.49	0.10
8	Fashion-MNIST	784	10800	1200	1389.91	**1037.25**	274.13	93	**0.98**	0.45	0.01
9	Loan (BOW)	18	11200	2800	**3548.16**	10069.39	42.22	60	**0.93**	0.46	111.21
10	Emo (BERT)	768	11678	1298	**2153.26**	2913.84	232.23	77	**0.97**	0.40	560.45
11	Essays (BOW)	300	11678	1298	**495.90**	2884.23	1664.89	88	0.21	**0.50**	0.00
12	Essays (BERT)	768	11678	1298	**350.85**	3761.81	1245.04	76	0.22	**0.39**	0.00
13	Tweet (BOW)	300	18000	1000	4256.26	**304.15**	216.32	99	**1.00**	0.54	7.70
14	Tweet (BERT)	768	18000	1000	3795.24	**352.38**	343.72	100	**0.99**	0.55	0.32
15	Census Income	41	80136	8905	**2542.06**	66003.10	7012.80	19	**0.33**	0.07	87.19

Table 1: Here, d, m – dimension and size of datasets, $\#\mathbf{x_t}$– count of test points, $\hat{r}$, $\check{r}$ – avg upper and lower bound robustness from RobustRange, $\hat{r}_{IPr}$(normalized), $\hat{r}_{IPr}$ – avg upper bound robustness from IP-relabel, %Found $\hat{r}_{IPr}$– % of robustness found by IP-relabel, and ρ, ρ_{IPr}– avg likelihood of getting desired classification by RobustRange and IP-relabel.

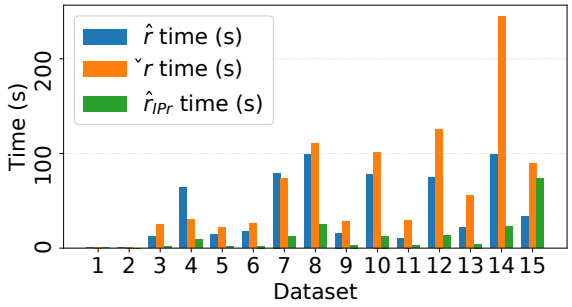

Figure 2: Average time taken in seconds per test point by RobustRange ($\hat{r}$, $\check{r}$) and IP-relabel ($\hat{r}_{IPr}$).

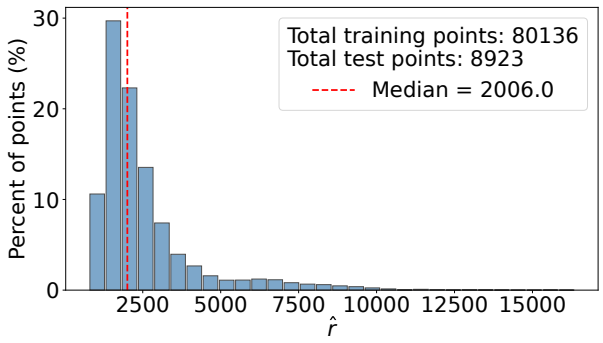

Figure 3: Histograms of the distribution of points across $\hat{r}$ values for the Census Income dataset.

RQ2: How does upper bound robustness $\hat{r}$ impact victim's training process?

To assess the impact of $\hat{r}$ on the victim's training process, we poisoned the training dataset by perturbing the labels of the training points corresponding to $\hat{r}$. We then trained a classifier on the poisoned dataset and compared its accuracy with the frequency of desired classification for test points (denoted by ρ). Since we assume the victim's training process is a black-box, we compared all pairs of loss functions.

More specifically, we assumed a loss function (l) for RobustRange and computed $\hat{r}$ for each test point. We then created poisoned datasets by perturbing the labels of 0, $\frac{\hat{r}}{4}$, $\frac{\hat{r}}{2}$, $\hat{r}$, $2\hat{r}$, and $4\hat{r}$ points (for $2\hat{r}$, $4\hat{r}$, additional random points were chosen). Each poisoned dataset was used to train a linear classifier with a loss function. We calculated the classifier's accuracy and ρ. This process was repeated for all possible pairs of loss functions, three to compute $\hat{r}$ and three for training the classifier.

Figure 4 shows results for the Census Income dataset: the optimal outcome for all loss functions is when the poisoned value is $\hat{r}$, as higher values sharply reduce accuracy without increasing flip likelihood. Similar trends appear in other datasets (see extended version), confirming that $\hat{r}$ computed from RobustRange is an effective poisoning measure.

RQ3: How does our technique for finding $\hat{r}$ compare against SOTA?

We compare our tool against the IP-relabel technique (Yang, Xu, and Yu 2023). While IP-relabel assumes a white-box victim's model and computes only an upper bound robustness, we include it in our comparison as it is the most closely related recent work. The average upper bound robustness computed by IP-relabel is presented in Table 1 (column $\hat{r}_{IPr}$). Notably, IP-relabel does not guarantee an upper bound robustness for all points. In our evaluation, it was able to compute robustness for all points in only 2/15 datasets (see column %Found $\hat{r}_{IPr}$). In contrast, RobustRange was able to find robustness for all test points across all datasets. For a fair comparison, when IP-relabel failed to generate an upper bound, we assigned the size of the training dataset as the bound (column $\hat{r}_{IPr}$(normalized)). Under this metric, RobustRange found lower average robustness for 8 datasets.

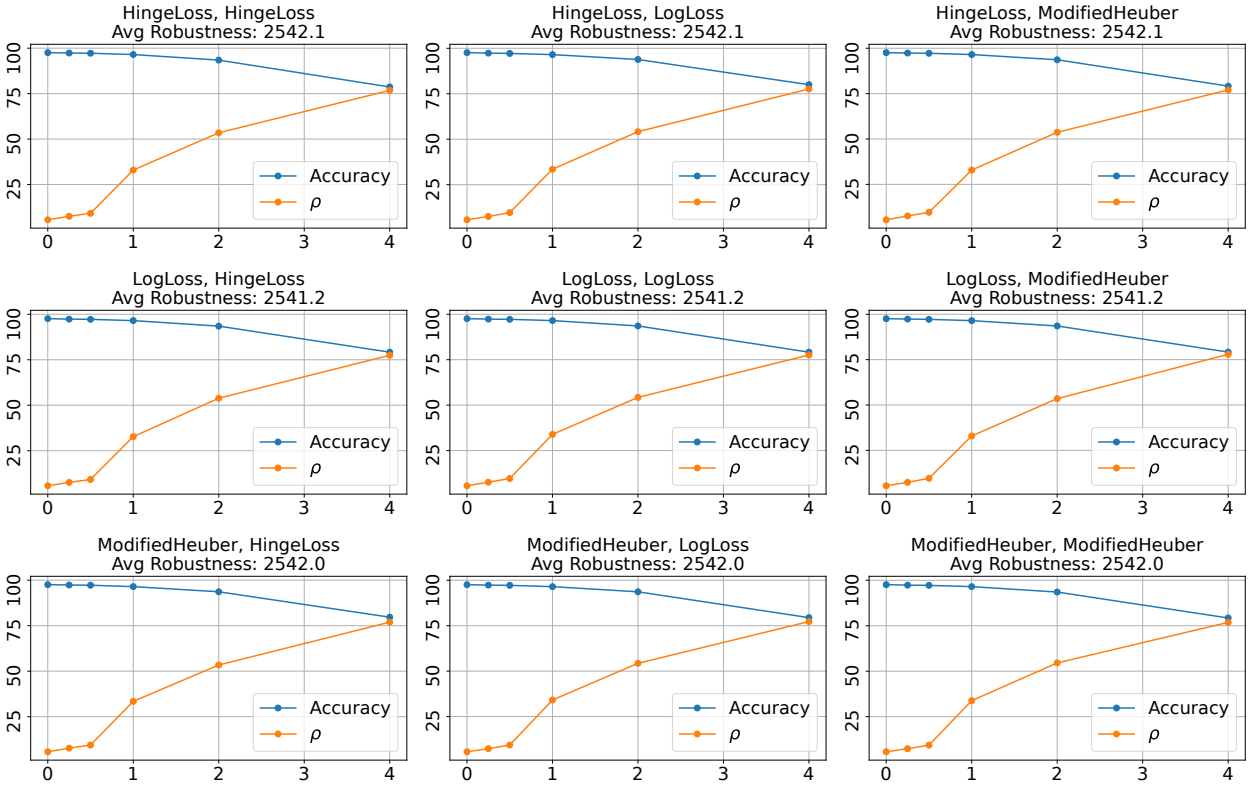

Figure 4: Comparison of average accuracy vs ρ for Census Income dataset with $\{0, \frac{\hat{r}}{4}, \frac{\hat{r}}{2}, \hat{r}, 2\hat{r}, 4\hat{r}\}$ and loss functions.

Additionally, we poison the dataset using each tool's robustness and compare the success rate of desired test classifications (columns ρ and ρ_{IP_r}). In this evaluation, RobustRange outperforms IP-relabel on 12 datasets, including 6 where its average robustness is lower.

In summary, our tool assumes a realistic black-box adversary, provides tighter robustness bounds, and more reliably achieves the desired test point classification.

RQ4: How effective is our technique in finding lower bound robustness $\check{r}$?

The average lower bound $\check{r}$ computed by RobustRange is presented in Table 1 (column $\check{r}$), along with the average time taken per test point in Figure 2. RobustRange generated a non-zero average $\check{r}$ for ten datasets. The average time taken per test point was under four minutes.

While the average lower bound robustness can be low for some datasets, our tool is capable of generating high low bounds. For example, consider the histograms in Figure 5 for Census Income dataset. Although the median robustness was 0, RobustRange was able to generate robustness values greater than 500 for 5% of the test points, and non-zero robustness for 15% of the points. Therefore, the lower bound robustness generated by our tool can still be useful for individual test points.

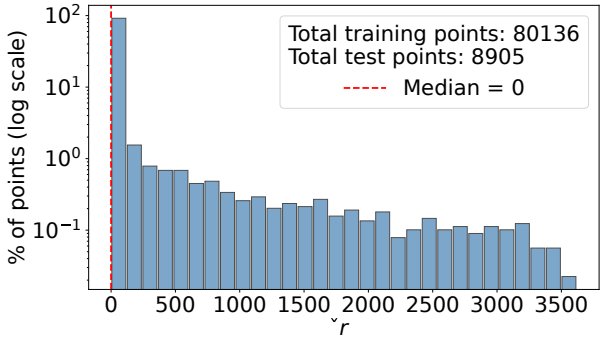

Figure 5: Histograms of the distribution of points across $\check{r}$ values for the Census Income dataset.

Conclusion and Future Work

In this paper, we have proven that the problem of finding robustness in targeted data poisoning is NP-complete in the setting considered, and have introduced effective techniques for computing lower and upper bound of robustness. An interesting future direction is to extend the lower bound algorithm for non-linear classifiers.

References

Adebayo, J.; Hall, M.; Yu, B.; and Chern, B. 2023. Quantifying and mitigating the impact of label errors on model disparity metrics. *arXiv preprint arXiv:2310.02533*.

Alpaydin, E.; and Alimoglu, F. 1996. Pen-Based Recognition of Handwritten Digits. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5MG6K.

Barreno, M.; Nelson, B.; Joseph, A. D.; and Tygar, J. D. 2010. The security of machine learning. *Machine learning*, 81: 121–148.

Bazaraa, M. S.; Jarvis, J. J.; and Sherali, H. D. 2011. *Linear programming and network flows*. John Wiley & Sons.

Biggio, B.; Nelson, B.; and Laskov, P. 2011. Support vector machines under adversarial label noise. In *Asian conference on machine learning*, 97–112. PMLR.

Biggio, B.; Nelson, B.; and Laskov, P. 2012. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*.

Bolusani, S.; Besançon, M.; Bestuzheva, K.; Chmiela, A.; Dionísio, J.; Donkiewicz, T.; van Doornmalen, J.; Eifler, L.; Ghannam, M.; Gleixner, A.; Graczyk, C.; Halbig, K.; Hedtke, I.; Hoen, A.; Hojny, C.; van der Hulst, R.; Kamp, D.; Koch, T.; Kofler, K.; Lentz, J.; Manns, J.; Mexi, G.; Mühmer, E.; Pfetsch, M. E.; Schlösser, F.; Serrano, F.; Shinano, Y.; Turner, M.; Vigerske, S.; Weninger, D.; and Xu, L. 2024. The SCIP Optimization Suite 9.0. Technical report, Optimization Online.

Chen, X.; Liu, C.; Li, B.; Lu, K.; and Song, D. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*.

Chen, Y.; Ding, Z.; and Wagner, D. 2023. Continuous learning for android malware detection. In *32nd USENIX Security Symposium (USENIX Security 23)*, 1127–1144.

Chen, Y.; Wang, S.; Qin, Y.; Liao, X.; Jana, S.; and Wagner, D. 2021. Learning security classifiers with verified global robustness properties. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 477–494.

Cinà, A. E.; Vascon, S.; Demontis, A.; Biggio, B.; Roli, F.; and Pelillo, M. 2021. The hammer and the nut: Is bilevel optimization really needed to poison linear classifiers? In *2021 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.

de Gibert, O.; Perez, N.; García-Pablos, A.; and Cuadros, M. 2018. Hate Speech Dataset from a White Supremacy Forum. In Fišer, D.; Huang, R.; Prabhakaran, V.; Voigt, R.; Waseem, Z.; and Wernimont, J., eds., *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, 11–20. Brussels, Belgium: Association for Computational Linguistics.

Ferrari Dacrema, M.; Cremonesi, P.; and Jannach, D. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM conference on recommender systems*, 101–109.

Gao, J.; Karbasi, A.; and Mahmoody, M. 2021. Learning and certification under instance-targeted poisoning. In *Uncertainty in Artificial Intelligence*, 2135–2145. PMLR.

Go, A.; Bhayani, R.; and Huang, L. 2009. Twitter sentiment classification using distant supervision. *Processing*, 150.

Gu, T.; Liu, K.; Dolan-Gavitt, B.; and Garg, S. 2019. Badnets: Evaluating backdooring attacks on deep neural networks. *Ieee Access*, 7: 47230–47244.

Hamner, B.; Morgan, J.; lynnvandev; Shermis, M.; and Ark, T. V. 2012. The Hewlett Foundation: Automated Essay Scoring. https://kaggle.com/competitions/asap-aes. Kaggle.

Hanneke, S.; Karbasi, A.; Mahmoody, M.; Mehalel, I.; and Moran, S. 2022. On optimal learning under targeted data poisoning. *Advances in Neural Information Processing Systems*, 35: 30770–30782.

Jia, J.; Liu, Y.; Cao, X.; and Gong, N. Z. 2022. Certified robustness of nearest neighbors against data poisoning and backdoor attacks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, 9575–9583.

Karp, R. M. 2009. Reducibility among combinatorial problems. In *50 Years of Integer Programming 1958-2008: from the Early Years to the State-of-the-Art*, 219–241. Springer.

Koh, P. W.; and Liang, P. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*, 1885–1894. PMLR.

Kohavi, R. 1996. Census Income. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5GP7S.

Kumar, R. S. S.; Nyström, M.; Lambert, J.; Marshall, A.; Goertzel, M.; Comissoneru, A.; Swann, M.; and Xia, S. 2020. Adversarial machine learning-industry perspectives. In *2020 IEEE security and privacy workshops (SPW)*, 69–75. IEEE.

Levine, A.; and Feizi, S. 2020. Deep partition aggregation: Provable defense against general poisoning attacks. *arXiv preprint arXiv:2006.14768*.

Paudice, A.; Muñoz-González, L.; and Lupu, E. C. 2019. Label sanitization against label flipping poisoning attacks. In *ECML PKDD 2018 Workshops: Nemesis 2018, UrbReas 2018, SoGood 2018, IWAISe 2018, and Green Data Mining 2018, Dublin, Ireland, September 10-14, 2018, Proceedings 18*, 5–15. Springer.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830.

Rosenfeld, E.; Winston, E.; Ravikumar, P.; and Kolter, Z. 2020. Certified robustness to label-flipping attacks via randomized smoothing. In *International Conference on Machine Learning*, 8230–8241. PMLR.

Saha, A.; Subramanya, A.; and Pirsiavash, H. 2020. Hidden trigger backdoor attacks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 11957–11965.

Shafahi, A.; Huang, W. R.; Najibi, M.; Suciu, O.; Studer, C.; Dumitras, T.; and Goldstein, T. 2018. Poison frogs! targeted clean-label poisoning attacks on neural networks. *Advances in neural information processing systems*, 31.

Slate, D. 1991. Letter Recognition. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5ZP40.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A.; and Potts, C. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In Yarowsky, D.; Baldwin, T.; Korhonen, A.; Livescu, K.; and Bethard, S., eds., *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1631–1642. Seattle, Washington, USA: Association for Computational Linguistics.

Suciu, O.; Marginean, R.; Kaya, Y.; Daume III, H.; and Dumitras, T. 2018. When does machine learning {FAIL}? generalized transferability for evasion and poisoning attacks. In *27th USENIX Security Symposium (USENIX Security 18)*, 1299–1316.

Surana, S. 2021. Loan Prediction based on Customer Behavior. https://www.kaggle.com/datasets/subhamjain/loan-prediction-based-on-customer-behavior. Accessed: 2025-07-31.

Şuvak, Z.; Anjos, M. F.; Brotcorne, L.; and Cattaruzza, D. 2022. Design of poisoning attacks on linear regression using bilevel optimization.

Suya, F.; Zhang, X.; Tian, Y.; and Evans, D. 2024. What Distributions are Robust to Indiscriminate Poisoning Attacks for Linear Learners? *Advances in neural information processing systems*, 36.

Tian, Z.; Cui, L.; Liang, J.; and Yu, S. 2022. A comprehensive survey on poisoning attacks and countermeasures in machine learning. *ACM Computing Surveys*, 55(8): 1–35.

Tramer, F.; and Boneh, D. 2020. Differentially private learning needs better features (or much more data). *arXiv preprint arXiv:2011.11660*.

Wang, W.; Levine, A.; and Feizi, S. 2022a. Lethal dose conjecture on data poisoning. *Advances in Neural Information Processing Systems*, 35: 1776–1789.

Wang, W.; Levine, A. J.; and Feizi, S. 2022b. Improved certified defenses against data poisoning with (deterministic) finite aggregation. In *International Conference on Machine Learning*, 22769–22783. PMLR.

Xiao, H.; Biggio, B.; Nelson, B.; Xiao, H.; Eckert, C.; and Roli, F. 2015. Support vector machines under adversarial label contamination. *Neurocomputing*, 160: 53–62.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:1708.07747.

Xiao, H.; Xiao, H.; and Eckert, C. 2012. Adversarial label flips attack on support vector machines. In *ECAI 2012*, 870–875. IOS Press.

Yang, J.; Jain, S.; and Wallace, B. C. 2023. How many and which training points would need to be removed to flip this prediction? *arXiv preprint arXiv:2302.02169*.

Yang, J.; Xu, L.; and Yu, L. 2023. Relabeling minimal training subset to flip a prediction. *arXiv preprint arXiv:2305.12809*.

Zhao, M.; An, B.; Gao, W.; and Zhang, T. 2017. Efficient label contamination attacks against black-box learning models. In *IJCAI*, 3945–3951.

Zhu, C.; Huang, W. R.; Li, H.; Taylor, G.; Studer, C.; and Goldstein, T. 2019. Transferable clean-label poisoning attacks on deep neural nets. In *International conference on machine learning*, 7614–7623. PMLR.