Machine Learning (2022) 111:1–47
https://doi.org/10.1007/s10994-021-06119-y

Stronger data poisoning attacks break data sanitization defenses

Pang Wei Koh[1] · **Jacob Steinhardt**[2] · **Percy Liang**[1]

Received: 24 March 2020 / Revised: 30 September 2021 / Accepted: 27 October 2021 /
Published online: 24 November 2021

Abstract
Machine learning models trained on data from the outside world can be corrupted by *data poisoning* attacks that inject malicious points into the models' training sets. A common defense against these attacks is *data sanitization*: first filter out anomalous training points before training the model. In this paper, we develop three attacks that can bypass a broad range of common data sanitization defenses, including anomaly detectors based on nearest neighbors, training loss, and singular-value decomposition. By adding just 3% poisoned data, our attacks successfully increase test error on the Enron spam detection dataset from 3 to 24% and on the IMDB sentiment classification dataset from 12 to 29%. In contrast, existing attacks which do not explicitly account for these data sanitization defenses are defeated by them. Our attacks are based on two ideas: (i) we coordinate our attacks to place poisoned points near one another, and (ii) we formulate each attack as a constrained optimization problem, with constraints designed to ensure that the poisoned points evade detection. As this optimization involves solving an expensive bilevel problem, our three attacks correspond to different ways of approximating this problem, based on influence functions; minimax duality; and the Karush–Kuhn–Tucker (KKT) conditions. Our results underscore the need to develop more robust defenses against data poisoning attacks.

Keywords Data poisoning · Data sanitization · Anomaly detection · Security

Pang Wei Koh and Jacob Steinhardt contributed equally.

Editors: Daniel Fremont, Mykel Kochenderfer, Alessio Lomuscio, Dragos Margineantu, Cheng Soon-Ong.

✉ Pang Wei Koh
pangwei@cs.stanford.edu

Jacob Steinhardt
jsteinhardt@berkeley.edu

Percy Liang
pliang@cs.stanford.edu

[1] Department of Computer Science, Stanford University, Stanford, USA

[2] Department of Statistics, UC Berkeley, Berkeley, USA

1 Introduction

In high-stakes settings like autonomous driving (Gu et al., 2017), biometrics (Chen et al., 2017), and cybersecurity (Rubinstein et al., 2009; Suciu et al., 2018), machine learning (ML) systems need to be secure against attacks by malicious actors. Securing ML systems is complicated by the fact that they are often trained on data obtained from the outside world, which makes them especially vulnerable. By attacking this data collection process, which could be as easy as creating a new user account, adversaries can inject malicious data into the system and cause it to fail.

These *data poisoning* attacks are the focus of the present work. We consider attacks against classifiers, wherein an attacker adds some small fraction of new training points to degrade the performance of the trained classifier on a test set. Figure 1 illustrates this setting: a model that might otherwise correctly classify most of the data (Fig. 1-Left) can be made to learn a significantly different decision boundary by an attacker who injects just a small amount of poisoned data (Fig. 1-Middle).

A common and often effective defense against data poisoning attacks is *data sanitization*, which use anomaly detectors to filter out training points that look suspicious (Hodge & Austin, 2004; Cretu et al., 2008; Paudice et al., 2018). Figure 1-Right illustrates a hypothetical defense: by removing the anomalous poisoned data, the defender can learn the correct decision boundary. In our experiments, common data sanitization defenses were able to defeat all of the existing data poisoning attacks we tested.

However, those attacks were naive in the sense that they did not explicitly try to evade the data sanitization defenses. This is typical in the literature: attackers might be optimized to act within an attack budget (Mei & Zhu, 2015b) and to only add points that belong to the input domain (e.g., word counts in a document should be integer-valued (Nelson et al., 2008; Newell et al., 2014), but not to evade defenses. Previous work has suggested that attacks optimized for evading data sanitization can in fact evade some types of defenses (Steinhardt et al., 2017), whereas attacks that are not optimized can be easily detected as anomalies (Frederickson et al., 2018). This leads to the question of whether data sanitization defenses are vulnerable to attackers who explicitly try to evade anomaly detection.

In this paper, we answer this question in the affirmative. We develop three data poisoning attacks that can simultaneously evade a broad range of common data sanitization

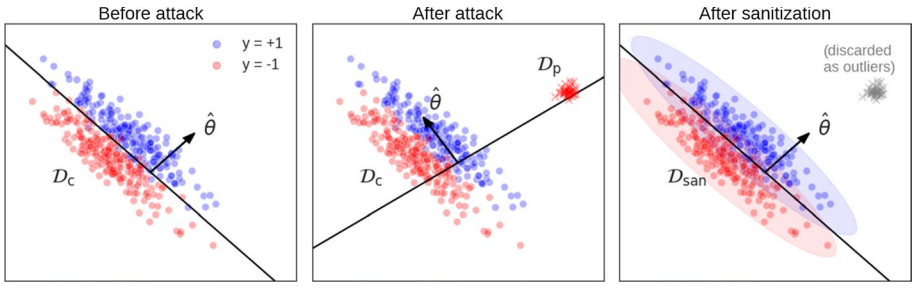


Fig. 1 Left: In the absence of any poisoned data, the defender can often learn model parameters $\hat{\theta}$ that fit the true data $\mathcal{D}_c$ well. Here, we show the decision boundary learned by a linear support vector machine on synthetic data. Middle: However, the addition of poisoned data $\mathcal{D}_p$ can significantly change the learned $\hat{\theta}$, leading to high test error $\mathcal{L}(\hat{\theta})$. Right: By discarding outliers from $\mathcal{D} = \mathcal{D}_c \cup \mathcal{D}_p$ and then training on the remaining $\mathcal{D}_{san}$, the defender can mitigate the effectiveness of the attacker. In this example, the defender discards all blue points outside the blue ellipse, and all red points outside the red ellipse (Color figure online)

defenses, including anomaly detectors based on nearest neighbors, training loss, singular-value decomposition, and the distance to class centroids. Our attacks are also able to deal with integer constraints on the input, which naturally arise in domains like natural language. For example, our attacks on a linear support vector machine increase test error on the Enron spam detection dataset from 3 to 24% and on the IMDB sentiment classification dataset from 12 to 29% by adding just 3% poisoned data, even in the presence of these data sanitization defenses.

We adopt two strategies to evade data sanitization defenses. The first strategy is targeted at defenses which use anomaly detectors that are highly sensitive to the presence of just a few points: e.g., an anomaly detector that throws out points which are far from their nearest neighbors will not recognize a point as anomalous if it is surrounded by a few other points, even if that small group of points is far from the rest of the data. Intuitively, such detectors tend to 'overfit' the training data. To evade these defenses, our attacks *concentrate* poisoned points in just a few distinct locations (Sect. 3). For example, poisoned data placed in a tight cluster will evade the nearest-neighbor-based anomaly detector that throws out points far away from other points. We show theoretically that we can concentrate all of the attack mass on just a few distinct points (e.g., only 2 points for 2-class support vector machines (SVMs) and logistic regression models) without any loss in attack effectiveness.

The second strategy is targeted at defenses which use anomaly detectors that are less sensitive and tend to be highly parametric: e.g., an anomaly detector that throws out points beyond some distance from the data centroid will not depend too much on the addition or removal of a few points from the data, as long as the data centroid does not change significantly. These defenses are more resistant to concentrated attacks, since they are less sensitive to small changes in the data (in this paper, we consider only attacks that inject a small fraction—3% or less—of poisoned data). To evade them, we formulate the attack as a constrained optimization problem, where the objective is to maximize the test loss of the model that the defender learns on the union of the clean and poisoned data; the optimization variables are the locations of the poisoned points; and the constraints are imposed by the defenses (such that a point that satisfies the constraints will be guaranteed to evade the defenses).

Formulating a data poisoning attack as an optimization problem is a common technique, first introduced in the context of support vector machines in Biggio et al. (2012b) and subsequently refined by Mei and Zhu (2015b) and later work. The central difficulty is that the bilevel problem—the attacker needs to reason about what model the defender would learn, which in turn requires solving an inner optimization problem—is non-convex and intractable to solve exactly (Bard 1991), and even local methods like gradient ascent can be slow (Koh & Liang, 2017). This is made even more challenging in our setting, compared to prior work, because we have additional constraints that encode the defenses that the attacker wishes to evade. To overcome this computational hurdle, we use two ideas:

1. We concentrate the attacks on a small number of distinct points. Beyond allowing us to evade some data sanitization defenses, as discussed above, it also significantly improves computational efficiency as we only need to optimize for the locations of a few distinct points. We use this to speed up gradient ascent on the bilevel optimization problem, leading to what we call the influence attack (Sect. 4.1).
2. The bilevel problem is expensive to solve because the effect of the optimization variables (poisoned points) on the objective (test loss) depends on the model parameters that the defender learns on the poisoned data, an intermediate quantity that is expensive to

compute. We break this dependency by first finding *decoy parameters*—model parameters that have high test error but low training error. Given fixed decoy parameters, we can then efficiently find poisoned points that yield the decoy parameters when trained on. We call this the KKT attack (Sect. 4.2), after the Karush–Kuhn–Tucker optimality conditions that used to derive this attack. Furthermore, we use these decoy parameters to adapt the attack introduced by Steinhardt et al. (2017), which in its original form is efficient but cannot evade loss-based defenses; this leads to our min–max attack (Sect. 4.3). These attacks mitigate some of the drawbacks of the influence attack, which is still too computationally intensive to scale to large datasets and sometimes gets stuck in local minima.

Finally, our study reveals several surprising facts about data poisoning:

1. Poisoned data does not have to look anomalous; if the poisoned points are carefully coordinated, each poisoned point can appear normal, as in the above example of the nearest-neighbor based anomaly detector.
2. Poisoned points need not have high loss under the poisoned model, so the defender cannot simply throw out points that have high loss. For example, given fixed decoy parameters, we can constrain our poisoned data to have low loss under those parameters.
3. Regularization reduces the effect that any single data point can have on the model and is therefore tempting to use as a defense against data poisoning. However, increasing regularization can actually make the defender more susceptible to attacks, because the defender becomes less able to fit the small fraction of poisoned points.

The success of our data poisoning attacks against common anomaly-based data sanitization defenses suggest that more work needs to be done on defending against data poisoning attacks. In particular, while anomaly detectors are well-suited to detect independently-generated anomalous points (e.g., due to some noise process in nature), a robust defense against data poisoning attacks will have to account for the ability of the attacker to place all of their poisoned data points in a coordinated fashion.

Beyond the merits of our specific attacks, our results underscore a broader point: data poisoning attacks need to account for defenses, and defenses correspondingly need to account for attacks that are specifically targeted against them. Attacks that do not consider defenses might work against a naive defender but be easily defeated by other defenses. Similarly, defenses that are only tested against basic attacks might give a false sense of security, as they might be broken by more determined and coordinated attackers.

2 Problem setting and defenses

2.1 General setting

We consider classification tasks. To simplify exposition, we focus on binary tasks, where we seek to learn a model $f_\theta : \mathcal{X} \to \{-1, +1\}$, parametrized by $\theta \in \mathbb{R}^d$, that maps from features $x \in \mathcal{X}$ to an output $y \in \{-1, +1\}$. A model f_θ is evaluated on a fixed test set $\mathcal{D}_{\text{test}} = \{(x_i, y_i)\}_{i=1}^{n_{\text{test}}}$ by its 0–1 test error $L_{0\text{-}1}(\theta; \mathcal{D}_{\text{test}})$, which is the proportion of $\mathcal{D}_{\text{test}}$ that it classifies wrongly:

$$L_{0\text{-}1}(\theta;\mathcal{D}_{\text{test}}) = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(x,y)\in\mathcal{D}_{\text{test}}} \mathbf{I}[f_\theta(x) \neq y]. \tag{1}$$

In the setting we consider, the *defender* aims to pick a $\hat{\theta}$ with low test error $L_{0\text{-}1}(\hat{\theta};\mathcal{D}_{\text{test}})$, while the *attacker* aims to mislead the defender into picking a $\hat{\theta}$ with high $L_{0\text{-}1}(\hat{\theta};\mathcal{D}_{\text{test}})$. The attacker observes the test set $\mathcal{D}_{\text{test}}$ as well as a clean training set $\mathcal{D}_{\text{c}} = \{(x_i, y_i)\}_{i=1}^{n}$, and chooses ϵn *poisoned* points $\mathcal{D}_{\text{p}}$ to add to $\mathcal{D}_{\text{c}}$. The defender observes the combined training set $\mathcal{D} = \mathcal{D}_{\text{c}} \cup \mathcal{D}_{\text{p}}$ consisting of the original n clean points and the ϵn additional poisoned points; uses a data sanitization defense to remove anomalous points; and then learns $\hat{\theta}$ from the remaining data.

The attacker has several advantages: it knows the test set in advance (whereas the defender does not); it knows the defender's training procedure; and it also gets to observe the clean training set $\mathcal{D}_{\text{c}}$. In reality, the attacker might not have access to all of this information. However, as defenders, we want to be robust even to attackers that might have the above information (this is the principle of security by design; see, e.g., Biggio et al. (2014)). For example, an attacker whose goal is to make the defender get a particular set of predictions wrong (e.g., the attacker might want to cause a "fake news" classifier to classify all websites from a certain domain as "real news") would accordingly choose, and therefore get to observe, $\mathcal{D}_{\text{test}}$. In contrast, the defender might not know the attacker's goal in advance, and therefore would not have access to $\mathcal{D}_{\text{test}}$.

Attacker

- Input: Clean training data $\mathcal{D}_{\text{c}}$ and test data $\mathcal{D}_{\text{test}}$.
- Output: Poisoned training data $\mathcal{D}_{\text{p}}$, with $|\mathcal{D}_{\text{p}}| = \epsilon|\mathcal{D}_{\text{c}}|$.
- Goal: Mislead defender into learning parameters $\hat{\theta}$ with high test error $L_{0\text{-}1}(\hat{\theta};\mathcal{D}_{\text{test}})$.

Defender

- Input: Combined training data $\mathcal{D} = \mathcal{D}_{\text{c}} \cup \mathcal{D}_{\text{p}}$.
- Output: Model parameters $\hat{\theta}$.
- Goal: Learn model parameters $\hat{\theta}$ with low test error $L_{0\text{-}1}(\hat{\theta};\mathcal{D}_{\text{test}})$ by filtering out poisoned points $\mathcal{D}_{\text{p}}$.

Further assumptions in experiments In our experiments, we assume that f_θ is a linear classifier, i.e., $f_\theta(x) = \text{sign}\,(\theta^\top x)$ for binary tasks. We consider both binary and multi-class classification. We also focus on *indiscriminate attacks* (Barreno et al., 2010), where the test data $\mathcal{D}_{\text{test}}$ is drawn from the same distribution as the clean training data $\mathcal{D}_{\text{c}}$, and the attacker tries to increase the average test error of the defender's model under this underlying data distribution. In Sect. 6.1, we show that the attacker can still construct strong indiscriminate attacks even when they do not know the test data $\mathcal{D}_{\text{test}}$. Most of our methods are also applicable to more general choices of $\mathcal{D}_{\text{test}}$; we discuss this further in Sect. 7.

2.2 Data sanitization defenses

To thwart the attacker, we assume the defender employs a *data sanitization* defense (Cretu et al., 2008), which first removes anomalous-looking points from $\mathcal{D} = \mathcal{D}_{\text{c}} \cup \mathcal{D}_{\text{p}}$ and then trains on the remaining points. The motivation is that intuitively, poisoned data that looks similar to the clean data will not be effective in changing the learned model;

therefore, the attacker would want to place poisoned points that are somehow different from the clean data. By discarding points that look too different (anomalous), the defender can thus protect itself against attack. Ideally—as in the hypothetical Fig. 1—a defense would discard the poisoned data $\mathcal{D}_{\text{p}}$ and leave the clean data $\mathcal{D}_{\text{c}}$, so that the defender learns model parameters $\hat{\theta}$ that have low test error.

Defenses differ in how they judge points as being anomalous. To formalize this, we represent each defense by a *score function* $s_\beta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that takes in a data point (x, y) and returns a number representing how anomalous that data point is. This score function is parametrized by *anomaly detector parameters* β that are derived from the combination of the clean and poisoned training data $\mathcal{D} = \mathcal{D}_{\text{c}} \cup \mathcal{D}_{\text{p}}$. As an example, in what we call the L2 defense, the defender discards points in $\mathcal{D}$ that are far from their respective class centroids. For this defense, $\beta = (\mu_+, \mu_-)$ would represent the class centroids in $\mathcal{D}$ and $s_\beta(x, y) = \|x - \beta_y\|_2$ would measure the distance of x to the centroid of class y.

Concretely, a defender (B, s_β):

1. Fits the *anomaly detector parameters* $\beta = B(\mathcal{D})$, where B is a function that takes in a dataset and returns a vector.
2. Constructs the *feasible set* $\mathcal{F}_\beta = \{(x, y) : (x, y) \in \mathcal{X} \times \mathcal{Y} \text{ with } s_\beta(x, y) < \tau_y\}$. The threshold τ_y is chosen such that a desired fraction of points from each class are discarded.
3. Forms the sanitized training dataset $\mathcal{D}_{\text{san}} = \mathcal{D} \cap \mathcal{F}_\beta$ by discarding all points that fall outside the feasible set.
4. Finds the $\hat{\theta}$ that minimizes the regularized training loss on $\mathcal{D}_{\text{san}}$:

$$\hat{\theta} = \operatorname*{argmin}_\theta L(\theta; \mathcal{D}_{\text{san}}) \stackrel{\text{def}}{=} \operatorname*{argmin}_\theta \frac{\lambda}{2}\|\theta\|_2^2 + \frac{1}{|\mathcal{D}_{\text{san}}|} \sum_{(x,y)\in\mathcal{D}_{\text{san}}} \ell(\theta; x, y), \tag{2}$$

   where λ is a hyperparameter controlling regularization strength and ℓ is a convex surrogate for the 0/1-loss that f_θ incurs.

We consider 5 different defenses that span existing approaches to data sanitization and anomaly detection:

– The L2 defense removes points far from their class centroids in L_2 distance:

$$\beta_y = \mathbb{E}_{\mathcal{D}}[x|y]$$
$$s_\beta(x, y) = \|x - \beta_y\|_2$$

– The slab defense (Steinhardt et al., 2017) projects points onto the line between the class centroids, then removes points too far from the centroids:

$$\beta_y = \mathbb{E}_{\mathcal{D}}[x|y]$$
$$s_\beta(x, y) = \left|(\beta_1 - \beta_{-1})^\top (x - \beta_y)\right|$$

  The idea is to use only the relevant dimensions in feature space to find outliers. The L2 defense treats all dimensions equally, whereas the slab defense treats the vector between the class centroids as the only relevant dimension.
– The loss defense discards points that are not well fit by a model trained (without any data sanitization) on the full dataset $\mathcal{D}$:

$$\beta = \operatorname{argmin}_\theta \mathbb{E}_{\mathcal{D}}[\ell_\theta(x, y)]$$
$$s_\beta(x, y) = \ell_\beta(x, y).$$

It is similar to the trimmed loss defense proposed in the context of regression in Jagielski et al. (2018). For a linear model, it is similar to the slab defense, except that the relevant dimension is learned using the loss function instead of being fixed as the direction between the class centroids.

– The SVD defense assumes that the clean data lies in some low-rank subspace, and that poisoned data therefore will have a large component out of this subspace (Rubinstein et al. 2009). Let X be the data matrix, with the i-th row containing x_i, the features of the i-th training point. Then:

$$\beta = \text{Matrix of top } k \text{ right singular vectors of } X$$
$$s_\beta(x, y) = \|(I - \beta\beta^\top)x\|_2$$

In our experiments, we choose the smallest k such that the normalized Frobenius approximation error (i.e., the normalized sum of the squared singular values) is < 0.05.

– The k-NN defense removes points that are far from their k nearest neighbors (e.g., Frederickson et al. (2018)).

$$\beta = \mathcal{D}_{\text{c}} \cup \mathcal{D}_{\text{p}}$$
$$s_\beta(x, y) = \text{Distance to } k\text{-th nearest neighbor in } \beta$$

In our experiments, we set $k = 5$.

Note that β is sometimes a simple set of summary statistics of the dataset (e.g., in the L2 and slab defenses), while at other times β can be the entire dataset (e.g., in the k-NN defense). We will handle these two types of defenses separately, as we discuss in Sect. 3.

The feasible set $\mathcal{F}_\beta$ encodes both the defenses and the input constraints of the dataset, since $\mathcal{F}_\beta \subseteq \mathcal{X} \times \mathcal{Y}$ and the input domain $\mathcal{X}$ only includes valid points. Thus, the defender will eliminate all input points that do not obey the input constraints of the dataset.

3 Attack framework

In this paper, we take on the role of the attacker. Recall that we are given a set of n clean training points $\mathcal{D}_{\text{c}}$ and a test set $\mathcal{D}_{\text{test}}$, and our goal is to come up with a set of ϵn poisoned training points $\mathcal{D}_{\text{p}}$ such that a defender following the procedure in Sect. 2.2 will choose model parameters $\hat{\theta}$ that incur high test error $\mathcal{L}(\hat{\theta})$. The difficulty lies in choosing poisoned points $\mathcal{D}_{\text{p}}$ that will both lead to high test error and also avoid being flagged as anomalous.

In this section, we describe our general approach to crafting attacks that can evade anomaly detectors. As mentioned in Sect. 1, we can roughly group anomaly detectors into two categories: those that are more sensitive to the data and tend to 'overfit' (like the k-NN defense), and those that are less sensitive to the data and tend to 'underfit' because they make strong parametric assumptions (like the L2 defense). In Sect. 3.1, we discuss how we can use concentrated attacks to evade defenses in the first group. In Sect. 3.2, we formulate the constrained optimization problem that we use to evade defenses in the second group. Finally, in Sect. 3.3, we introduce a randomized rounding procedure to handle problem settings where the input features are constrained to be integers.

3.1 Concentrated attacks

To bypass anomaly detectors that are sensitive to small changes in the data, we rely on the simple observation that poisoned data that is *concentrated* on a few locations tends to appear normal to anomaly detectors. For example:

- For the k-NN defense and other similar nonparametric detectors, this is trivially true: if several poisoned points are placed very near each other, then by definition, the distances to their nearest neighbors will be small.
- For the SVD defense, it is more likely that the low-rank representation of $\mathcal{D}$ will include the poisoned points, reducing their out-of-projection components.
- For the loss defense, if the poisoned points are concentrated in a similar location, the model will have more incentive to fit those points (because fitting one of them would imply fitting all of them, which would reduce the training loss more than fitting a single isolated point).

A potential issue for the attacker is that being constrained to place points in concentrated groups might make the attack less effective, in the sense of requiring more poisoned points to make the defender learn some target parameters. For example, it might be the case that a more efficient attack would involve spreading out each poisoned point throughout the feasible set.

Fortunately for the attacker, we show that if the feasible sets for each class are convex, and if the defender is using a 2-class SVM or logistic regression model, then the above scenario will not occur. Instead, we would only need two distinct points (one per class) to realize any attack:

Theorem 1 (2 points suffice for 2-class SVMs and logistic regression) *Consider a defender that learns a 2-class SVM or logistic regression model by first discarding all points outside a fixed feasible set $\mathcal{F}$ and then minimizing the average (regularized) training loss. Suppose that for each class $y = -1, +1$, the feasible set $\mathcal{F}_y \stackrel{\text{def}}{=} \{x : (x, y) \in \mathcal{F}\}$ is a convex set. If a parameter $\hat{\theta}$ is attainable by any set of $\tilde{n}$ poisoned points $\mathcal{D}_{\text{p}} = \{(\tilde{x}_1, \tilde{y}_1), \ldots, (\tilde{x}_{\tilde{n}}, \tilde{y}_{\tilde{n}})\} \subseteq \mathcal{F}$,*

then there exists a set of at most ñ poisoned points $\tilde{\mathcal{D}}_p$ (possibly with fractional copies) that also attains $\hat{\theta}$ but only contains 2 distinct points, one from each class.

More generally, the above statement is true for any margin-based model with loss of the form $\ell(\theta;x,y) = c(-y\theta^\top x)$, where $c : \mathbb{R} \to \mathbb{R}$is a convex, monotone increasing, and twice-differentiable function, and the ratio of second to first derivatives $c''/c'$is monotone non-increasing.

Remark 1 If the attacker concentrates all of the poisoned points in only two distinct locations, we will trivially evade the k-NN defense. The remaining defenses—L2, slab, loss, and SVD—all have convex feasible sets, so the conditions of the theorem hold. One technicality is that in reality, the feasible set will not remain fixed as the theorem assumes: if an attack $\mathcal{D}_p$ is collapsed into a different attack $\tilde{\mathcal{D}}_p$ with at most 2 distinct points, the feasible set $\mathcal{F}_\beta$ will also change. However, as discussed above, it will tend to change in a way that makes the poisoned points feasible, so if the original attack $\mathcal{D}_p$ was already feasible, the new attack $\tilde{\mathcal{D}}_p$ will likely also be feasible.

We defer the full proof to "Appendix 1". As a short proof sketch, the proof consists of two parts. We first relate the number of distinct points necessary to achieve an attack to the geometry of the set of gradients of points within the feasible set, using the notion of Carathéodory numbers from convex geometry. We then show, for the specific losses considered above (the hinge and logistic loss), that this set of feasible gradients has the necessary geometry. Our method is general and can be extended to different loss functions and feasible sets; we provide one such extension, to a multi-class SVM setting, in "Appendix 1".

One objection to attacks with only two distinct poisoned points is that they can be easily defeated by a defender that throws out repeated points. However, the attacker can add a small amount of random noise to fuzz up poisoned points without sacrificing the concentrated nature of the attack. Randomized rounding, which we use to handle datasets with integer input constraints in Sect. 3.3, is a version of this procedure.

3.2 Constrained optimization

Anomaly detectors that are more robust to small changes in the training data, such as those that rely on simple sufficient statistics of the data, tend to be less vulnerable to concentrated attacks. In our setting, the L2 and slab defenses in particular are not fooled by concentrated attacks: the class centroid, which is used in both defenses, cannot be moved too much by an ϵ fraction of poisoned points if ϵ is small and the clean training data $\mathcal{D}_c$ is well-clustered, since poisoned points that are too far away would be filtered out by the L2 defense.

To handle these defenses, we formulate the attacker's goal as a constrained optimization problem:

$$\begin{aligned} \underset{\mathcal{D}_p}{\text{maximize}} \quad & L_{0\text{-}1}(\hat{\theta};\mathcal{D}_{\text{test}}) \\ \text{s.t.} \quad & |\mathcal{D}_p| = \epsilon|\mathcal{D}_c|, && (\epsilon \text{ fraction of poisoned points}) \\ \text{where} \quad & \beta = B(\mathcal{D}_c \cup \mathcal{D}_p) && (\mathcal{F}_\beta \text{ is fit on clean and poisoned data}) \\ & \hat{\theta} \overset{\text{def}}{=} \operatorname{argmin}_\theta L\big(\theta;(\mathcal{D}_c \cup \mathcal{D}_p) \cap \mathcal{F}_\beta\big). && (\text{Defender trains on remaining data}) \end{aligned} \tag{3}$$

The first **constraint corresponds to the attacker only being able to add in an ϵ fraction of poisoned points; the second constraint corresponds to the defender fitting the anomaly detector on the entire training set $\mathcal{D}_c \cup \mathcal{D}_p$; and the final equality corresponds to the defender learning model parameters $\hat{\theta}$ that minimize training loss.

To make this problem more tractable, we make three approximations:

1. We assume that the defender does not discard any clean points. Thus, if all poisoned points are constrained to lie within the feasible set $\mathcal{F}_\beta$ and therefore evade sanitization, then the defender trains on the entirety of $\mathcal{D} = \mathcal{D}_c \cup \mathcal{D}_p$.[1]
2. We break the dependence of the feasible set $\mathcal{F}_\beta$ on the poisoned data $\mathcal{D}_p$ by fixing the anomaly detector $\beta = B(\mathcal{D}_c)$ on the clean data. This is a reasonable approximation for the L2 and slab defenses, as their feasible sets depend only on the class centroids, which are very stable with respect to small amounts of poisoned data $\mathcal{D}_p$.
3. Finally, as is standard, we replace the 0–1 test error $L_{0\text{-}1}(\hat{\theta};\mathcal{D}_{test})$ with its convex surrogate $L(\hat{\theta};\mathcal{D}_{test}) \stackrel{\text{def}}{=} \mathbb{E}_{\mathcal{D}_{test}} \ell(\hat{\theta};x,y)$, which is continuous and easier to optimize.

These approximations let us rewrite the attacker's goal as the following optimization problem:

$$\begin{aligned} \underset{\mathcal{D}_p}{\text{maximize}} \quad & L(\hat{\theta};\mathcal{D}_{test}) \\ \text{s.t.} \quad & |\mathcal{D}_p| = \epsilon|\mathcal{D}_c| \\ & \mathcal{D}_p \subseteq \mathcal{F}_\beta \\ \text{where} \quad & \hat{\theta} \stackrel{\text{def}}{=} \operatorname{argmin}_\theta L(\theta;\mathcal{D}_c \cup \mathcal{D}_p) \\ & \beta = B(\mathcal{D}_c). \end{aligned} \tag{4}$$

This constrained optimization formulation has been used in prior work (e.g., Steinhardt et al., 2017), and despite the approximations, it is still a bilevel problem that is non-convex and intractable to solve exactly (Bard, 1991, 1999). Our contribution in this regard is developing more effective and computationally efficient ways of solving it, which will be the focus of Sect. 4.

Instead of fixing the feasible set based on only the clean data, $\beta = B(\mathcal{D}_c)$, we can also adopt an iterative optimization approach, where we alternate between optimizing over $\mathcal{D}_p$ for a fixed β, and then updating β to reflect the new $\mathcal{D}_p$ (Algorithm 1). This iterative optimization procedure is guaranteed to make progress so long as the poisoned points $\mathcal{D}_p$ remain valid even after re-fitting β. In Sect. 5.4, we show that this slightly improves the attacks obtained, though it is not necessary to obtain effective attacks.

3.3 Handling integer input constraints with randomized rounding

Each of the three data poisoning attacks that we will develop use some form of gradient descent on the poisoned points to solve the attacker's optimization problem (4). However, gradient descent cannot be directly applied in settings where the input features are

[1] This favors the defender, since we do not consider the case in which a savvy attacker might place poisoned points in such a way as to cause the defender to throw out particularly good points in $\mathcal{D}_c$ and therefore learn a bad model.

constrained to be non-negative integers (e.g., with bag-of-word models in natural language tasks).

Algorithm 1 Pseudocode for attack via iterative constrained optimization.

Input: clean data $\mathcal{D}_c$, poisoned fraction ϵ.
Initialize anomaly detector on clean data: $\beta \leftarrow B(\mathcal{D}_c)$
repeat
  $\mathcal{D}_p \leftarrow \underset{\substack{|\mathcal{D}_p|=\epsilon|\mathcal{D}_c| \\ \mathcal{D}_p \subseteq \mathcal{F}_\beta}}{\text{argmax}} L(\hat{\theta}; \mathcal{D}_{\text{test}})$ (Solve (4) with fixed β; see Sections 4.1-4.3)
  $\beta \leftarrow B(\mathcal{D}_c \cup \mathcal{D}_p)$ (Update anomaly detector β with new $\mathcal{D}_p$)
until $\mathcal{D}_p$ converges
Output $\mathcal{D}_p$.

To handle this, Steinhardt et al. (2017) relaxed the integrality constraint to allow non-negative real-valued points, and then performed randomized rounding on these real-valued points to obtain integer-valued points:

1. Solve the optimization problem (4) while allowing all poisoned points $(x, y) \in \mathcal{D}_p$ to have real-valued x.
2. Then, for each $(x, y) \in \mathcal{D}_p$, we construct a rounded $\hat{x}$ as follows: for each coordinate i, $\hat{x}_i = \lceil x_i \rceil$ with probability $x_i - \lfloor x_i \rfloor$, and $\hat{x}_i = \lfloor x_i \rfloor$ otherwise. This procedure preserves the mean: $\mathbb{E}[\hat{x}] = x$.

However, this approach can produce poisoned points that get detected by data sanitization defenses. We adopt their approach but introduce two techniques to avoid detection:

Repeated points In high dimensions, randomized rounding can result in poisoned points that are far away from each other. This can make the poisoned points vulnerable to being filtered out by the defender, as they would no longer be concentrated in a few distinct locations (Sect. 3.1). To address this, we adopt a heuristic of repeating poisoned points r times after rounding (keeping the overall fraction of poisoned data, ϵ, the same). This means that we find $\epsilon n/r$ poisoned points $\{x_1, x_2, \ldots, x_{\epsilon n/r}\}$, do randomized rounding to get $\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{\epsilon n/r}\}$, and then form the multiset $\mathcal{D}_p$ by taking the union of r copies of this set. In practice, we find that setting $r = 2$ or 3 works well. This heuristic better concentrates the poisoned points while still preserving their expected mean $\mathbb{E}[\hat{x}]$.

Linear programming (LP) relaxation for the L2 defense Randomized rounding violates the L_2 constraint used in the L2 defense. Recall that in the L2 defense, we wish to play points (x, y) such that $\|x - \mu_y\|_2 \leq \tau_y$, where μ_y is the mean of class y and τ_y is some threshold (Sect. 2.2). The issue is Jensen's inequality: since the L_2 norm is convex, $\mathbb{E}[\|\hat{x} - \mu_y\|_2] \geq \|\mathbb{E}[\hat{x}] - \mu_y\|_2 = \|x - \mu_y\|_2$. This means that even if we control the norm of the continuous x by having $\|x - \mu_y\|_2 \leq \tau_y$, we could still have the randomly-rounded $\hat{x}$ violate this constraint on expectation: $\mathbb{E}[\|\hat{x} - \mu_y\|_2] > \tau_y$.

Steinhardt et al. (2017) deal with this problem by setting τ_y conservatively, so that $\hat{x}$ might avoid detection even if $\|\hat{x} - \mu_y\|_2 > \|x - \mu_y\|_2$. However, the conservative threshold reduces the attacker's options, resulting in a less effective attack. Instead, our approach is to control

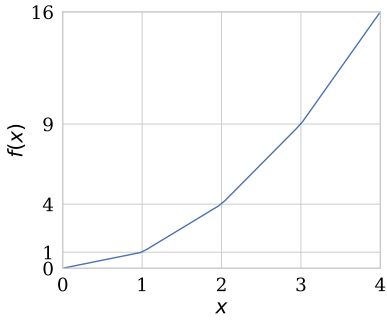


Fig. 2 Plot of $\mathbb{E}[\|\hat{x}\|_2^2] = f(x)$ against x for scalar x

the expected squared norm $\mathbb{E}[\|\hat{x} - \mu_y\|_2^2]$: if $\mathbb{E}[\|\hat{x} - \mu_y\|_2^2] < \tau_y^2$, then by Jensen's inequality, $\mathbb{E}[\|\hat{x} - \mu_y\|_2] < \tau_y$. To compute the expected squared norm, we first write

$$\mathbb{E}[\|\hat{x} - \mu_y\|_2^2] = \mathbb{E}[\|\hat{x}\|_2^2] - 2\langle x, \mu_y \rangle + \|\mu_y\|_2^2. \tag{5}$$

Note that $\mathbb{E}[\|\hat{x}\|_2^2]$ can in general be substantially larger than $\|x\|_2^2$ due to the variance from rounding. We can compute $\mathbb{E}[\|\hat{x}\|_2^2]$ explicitly as

$$\mathbb{E}[\|\hat{x}\|_2^2] = \sum_{i=1}^{d} x_i(\lceil x_i \rceil + \lfloor x_i \rfloor) - \lceil x_i \rceil \lfloor x_i \rfloor. \tag{6}$$

While the function $f(x) \stackrel{\text{def}}{=} x(\lceil x \rceil + \lfloor x \rfloor) - \lceil x \rceil \lfloor x \rfloor$ looks complicated, intuitively, we expect $f(x)$ to be close to x^2. Indeed, as Fig. 2 shows, it is a piecewise-linear function where the k-th piece linearly interpolates between k^2 and $(k+1)^2$, and we can write it as the maximum over a set of linear equations:

$$f(x) = \max_{k=0}^{\infty} (2k+1)x - k(k+1). \tag{7}$$

Thus, when solving the attacker optimization (Eq. (4)) for datasets with non-negative integer constraints, we replace the standard L2 feasible set $\mathcal{F}_\beta = \{(x, y) : \|x - \mu_y\|_2 \leq \tau_2 \text{ and } x \in \mathbf{R}_{\geq 0}\}$ with the modified constraint set

$$\mathcal{F}_{\text{LP}} = \{(x, y) : \mathbb{E}[\|\hat{x} - \mu_y\|_2^2] \leq \tau_y^2 \text{ and } x \in \mathbf{R}_{\geq 0}\}. \tag{8}$$

If we approximate the infinite maximum in (7) by its first M terms, then the corresponding approximation of $\mathcal{F}_{\text{LP}}$ can be represented via a linear program. In our experiments, we choose M adaptively for each coordinate i to be equal to the largest value that x_i attains across the dataset. This formulation allows us to express the L_2 norm constraint $\mathbb{E}[\|\hat{x} - \mu_y\|_2^2] \leq \tau_y^2$ as a set of linear constraints on the continuous x, allowing us to control the expected L_2 norm of the poisoned points after rounding.

Randomized rounding has some negative impact: it makes attacks less concentrated, as discussed above, and can also result in a few unlucky poisoned points getting filtered by other defenses (e.g., by the loss defense if the rounding happens to increase the loss on the point). Another advantage of the above LP relaxation is that in practice, the linear constraints tend to lead to nearly-integer x, which further reduces the negative impact of having to do randomized rounding.

4 Specific attacks

In this section, we introduce three different methods for efficiently solving the attacker's optimization problem (Eq. (4)) and generating an attack. As discussed in Sect. 3, these methods all generate concentrated attacks within our constrained optimization framework and use the randomized rounding procedure when necessary. We start with the influence attack in Sect. 4.1, which is direct but computationally slower, and then introduce the KKT and min–max attacks in Sects. 4.2and 4.3, which both use the idea of decoy parameters to speed up the attack.

4.1 The influence attack

Recall that solving Eq. (4) involves finding poisoned data $\mathcal{D}_\text{p}$ that maximizes the defender's test loss $L(\hat{\theta};\mathcal{D}_\text{test})$ for a fixed feasible set $\mathcal{F}_\beta$. The influence attack tackles this problem via projected gradient ascent.

At a high level, we can find a local maximum of Eq. (4) by iteratively taking gradient steps on the features of each poisoned point in $\mathcal{D}_\text{p}$, projecting each point onto the feasible set $\mathcal{F}_\beta$ after each iteration. This type of gradient-based data poisoning attack was first studied in the context of SVMs by Biggio et al. (2012b), and has subsequently been extended to linear and logistic regression (Mei and Zhu, 2015b), topic modeling (Mei & Zhu, 2015a), collaborative filtering (Li et al., 2016), and neural networks (Koh & Liang, 2017; Yang et al., 2017; Muñoz-González et al. 2017). We call this projected gradient ascent method the influence attack after Koh and Liang (2017), who use influence functions to compute this gradient. Our method builds upon previous work by incorporating the techniques mentioned in Sect. 3—concentrating the attack and using randomized rounding with the LP relaxation.

4.1.1 The basic influence attack

First, we review the basic influence attack, borrowing from the presentation in Koh and Liang (2017). Our goal is to perform gradient ascent on each poisoned point $(\tilde{x}, \tilde{y}) \in \mathcal{D}_\text{p}$ to maximize the test loss $L(\hat{\theta};\mathcal{D}_\text{test})$. The influence-basic algorithm (Algorithm 2) iteratively computes the gradient $\frac{\partial L}{\partial \tilde{x}}$ for each poisoned point $(\tilde{x}, \tilde{y}) \in \mathcal{D}_\text{p}$, moves each point in the direction of its gradient, and then projects each point back onto the feasible set $\mathcal{F}_\beta$. We only optimize over the input features $\tilde{x}$; since the labels $\tilde{y}$ are discrete, we cannot compute gradients on them. Instead, we fix the labels $\tilde{y}$ at the beginning of the algorithm, grid searching over the fraction of positive versus negative labels. We provide more implementation details in "Appendix 2.1".

Computing the gradient The difficulty in computing the gradient of the test loss $L(\hat{\theta};\mathcal{D}_\text{test})$ w.r.t. each $\tilde{x}$ in $\mathcal{D}_\text{p}$ is that L depends on $\tilde{x}$ only through the model parameters $\hat{\theta}$, which is a complicated function of $\mathcal{D}_\text{p}$. The influence attack uses a closed-form estimate of $\frac{\partial \hat{\theta}}{\partial \tilde{x}}$, which measures how much the model parameters $\hat{\theta}$ change with a small change to $\tilde{x}$. The desired derivative $\frac{\partial L}{\partial \tilde{x}}$ can then be computed via the chain rule: $\frac{\partial L}{\partial \tilde{x}} = \frac{\partial L}{\partial \hat{\theta}} \frac{\partial \hat{\theta}}{\partial \tilde{x}}$.

The quantity $\frac{\partial L}{\partial \hat{\theta}}$ is the average gradient of the test loss, which we denote as $g_{\hat{\theta},\mathcal{D}_\text{test}}$ for convenience, and it can be computed straightforwardly as

$$g_{\hat{\theta},\mathcal{D}_\text{test}} \stackrel{\text{def}}{=} \frac{\partial L}{\partial \hat{\theta}} = \frac{1}{|\mathcal{D}_\text{test}|} \sum_{(x,y)\in\mathcal{D}_\text{test}} \nabla \ell(\hat{\theta};x,y). \tag{9}$$

Calculating $\frac{\partial \hat{\theta}}{\partial \tilde{x}}$ is more involved, but a standard result (see, e.g., Section 2.2 of Koh and Liang (2017)) gives the expression

$$\frac{\partial \hat{\theta}}{\partial \tilde{x}} = -H_{\hat{\theta}}^{-1} \frac{\partial^2 \ell(\hat{\theta}; \tilde{x}, \tilde{y})}{\partial \hat{\theta} \, \partial \tilde{x}}, \tag{10}$$

where $H_{\hat{\theta}}$ is the Hessian of the training loss at $\hat{\theta}$:

$$H_{\hat{\theta}} \stackrel{\text{def}}{=} \lambda I + \frac{1}{|\mathcal{D}_c \cup \mathcal{D}_p|} \sum_{(x,y) \in \mathcal{D}_c \cup \mathcal{D}_p} \frac{\partial^2 \ell(\hat{\theta}; x, y)}{\partial \hat{\theta}^2}. \tag{11}$$

Combining Eqs. (9) to (11), the gradient of the test loss w.r.t. an attack point $\tilde{x}$ is

$$\frac{\partial L(\hat{\theta})}{\partial \tilde{x}} = -g_{\hat{\theta}, \mathcal{D}_{\text{test}}}^{\top} H_{\hat{\theta}}^{-1} \frac{\partial^2 \ell(\hat{\theta}; \tilde{x}, \tilde{y})}{\partial \hat{\theta} \, \partial \tilde{x}}.$$

Projecting onto the feasible set To prevent the poisoned points from being sanitized, we project each poisoned point $(\tilde{x}, \tilde{y}) \in \mathcal{D}_p$ onto the feasible set $\mathcal{F}_\beta$. The projection is well-defined for the L2 and slab defenses, as their feasible sets are convex, and finding the projection is a convex optimization problem that can be solved efficiently by a general-purpose convex solver. When the $\tilde{x}$'s are constrained to take on integer values, the influence-basic attack uses the vanilla randomized rounding procedure described in Sect. 3.3 (without the repeated points heuristic or linear programming relaxation).

Algorithm 2 The influence-basic attack.

Input: clean data set $\mathcal{D}_c$, poisoning fraction ϵ, step size η, feasible set $\mathcal{F}_\beta$.
Initialize poisoned data set $\mathcal{D}_p \leftarrow \{(\tilde{x}_1, \tilde{y}_1), \ldots, (\tilde{x}_{\epsilon n}, \tilde{y}_{\epsilon n})\}$.
for $t = 1, 2, \ldots$ **do**
  Compute model parameters $\hat{\theta} \leftarrow \text{argmin}_\theta \, L\left(\theta; (\mathcal{D}_c \cup \mathcal{D}_p) \cap \mathcal{F}_\beta\right)$.
  Pre-compute $g_{\hat{\theta}, \mathcal{D}_{\text{test}}}^{\top} H_{\hat{\theta}}^{-1}$ as in (9) and (11).
  for $i = 1, \ldots, \epsilon n$ **do**
    Set $\tilde{x}_i^0 \leftarrow \tilde{x}_i - \eta g_{\hat{\theta}, \mathcal{D}_{\text{test}}}^{\top} H_{\hat{\theta}}^{-1} \frac{\partial^2 \ell(\hat{\theta}; \tilde{x}_i, \tilde{y}_i)}{\partial \hat{\theta} \, \partial \tilde{x}_i}$. (gradient update)
    Set $\tilde{x}_i \leftarrow \text{argmin}_{x \in \mathcal{F}_\beta} \|x - \tilde{x}_i^0\|_2$. (projection)
  end for
end for
Output $\mathcal{D}_p$.

4.1.2 Improvements to the basic algorithm

We introduce the influence attack, which improves upon the influence-basic attack in two ways. First, it incorporates randomized rounding with the LP relaxation, as described in Sect. 3.3. Second, it concentrates the attack (Sect. 3.1), which makes the attack stronger and more computationally efficient. The influence-basic attack optimizes over each of the ϵn poisoned points separately, which is slow and results in poisoned points that are often quite far from each other (because of differences in initialization), leaving them vulnerable to being detected as anomalies. As Theorem 1 shows, we can modify the algorithm to instead only consider copies of two distinct points $(\tilde{x}_+, 1)$ and $(\tilde{x}_-, -1)$, one from each class,

without any loss in potential attack effectiveness. This is faster, as at each iteration, we only need to compute the gradients and do the projection twice (vs. ϵn times). Moreover, the resulting attack is by construction concentrated on only two distinct points, helping it evade detection.

However, even after these improvements, the influence attack is slow especially in high dimensions: each iteration of gradient descent requires computing an expensive inverse Hessian-vector product (10) and a projection onto the feasible set. Moreover, the influence attack relies on local optimization and can sometimes get stuck in poor local minima, even when the underlying model loss is convex. To mitigate these shortcomings, we propose the KKT attack, which we discuss next.

4.2 The KKT attack

The KKT attack is based on the observation that the attacker's optimization problem (4) is difficult to solve because the optimization variable $\mathcal{D}_\text{p}$ only affects the objective (test loss $L(\hat{\theta};\mathcal{D}_\text{test})$) through the model parameters $\hat{\theta}$, which are themselves a complicated function of $\mathcal{D}_\text{p}$. In general, we do not know what $\hat{\theta}$ would lead to an attack that is both effective and realizable; but if we did know which $\hat{\theta}$ we were after, then the attacker's optimization problem simplifies to finding $\mathcal{D}_\text{p}$ such that $\hat{\theta} = \operatorname{argmin}_\theta L(\theta;\mathcal{D}_\text{c} \cup \mathcal{D}_\text{p})$. As we will show in this section, this simplified problem can be solved much more efficiently than the original bilevel problem.

The KKT attack has two parts:

1. Using fast heuristics to find *decoy parameters* θ_decoy that we want the defender to learn, and then
2. Finding poisoned data $\mathcal{D}_\text{p}$ that tricks the defender into learning those decoy parameters θ_decoy.

The name of this attack comes from the use of the Karush–Kuhn–Tucker (KKT) first-order necessary conditions for optimality in the second step.

4.2.1 Finding decoy parameters θ_decoy

Good decoy parameters, from the perspective of the attacker, should have high test error while still being achievable by some poisoned data $\mathcal{D}_\text{p}$. Decoy parameters that have a high loss on the clean data $\mathcal{D}_\text{c}$ are unlikely to be achievable by an ϵ fraction of poisoned data $\mathcal{D}_\text{p}$, since it is likely that there exist other parameters that would have a lower training loss on the combined data $\mathcal{D}_\text{c} \cup \mathcal{D}_\text{p}$ and would therefore be learned instead by the defender.

Algorithm 3 Finding decoy parameters θ_{decoy}

Input: clean data set $\mathcal{D}_{\text{c}}$, loss threshold γ, number of repeats r.
$\theta_c \leftarrow \text{argmin}_\theta\, L(\theta; \mathcal{D}_{\text{c}})$
$\mathcal{D}_{\text{flip}} \leftarrow r$ copies of $\{(x, y) \in \mathcal{D}_{\text{test}} : \ell(\theta_c; x, y) \geq \gamma\}$
$\mathcal{D}_{\text{decoy}} \leftarrow \mathcal{D}_{\text{c}} \cup \mathcal{D}_{\text{flip}}$
$\theta_{\text{decoy}} \leftarrow \text{argmin}_\theta\, L(\theta; \mathcal{D}_{\text{decoy}})$
Output $\mathcal{D}_{\text{decoy}}$.

Our heuristic is to augment the clean data $\mathcal{D}_{\text{c}}$ with a dataset $\mathcal{D}_{\text{flip}}$ comprising label-flipped examples from $\mathcal{D}_{\text{test}}$, and then find the parameters θ_{decoy} that minimize the training loss on this modified training set $\mathcal{D}_{\text{decoy}} = \mathcal{D}_{\text{c}} \cup \mathcal{D}_{\text{flip}}$ (Algorithm 3). The idea is that since the decoy parameters θ_{decoy} were trained on $\mathcal{D}_{\text{decoy}}$, which incorporates flipped points from $\mathcal{D}_{\text{test}}$, it might achieve high test loss. At the same time, the following informal argument suggests that on the clean data $\mathcal{D}_{\text{c}}$, the decoy parameters θ_{decoy} are likely to be not much worse than the optimal parameters for the clean data, θ_c. By construction,

$$\begin{aligned}|\mathcal{D}_{\text{c}}| \cdot L(\theta_{\text{decoy}}; \mathcal{D}_{\text{c}}) + |\mathcal{D}_{\text{flip}}| \cdot L(\theta_{\text{decoy}}; \mathcal{D}_{\text{flip}}) &= |\mathcal{D}_{\text{decoy}}| \cdot L(\theta_{\text{decoy}}; \mathcal{D}_{\text{decoy}}) \\ &\leq |\mathcal{D}_{\text{decoy}}| \cdot L(\theta_c; \mathcal{D}_{\text{decoy}}) \\ &= |\mathcal{D}_{\text{c}}| \cdot L(\theta_c; \mathcal{D}_{\text{c}}) + |\mathcal{D}_{\text{flip}}| \cdot L(\theta_c; \mathcal{D}_{\text{flip}}).\end{aligned}$$

Rearranging terms, this implies that

$$\begin{aligned}L(\theta_{\text{decoy}}; \mathcal{D}_{\text{c}}) &\leq L(\theta_c; \mathcal{D}_{\text{c}}) + \frac{|\mathcal{D}_{\text{flip}}|}{|\mathcal{D}_{\text{c}}|} \cdot \left(L(\theta_c; \mathcal{D}_{\text{flip}}) - L(\theta_{\text{decoy}}; \mathcal{D}_{\text{flip}})\right) \\ &\leq L(\theta_c; \mathcal{D}_{\text{c}}) + \frac{|\mathcal{D}_{\text{flip}}|}{|\mathcal{D}_{\text{c}}|} \cdot L(\theta_c; \mathcal{D}_{\text{flip}}),\end{aligned}$$

where the last inequality comes from the non-negativity of the loss L. The second term on the right-hand side, $L(\theta_c; \mathcal{D}_{\text{flip}})$, is likely to be small: $\mathcal{D}_{\text{flip}}$ comprises of points that originally had a high loss under θ_c before their labels were flipped (which implies that their label-flipped versions are likely to have a lower loss), and we can choose $|\mathcal{D}_{\text{flip}}|$ to be small compared to $|\mathcal{D}_{\text{c}}|$. Thus, the average loss $L(\theta_{\text{decoy}}; \mathcal{D}_{\text{c}})$ of the decoy parameters θ_{decoy} on the clean data $\mathcal{D}_{\text{c}}$ is not likely to be too much higher than $L(\theta_c; \mathcal{D}_{\text{c}})$, which is the best possible average loss on $\mathcal{D}_{\text{c}}$ within the model family.

By varying the loss threshold γ and number of repeats r used in Algorithm 3, we obtain different candidates for θ_{decoy}. As we will discuss next, finding an attack $\mathcal{D}_{\text{p}}$ for each candidate θ_{decoy} is fast, so we simply generate a set of candidate decoy parameters and try all of them, picking the θ_{decoy} that achieves the highest test loss.

4.2.2 Attacking with known θ_{decoy}

For given decoy parameters θ_{decoy}, the next step for the attacker is to find poisoned data $\mathcal{D}_{\text{p}}$ such that $\mathcal{D}_{\text{p}}$ evades data sanitization and θ_{decoy} minimizes the overall training loss over both $\mathcal{D}_{\text{p}}$ and the clean data $\mathcal{D}_{\text{c}}$. We can formulate this task as the following optimization problem:

$$\begin{aligned} \text{find} \quad & \mathcal{D}_\text{p} \\ \text{s.t.} \quad & |\mathcal{D}_\text{p}| = \epsilon|\mathcal{D}_\text{c}| \\ & \mathcal{D}_\text{p} \subseteq \mathcal{F}_\beta \\ & \theta_\text{decoy} = \operatorname{argmin}_\theta L(\theta; \mathcal{D}_\text{c} \cup \mathcal{D}_\text{p}). \end{aligned} \tag{12}$$

Since θ_decoy is pre-specified, we can rewrite the inner optimization as a simple equality. Specifically, if the loss ℓ is strictly convex and differentiable in θ, we can rewrite the condition

$$\begin{aligned} \theta_\text{decoy} &= \operatorname{argmin}_\theta L(\theta; \mathcal{D}_\text{c} \cup \mathcal{D}_\text{p}) \\ &= \operatorname{argmin}_\theta \sum_{(x,y)\in\mathcal{D}_\text{c}} \ell(\theta; x, y) + \sum_{(\tilde{x},\tilde{y})\in\mathcal{D}_\text{p}} \ell(\theta; \tilde{x}, \tilde{y}) \end{aligned}$$

as the equivalent KKT optimality condition

$$\sum_{(x,y)\in\mathcal{D}_\text{c}} \nabla_\theta \ell(\theta_\text{decoy}; x, y) + \sum_{(\tilde{x},\tilde{y})\in\mathcal{D}_\text{p}} \nabla_\theta \ell(\theta_\text{decoy}; \tilde{x}, \tilde{y}) = 0. \tag{13}$$

If the loss ℓ is not differentiable, e.g., the hinge loss, we can replace this with a similar subgradient condition.

Since the first term in (13) is fixed and does not depend on the optimization variable $\mathcal{D}_\text{p}$, we can treat it as a constant:

$$g_{\theta_\text{decoy},\mathcal{D}_\text{c}} \stackrel{\text{def}}{=} \frac{1}{|\mathcal{D}_\text{c}|} \sum_{(x,y)\in\mathcal{D}_\text{c}} \nabla_\theta \ell(\theta_\text{decoy}; x, y).$$

Rewriting (13) as $g_{\theta_\text{decoy},\mathcal{D}_\text{c}} + \frac{1}{|\mathcal{D}_\text{c}|} \sum_{(\tilde{x},\tilde{y})\in\mathcal{D}_\text{p}} \nabla_\theta \ell(\theta_\text{decoy}; \tilde{x}, \tilde{y}) = 0$ and substituting it into (12) gives us

$$\begin{aligned} \text{find} \quad & \mathcal{D}_\text{p} \\ \text{s.t.} \quad & |\mathcal{D}_\text{p}| = \epsilon|\mathcal{D}_\text{c}| \\ & \mathcal{D}_\text{p} \subseteq \mathcal{F}_\beta \\ & g_{\theta_\text{decoy},\mathcal{D}_\text{c}} + \frac{1}{|\mathcal{D}_\text{c}|} \sum_{(\tilde{x},\tilde{y})\in\mathcal{D}_\text{p}} \nabla_\theta \ell(\theta_\text{decoy}; \tilde{x}, \tilde{y}) = 0. \end{aligned} \tag{14}$$

If this optimization problem (14) has a solution, we can find it by solving the equivalent norm-minimization problem

$$\begin{aligned} \underset{\mathcal{D}_\text{p}}{\text{minimize}} \quad & \left\| g_{\theta_\text{decoy},\mathcal{D}_\text{c}} + \frac{1}{|\mathcal{D}_\text{c}|} \sum_{(\tilde{x},\tilde{y})\in\mathcal{D}_\text{p}} \nabla_\theta \ell(\theta_\text{decoy}; \tilde{x}, \tilde{y}) \right\|_2^2 \\ \text{s.t.} \quad & |\mathcal{D}_\text{p}| = \epsilon|\mathcal{D}_\text{c}| \\ & \mathcal{D}_\text{p} \subseteq \mathcal{F}_\beta, \end{aligned} \tag{15}$$

which moves the KKT constraint into the objective, relying on the fact that the norm of a vector is minimized when the vector is 0.

Next, we make use of Theorem 1, which shows that for the binary classification problems we consider, we can concentrate our attacks by placing all of the poisoned points at two

distinct locations $\tilde{x}_+$ and $\tilde{x}_-$ without any loss in attack effectiveness. If we let $\epsilon_+ \cdot n$ and $\epsilon_- \cdot n$ be the number of positive and negative poisoned points added, respectively, we can write (15) as

$$\begin{aligned} \underset{\tilde{x}_+, \tilde{x}_-, \epsilon_+, \epsilon_-}{\text{minimize}} \quad & \left\| g_{\theta_{\text{decoy}}, \mathcal{D}_c} + \epsilon_+ \nabla_\theta \ell(\theta_{\text{decoy}}; \tilde{x}_+, 1) + \epsilon_- \nabla_\theta \ell(\theta_{\text{decoy}}; \tilde{x}_-, -1) \right\|_2^2 \\ \text{s.t.} \quad & \epsilon_+ + \epsilon_- = \epsilon \\ & (\tilde{x}_+, 1), (\tilde{x}_-, -1) \in \mathcal{F}_\beta. \end{aligned} \tag{16}$$

For general losses, this optimization problem is non-convex but can be solved by local methods like gradient descent. In our experiments, the defender uses the hinge loss $\ell(\theta; x, y) = \max(0, 1 - y\theta^\top x)$ and applies ℓ_2 regularization with regularization parameter λ ((2), Sect. 2.2). This setting allows us to further rewrite (16) as the following:

$$\begin{aligned} \underset{\tilde{x}_+, \tilde{x}_-, \epsilon_+, \epsilon_-}{\text{minimize}} \quad & \left\| g_{\theta_{\text{decoy}}, \mathcal{D}_c} - \epsilon_+ \tilde{x}_+ + \epsilon_- \tilde{x}_- + \lambda \theta_{\text{decoy}} \right\|_2^2 \\ \text{s.t.} \quad & 1 - \theta^\top \tilde{x}_+ \geq 0 \\ & 1 + \theta^\top \tilde{x}_+ \geq 0 \\ & \epsilon_+ + \epsilon_- = \epsilon \\ & (\tilde{x}_+, 1), (\tilde{x}_-, -1) \in \mathcal{F}_\beta, \end{aligned} \tag{17}$$

where the first two constraints ensure that $(\tilde{x}_+, 1)$ and $(\tilde{x}_-, -1)$ are both support vectors. This problem is convex when the feasible set $\mathcal{F}_\beta$ is convex and ϵ_+ and ϵ_- are fixed; since $\mathcal{F}_\beta$ is convex in our setting, we can grid search over ϵ_+ and ϵ_- and call a generic solver for the resulting convex problem. Pseudocode is given in Algorithm 4.

Algorithm 4 KKT attack with grid search.

Input: clean data set $\mathcal{D}_c$, feasible set $\mathcal{F}_\beta$, poisoning fraction ϵ, decoy parameter θ_{decoy}, grid search size T.
for $t = 0, \ldots, T$ **do**
  Set $\epsilon_+ \leftarrow t\epsilon/T$, $\epsilon_- \leftarrow \epsilon - \epsilon_+$.
  Obtain $\hat{\theta}, \tilde{x}_+$, and $\tilde{x}_-$ by solving (17) with fixed values of ϵ_+, ϵ_-.
  Evaluate test error $\mathcal{L}(\hat{\theta})$.
end for
Pick $\tilde{x}_+, \tilde{x}_-, \epsilon_+, \epsilon_-$ corresponding to highest test error $\mathcal{L}(\hat{\theta})$ found in grid search.
Output $\mathcal{D}_p = \{\epsilon_+ |\mathcal{D}_c| \text{ copies of } \tilde{x}_+ \text{ and } \epsilon_- |\mathcal{D}_c| \text{ copies of } \tilde{x}_-\}$.

Evading the loss defense Defenses like the loss defense have feasible sets that depend on the model parameters $\hat{\theta}$ that the defender learns, which in turn depend on the poisoned points. This dependence makes it difficult for the attacker to explicitly constrain their poisoned points to lie within such feasible sets. In the influence attack, we relied on concentrated attacks (Sect. 3.1) to evade the loss defense. This approach is empirically effective, but it relies to some extent on luck, as the attacker cannot guarantee that its poisoned points will have low loss.

One advantage of decoy parameters is that they give attackers a computationally tractable handle on parameter-dependent defenses like the loss defense. With decoy parameters, the attacker can approximately specify the feasible set $\mathcal{F}_\beta$ independently of the learned parameters $\hat{\theta}$, since we know that if the attack works, the learned parameters $\hat{\theta}$ should be

close to the decoy parameters θ_{decoy}. For example, we can handle the loss defense by adding the constraint $\ell(\theta_{\text{decoy}};\tilde{x},\tilde{y}) < \tau_{\tilde{y}}$ to the feasible set $\mathcal{F}_\beta$.

4.3 Improved min–max attack

Our third and final attack is the min–max attack, which improves on what we call the min–max-basic attack from prior work (Steinhardt et al., 2017). The min–max attack relies on the same decoy parameters introduced in Sect. 4.2, but unlike the influence and KKT attacks, it naturally handles multi-class problems without a grid search, and it does not require convexity of the feasible set. Its drawback is assuming that the clean data $\mathcal{D}_{\text{c}}$ and the test data $\mathcal{D}_{\text{test}}$ are drawn from the same distribution (i.e., that the attacker is performing an *indiscriminate* attack; see Sects. 2and 7).

4.3.1 The min–max-basic attack

We start by reviewing the min–max-basic attack, as it was introduced in Steinhardt et al. (2017). Recall that the attacker's goal is to find poisoned points $\mathcal{D}_{\text{p}}$ that maximize the test loss $L(\hat{\theta};\mathcal{D}_{\text{test}})$ that the defender incurs, where the parameters $\hat{\theta}$ are chosen to minimize the training loss $L(\hat{\theta};\mathcal{D}_{\text{c}} \cup \mathcal{D}_{\text{p}})$ (Eq. (4)). As we discussed in Sects. 3.2, 4.1, and 4.2, the bilevel nature of this optimization problem—maximizing the loss involves an inner minimization to find the parameters $\hat{\theta}$—makes it difficult to solve.

The key insight in Steinhardt et al. (2017) was that we can make this problem tractable by replacing the test loss $L(\hat{\theta};\mathcal{D}_{\text{test}})$ with the training loss $L(\hat{\theta};\mathcal{D}_{\text{c}} \cup \mathcal{D}_{\text{p}})$. This substitution changes the bilevel problem into a *saddle-point problem*—i.e., one that can be expressed in the form $\min_u \max_v f(u,v)$—that can be solved efficiently via gradient descent.

To do so, we first approximate the average test loss with the average clean training loss:

$$L(\theta;\mathcal{D}_{\text{test}}) \approx L(\theta;\mathcal{D}_{\text{c}}). \tag{18}$$

This approximation only works in the setting where the test data $\mathcal{D}_{\text{test}}$ is drawn from the same distribution as the (clean) training data $\mathcal{D}_{\text{c}}$, and relies on the training set being sufficiently big and the model being appropriately regularized, such that test loss is similar to training loss. Next, we make use of the non-negativity of the loss to upper bound the average clean training loss with the average combined loss on the clean and poisoned data:

$$L(\theta;\mathcal{D}_{\text{c}}) \le L(\theta;\mathcal{D}_{\text{c}}) + \epsilon L(\theta;\mathcal{D}_{\text{p}}) = (1+\epsilon)L(\theta;\mathcal{D}_{\text{c}} \cup \mathcal{D}_{\text{p}}), \tag{19}$$

where, as usual, ϵ is the relative ratio of poisoned points $\epsilon = \frac{|\mathcal{D}_{\text{p}}|}{|\mathcal{D}_{\text{c}}|}$.

By combining (18) and (19), we can approximately upper-bound the average test loss $L(\theta;\mathcal{D}_{\text{test}})$ by $(1+\epsilon)$ times the average loss on the combined training data $L(\theta;\mathcal{D}_{\text{c}} \cup \mathcal{D}_{\text{p}})$. Instead of directly optimizing for $L(\theta;\mathcal{D}_{\text{test}})$ as the attacker (Eq. (4)), we can therefore optimize for $L(\theta;\mathcal{D}_{\text{c}} \cup \mathcal{D}_{\text{p}})$, which gives us

$$\begin{aligned} \underset{\mathcal{D}_{\text{p}} \subseteq \mathcal{F}_\beta}{\text{maximize}} \quad & L(\theta;\mathcal{D}_{\text{c}} \cup \mathcal{D}_{\text{p}}) \\ \text{where} \quad & \hat{\theta} \overset{\text{def}}{=} \operatorname{argmin}_\theta L(\theta;\mathcal{D}_{\text{c}} \cup \mathcal{D}_{\text{p}}). \end{aligned}$$

The advantage of this formulation is that the outer maximization and inner minimization are over the same function $L(\theta;\mathcal{D}_c \cup \mathcal{D}_p)$, which lets us rewrite it as the saddle-point problem

$$\max_{\mathcal{D}_p \subseteq \mathcal{F}_\beta} \min_{\theta} L(\hat{\theta};\mathcal{D}_c \cup \mathcal{D}_p). \tag{20}$$

When the loss ℓ is convex, we can solve (20) by swapping min and max and solving the resulting problem $\min_\theta \max_{\mathcal{D}_p \subseteq \mathcal{F}_\beta} L(\theta;\mathcal{D}_c \cup \mathcal{D}_p)$, which expands out to

$$\min_{\theta} \left[\frac{\lambda(1+\epsilon)}{2}\|\theta\|_2^2 + \frac{1}{|\mathcal{D}_c|} \sum_{(x,y)\in\mathcal{D}_c} \ell(\theta;x,y) + \epsilon \max_{(\tilde{x},\tilde{y})\in\mathcal{F}_\beta} \ell(\theta;\tilde{x},\tilde{y}) \right]. \tag{21}$$

This problem is convex when the loss ℓ is convex, and we can solve it via subgradient descent by iteratively finding $(\tilde{x},\tilde{y}) \in \mathcal{F}_\beta$ that maximizes $\ell(\theta;\tilde{x},\tilde{y})$, then taking the subgradient of the outer expression w.r.t. $(\tilde{x},\tilde{y})$.

To solve the inner problem of finding $(\tilde{x},\tilde{y}) \in \mathcal{F}_\beta$ that maximizes $\ell(\theta;\tilde{x},\tilde{y})$, we note that if the model is margin-based, i.e., $\ell(\theta;\tilde{x},\tilde{y}) = c(-y\theta^\top x)$ for some monotone increasing function c (which is the case for SVMs and logistic regression), then maximizing $\ell(\theta;\tilde{x},\tilde{y})$ is equivalent to minimizing the margin $y\theta^\top x$. For a fixed $\tilde{y}$, we can solve the convex problem

$$\begin{aligned} \underset{\tilde{x}}{\text{minimize}} \quad & \tilde{y}\theta^\top \tilde{x} \\ \text{s.t.} \quad & (\tilde{x},\tilde{y}) \in \mathcal{F}_\beta. \end{aligned}$$

To find the $(\tilde{x},\tilde{y}) \in \mathcal{F}_\beta$ that maximizes the loss $\ell(\theta;\tilde{x},\tilde{y})$, we therefore enumerate over the possible choices of $\tilde{y}$, solving the above convex problem for each $\tilde{y}$, and pick the one that gives the smallest (most negative) margin.

Steinhardt et al. (2017) show that if (21) is minimized incrementally via ϵn iterations of gradient descent, then we can form a strong attack $\mathcal{D}_p$ out of the corresponding set of ϵn maximizers $\{(\tilde{x},\tilde{y})\}$. Pseudocode is given in Algorithm 5.

Algorithm 5 min-max-basic attack.

Input: clean data $\mathcal{D}_c$, poisoned fraction ϵ, burn-in n_{burn}, feasible set $\mathcal{F}_\beta$.
Initialize: $\theta \in \mathbf{R}^d$, $\mathcal{D}_p \leftarrow \emptyset$.
for $t = 1, \ldots, n_{\text{burn}} + \epsilon n$ **do**
  $Pick(\tilde{x},\tilde{y}) \in \text{argmax}_{(x,y)\in\mathcal{F}_\beta} \ell(\theta;x,y)$. (Find highest-loss point in $\mathcal{F}_\beta$)
  $\theta \leftarrow \theta - \eta(\lambda\theta + \nabla L(\theta) + \epsilon\nabla\ell(\theta;\tilde{x},\tilde{y}))$ (Gradient update)
  if $t > n_{\text{burn}}$ **then**
    $\mathcal{D}_p \leftarrow \mathcal{D}_p \cup \{(\tilde{x},\tilde{y})\}$ (Add point to attack set)
  end if
end for
Output $\mathcal{D}_p$.

This algorithm automatically handles class balance, since at each iteration it chooses to add either a positive or negative point; it can thus handle multi-class attacks without additional difficulty, unlike the KKT or influence attacks. Moreover, unlike the influence attack, it avoids solving the expensive bilevel optimization problem.

Table 1 Characteristics of the datasets we consider, together with the base test errors that an SVM achieves on them (with regularization parameters λ selected by validation). The input covariates for Enron and IMDB must be non-negative integers

Dataset	Classes	n	n_{test}	d	Base Error	Input constraints
Enron	2	4137	1035	511	2.9% ($\lambda = 0.09$)	$\mathbf{Z}_{\geq 0}$
IMDB	2	25000	25000	89527	11.9% ($\lambda = 0.01$)	$\mathbf{Z}_{\geq 0}$
MNIST	10	60000	10000	784	7.5%[a]	[0, 1]
MNIST-1-7	2	13007	2163	784	0.7% ($\lambda = 0.01$)	[0, 1]
Dogfish	2	1800	600	2048	1.3% ($\lambda = 1.10$)	$\mathbf{R}$

[a]We used the multi-class SVM formulation in Crammer and Singer (2002) with no explicit regularization for the 10-class MNIST dataset. We used AdaGrad (Duchi et al., 2010), which provides implicit regularization, as the optimizer.

4.3.2 Improvements to the basic algorithm

We improve the min–max-basic attack by incorporating the decoy parameters introduced in Sect. 4.2 (Algorithm 3). The problem with the min–max-basic attack, which repeatedly adds the highest-loss point that lies in the feasible set $\mathcal{F}_\beta$, is that at low ϵ, the attack might end up picking poisoned points $\mathcal{D}_\text{p}$ that are not fit well by the model, i.e., with high $L(\hat{\theta};\mathcal{D}_\text{p})$. These points could still lead to a high combined loss $L(\hat{\theta};\mathcal{D}_\text{c} \cup \mathcal{D}_\text{p})$, but such an attack would be ineffective for two reasons:

1. If the poisoned points have high loss compared to the clean points, they are likely to be filtered by the loss defense.
2. Even if the poisoned points are not filtered out, the loss on the clean data $L(\hat{\theta};\mathcal{D}_\text{c})$ might still be low, implying that the test loss would also be low. Such a scenario could happen if there is no model that fits both the poisoned points $\mathcal{D}_\text{p}$ and the clean points $\mathcal{D}_\text{c}$ well; since ϵ is small, overall training loss could then be minimized by fitting $\mathcal{D}_\text{c}$ well at the expense of $\mathcal{D}_\text{p}$.

We therefore want to keep the loss on the poisoned points, $L(\hat{\theta};\mathcal{D}_\text{p})$, small. To do so, we use the *decoy parameters* θ_decoy from Sect. 4.2 (Algorithm 3), augmenting the feasible set $\mathcal{F}_\beta$ with the constraint

$$\ell(\theta_\text{decoy};x,y) \leq \tau, \tag{22}$$

for some fixed threshold τ. At each iteration, the attacker thus searches for poisoned points $(\tilde{x},\tilde{y})$ that maximize loss under the current parameters θ while having low loss under the decoy parameters θ_decoy. This procedure addresses the two issues above:

1. If the learned parameters are driven towards θ_decoy, the poisoned points in $\mathcal{D}_\text{p}$ will have low loss due to the constraint (22), and hence will not get filtered by the loss defense.
2. Adding poisoned points with high loss under the current parameters but low loss under the decoy parameters θ_decoy is likely to drive the learned parameters towards θ_decoy. In turn, this will increase the test loss, since θ_decoy is chosen to have high test loss Sect. 4.2.1.

We find that empirically, the min–max-basic attack naturally yields attacks that are quite concentrated. For datasets with integer input constraints, we additionally use the linear programming relaxation and repeated points heuristic (Sect. 3.3). Altogether, these changes to the min–max-basic attack yield what we call the min–max attack.

5 Experiments: attackers with complete information

5.1 Datasets

Our experiments focus on two binary classification datasets. Summaries of each dataset are given in Table 1, including the number of training points n, the dimension of each point d, and the base accuracy of an SVM trained only on the clean data.

1. The Enron spam classification text dataset (Metsis et al., 2006), which requires input to be non-negative and integer valued, as each feature represents word frequency. The Enron dataset has $n \approx d$ and a relatively low base error of 3.0%.
2. The IMDB sentiment classification text dataset (Maas et al., 2011), which similarly requires input to be non-negative and integer valued. Compared to the other datasets, the IMDB dataset has significantly larger n and d, presenting computational challenges. It also has $n \ll d$ and is not as linearly separable, with a high base error of 11.9%.

In addition, we use the standard 10-class MNIST dataset (LeCun et al., 1998) as an illustration of a multi-class setting.

These datasets are considered in Steinhardt et al. (2017), which also studied data poisoning. In "Appendix 3", we also consider experiments on the other two datasets studied in that work: MNIST-1-7, a binary version of MNIST (LeCun et al., 1998), and Dogfish (Koh & Liang, 2017). These datasets were shown by Steinhardt et al. (2017) to have some certificates of defensibility using the L2 and slab defenses, and indeed, our attacks were not as effective on them as they were for the other datasets above.

5.2 Setup

We used the non-iterative version of Algorithm 1 to carry out the data poisoning attacks. We assumed that the attacker has limited control over the training data: in our experiments, we allowed the attacker to only add up to $\epsilon = 3\%$ poisoned data, and we set the data sanitization threshold τ such that the defender removes $p = 5\%$ of the training data from each class after training its anomaly detector on the combined clean and poisoned dataset $\mathcal{D}_c \cup \mathcal{D}_p$.

As the attacker's goal is to increase test error regardless of which defense is deployed against it, we evaluated an attack $\mathcal{D}_p$ by running each of the 5 defenses in Sect. 2.2 separately against it, and measuring the minimum increase in test error it achieves over all of the defenses.

We optimized each attack against all of the defenses. Specifically, for the influence attack, we took the feasible set $\mathcal{F}_\beta$ to be the intersection of the feasible sets under the L2 and slab defenses, plus any additional input constraints that each dataset imposed. For the KKT and min–max attacks, we used the decoy parameters to expand the feasible set $\mathcal{F}_\beta$ to incorporate the loss defense as well. We relied on concentrated attacks to evade the remaining defenses.

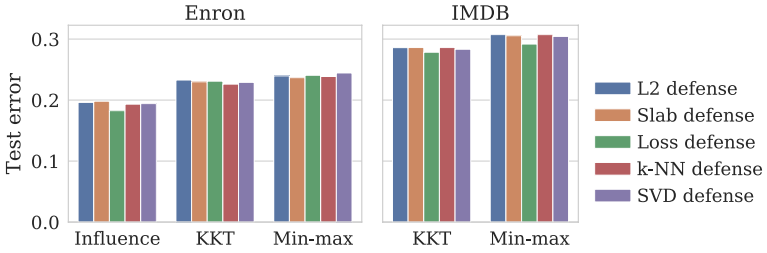
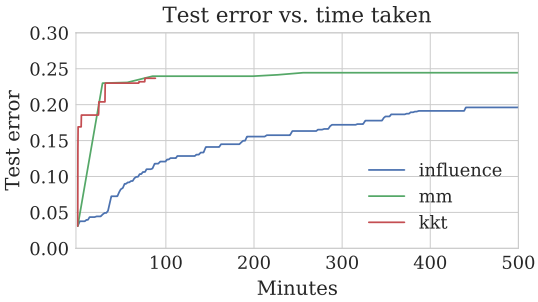


Fig. 3 The KKT and min–max attacks give slightly higher test error than the influence attack on the Enron dataset. Moreover, they are more computationally efficient, and can be run on the larger and higher-dimensional IMDB dataset

Fig. 4 The test error achieved by the different attacks (against the L2 defense), versus the number of minutes taken to generate the attacks. Each step increase in test error represents the processing of one choice of decoy parameters (for the KKT and min–max attacks) or 10 gradient steps (for the influence attack)



For the binary classification tasks, we used support vector machines (SVMs) with the hinge loss $\ell(\theta;x,y) = \max(0, 1 - y\theta^\top x)$ and with L_2 regularization, fixing the regularization parameter via cross-validation on the clean data.[2] For the multi-class task, we used the multi-class SVM formulation in Crammer and Singer (2002). Further implementation details of our attacks are in "Appendix 2".

5.3 Comparing attacks on Enron and IMDB

We tested all three attacks on the Enron spam classification dataset, and the KKT and min–max attack on the IMDB sentiment classification dataset (which was too large to run the influence attack on). All of the attacks were successful, with the KKT and min–max attack achieving slightly higher test error than the influence attack on the Enron dataset (Fig. 3-Left). As each defense is evaluated separately against the attack, we plot a bar for each defense. However, our attacks are constructed to avoid all of the defenses; this simulates the fact that the attacker might not know which defense will be deployed ahead of time, and therefore strives to evade all of them.

For the KKT and min–max attacks, successful attacks did not need to exactly reach the decoy parameters; in fact, trying to get to ambitious (i.e., high test error but unattainable) decoy parameters could sometimes outperform exactly reaching unambitious decoy

[2] In practice, the defender does not have access to the clean data, so its regularization parameter will be chosen based on the full dataset $\mathcal{D}_c \cup \mathcal{D}_p$. However, our framework assumes that the attacker knows the regularization parameter in advance. This is a potential disadvantage for the defender. In Sect. 6.2, we study what happens if the attacker does not know exactly how much regularization the defender will use.

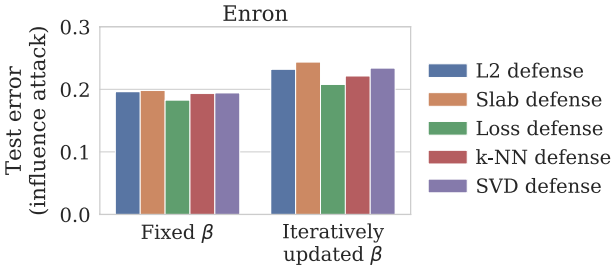


Fig. 5 Iteratively updating the feasible set $\mathcal{F}_\beta$ increases test error by a few percentage points on the Enron dataset (with $\epsilon = 3\%$ poisoned data), compared to fixing the feasible set based on just the clean data $\mathcal{D}_c$

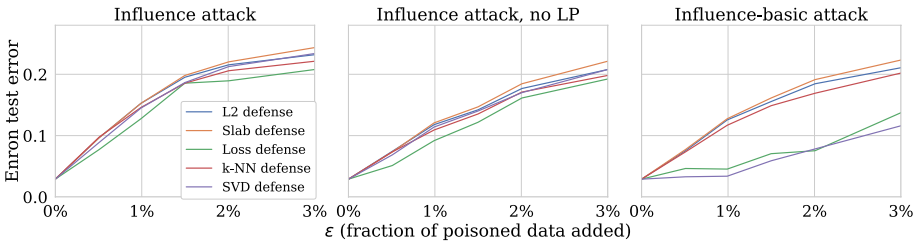


Fig. 6 Ablative study of the changes we made to the influence attack, evaluated on the Enron dataset. Left: results of the influence** attack. Middle: results of the influence attack, without the linear programming (LP) relaxation described in Sect. 3.3. Right: results of the influence-basic attack, which does not use the LP relaxation nor a concentrated attack

parameters. (Of course, the ideal choice of decoy parameters would have high test error but also be attainable.)

Timing

We measured the speed of each attack against the Enron dataset by running them each on 2 cores of an Intel Xeon E5 processor. Additionally, the influence attack used a GeForce GTX Titan X for GPU-based calculations. Despite not using a GPU, the KKT and min–max attacks were significantly faster: while the influence attack took 286 min to reach 17% error, the KKT attack only took 27 s (Fig. 4). The min–max attack took 28 min to process its first decoy parameter, which gave 23.0% error. The main computational bottleneck for the influence attack is the inverse Hessian-vector product calculation in Eq. (10). In contrast, the other two attacks solve convex subproblems (as opposed to the non-convex problem that the influence attack is doing gradient descent on) and admit more efficient general-purpose convex optimization solvers.

5.4 Iterative optimization

The above experiments simply fix the feasible set $\mathcal{F}_\beta$ based on the clean training data, as described in Sect. 3.2. To study the effect of refining the attacks by iteratively $\mathcal{F}_\beta$ (Algorithm 1), we ran an iterative version of the influence attack against the Enron dataset. Figure 5 shows that iterative optimization does only slightly better: at the low levels of ϵ that we chose, the attack does not shift the centroids of the data that much, and therefore the feasible set $\mathcal{F}_\beta$ for both the L2 and slab defenses stay somewhat constant.

One perspective on iterative optimization in our setting is that it is targeting the slab defense by trying to rotate the vector between the two class centroids; as Steinhardt et al.

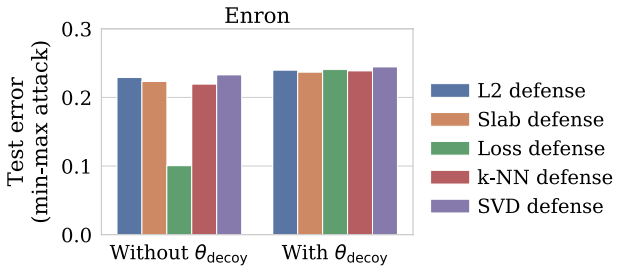


Fig. 7 The use of decoy parameters allows the min–max attack to evade the loss defense

(2017) show, at large ϵ (e.g., $\epsilon = 0.3$, which is 10 times larger than what we consider), this vector can be significantly changed by the poisoned points, whereas the $L2$ feasible set is harder to perturb.[3] On the Enron dataset and at the low ϵ settings that we consider, the slab defense only decreases test error by a few percentage points, so iterative optimization only increases test loss by a few percentage points. To illustrate this point, we ran an attack with $\epsilon = 0.3$ on the MNIST-1-7 dataset: the influence attack without iterative optimization achieved an increase in test error of only 1.1%, while the influence attack with iterative optimization achieved a larger increase in test error of 7.5%.

5.5 Ablations for the influence attack

We studied the effect of the two improvements made to the influence-basic attack—the linear programming (LP) relaxation and the concentration of the attack—on the Enron dataset. Removing the linear programming (LP) relaxation decreased the achieved test error by a few percentage points (Fig. 6-Left vs. Mid). Further removing the concentrated attack decreased test error substantially (Fig. 6-Right). The influence-basic attack is still optimized to evade the L2 and slab defenses, but because its poisoned points are not concentrated, many of them get filtered out by the loss and SVD defenses. Consequently, it does not manage to increase test error beyond 11% under those defenses.

5.6 Ablations for the min–max attack

To measure the effect of using decoy parameters in the min–max attack, we ran the min–max-basic attack from Steinhardt et al. (2017), augmented with the linear programming relaxation and repeated points heuristic. (The unaugmented version of the min–max-basic attack from Steinhardt et al. (2017) performs worse.) In contrast to the min–max attack, the min–max-basic attack gets defeated by the loss defense (test error 10.0%, Fig. 7). As discussed in Sect. 4.3.2, without the constraints imposed by the decoy parameters, the

[3] Note that our influence attacks on MNIST-1-7 at high ϵ are considerably weaker than the attacks in Steinhardt et al. (2017), which uses a specialized semi-definite program that relies on poisoning the anomaly detector (i.e., placing poisoned points to move the class centroids in a way that renders the slab defense ineffective). They achieve an increase in test error of 39% for $\epsilon = 0.3$. The high-ϵ setting is not our focus in this paper, since it is less realistic; this performance gap could be a result of poor step size tuning or initialization on our part, or it could signify a weakness in the applicability of iterative optimization and/or gradient descent to the high-ϵ setting.

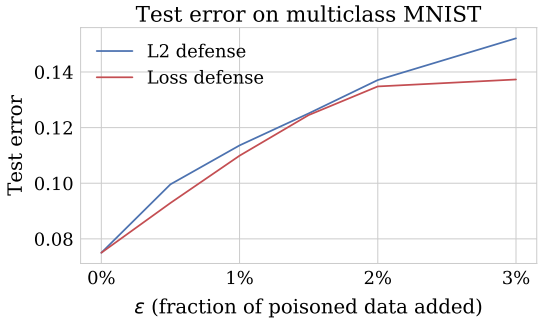


Fig. 8 The min–max attack scales to handle multi-class attacks, as it automatically chooses the class balance / relative proportion of poisoned points. Here, we show that the min–max attack can increase test error on the MNIST dataset to 13.7% with $\epsilon = 3\%$ poisoned data

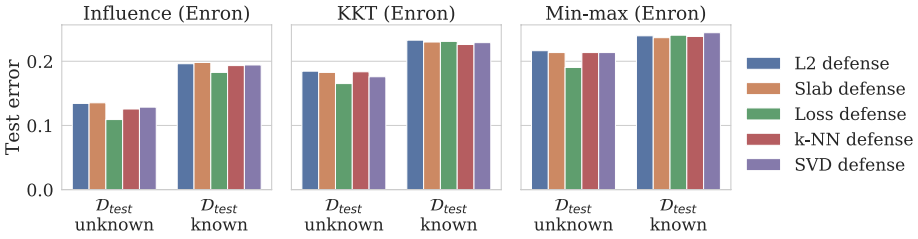


Fig. 9 The performance of each attack when the test data $\mathcal{D}_{\text{test}}$ is unknown and the attack is optimized against the training data (left columns) versus in the setting where $\mathcal{D}_{\text{test}}$ is known (right columns)

poisoned points found by the min–max-basic attack have high loss and consequently get filtered out by the loss defense.

5.7 Attacks on multi-class tasks with the min–max attack on MNIST

Finally, one advantage of the min–max attack is that it works on a variety of input domains $\mathcal{X} \times \mathcal{Y}$ and non-convex feasible sets $\mathcal{F}_\beta \subset \mathcal{X} \times \mathcal{Y}$, so long as we can still efficiently solve $\max_{(x,y)\in\mathcal{F}_\beta} \ell(\theta; x, y)$. In particular, for multi-class problems, the min–max attack can search over different choices of $\tilde{y}$ for each poisoned point $(\tilde{x}, \tilde{y})$, whereas the influence and KKT attacks require us to grid search over the relative proportion of the different classes. As we need $k(k-1)$ distinct points to carry out any data poisoning attack against a k-class SVM (Proposition 4 in "Appendix 1"), this grid search would take time exponential in k^2.

We illustrate a multi-class attack by running the min–max attack on the 10-class MNIST dataset (LeCun et al. 1998). Using $\epsilon = 3\%$ poisoned data, the min–max attack obtains 15.2% test error against the L2 defense and 13.7% test error against the loss defense, demonstrating a high-leverage attack in a multi-class setting (Fig. 8).

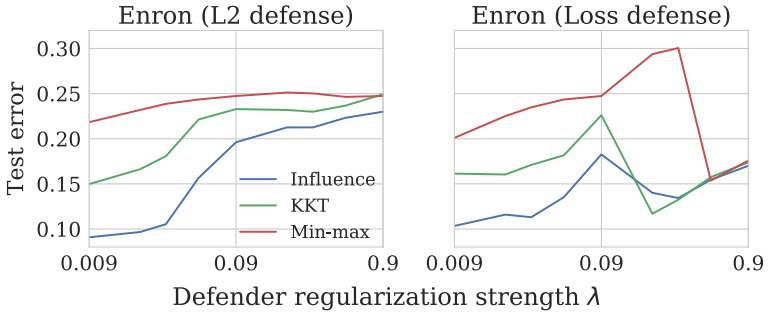


Fig. 10 The effect of changing the defender's regularization strength on the test error achieved by each attack. The attacker's regularization is fixed at $\lambda = 0.09$

6 Experiments: attackers with incomplete information

In Sect. 5, we saw that the influence, KKT, and min–max attacks are effective if the attacker has complete information about the model and defenses that the defender is using. In this section, we study transferability—are these attacks still effective when the attacker does not have complete information about the defender? Specifically, we use the Enron dataset to explore what happens when the attacker does not have knowledge of 1) the test set (i.e., they only see the train set); 2) the amount of regularization that the defender uses; 3) their optimization algorithm; and 4) their loss function.

In general, our attacks are still effective under these changes, with the min–max attack generally being the most robust and the influence attack being the least. However, the loss defense poses problems for all three attacks when the optimization algorithm or the loss function are changed, suggesting that attackers should set conservative thresholds against the loss defense.

6.1 Unobserved test data

An attacker might not have a specific test set in mind; for example, they might aim to make the defender incur high expected loss on test points from the same distribution as the training set. For such an attacker, optimizing specifically for some known test data $\mathcal{D}_{\text{test}}$ might not translate into an effective attack on a different sample of test data. We tested if our attacks would still be effective if the attacker only knew the clean training data $\mathcal{D}_c$ but not the test data $\mathcal{D}_{\text{test}}$.[4] Specifically, we generated attacks by simply using $\mathcal{D}_c$ in place of $\mathcal{D}_{\text{test}}$ (i.e., we optimized for high error on the training data $\mathcal{D}_c$).

Figure 9 shows the test error that the resulting attacks incurred on $\mathcal{D}_{\text{test}}$, compared with attacks that assume knowledge of $\mathcal{D}_{\text{test}}$, as used in the rest of the paper. Overall, the attacks still significantly increase test error even without knowing $\mathcal{D}_{\text{test}}$. However, the influence attack is comparatively less effective when the test dataset is not known (reaching 10.9% test error instead of 18.3% test error). In contrast, the KKT (16.5% vs. 22.6% test error) and min–max attacks (19.0% vs 23.7% test error) are more robust to using the training data

[4] A different setting is if the attacker knows the test data $\mathcal{D}_{\text{test}}$ but not the clean training data $\mathcal{D}_c$. For example, the attacker might only have access to a similarly distributed but distinct dataset $\mathcal{D}'_c$. As our attacks depend on the clean training data primarily through the average gradient of the loss computed over $\mathcal{D}_c$, we expect that swapping $\mathcal{D}_c$ with $\mathcal{D}'_c$ should not matter for sufficiently large training sets.

in place of the test data. These results suggest that the influence attack, which explicitly optimizes for loss on the test set, overfits more strongly to the test set compared to the KKT and min–max attacks, which rely on the test set mainly through the construction of decoy parameters.

A variant of the above setting is when the attacker does not know the exact test data $\mathcal{D}_{\text{test}}$ but nonetheless has access to some validation data $\mathcal{D}_{\text{val}}$ from the same distribution, as well as the training data $\mathcal{D}_{\text{c}}$. In this setting, the attacker could optimize an attack against $\mathcal{D}_{\text{val}}$; we expect that doing so will result in an attack that is more effective than optimizing against the training data $\mathcal{D}_{\text{c}}$, as we do above, though still slightly less effective than optimizing against $\mathcal{D}_{\text{test}}$ itself.

6.2 Regularization

Do attacks that are optimized for one level of regularization still work well at other levels of regularization? Recall that the defender uses L_2 regularization with the hyperparameter λ controlling the amount of regularization (Eq. (2)); in particular, we use $\lambda = 0.09$ for the Enron dataset (Table 1). To test the effect of the defender's choice of λ, we varied it from 0.009 to 0.9 while keeping the attacker's λ constant at 0.09. Figure 10 shows the results:

- Against the L2 defense, test error generally increased with L_2 regularization strength (Fig. 10-Left). The L2 feasible set depends only on the location of the poisoned points and not on the model parameters that the defender learns. Thus, the test error changes because the defender learns a different set of parameters given exactly the same set of poisoned points, and not because a different set of poisoned points get filtered out by the defenses.
- The amount of regularization had different effects on each attack's effectiveness against the L2 defense. The influence attack became less effective as we reduced defender regularization ($< 10\%$ test error against the L2 defense at $\lambda = 0.009$), while the min–max attack was robust to changes in defender regularization (22% test error at $\lambda = 0.009$).
- Increasing regularization can make the loss defense more effective (Fig. 10-Right). The loss feasible set is sensitive to changes in the model parameters that the defender learns, so some poisoned points that evaded this defense under the original model (with $\lambda = 0.09$) are now detected under the changed model.
- Unlike the other attacks, the min–max attack initially gets more effective against the loss defense as defender regularization is increased from $\lambda = 0.09$. We suspect that this is due to the min–max attack using a fixed loss threshold τ (see Eq. (22) in Sect. 4.3.2) that is more conservative than the KKT attack (which uses an adaptive threshold based on the quantiles of the loss under the decoy parameters) and the influence attack (which solely relies on concentrated attacks to overcome the loss defense).

These results imply that attackers should optimize for lower levels of regularization, in case the defender uses a lower level (and conversely, it suggests that defenders might want to use lower levels of regularization than they might otherwise). Attackers using decoy parameters might also decide to set their loss thresholds more conservatively.

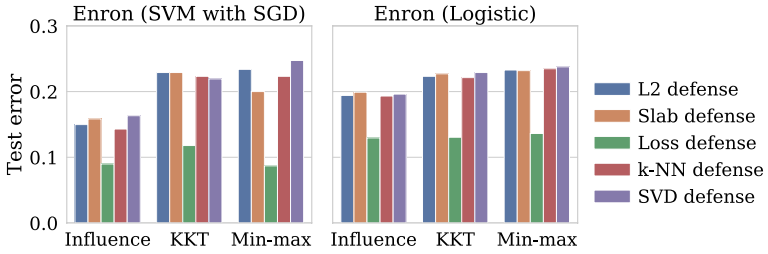


Fig. 11 With the significant exception of the loss defense, the attack results are robust to shifts in the optimization algorithm (from an exact solution to a single pass of stochastic gradient descent) and the loss function (from the hinge loss to the logistic loss)

These results also suggest that the data poisoning attacks are not exploiting overfitting. If that were the case, we would expect increasing regularization to decrease overfitting and thus reduce attack effectiveness. Instead, we observe the opposite: the defender generally suffers when increasing regularizatio, as it is harder for the defender's model to fit both the poisoned training points and clean training points well, and if the clean training points are not fit well, the test error will consequently increase.

We note that Demontis et al. (2019) studied the transferability of gradient-based data poisoning attacks for image recognition and found that attacks were more effective when the defender used less regularization. The differences from our work is that they assumed that there are no defenses (i.e., the only constraint on the attacker is to generate a valid image) and that there are a small number of training points relative to dimension (e.g., $n = 500$ for a binary MNIST classification problem).

6.3 Optimization and loss function

In our previous experiments, we assumed that the defender would learn the model parameters $\hat{\theta}$ that globally minimized training loss. In practice, defenders might use stochastic optimization and/or early stopping, leading to parameters $\hat{\theta}$ that are close to but not exactly at the optimum.

Figure 11-Left shows the results of our attacks on a defender that learned a model by doing a single pass of stochastic gradient descent over the dataset (i.e., each sample is looked at exactly once). We also tested how an attacker using the hinge loss would fare against a defender who uses the logistic loss (Fig. 11-Right). All three attacks stayed effective against all of the defenses except the loss defense, which managed to significantly reduce the damage inflicted by the attacker.

As in Sect. 6.2, these results suggest that attackers should use a conservative loss threshold to harden their attacks against loss-based defenses. It also suggests that the attacker's ability to evade loss-based defenses is especially sensitive (relative to other defenses) to getting the defender's model correct.

7 Related work

In this section, we discuss other attack settings and defense strategies that have been studied in the literature. For broad surveys on this topic, see e.g., Barreno et al. (2010), Biggio et al. (2014), Gardiner and Nagaraja (2016), Papernot et al. (2016b), and Vorobeychik and Kantarcioglu (2018).

7.1 Attacks

Label-flip attacks The attacks presented in this work control both the label $\tilde{y}$ and input features $\tilde{x}$ of the poisoned points. Attacks that only control the label $\tilde{y}$ are known as *label-flip attacks*. In a typical label-flip attack, the attacker gets to change some ϵ fraction of the labels y of the training set but is unable to change the features x (Biggio et al., 2011; Xiao et al., 2012, 2015). We experimented with a variant of the label flip attack described in Xiao et al. (2012), where we allowed the attacker to pick examples from the test set, flip their labels, and add them to the training set ("Appendix 4"). We found that this attack, though fast to run, was significantly less effective than our proposed attacks; control of $\tilde{x}$ seems to be necessary to carry out high-leverage attacks in the presence of data sanitization defenses.

Targeted versus indiscriminate attacks In our experiments, the attacker sought to increase error on a test set $\mathcal{D}_{\text{test}}$ that was drawn from the same distribution as the clean training data $\mathcal{D}_{\text{c}}$. This type of attack is known as an *indiscriminate* attack (Barreno et al., 2010) and is akin to a denial-of-service attack.

Indiscriminate attacks seek to change the predictions of the learned model on a good fraction of the entire data distribution and therefore require substantial changes to the model. This makes indiscriminate attacks statistically interesting, as they get at fundamental properties of the model: how might an attacker that only controls 1% of the training data bring about a 10% increase in test error?

A different type of attack is a *targeted* attack, in which the attacker seeks to cause errors on specific test examples or small sub-populations of test examples (Gu et al., 2017; Chen et al., 2017; Burkard & Lagesse, 2017; Koh & Liang, 2017; Shafahi et al., 2018; Suciu et al., 2018). For example, an attacker might seek to have all of the emails that they send marked as non-spam while leaving other emails unaffected; or an attacker might seek to cause a face recognition system to recognize their face as that of a particular victim's (as in Biggio et al. (2012a) and Biggio et al. (2013), which build off the attacks in Kloft and Laskov (2012)). Targeted attackers only seek to change the predictions of the model on a small number of instances, and therefore might be able to add in 50 poisoned training points to cause an error on a single test point (Shafahi et al., 2018). Targeted attacks are well-motivated from a security perspective: attackers might only care about a subset of the model's prediction, and targeted attacks require less control over the training set and are therefore easier to carry out.

The influence and KKT attacks in Sects. 4.1 and 4.2 do not make any assumptions on the nature of the test set $\mathcal{D}_{\text{test}}$, and can therefore handle the targeted attack setting without modification. In contrast, the min–max attack in Sect. 4.3 assumes that the training error is a good approximation to the test error, and is thus only appropriate in the indiscriminate attack setting.

Backdoor attacks A backdoor attack is a targeted attack that seeks to cause examples that contain a specific backdoor pattern, e.g., a bright sticker (Gu et al., 2017) or a particular type of sunglasses (Chen et al., 2017), to be misclassified. Backdoor attacks work by superimposing the chosen backdoor pattern onto particular training examples from a given class, which causes the model to associate the backdoor pattern with that class (and therefore misclassify, at test time, examples of a different class that also contain the backdoor pattern). The attackers in Gu et al. (2017) and Chen et al. (2017) do not need to know the model that the defender is using; in fact, the attacker in Chen et al. (2017) does not even need any knowledge of the training set, instead adding examples from a external dataset. These weaker assumptions on attacker capabilities are common in targeted attacks. In contrast, the indiscriminate attacks that we develop in this paper make use of knowledge of the model and the training set in order to have high leverage.

Clean-label attacks Clean-label attacks are attacks that "do not require control over the labeling function; the poisoned training data appear to be labeled correctly according to an expert observer" (Shafahi et al., 2018). Not requiring control over labeling makes it easier for the attacker to practically conduct such an attack, as the attacker only needs to introduce the unlabeled poisoned data into the general pool of data (e.g., uploading poisoned images or sending poisoned emails, as Shafahi et al., 2018 discusses) and wait for the defender to label and ingest the poisoned data.

The backdoor attacks discussed above are examples of clean-label attacks, provided the backdoor pattern is chosen to be innocuous enough to avoid human suspicion. Another way of constructing clean-label attacks is by constraining the poisoned points to be close, in some metric, to a clean training point of the same class; Shafahi et al. (2018) does this with the ℓ_2 norm, while Suciu et al. (2018) and Koh and Liang (2017) use the ℓ_∞ norm.

Our attacks are not clean-label attacks, in that the poisoned points will not necessarily be labeled as the correct class by a human expert. On the other hand, our poisoned points are designed to evade detection by automatic outlier detectors; note that "clean-label" points can fool human experts but still look like statistical outliers.

Adversarial examples and test-time attacks The bulk of recent research in machine learning security has focused on test-time attacks, where the attacker perturbs the test example to obtain a desired classification, leaving the training data and the model unchanged. This line of research was sparked by the striking discovery that test images could be perturbed in a visually-imperceptible way and yet fool state-of-the-art neural network image classifiers (Szegedy et al., 2014; Goodfellow et al. 2015; Carlini et al., 2016; Kurakin et al., 2016; Papernot et al., 2016a, 2017; Moosavi-Dezfooli et al., 2016). Designing models that are robust to such attacks, as well as coming up with more effective attacks, is an active area of research (Papernot et al., 2016c; Madry et al., 2017; Tramèr et al., 2017; Wong & Kolter, 2018; Raghunathan et al., 2018; Sinha et al., 2018; Athalye et al., 2018; Papernot & McDaniel, 2018).

In contrast to these test-time attacks, data poisoning attacks are train-time attacks: the attacker leaves the test example unchanged, and instead perturbs the training data so as to affect the learned model. Data poisoning is less well-studied, and compared to test-time attacks, it is harder to both attack and defend in the data poisoning setting: data poisoning attacks have to depend on the entire training set, whereas test-time attacks only depend on the learned parameters; and common defense techniques against test-time attacks, such as 'adversarial training' (Goodfellow et al., 2015), do not have analogues in the data poisoning setting.

7.2 Defenses

In the literature, effective defenses typically require additional information than the defenders we consider in this paper, e.g., having a labeled set of outliers or having a trusted dataset.

Using labeled outlier data and other training metadata If the defender has access to data that has been labeled as 'normal' versus 'outlier', then outlier detection can be treated as a standard supervised classification problem (Hodge & Austin, 2004). For example, an online retailer might have a set of transactions that had been previously labeled as fraudulent, and could train a separate outlier detection model to detect and throw out other fraudulent-looking transactions from the dataset. One drawback is that in an adversarial setting there is no assumption that future poisoned points might look like previous poisoned points. Such methods are therefore more suited for detecting outliers caused by natural noise processes rather than adversaries.

Instead of directly using 'normal' versus 'outlier' labels, defenders can instead rely on other types of training metadata. For example, Cretu et al. (2008)—which introduced the term 'data sanitization' in the context of machine learning—uses information on the time at which each training point was added to the training set. The intuition is that "in a training set spanning a sufficiently large time interval, an attack or an abnormality will appear only in small and relatively confined time intervals" (Cretu et al., 2008).

Using trusted data Other defense methods rely on having a trusted subset $\mathcal{T}$ of the training data that only contains clean data (obtained for example by human curation). One example is the Reject on Negative Impact (RONI) defense proposed by Nelson et al. (2008), which was one of the first papers studying data poisoning attacks and defenses. The RONI defense iterates over training points (x, y) and rejects points if the model learned on just the trusted data $\mathcal{T}$ is significantly different from the model learned on $\mathcal{T} \cup \{(x, y)\}$. Another example is the outlier detector introduced in Paudice et al. (2018), which operates similarly to our k-NN defense except that it measures distances only to the points in the trusted subset.

Having a trusted dataset makes it easier for the defender, though such a dataset might be expensive or even impossible to collect; if the defender has enough resources to collect a large amount of trusted data, then they can train a model on only the trusted data, solving the problem of data poisoning. The question of whether a small amount of trusted data is sufficient for defeating attackers while maintaining high overall performance (i.e., not rejecting clean training points that are not similar to the trusted data) is an open one. Finally, defenses that rely on trusted data are particularly vulnerable to attackers that manage to compromise the trusted data (e.g., through clean-label attacks that escape human notice).

High-dimensional robust estimation The theoretical computer science community has studied robust estimators in high dimensions, which seek to work well even in the presence of outliers (Kearns & Li, 1993). A key issue is that many traditional robust estimators incur a $\sqrt{d}$ increase in error in d dimensions. This theoretical insight aligns with the empirical results in this paper showing that it is often possible to attack classifiers with only a small fraction of poisoned data.

Motivated by these issues, Klivans et al. (2009) and later Awasthi et al. (2014) and Diakonikolas et al. (2017b) design robust classification algorithms that avoid the poor dimension-dependence of traditional estimators, although only under strong distributional assumptions such as log-concavity. Separately, Lai et al. (2016) and Diakonikolas

Table 2 Comparison of attacks. Enron test error is reported at $\epsilon = 3\%$, and the time taken is how long each attack took to reach 17% Enron test error

Attack	Enron test error (%)	Time taken	Pros and cons
Influence	18.3	286 min	+ No need to find θ_{decoy} − Slow − Less transferable − Does not handle loss defense
KKT	22.5	27s	+ θ_{decoy} handles loss defense
Min–max	23.7	28 min	+ θ_{decoy} handles loss defense + Handles multi-class setting − Assumes indiscriminate attack

et al. (2016) designed robust procedures for mean estimation, which again required strong distributional assumptions. Later work (Charikar et al. 2017; Diakonikolas et al. 2017a; Steinhardt et al. 2018) showed how to perform mean estimation under much more mild assumptions, and Diakonikolas et al. (2017a) showed that their procedure could yield robust estimates in practice. In recent concurrent work, Diakonikolas et al. (2018) showed that robust estimation techniques can be adapted to classification and used this to design a practical algorithm that appears more robust than many traditional alternatives. It would be interesting future work to attack this latter algorithm in order to better vet its robustness.

*Certified defenses*Steinhardt et al. (2017) explore the task of provably certifying defenses, i.e., computing a dataset-dependent upper bound on the maximum test loss that an attacker can cause the defender to incur. Their method—from which we adopt the min–max-basic attack—shows that the L2 and slab defenses are sufficient for defending models trained on the MNIST-1-7 and Dogfish datasets but cannot certifiably protect models trained on the Enron and IMDB datasets, which matches with our experimental results. Open questions are whether our improvements to the min–max-basic (e.g., decoy parameters) can be used in their framework to derive tighter upper bounds on attack effectiveness, and whether the other defenses (e.g., the loss defense) can be incorporated into their framework.

8 Discussion

In this paper, we developed three distinct attacks that could evade data sanitization defenses while significantly increasing test error on the Enron and IMDB datasets. The influence attack directly optimizes for increasing the test loss through gradient ascent on the poisoned points; the KKT attack chooses poisoned points to achieve pre-determined decoy parameters; and the min–max attack efficiently solves for the poisoned points that maximize train loss, as a proxy for test loss.

We summarize the relative merits of these attacks in Table 2. The influence attack is direct but slow, less effective against model-based defenses (such as the loss defense), and less robust. The KKT attack is much faster, at least in our setting where it can heavily exploit convexity; however, its reliance on decoy parameters is also a limitation, as our heuristic for generating decoy parameters might fail under more sophisticated defenses. The min–max attack shares the same reliance on decoy parameters and is slower than the KKT attack, and it only works in the indiscriminate setting, but it is more robust and can handle multi-class settings more efficiently.

We expect that more sophisticated data sanitization defenses could defeat the attacks developed in this paper, which did not account for them. However, these defenses might in turn be broken by attacks that are specifically geared for those defenses. The results in this paper show that data poisoning defenses need to be tested against attackers that are designed to evade them. We end by discussing some directions for future work.

What makes datasets and models attackable? The effectiveness of our attacks vary significantly from dataset to dataset (e.g., linear models trained on the Enron and IMDB datasets are more vulnerable than linear models trained on the MNIST-1-7 and Dogfish datasets). What conditions make certain models on certain datasets attackable, but not others? This is an open question; we speculate that linear models on the Enron and IMDB datasets are easier to attack because they have higher dimensionality and are less linearly separable, but this question deserves more study.

Non-convex models One limitation of our attacks is that they rely on the attacker and defender being able to find the model parameters θ that globally minimize the training loss. This is a reasonable assumption if the loss is convex, but most models in practice today are non-convex. Analysis of attack algorithms in the non-convex setting is substantially more difficult, e.g., a poisoning attack that works against a neural net trained with a given random seed might fail when the random seed is changed, or a single poisoned point might cause the defender to get stuck at a very bad local minimum. The attacks that we present in this paper can, at least empirically, be applied in some form to non-convex models. For example, in the influence attack, we can still move poisoned data points along the gradient of the test loss. It is an open question whether our attacks remain effective in the non-convex setting.

Strategies for stronger defenses How might we build stronger defenses that are robust against determined attackers? We outline several approaches, as well as potential difficulties.

One strategy would be to try to design better outlier detectors—perhaps the L2 and slab defenses provide too crude a measure of whether a point is realistic, and sophisticated generative models such as generative adversarial nets (Goodfellow et al., 2014) could better rule out poisoned data. We are skeptical of this approach, as there are many natural distributions (such as multivariate Gaussians) where even a perfect generative model cannot prevent an adversary from substantially skewing empirical statistics of the data (see Steinhardt, 2018, Section 1.3). The existence of adversarial test examples for image classifiers (Szegedy et al., 2014; Goodfellow et al., 2015) also weights against this approach, since such examples are generated using a method for inducing high loss under a target model. This method could likely be adapted for use in the min–max attack, as the the main sub-routine in that attack involves generating examples that induce high loss under a target model.

A different strategy is to learn multiple models on different (random or otherwise) subsets of data, in the hopes that the data will be relatively clean at least on some subsets (Fischler & Bolles, 1981; Kearns & Li, 1993; Cretu et al., 2008). In general, these methods are variants of loss-based defenses, in that they assume that poisoned points tend to be poorly fit by the model, and could therefore still be vulnerable to attacks that specifically target loss-based defenses. Especially with larger models and datasets, these methods also incur the additional computational cost from needing to fit multiple models.

Another strategy rests on the observation that if we could directly minimize the 0–1 test error (rather than using a convex proxy), then an adversary controlling an ϵ-fraction of the data could always induce at most additional $\frac{\epsilon}{1-\epsilon}$ error, at least on the training set.[5] The key issue with convex proxies for the 0/1-loss is that they are unbounded and so an adversary can cause the loss on $\mathcal{D}_{\text{p}}$ to be very large. One could perhaps do better by using non-convex but bounded proxies for the 0/1-loss, which would make optimization of the training loss more difficult, but might pay off with higher robustness. However, it also opens up a new avenue for attack—the attacker could try to push the learner towards a bad local minimum. There are also known hardness results for even approximately minimizing the 0/1-loss (Feldman et al., 2009; Guruswami & Raghavendra, 2009), but it is possible that they do not apply in practice.

Finally, as noted in Sect. 7, there is recent work seeking to design estimators that are *provably robust* to adversarial training data under suitable distributional assumptions. Diakonikolas et al. (2018) recently presented a practical implementation of these methods for classification and regression tasks and showed promising initial results. Their method iteratively removes points with outlying gradients and refits the model, and can be viewed as a more sophisticated version of an iterative loss-based defense; Liu et al. (2017) and Jagielski et al. (2018) also present similar iterative algorithms for the regression setting. We view provable security under a well-defined threat model as the gold standard, and encourage further work along this direction (see Li (2018) or Steinhardt (2018) for two recent overviews). There appears to be plenty of room both to improve the practical implementation of this family of defenses and to devise new theoretically-grounded procedures.

Appendix 1: How many distinct points are needed for data poisoning attacks?

Consider some attack $\mathcal{D}_{\text{p}}$ which makes a defender learn model parameters $\hat{\theta}$. Under what conditions does there exist some other attack $\mathcal{D}'_{\text{p}}$ that contains at most as many points ($|\mathcal{D}_{\text{p}}| \geq |\mathcal{D}'_{\text{p}}|$), but with fewer *distinct* points (i.e., $\mathcal{D}'_{\text{p}}$ contains repeated copies of points)?

If the attacker could place poisoned points at arbitrary locations, and if the model's loss function is unbounded (as is the case in most models, e.g., SVMs or logistic regression), then very few poisoned points (distinct or otherwise) are generally needed since the attacker can get high leverage over the model by placing a poisoned point far away. However, in our setting, the attacker is constrained to play poisoned points that are in the feasible set $\mathcal{F}$.

In this section, we provide a general method for finding the minimum number of distinct points necessary for achieving any attack, given a model with a strictly convex loss function and some feasible set $\mathcal{F}$. We show that for binary SVMs and logistic regression, if $\mathcal{F}$ is convex for each class—as is the case for the L2, slab, loss, and SVD—then only 2 distinct points are necessary.

As a high-level sketch, our proof proceeds as follows:

1. (Proposition 1) We consider the set of scaled feasible gradients $\mathcal{G}_{\hat{\theta}} \overset{\text{def}}{=} \{\alpha \nabla_{\theta} \ell(\hat{\theta}; x, y) : 0 \leq \alpha \leq 1, (x, y) \in \mathcal{F}\}$. This set corresponds to the gradients of all feasible points, scaled by some $0 \leq \alpha \leq 1$. We show that the number of distinct points

[5] To see this, note that the 0/1-loss of θ^* averaged across $\mathcal{D}_{\text{c}} \cup \mathcal{D}_{\text{p}}$ is at most at most ϵ larger than across $\mathcal{D}_{\text{c}}$, so any $\hat{\theta}$ outperforming θ^* can only have slightly higher loss than $\hat{\theta}$ across $\mathcal{D}_{\text{c}}$.

needed for an attack relates to the geometry of this set $\mathcal{G}_{\hat{\theta}}$. In particular, if $\mathcal{G}_{\hat{\theta}}$ is convex for each class, then only 2 points are needed.
2. (Proposition 2) We check that for SVMs, $\mathcal{G}_{\hat{\theta}}$ is convex for each class if the original feasible set $\mathcal{F}$ is convex for each class.
3. (Proposition 3) More generally, we establish conditions under which differentiable margin-based losses have $\mathcal{G}_{\hat{\theta}}$ convex for each class, and we show that logistic regression satisfies these conditions.

For convenience, in the sequel we will assume that these models are trained by finding

$$\hat{\theta} = \underset{\theta \in \Theta}{\arg\min} \ \frac{\lambda}{2}\|\theta\|_2^2 + \sum_{(x,y)\in\mathcal{D}_c} \ell(\theta;x,y) + \sum_{(\tilde{x},\tilde{y})\in\mathcal{D}_p} \ell(\theta;\tilde{x},\tilde{y}). \tag{23}$$

In other words, the degree of regularization is not explicitly affected by the total number of data points $|\mathcal{D}_c| + |\mathcal{D}_p|$. Moreover, the overall loss is strictly convex due to regularization, even if $\ell(\theta;x,y)$ is only convex (and not strictly convex) in θ, as is the case with the hinge loss. We also assume that $\mathcal{D}_p \subseteq \mathcal{F}$ (otherwise, the poisoned points will simply be thrown out by the defender).

We start by establishing the equivalence between the number of distinct points needed to poison a given model and the geometry of the set of feasible gradients of that model.

Definition 1 The Carathéodory number of a set $\mathcal{G} \subseteq \mathbb{R}^n$ is the smallest number c such that each $\tilde{g} \in \text{conv}(\mathcal{G})$ can be written as a convex combination of at most c points in $\mathcal{G}$. (Each $\tilde{g}$ may be a convex combination of a different set of c points.)

Proposition 1 *Consider a defender who learns a model by first discarding all points outside a fixed feasible set $\mathcal{F}$, and then finding the parameters that minimize a strictly convex loss $\ell(\theta;x,y)$ averaged over the training set. If a parameter $\hat{\theta}$ is attainable by any set of $\tilde{n}$ poisoned points $\mathcal{D}_p = \{(\tilde{x}_1,\tilde{y}_1),\ldots,(\tilde{x}_{\tilde{n}},\tilde{y}_{\tilde{n}})\} \subseteq \mathcal{F}$, then there exists a set $\tilde{\mathcal{D}}_p$ that also attains $\hat{\theta}$ with at most $\tilde{n}$ poisoned points but only contains c distinct points (with a potentially fractional number of repeats of each point), where c is the Carathéodory number of the set of possible scaled gradients $\mathcal{G}_{\hat{\theta}} \overset{\text{def}}{=} \{\alpha\nabla_\theta \ell(\hat{\theta};x,y) : 0 \le \alpha \le 1, (x,y) \in \mathcal{F}\}$.*

Remark 2 If $\ell(\theta;x,y)$ is not differentiable in θ, we obtain an equivalent result by considering the subgradient sets $\partial_\theta \ell$. In this case, we can define $\mathcal{G}_{\hat{\theta}} \overset{\text{def}}{=} \{\alpha g : 0 \le \alpha \le 1, g \in \bigcup_{(x,y)\in\mathcal{F}} \partial_\theta \ell(\hat{\theta};x,y)\}$. For clarity in the following proof, we will assume that $\ell(\theta;x,y)$ is differentiable, but the argument for the non-differentiable case is almost identical.

Proof Assume that we are given a set of n clean training points $\mathcal{D}_c$ and a set of $\tilde{n}$ poisoned points $\mathcal{D}_p$. Without loss of generality, we assume that each poisoned point $(\tilde{x},\tilde{y}) \in \mathcal{D}_p$ lies in the feasible set $\mathcal{F}$ (otherwise, the poisoned point will be filtered out and have no effect).

The defender learns parameters $\hat{\theta}$ that minimize the training loss

$$\frac{\lambda}{2}\|\theta\|_2^2 + \sum_{(x,y)\in\mathcal{D}_c} \ell(\theta;x,y) + \sum_{(\tilde{x},\tilde{y})\in\mathcal{D}_p} \ell(\theta;\tilde{x},\tilde{y}). \tag{24}$$

Since $\hat{\theta}$ is a minimum of the loss, we have that

$$0 = \lambda\hat{\theta} + \sum_{(x,y)\in\mathcal{D}_c} \nabla_\theta \ell(\hat{\theta};x,y) + \sum_{(\tilde{x},\tilde{y})\in\mathcal{D}_p} \nabla_\theta \ell(\hat{\theta};\tilde{x},\tilde{y}), \tag{25}$$

where $\nabla_\theta \ell(\hat{\theta};x,y)$ denotes the gradient of the loss at the point (x, y) with parameters $\hat{\theta}$.

Our goal is to find the minimum number of distinct points c such that given any clean dataset $\mathcal{D}_c$, attack $\mathcal{D}_p$, and consequent model parameters $\hat{\theta}$, we can find some other attack $\tilde{\mathcal{D}}_p$ with at most c distinct points such that $\tilde{\mathcal{D}}_p$ also makes the defender learn $\hat{\theta}$. The key observation is that because the loss is strictly convex (by assumption), (25) is a necessary and sufficient condition for the defender to learn $\hat{\theta}$. In particular, if we define $g \overset{\text{def}}{=} \sum_{(\tilde{x},\tilde{y})\in\mathcal{D}_p} \nabla_\theta \ell(\hat{\theta};\tilde{x},\tilde{y})$, then any other attack $\tilde{\mathcal{D}}_p$ that satisfies $g = \sum_{(\tilde{x},\tilde{y})\in\tilde{\mathcal{D}}_p} \nabla_\theta \ell(\hat{\theta};\tilde{x},\tilde{y})$ —note that we have $\tilde{\mathcal{D}}_p$ in place of $\mathcal{D}_p$—will also make the defender learn $\hat{\theta}$.

What values can g take on? Since $g \overset{\text{def}}{=} \sum_{(\tilde{x},\tilde{y})\in\mathcal{D}_p} \nabla_\theta \ell(\hat{\theta};\tilde{x},\tilde{y})$ by construction, and each point $(\tilde{x},\tilde{y})$ in $\mathcal{D}_p$ lies in the feasible set $\mathcal{F}$, we know that the normalized vector $\tilde{g} \overset{\text{def}}{=} g/\tilde{n}$ (where $\tilde{n}$ is the number of points in $\tilde{\mathcal{D}}_p$) is contained in the convex hull $\text{conv}(\mathcal{G}_{\hat{\theta}})$, where $\mathcal{G}_{\hat{\theta}}$ is the set of scaled gradients that are possible in the feasible set, $\mathcal{G}_{\hat{\theta}} \overset{\text{def}}{=} \{\alpha\nabla_\theta \ell(\hat{\theta};x,y) : 0 \le \alpha \le 1, (x,y) \in \mathcal{F}\}$.[6]

Now, say we can find c points $g_1, g_2, \ldots, g_c \in \mathcal{G}_{\hat{\theta}}$ such that $\tilde{g}$ is a convex combination of these points (i.e., $\tilde{g} \in \gamma_1 g_1 + \gamma_2 g_2 + \cdots + \gamma_k g_c$, with $\gamma_i \ge 0$ and $\sum_i \gamma_i = 1$). Since each point $g_i \in \mathcal{G}_{\hat{\theta}}$ can be written as $\alpha_i \nabla_\theta \ell(\hat{\theta};\tilde{x}_i,\tilde{y}_i)$ for some $(\tilde{x}_i,\tilde{y}_i) \in \mathcal{F}$ with $0 \le \alpha_i \le 1$, we can construct a dataset $\tilde{\mathcal{D}}_p$ comprising $\alpha_1\gamma_1\tilde{n}$ copies of $(\tilde{x}_1,\tilde{y}_1)$, $\alpha_2\gamma_2\tilde{n}$ copies of $(\tilde{x}_2,\tilde{y}_2)$, and so on, such that $g = \tilde{n}\tilde{g} = \tilde{n}\sum_{i=1}^{c} \alpha_i\gamma_i \nabla_\theta \ell(\hat{\theta};\tilde{x}_i,\tilde{y}_i)$. This constructed dataset $\tilde{\mathcal{D}}_p$ will therefore attain $\hat{\theta}$ with only c distinct points, and since $\sum_{i=1}^{c} \alpha_i\gamma_i \le 1$, the total weight of all of these points will be less than or equal to $\tilde{n}$.

We thus want to find the smallest number c such that for any $\tilde{g} \in \text{conv}(\mathcal{G}_{\hat{\theta}})$, we can write $\tilde{g}$ as a convex combination of at most c points in $\mathcal{G}_{\hat{\theta}}$. This is the definition of the Carathéodory number of $\mathcal{G}_{\hat{\theta}}$, as desired.

□

Proposition 1 tells us that to find the number of distinct points required for data poisoning attacks on a given model and feasible set, it suffices to find the Carathéodory number of the set of feasible gradients of that model. Finding the Carathéodory number of a set is a well-studied problem (see, e.g., Bárány and Karasev (2012), or Mirrokni et al. (2015) for an approximate version of the problem). In our setting, each feasible set can be written as the union of a small number of convex sets, which simplifies the analysis of its Carathéodory number. We start by establishing the following lemma:

Lemma 1 *If a set $\mathcal{G}$ is the union of k convex sets, $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2 \cup \cdots \mathcal{G}_k$ where each $\mathcal{G}_i$ is convex, then the Carathéodory number of $\mathcal{G}$ is at most k.*

[6] g is actually contained in the convex hull of the unscaled gradient set $\{\nabla_\theta \ell(\hat{\theta};x,y) : (x,y) \in \mathcal{F}\}$, which is a subset of the scaled gradient set, so the above proof also goes through if we consider the unscaled gradient set in place of the scaled gradient set. However, as we will see later in this section, adding the α scaling term reduces the Carathéodory number, which gives a stronger result.

Proof Pick any $\tilde{g} \in \text{conv}(\mathcal{G})$. By construction, we can write $\tilde{g} = \sum_{i=1}^{k} \sum_{j=1}^{n_i} \alpha_{ij} x_{ij}$ where $x_{ij} \in \mathcal{G}_i$, $\alpha_{ij} \geq 0$, and $\sum_i \sum_j \alpha_{ij} = 1$. Since the $\mathcal{G}_i$ are convex sets, we can find $\tilde{x}_i \in \mathcal{G}_i \subseteq \mathcal{G}$ such that $\tilde{x}_i = \sum_{j=1}^{n_i} \alpha_{ij} x_{ij} / \sum_{j=1}^{n_i} \alpha_{ij}$, allowing us to write $\tilde{g} = \sum_{i=1}^{k} \left(\sum_{j=1}^{n_i} \alpha_{ij} \right) \tilde{x}_i$. Since any $\tilde{g} \in \text{conv}(\mathcal{G})$ can be written as the convex combination of at most k points in $\mathcal{G}$, the Carathéodory number of $\mathcal{G}$ is k.

□

We use this lemma to establish the Carathéodory number of the set of scaled gradients for a binary SVM.

Proposition 2 *Consider the setting of Proposition 1, and let the loss function be the ℓ_2 -regularized hinge loss on data with binary labels. Suppose that for each class $y = -1, +1$, the feasible set $\mathcal{F}_y \stackrel{\text{def}}{=} \{x : (x, y) \in \mathcal{F}\}$ is a convex set. Then the Carathéodory number of $\mathcal{G}_{\hat{\theta}} \stackrel{\text{def}}{=} \{\alpha g : 0 \leq \alpha \leq 1, g \in \bigcup_{(x,y)\in\mathcal{F}} \partial_\theta \ell(\hat{\theta}; x, y)\}$ is at most 2, independent of $\hat{\theta}$.*

Proof Recall that in a binary SVM, the loss on an individual point is given by

$$\ell(\theta; x, y) = \max(0, 1 - y\theta^\top x). \tag{26}$$

For convenience, we have folded the regularization term into the loss on each point.

From Proposition 1, we want to find the Carathéodory number of the set of all possible scaled (sub)gradients of poisoned points $\mathcal{G} \stackrel{\text{def}}{=} \{\alpha g : 0 \leq \alpha \leq 1, g \in \bigcup_{(x,y)\in\mathcal{F}} \partial_\theta \ell(\hat{\theta}; x, y)\}$. For a binary SVM, the subgradient sets are:

$$\partial_\theta \ell(\hat{\theta}; x, y) = \begin{cases} \{0\}, & \text{if } y\hat{\theta}^\top x > 1 \\ \{-\gamma y x : 0 \leq \gamma \leq 1\} & \text{if } y\hat{\theta}^\top x = 1 \\ \{-yx\} & \text{if } y\hat{\theta}^\top x < 1. \end{cases} \tag{27}$$

Plugging this into the expression for $\mathcal{G}_{\hat{\theta}}$, we get that

$$\mathcal{G}_{\hat{\theta}} = \{-\alpha y x : 0 \leq \alpha \leq 1, (x, y) \in \mathcal{F}, y\hat{\theta}^\top x \leq 1\}, \tag{28}$$

which we can rewrite as the union of two convex sets, one for each class:

$$\mathcal{G}_{\hat{\theta}} = \{-\alpha x : 0 \leq \alpha \leq 1, (x, +1) \in \mathcal{F}_{+1}, \hat{\theta}^\top x \leq 1\} \cup \{\alpha x : 0 \leq \alpha \leq 1, (x, -1) \in \mathcal{F}_{-1}, -\hat{\theta}^\top x \leq 1\}.$$

By Lemma 1, the Carathéodory number of $\mathcal{G}_{\hat{\theta}}$ is at most 2, regardless of what $\hat{\theta}$ is.

□

More generally, we can bound the Carathéodory number of the set of scaled gradients for a particular class of convex, differentiable, margin-based losses.

Proposition 3 (Carathéodory number of $\mathcal{G}$ for margin-based losses) *Consider the setting of Proposition 1 on binary data, and let the loss function be $\ell(\theta; x, y) = c(-y\theta^\top x)$, where $c : \mathbb{R} \to \mathbb{R}$ is a convex, monotone increasing, and twice-differentiable function. Suppose that the ratio of the second to the first derivative of c, c''/c' is a monotone non-increasing function, and that for each class $y = -1, +1$, the feasible set $\mathcal{F}_y \stackrel{\text{def}}{=} \{x : (x, y) \in \mathcal{F}\}$ is a convex set. Then the Carathéodory number of $\mathcal{G}_{\hat{\theta}} \stackrel{\text{def}}{=} \{\alpha \nabla_\theta \ell(\hat{\theta}; x, y) : 0 \leq \alpha \leq 1, (x, y) \in \mathcal{F}\}$ is at most 2, independent of $\hat{\theta}$.*

Proof We can write $\mathcal{G}_{\hat{\theta}}$ as the union of two sets $\mathcal{G}_{\hat{\theta},+1}$ and $\mathcal{G}_{\hat{\theta},-1}$, one for each class, where $\mathcal{G}_{\hat{\theta},y} \stackrel{\text{def}}{=} \{\alpha \nabla_\theta \ell(\hat{\theta};x,y) : 0 \le \alpha \le 1, x \in \mathcal{F}_y\}$. To show that the Carathéodory number of $\mathcal{G}_{\hat{\theta}}$ is at most 2, it suffices to show that each class set $\mathcal{G}_{\hat{\theta},y}$ is convex and then apply Lemma 1.

Pick $\hat{\theta}$ arbitrarily and fix y to be -1 or $+1$. To check that $\mathcal{G}_{\hat{\theta},y}$ is convex, it suffices to check that for each possible choice of x_1 and x_2 in $\mathcal{F}_y$ and $0 \le \gamma \le 1$, there exists some $\tilde{x} \in \mathcal{F}_y$ and $0 \le \alpha \le 1$ such that

$$\alpha \nabla_\theta \ell(\hat{\theta};\tilde{x},y) = \gamma \nabla_\theta \ell(\hat{\theta};x_1,y) + (1-\gamma)\nabla_\theta \ell(\hat{\theta};x_2,y). \tag{29}$$

Since our loss function has the form $\ell(\theta;x,y) = c(-y\theta^\top x)$, we have that

$$\nabla_\theta \ell(\hat{\theta};x,y) = c'(-y\theta^\top x)(-yx), \tag{30}$$

where c' is the derivative of c. Substituting this into (29) and cancelling out the $-y$ terms on both sides gives us the equivalent condition

$$\alpha c'(-y\theta^\top \tilde{x})\tilde{x} = \gamma c'(-y\theta^\top x_1)x_1 + (1-\gamma)c'(-y\theta^\top x_2)x_2. \tag{31}$$

To satisfy the above condition, we will take

$$\tilde{x} = \frac{\gamma c'(-y\theta^\top x_1)}{\gamma c'(-y\theta^\top x_1) + (1-\gamma)c'(-y\theta^\top x_2)}x_1 + \frac{(1-\gamma)c'(-y\theta^\top x_2)}{\gamma c'(-y\theta^\top x_1) + (1-\gamma)c'(-y\theta^\top x_2)}x_2, \tag{32}$$

which in turn implies that

$$\alpha = \frac{\gamma c'(-y\theta^\top x_1) + (1-\gamma)c'(-y\theta^\top x_2)}{c'(-y\theta^\top \tilde{x})}. \tag{33}$$

Since $\tilde{x}$ is a convex combination of x_1 and x_2, the convexity of $\mathcal{F}_y$ implies that $\tilde{x} \in \mathcal{F}_y$, so it remains to check that $0 \le \alpha \le 1$.[7] Moreover, since c is monotone increasing by assumption, c' is positive, and therefore α is positive. It remains to check that $\alpha \le 1$.

First, note that $\log c'(\cdot)$ is a concave function, as its derivative c''/c' is monotone non-increasing by assumption. For notational convenience, let $s_1 \stackrel{\text{def}}{=} -y\theta^\top x_1$ and $s_2 \stackrel{\text{def}}{=} -y\theta^\top x_2$, and let $T \stackrel{\text{def}}{=} \gamma c'(s_1) + (1-\gamma)c'(s_2)$. We then have that

$$\begin{aligned}
\log c'(-y\theta^\top \tilde{x}) &= \log c'\left(\frac{\gamma c'(s_1)s_1 + (1-\gamma)c'(s_2)s_2}{T}\right) \\
&\ge (\gamma c'(s_1)/T)\log c'(s_1) + ((1-\gamma)c'(s_2)/T)\log c'(s_2) \text{ (Jensen's)} \\
&= (\gamma c'(s_1)/T)\log \frac{\gamma c'(s_1)/T}{\gamma} + ((1-\gamma)c'(s_2)/T)\log \frac{(1-\gamma)c'(s_2)/T}{\gamma} + \log T \\
&\ge \log T \text{ (non-negativity of KL divergence)}
\end{aligned}$$

Exponentiating both sides and rearranging gives us

$$\alpha = \frac{T}{c'(-y\theta^\top \tilde{x})} \le 1.$$

[7] Note that since $\alpha c'(-y\theta^\top \tilde{x})$ is a scalar, so for (31) to hold, $\tilde{x}$ needs to point in the same direction as $\gamma c'(-y\theta^\top x_1)x_1 + (1-\gamma)c'(-y\theta^\top x_2)x_2$. Moreover, since we need $\tilde{x}$ to be in $\mathcal{F}_y$, we want $\tilde{x}$ to be a convex combination of x_1 and x_2. This choice of $\tilde{x}$ is the only choice that satisfies these two considerations.

The above argument shows that $\mathcal{G}_{\hat{\theta},y}$ is a convex set. Since we picked $\hat{\theta}$ and y arbitrarily, we can apply Lemma 1 to conclude that $\mathcal{G}_{\hat{\theta}} = \mathcal{G}_{\hat{\theta},+1} \cup \mathcal{G}_{\hat{\theta},-1}$ has Carathéodory number at most 2 regardless of $\hat{\theta}$. □

Corollary 1 (Carathéodory number of $\mathcal{G}$ for logistic regression) *Consider a logistic regression model in the setting of Proposition 3, where* $\ell(\theta;x,y) = \log(1+\exp(-y\theta^\top x))$. *Then the Carathéodory number of* $\mathcal{G}_{\hat{\theta}} \stackrel{\text{def}}{=} \{\alpha\nabla_\theta \ell(\hat{\theta};x,y) : 0 \le \alpha \le 1, (x,y) \in \mathcal{F}\}$ *is at most 2, independent of* $\hat{\theta}$.

Proof In logistic regression, we have that $c(v) = \log(1+\exp(v))$; $c'(v) = \sigma(v)$ where σ is the sigmoid function, $\sigma(v) \stackrel{\text{def}}{=} \frac{1}{1+\exp(-v)}$; and $c''(v) = \sigma(v)(1-\sigma(v))$. Thus, $\frac{c''}{c'}$ is a monotone decreasing function, and c is convex, monotone increasing, and twice-differentiable, so Proposition 3 applies. □

We can collect all of the above results into the following theorem, which appears in the main text.

Theorem 1(2 points suffice for 2-class SVMs and logistic regression) Consider a defender that learns a 2-class SVM or logistic regression model by first discarding all points outside a fixed feasible set $\mathcal{F}$ and then minimizing the average (regularized) training loss. Suppose that for each class $y = -1, +1$, the feasible set $\mathcal{F}_y \stackrel{\text{def}}{=} \{x : (x,y) \in \mathcal{F}\}$ is a convex set. If a parameter $\hat{\theta}$ is attainable by any set of $\tilde{n}$ poisoned points $\mathcal{D}_\text{p} = \{(\tilde{x}_1, \tilde{y}_1), \ldots, (\tilde{x}_{\tilde{n}}, \tilde{y}_{\tilde{n}})\} \subseteq \mathcal{F}$, then there exists a set of at most $\tilde{n}$ poisoned points $\tilde{\mathcal{D}}_\text{p}$ (possibly with fractional copies) that also attains $\hat{\theta}$ but only contains 2 distinct points, one from each class.

More generally, the above statement is true for any margin-based model with loss of the form $\ell(\theta;x,y) = c(-y\theta^\top x)$, where $c : \mathbb{R} \to \mathbb{R}$ is a convex, monotone increasing, and twice-differentiable function, and the ratio of second to first derivatives c''/c' is monotone non-increasing.

Proof From Proposition 2 (SVMs), Proposition 3 (margin-based losses), and Corollary 1 (logistic regression), we have that the Carathéodory number of $\mathcal{G}_{\hat{\theta}}$ (the set of scaled possible gradients) for each of these models is at most 2, regardless of $\hat{\theta}$. By Proposition 1, we conclude that only 2 distinct points are necessary. In particular, since $\mathcal{G}_{\hat{\theta}}$ can be represented in each of these cases as the union of two convex sets, one for each class, we need 1 distinct point from each class to realize any data poisoning attack. □

The general approach of finding the Carathéodory number of the set of scaled possible gradient can be applied to other models beyond those that we consider in this paper. As one example, we can extend the above approach to the setting of a multi-class SVM:

Proposition 4 ($k(k-1)$ distinct points suffice for a k-class SVM) *Consider the setting of Proposition 2, but with a k-class SVM. If a parameter* $\hat{\theta}$ *is attainable by any set of* $\tilde{n}$ *poisoned points* $\mathcal{D}_\text{p} = \{(\tilde{x}_1, \tilde{y}_1), \ldots, (\tilde{x}_{\tilde{n}}, \tilde{y}_{\tilde{n}})\} \subseteq \mathcal{F}$, *then there exists a set of at most* $\tilde{n}$ *poisoned*

points $\tilde{\mathcal{D}}_p$ that also attains $\hat{\theta}$, but that only has $k-1$ distinct values of $\tilde{x}$ for each distinct $\tilde{y}$, for a total of $k(k-1)$ distinct points.

Proof We follow the multi-class SVM formulation described in Crammer and Singer (2002), which has parameters $\theta = [\theta_1;\theta_2;\ldots;\theta_k] \in \mathbb{R}^{kd}$ for each class $y = 1, 2, \ldots, k$, where d is the dimensionality of the feature space $\mathcal{X}$. The loss on an individual point is given by

$$\ell(\theta;x,y) = \max(0, 1 - \theta_y^\top x + \max_{i \neq y} \theta_i^\top x). \tag{34}$$

This reduces to the above formulation for a binary (2-class) SVM by setting $\theta_{-1} = -\theta_{+1}$.

Let $i_{x,y} = \arg\max_{i \neq y} \theta_i^\top x$, and let $x^{(i)} = [\underbrace{0,\ldots,0}_{(i-1)d \text{ zeroes}}, x, \underbrace{0,\ldots,0}_{(n-i)d \text{ zeroes}}]$. Without loss of generality, we can ignore subgradients and take $\nabla_\theta \ell(\theta;x,y) = -x^{(y)} + x^{(i_{x,y})}$, following a similar argument to that used in Proposition 2.

Define $\mathcal{G}_{i,j} = \{-\alpha x^{(i)} + \alpha x^{(j)} : 0 \leq \alpha \leq 1, (x,i) \in \mathcal{F}\}$. We have that $\mathcal{G}_{\hat{\theta}} = \bigcup_{i=1}^k \bigcup_{j \neq i}^k \mathcal{G}_{i,j}$, and since each $\mathcal{G}_{i,j}$ is a convex set, by Lemma 1, the Carathéodory number of $\mathcal{G}$ is at most $k(k-1)$. □

Appendix 2: Attack implementation details

Appendix 2.1: The influence attack

The following details apply to both the influence-basic and influence attacks.

Smoothed hinge loss In our setting, the loss $\ell(\theta;x,y)$ is the hinge loss $\max(0, 1 - y\theta^\top x)$. One issue is that this loss is piecewise linear, which means that its gradient is zero or one, and its Hessian zero, almost everywhere. As a result, the gradient of the test loss L w.r.t. $\tilde{x}$ —which involves an inverse-Hessian-gradient-product, as in (10)—gives a poor indication of which directions to perturb $\tilde{x}$, and can easily get stuck. To mitigate this problem, we follow Koh and Liang (2017) and smooth the hinge loss for the purposes of computing the gradient (the actual training of the model is still done using the hinge loss). Specifically, we replace it with the function $\ell_{\text{smooth}}(\theta;x,y) = \delta \log(1 + \exp(\frac{1-y\theta^\top x}{\delta}))$ for some small δ. The function ℓ_{smooth} converges to ℓ as $\delta \to 0$, but has derivatives of all orders whenever $\delta > 0$. We note that there are other ways to get around the non-differentiability of the hinge loss. For example, Biggio et al. (2012b), Mei and Zhu (2015b), and their subsequent work make use of the dual formulation of SVMs to derive the required gradient, under the assumption that the set of support vectors (defined as training points that are exactly at the hinge) remain unchanged by the gradient update. For our purposes, we expect that such methods would perform similarly if the step size of the gradient update is small enough.

Computing inverse Hessian-vector products To efficiently compute $\frac{\partial \hat{\theta}}{\partial \tilde{x}}$ in (10), we follow Koh and Liang (2017) and use a combination of fast Hessian-vector products (Pearlmutter 1994) and a conjugate gradient solver (Martens, 2010); see also Agarwal et al. (2016) and Muñoz-González et al. (2017) for other tractable approaches. We select the step size η by trying a range of options and selecting the best-performing one on the test set (since, in our setting, the attacker knows the test set in advance).

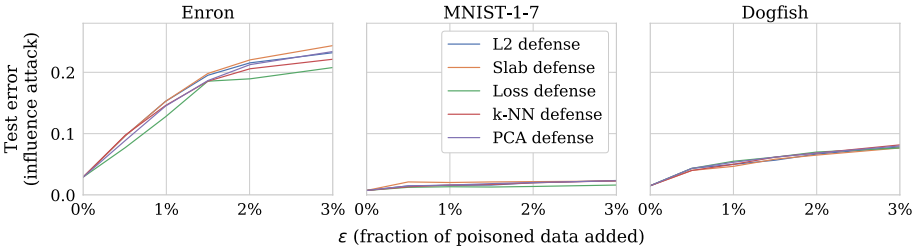


Fig. 12 The influence attack successfully drives test error on the Enron dataset from 3 to 23% with just $\epsilon = 3\%$ poisoned data, and on the Dogfish dataset from 1 to 8%, but does not manage to significantly affect the MNIST-1-7 dataset

Choosing the labels of poisoned points For the influence-basic attack, we initialize both $\tilde{x}_i$ and $\tilde{y}_i$ through random label flips on the training set. In other words, we select ϵn data points from the clean data $\mathcal{D}_c$ uniformly at random, with replacement, to flip and add to the poisoned data $\mathcal{D}_p$. We only consider data points that would still lie in $\mathcal{F}_\beta$ after the label flip. For the influence attack, there is only one distinct positive poisoned point $\tilde{x}_+$ and one distinct negative poisoned point $\tilde{x}_-$. We weight these two points inversely proportionally to the class balance: i.e., if there are P positive and N negative points in $\mathcal{D}_c$, then we place $\frac{N}{P+N} \cdot \epsilon$ weight on $(\tilde{x}_+, 1)$ and $\frac{P}{P+N} \cdot \epsilon$ weight on $(\tilde{x}_-, -1)$. This maintains the same class balance, on average, as the label flip initialization for the influence-basic attack.

Appendix 2.2: The KKT attack

Generating decoy parameters For each dataset, we first generated candidate decoy parameters as in Sect. 4.2.1—adding r copies of each test point whose flipped label has loss greater than γ. For Enron, we swept over $r \in \{1, 2, 3, 5, 8, 12, 18, 25, 33\}$ and γ set to the qth quantile of the loss (over the flipped test set), for $q \in \{0.05, 0.10, \ldots, 0.55\}$. This yielded 99 candidates θ_{decoy}; for efficiency we removed all parameters that had lower test error and higher training loss than some other candidate θ'_{decoy}. This left us with 48 parameters total. For IMDB, we applied a similar procedure but took $r \in \{1, 2, 3, 4, 5\}$ and $q \in \{0.1, 0.2, \ldots, 0.6\}$; this yielded 18 candidates after pruning (we sought fewer candidates for IMDB because it is bigger and slower to attack).

Choosing the labels of poisoned points Given the fraction of poisoned data $\epsilon = 3\%$, for each set of decoy parameters, we grid searched over 7 different ratios of positive versus negative poisoned points, ranging from $\epsilon_+ = 3\%, \epsilon_- = 0\%$ to $\epsilon_+ = 0\%, \epsilon_- = 3\%$.

Appendix 2.3: The min–max attack

Loss defense threshold To evade the loss defense, we used the threshold $\tau = 0.25$ across all experiments. The results were fairly robust to this choice of threshold.

Multi-class training For the MNIST dataset, we trained the multi-class SVM with AdaGrad (Duchi et al., 2010), with a batch size of 20, step size $\eta = 0.02$, and 3 passes over the training data.

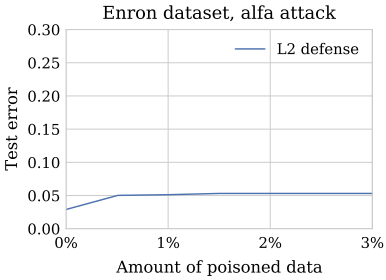


Fig. 13 Results of the alfa attack on the Enron dataset and L2 defense

Appendix 3: Experiments on MNIST-1-7 and Dogfish

In this section, we study the effectiveness of the influence attack on the following two datasets:

1. The MNIST-1-7 image dataset (LeCun et al., 1998), which requires each input feature to lie within the interval [0, 1] (representing normalized pixels). It is derived from the standard 10-class MNIST dataset by taking just the images labeled '1' or '7'. It is easily linearly separable: an SVM achieves 0.7% error on the clean data.
2. The Dogfish image dataset (Koh & Liang, 2017), which has no input constraints; its features are neural network representations, which for the purposes of this paper we allow to take any value in $\mathbb{R}^d$. Compared to the MNIST-1-7 dataset (where $n \gg d$), the Dogfish dataset has $n \approx d$, allowing attackers to potentially exploit overfitting. The Dogfish dataset also has a low base error of 1.3%.

As discussed in Sect. 5, Steinhardt et al. (2017) showed that the L2 and slab defenses are certifiably effective at defending the MNIST-1-7 dataset, and to a lesser extent the Dogfish dataset, for low values of ϵ. The results in Fig. 12 are consistent with this: with $\epsilon = 3\%$, the influence attack increases Dogfish test error from 1 to 8% and does not appreciably affect MNIST-1-7 test error. In contrast, it drives test error on the Enron dataset from 3 to 23%.

Appendix 4: Label flip attacks

Label flip attacks are a popular strategy in the literature for executing data poisoning attacks (Biggio et al., 2011; Xiao et al., 2012, 2015). In a label flip attack, the attacker is given a set of real data $\mathcal{D}_{\text{pool}} = \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^{n_{\text{pool}}}$. To form the poisoned data $\mathcal{D}_{\text{p}}$, the attacker chooses ϵn points (possibly repeated) from $\mathcal{D}_{\text{pool}}$ and flips their labels.

Many ways of choosing which points to flip have been proposed (see Xiao et al., 2015 for a review). Here, we consider the alfa attack from Xiao et al. (2012). alfa seeks to add points that have a high loss under the original (clean) model $\theta^* \stackrel{\text{def}}{=} \operatorname{argmin}_\theta L(\theta; \mathcal{D}_{\text{c}})$ but a low loss under the final (poisoned) model $\tilde{\theta} \stackrel{\text{def}}{=} \operatorname{argmin}_\theta L(\theta; \mathcal{D}_{\text{c}} \cup \mathcal{D}_{\text{p}})$. Concretely, it solves:

$$\begin{aligned} \underset{\mathcal{D}_\mathrm{p}}{\text{maximize}} \quad & L(\hat{\theta};\mathcal{D}_\mathrm{p}) - L(\theta^*;\mathcal{D}_\mathrm{p}) \\ \text{s.t.} \quad & |\mathcal{D}_\mathrm{p}| = \epsilon|\mathcal{D}_\mathrm{c}| \\ & (\tilde{x}, -\tilde{y}) \in \mathcal{D}_\mathrm{pool} \quad \forall(\tilde{x}, \tilde{y}) \in \mathcal{D}_\mathrm{p} \\ & \hat{\theta} = \operatorname{argmin}_\theta L(\theta;\mathcal{D}_\mathrm{c} \cup \mathcal{D}_\mathrm{p}) \end{aligned}$$

We adapt the original alfa attack to our setting in the following two ways:

1. We constrain all poisoned points in $\mathcal{D}_\mathrm{p}$ to lie in the feasible set $\mathcal{F}_\beta$.
2. We set $\mathcal{D}_\mathrm{pool} = \mathcal{D}_\mathrm{test}$; that is, the attacker gets to add points from the flipped test set. Intuitively, adding flipped versions of test points to the training set should cause the model to wrongly classify those test points, in line with the attacker's goal of increasing the model's loss on the test set.

We evaluated this variant of alfa on the Enron dataset. To make it easier for the attacker, we only considered the L2 defense. Our implementation of the alfa attack was not able to increase the test error much, achieving a 2% increase in test error with $\epsilon = 3\%$ (Fig. 13). In contrast, the attacks we introduce in the main text achieve an increase in test error of 15–20%, despite having to deal with more sophisticated defenses. Our conclusion is that only modifying the labels $\tilde{y}$, as in label flip attacks, does not give the attacker much power relative to being able to modify $\tilde{x}$ as well.

Acknowledgements We are grateful to Steve Mussmann, Zhenghao Chen, Marc Rasi, Robin Jia, and our anonymous reviewers for helpful comments and discussion.

Author Contributions PWK, JS, and PL conceptualized the project and wrote the paper. PWK and JS implemented the experiments.

Funding This work was partially funded by an Open Philanthropy Project Award. PWK was supported by the Facebook Fellowship Program. JS was supported by the Fannie and John Hertz Foundation Fellowship.

Availability of data and code Code and data for replicating our experiments are available at https://github.com/kohpangwei/data-poisoning-journal-release.

References

Agarwal, N., Bullins, B., & Hazan, E. (2016). *Second order stochastic optimization in linear time*. arXiv:160203943

Athalye, A., Carlini, N.&, Wagner, D. (2018). Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning (ICML)*.

Awasthi, P., Balcan, M. F., & Long, P. M. (2014). The power of localization for efficiently learning linear separators with noise. In *Symposium on theory of computing (STOC)* (pp. 449–458).

Bárány, I., & Karasev, R. (2012). Notes about the Carathéodory number. *Discrete & Computational Geometry, 48*(3), 783–792.

Bard, J. F. (1999). *Practical bilevel optimization: Algorithms and applications*. Springer.

Bard, J. F. (1991). Some properties of the bilevel programming problem. *Journal of Optimization Theory and Applications, 68*(2), 371–378.

Barreno, M., Nelson, B., Joseph, A. D., & Tygar, J. D. (2010). The security of machine learning. *Machine Learning, 81*(2), 121–148.

Biggio, B., Didaci, L., Fumera, G., & Roli, F. (2013). Poisoning attacks to compromise face templates. In *2013 international conference on biometrics (ICB)* (pp. 1–7)

Biggio, B., Fumera, G., Roli, F., & Didaci, L. (2012a). Poisoning adaptive biometric systems. In *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)* (pp. 417–425).

Biggio, B., Nelson, B., & Laskov, P. (2012b). Poisoning attacks against support vector machines. In *International conference on machine learning (ICML)* (pp. 1467–1474).

Biggio, B., Fumera, G., & Roli, F. (2014). Security evaluation of pattern classifiers under attack. *IEEE Transactions on Knowledge and Data Engineering, 26*(4), 984–996.

Biggio, B., Nelson, B., & Laskov, P. (2011). Support vector machines under adversarial label noise. *ACML, 20,* 97–112.

Burkard, C., & Lagesse, B. (2017). Analysis of causative attacks against SVMs learning from data streams. In *International workshop on security and privacy analytics*.

Carlini, N., Mishra, P., Vaidya, T., Zhang, Y., Sherr, M., Shields, C., Wagner, D. & Zhou, W. (2016). Hidden voice commands. In: *USENIX security*.

Charikar, M., Steinhardt, J., & Valiant, G. (2017). Learning from untrusted data. In *Symposium on theory of computing (STOC)*.

Chen, X., Liu, C., Li, B., Lu, K., & Song, D. (2017). *Targeted backdoor attacks on deep learning systems using data poisoning*. arXiv:171205526

Crammer, K., & Singer, Y. (2002). On the learnability and design of output codes for multiclass problems. *Machine Learning, 47*(2), 201–233.

Cretu, G. F., Stavrou, A., Locasto, M. E., Stolfo, S. J., & Keromytis, A. D. (2008). Casting out demons: Sanitizing training data for anomaly sensors. In *IEEE symposium on security and privacy* (pp. 81–95).

Demontis, A., Melis, M., Pintor, M., Jagielski, M., Biggio, B., Oprea, A., Nita-Rotaru, C., & Roli, F. (2019). Why do adversarial attacks transfer? Explaining transferability of evasion and poisoning attacks. In *28th USENIX security symposium (USENIX security 19)* (pp. 321–338).

Diakonikolas, I., Kamath, G., Kane, D., Li, J., Moitra, A., & Stewart, A. (2016). Robust estimators in high dimensions without the computational intractability. In *Foundations of computer science (FOCS)*.

Diakonikolas, I., Kamath, G., Kane, D.M., Li, J., Steinhardt, J., & Stewart, A. (2018). *Sever: A robust meta-algorithm for stochastic optimization*. arXiv:180302815

Diakonikolas, I., Kamath, G., Kane, D., Lim J., Moitra, A., & Stewart, A. (2017a). *Being robust (in high dimensions) can be practical*. arXiv.

Diakonikolas, I., Kane, D. M., & Stewart, A. (2017b). *Learning geometric concepts with nasty noise*. arXiv.

Duchi, J., Hazan, E., & Singer, Y. (2010). Adaptive subgradient methods for online learning and stochastic optimization. In *Conference onlearning theory (COLT)*.

Feldman, V., Gopalan, P., Khot, S., & Ponnuswami, A. K. (2009). On agnostic learning of parities, monomials, and halfspaces. *SIAM Journal on Computing, 39*(2), 606–645.

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

Frederickson, C., Moore, M., Dawson, G., & Polikar, R. (2018). Attack strength vs. detectability dilemma in adversarial machine learning. In *2018 international joint conference on neural networks (IJCNN)* (pp. 1–8).

Gardiner, J., & Nagaraja, S. (2016). On the security of machine learning in malware C&C detection: A survey. *ACM Computing Surveys (CSUR),49*(3)**

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems (NeurIPS)*.

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *International conference on learning representations (ICLR)*.

Gu, T., Dolan-Gavitt, B., Garg, S. (2017). *Badnets: Identifying vulnerabilities in the machine learning model supply chain*. arXiv:170806733

Guruswami, V., & Raghavendra, P. (2009). Hardness of learning halfspaces with noise. *SIAM Journal on Computing, 39*(2), 742–765.

Hodge, V., & Austin, J. (2004). A survey of outlier detection methodologies. *Artificial Intelligence Review, 22*(2), 85–126.

Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., & Li, B. (2018). Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE symposium on security and privacy (SP)* (pp. 19–35).

Kearns, M., & Li, M. (1993). Learning in the presence of malicious errors. *SIAM Journal on Computing, 22*(4), 807–837.

Klivans, A. R., Long, P. M., & Servedio, R. A. (2009). Learning halfspaces with malicious noise. *Journal of Machine Learning Research (JMLR), 10,* 2715–2740.
Kloft, M., & Laskov, P. (2012). Security analysis of online centroid anomaly detection. *Journal of Machine Learning Research (JMLR), 13,* 3681–3724.
Koh, P. W., & Liang, P. (2017). Understanding black-box predictions via influence functions. In *International conference on machine learning (ICML)*.
Kurakin, A., Goodfellow, I., & Bengio, S. (2016). *Adversarial examples in the physical world*. arXiv.
Lai, K. A., Rao, A. B., & Vempala, S. (2016). Agnostic estimation of mean and covariance. In *Foundations of computer science (FOCS)*.
LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324.
Li, J. (2018). *Principled approaches to robust machine learning and beyond*. Ph.D. thesis, Massachusetts Institute of Technology.
Li, B., Wang, Y., Singh, A., & Vorobeychik, Y. (2016). Data poisoning attacks on factorization-based collaborative filtering. In *Advances in neural information processing systems (NeurIPS)*.
Liu, C., Li, B., Vorobeychik, Y., & Oprea, A. (2017). Robust linear regression against training data poisoning. In *Proceedings of the 10th ACM workshop on artificial intelligence and security* (pp. 91–102).
Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y.,&Potts, C. (2011). Learning word vectors for sentiment analysis. In *Association for computational linguistics (ACL)*
Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). *Towards deep learning models resistant to adversarial attacks (published at ICLR 2018)*. arXiv.
Martens, J. (2010). Deep learning via hessian-free optimization. In *International conference on machine learning (ICML)* (pp. 735–742).
Mei, S., & Zhu, X. (2015a). The security of latent Dirichlet allocation. In *Artificial intelligence and statistics (AISTATS)*.
Mei, S., & Zhu, X. (2015b). Using machine teaching to identify optimal training-set attacks on machine learners. In *Association for the advancement of artificial intelligence (AAAI)*.
Metsis, V., Androutsopoulos, I., & Paliouras, G. (2006). Spam filtering with naive Bayes—Which naive Bayes? *CEAS, 17,* 28–69.
Mirrokni, V., Leme, R. P., Vladu, A., & Wai Wong, S.C. (2015). *Tight bounds for approximate Carathéodory and beyond*. arXiv:151208602.
Moosavi-Dezfooli, S. M., Fawzi, A., & Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In *Computer vision and pattern recognition (CVPR)* (pp. 2574–2582).
Muñoz-González, L., Biggio, B., Demontis, A., Paudice, A., Wongrassamee, V., Lupu, E. C., & Roli, F. (2017). Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM workshop on artificial intelligence and security* (pp. 27–38).
Nelson, B., Barreno, M., Chi, F. J., Joseph, A. D., Rubinstein, B. I., Saini, U., et al. (2008). Exploiting machine learning to subvert your spam filter. *LEET, 8,* 1–9.
Newell, A., Potharaju, R., Xiang, L., & Nita-Rotaru, C. (2014). On the practicality of integrity attacks on document-level sentiment analysis. In *Workshop on artificial intelligence and security (AISec)* (pp. 83–93)
Papernot, N., & McDaniel, P. (2018). *Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning*. arXiv:180304765
Papernot, N., McDaniel, P., & Goodfellow, I. (2016a). *Transferability in machine learning: from phenomena to black-box attacks using adversarial samples*. arXiv.
Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, B. Z., & Swami, A. (2017). Practical black-box attacks against machine learning. In *Asia conference on computer and communications security* (pp. 506–519).
Papernot, N., McDaniel, P., Sinha, A., & Wellman, M. (2016b). *Towards the science of security and privacy in machine learning*. arXiv.
Papernot, N., McDaniel, P., Wu, X., Jha, S., & Swami, A. (2016c). as a defense to adversarial perturbations against deep neural networks. In *IEEE symposium on security and privacy* (pp. 582–597).
Paudice, A., Muñoz-González, L., Gyorgy, A., & Lupu, E. C. (2018). *Detection of adversarial training examples in poisoning attacks through anomaly detection*. arXiv:180203041
Pearlmutter, B. A. (1994). Fast exact multiplication by the Hessian. *Neural Computation, 6*(1), 147–160.
Raghunathan, A., Steinhardt, J., & Liang, P. (2018). Certified defenses against adversarial examples. In *International conference on learning representations (ICLR)*.
Rubinstein, B., Nelson, B., Huang, L., Joseph, A. D., Lau, S. H., Rao, S., Taft, N., & Tygar, J. (2009). Certified defenses against adversarial examples. In *International conference on learning representations (ICLR)*.

Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., & Goldstein, T. (2018). *Poison Frogs! Targeted clean-label poisoning attacks on neural networks*. arXiv:180400792

Sinha, A., Namkoong, H., & Duchi, J. (2018). Certifiable distributional robustness with principled adversarial training. In *International conference on learning representations (ICLR)*.

Steinhardt, J. (2018). *Robust learning: Information theory and algorithms*. Ph.D. thesis, Stanford University.

Steinhardt, J., Charikar, M., & Valiant, G. (2018). Resilience: A criterion for learning in the presence of arbitrary outliers. In *Innovations in theoretical computer science (ITCS)*.

Steinhardt, J., Koh, P. W., & Liang, P. (2017). Certified defenses for data poisoning attacks. In *Advances in neural information processing systems (NeurIPS)*.

Suciu, O., Mărginean, R., Kaya, Y., III, H. D., & Dumitraş, T. (2018). *When does machine learning fail? generalized transferability for evasion and poisoning attacks*. arXiv:180306975

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*.

Tramèr, F., Kurakin, A., Papernot, N., Boneh, D., & McDaniel, P. (2017). *Ensemble adversarial training: Attacks and defenses.* arXiv:170507204

Vorobeychik, Y., & Kantarcioglu, M. (2018). Adversarial machine learning. *Machine Learning, 12*(3), 1–169.

Wong, E., & Kolter, J. Z. (2018). Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International conference on machine learning (ICML)*.

Xiao, H., Xiao, H., & Eckert, C. (2012). Adversarial label flips attack on support vector machines. In *European conference on artificial intelligence*.

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

Yang, C., Wu, Q., Li, H., & Chen, Y. (2017). *Generative poisoning attack method against neural networks*. arXiv.