Model-Targeted Poisoning Attacks with Provable Convergence

Fnu Suya [1] **Saeed Mahloujifar** [2] **Anshuman Suri** [1] **David Evans** [1] **Yuan Tian** [1]

Abstract

In a poisoning attack, an adversary with control over a small fraction of the training data attempts to select that data in a way that induces a corrupted model that misbehaves in favor of the adversary. We consider poisoning attacks against convex machine learning models and propose an efficient poisoning attack designed to induce a specified model. Unlike previous model-targeted poisoning attacks, our attack comes with provable convergence to *any* attainable target classifier. The distance from the induced classifier to the target classifier is inversely proportional to the square root of the number of poisoning points. We also provide a lower bound on the minimum number of poisoning points needed to achieve a given target classifier. Our method uses online convex optimization, so finds poisoning points incrementally. This provides more flexibility than previous attacks which require a priori assumption about the number of poisoning points. Our attack is the first model-targeted poisoning attack that provides provable convergence for convex models, and in our experiments, it either exceeds or matches state-of-the-art attacks in terms of attack success rate and distance to the target model.

1. Introduction

Machine learning often requires a large amount of labeled training data, which is collected from untrusted sources. A typical application is email spam filtering, where a spam detector filters out spam messages based on features (e.g., presence of certain words) and periodically updates the model based on newly received emails labeled by users. In such a setting, spammers can generate spam messages that inject benign words likely to occur in legitimate emails, and when models are trained on these spam messages, the filtering accuracy drops significantly (Nelson et al., 2008; Huang et al., 2011). Such attacks are known as *poisoning attacks*, and a training process that uses labels or data from untrusted sources is potentially vulnerable to them.

Poisoning attacks can be categorized as *objective-driven* or *model-targeted*. Objective-driven poisoning attacks have a specified attacker objective (such as reducing the overall accuracy of the victim model) and aim to induce a model that maximizes that objective. Model-targeted attacks have a specific target model in mind and aim to induce a victim model as close as possible to that target model. Objective-driven attacks are most commonly studied in the existing literature, and indeed, it is natural to think about attacks in terms of the goals of an adversary. We argue, though, that breaking poisoning attacks into the two steps of first finding a model to target and then selecting poisoning points to induce that model has significant advantages. This view leads to improvements in our understanding of poisoning attacks and simplifies the task of designing effective attacks for a variety of different objectives. Importantly, it can also lead to more effective poisoning attacks.

Attacker objectives for realistic attacks are diverse, and designing a unified and effective attack strategy for different attacker objectives is hard. Most work has considered one of two extremal attacker objectives: *indiscriminate* attacks, where the adversary's goal is simply to decrease the overall accuracy of the model (Biggio et al., 2012; Xiao et al., 2012; Mei & Zhu, 2015b; Steinhardt et al., 2017; Koh et al., 2018); and *instance-targeted* attacks, where the goal is to induce a classifier that misclassifies a particular known input (Shafahi et al., 2018; Zhu et al., 2019; Koh & Liang, 2017; Geiping et al., 2020; Huang et al., 2020). Recently, Jagielski et al. (2019) introduced a more realistic attacker objective known as a *subpopulation* attack, where the goal is to increase the error rate or obtain a particular output for a defined subpopulation of the data

[1]University of Virginia [2]Princeton University. Correspondence to: Fnu Suya <suya@virginia.edu>, Saeed Mahloujifar <sfar@princeton.edu>.

Preprint.

distribution. Gradient-based local optimization is most commonly used to construct poisoning points for a particular attacker objective (Biggio et al., 2012; Xiao et al., 2012; Mei & Zhu, 2015b; Koh & Liang, 2017; Shafahi et al., 2018; Zhu et al., 2019). These attacks can be modified to fit other attacker objectives, but since they are based on local optimization techniques they often get stuck into bad local optima and fail to find effective sets of poisoning points (Steinhardt et al., 2017; Koh et al., 2018). To circumvent the issue of local optima, Steinhardt et al. (2017) formulate the indiscriminate attack as a min-max optimization and solve it efficiently using online convex optimization techniques. However, this attack only applies to the indiscriminate setting.

In contrast, *model-targeted attacks* incorporate the attacker objective into a target model and hence, the target model can reflect any attacker objective. Thus, the same model-targeted attack methods can be directly applied to a range of indiscriminate and subpopulation attacks just by finding a suitable target model. Mei & Zhu (2015b) first introduced a target model into a poisoning attack and then utilized the KKT condition to transform the problem into a tractable form, but their attack is still based on gradient-based local optimization techniques and suffers from bad local optima (Steinhardt et al., 2017; Koh et al., 2018). Koh et al. (2018) proposed the KKT attack, which converts the complicated bi-level optimization into a simple convex optimization problem utilizing the KKT condition and the Carathéodory number of the set of scaled gradients, avoiding the local optima issues. However, their attack only works for margin-based losses and does not provide any guarantee on the number of poisoning points required to converge to the target classifier. Additionally, these attacks require knowing the number of poisoning points before running the attack, which is often not available in practical applications.

We study poisoning attacks on simple convex models because poisoning attacks are still not fully understood in these settings. In addition, many important industrial applications continue to rely on simple models due to their easiness in model debugging, low computational cost, and for many applications, such simple convex models also have either comparable or better performances than the complex deep neural networks (Dacrema et al., 2019; Tramèr & Boneh, 2020).

Contributions. Our main contribution is a principled and general model-targeted poisoning method, along with proof that the model it induces converges to the target model. In this work, we focus on effectiveness in inducing a given target model and defer to future work a full exploration of how to select good target models for particular attacker objectives. Our focus also aligns with the goal of previous model-targeted poisoning attacks (Koh et al., 2018; Mei & Zhu, 2015b).

We prove, for settings where the loss function is convex and proper regularization is adopted in training, that the model induced by training on the original training data with these points added, converges to the target classifier as the number of poison points increases (Theorem 1). Previous model-targeted attacks lack such convergence guarantees. We then prove a lower bound on the minimum number of poisoning points needed to reach the target model (Theorem 2). Such a lower bound can be used to estimate the optimality of model-targeted poisoning attacks and also indicate the intrinsic hardness of attacking different targets. Our attack applies to incremental poisoning scenarios as it works in an online fashion to find effective poisoning points without a predetermined poisoning rate. Previous model-targeted attacks assume a priori number of poisoning points.

We evaluate our attack and compare it to the state-of-the-art model-targeted attack (Koh et al., 2018). We evaluate the convergence of our attack to the target model and find that for the same number of poisoning points, our attack is able to induce models closer to the target model, for all target classifiers we tried. The success rate of our attack exceeds that of the state-of-the-art attack in subpopulation attack scenarios and is comparable for indiscriminate attacks (Section 5).

2. Problem Setup

The poisoning attack proposed in this paper applies to multi-class prediction tasks or regression problems (by treating the response variable as an additional data feature), but for simplicity of presentation we consider a binary prediction task, $h : \mathcal{X} \to \mathcal{Y}$, where $X \subseteq \mathbb{R}^d$ and $\mathcal{Y} = \{+1, -1\}$. The prediction model h is characterized by parameters $\theta \in \Theta \subseteq \mathbb{R}^d$. We define the non-negative convex loss on an individual point, (x, y), as $l(\theta; x, y)$ (e.g., hinge loss for SVM model). We also define the empirical loss over a set of points A as $L(\theta; A) = \sum_{(x,y)\in A} l(\theta; x, y)$.

We adopt the game-theoretic formalization of the poisoning attack process from Steinhardt et al. (2017) to describe our model-targeted attack scenario:

1. N data points are drawn uniformly at random from the true data distribution over $\mathcal{X} \times \mathcal{Y}$ and form the clean training set, $\mathcal{D}_c$.

2. The adversary, with knowledge of $\mathcal{D}_c$, the model training process and the model space Θ, generates a target classifier $\theta_p \in \Theta$ that satisfies the attack goal.

3. The adversary produces a set of poisoning points, $\mathcal{D}_p$, with the knowledge of $\mathcal{D}_c$, model training process, Θ and θ_p.

4. Model builder trains the model on $\mathcal{D}_c \cup \mathcal{D}_p$ and produces a classifier, θ_{atk}.

The adversary's goal is that the induced classifier, θ_{atk}, is close to the desired target classifier, θ_p (Section 4.2 discusses how this distance is measured). Step 2 corresponds to the target classifier generation process. Our attack works for any target classifier, and in the paper we do not focus on the question of how to find the best target classifier to achieve a particular adversarial goal but simply adopt the heuristic target classifier generation process from Koh et al. (2018). Step 3 corresponds to our model-targeted poisoning attack and is also the main contribution of the paper.

We assume the model builder trains a model through empirical risk minimization (ERM) and the training process details are known to the attacker:

$$\theta_c = \arg\min_{\theta \in \Theta} \frac{1}{|\mathcal{D}_c|} L(\theta; \mathcal{D}_c) + C_R \cdot R(\theta) \tag{1}$$

where $R(\theta)$ is the nonnegative regularization function (e.g., $\frac{1}{2}\|\theta\|_2^2$ for SVM model).

Threat Model. We assume an adversary with full knowledge of training data, model space, and training process. Although this may be unrealistic for many scenarios, this setting allows us to focus on a particular aspect of poisoning attacks and is the setting used in many prior works (Biggio et al., 2011; Mei & Zhu, 2015b; Steinhardt et al., 2017; Koh et al., 2018; Shafahi et al., 2018). We assume an addition-only attack where the attacker only adds poisoning points into the clean training set. A stronger attacker may be able to modify or remove existing points, but this typically requires administrative access to the system. The added points are unconstrained, other than being value elements of the input space. They can have arbitrary features and labels, which enables us to perform the worst-case analysis on the robustness of models against addition-only poisoning attacks. Although some previous works also allow arbitrary selection of the poisoning points (Biggio et al., 2011; Mei & Zhu, 2015b; Steinhardt et al., 2017; Koh et al., 2018), others put different restrictions on the poisoning points. A clean-label attack assumes adversaries can only perturb the features of the data, but the label is given by an oracle labeler (Koh & Liang, 2017; Shafahi et al., 2018; Zhu et al., 2019; Huang et al., 2020). In label-flipping attacks, adversaries are only allowed to change the labels (Biggio et al., 2011; Xiao et al., 2012; 2015; Jagielski et al., 2019). These restricted attacks are weaker than the poisoning attacks without restrictions (Koh et al., 2018; Hong et al., 2020).

3. Related Work

The most commonly used poisoning strategy is gradient-based attack. Gradient-based attacks iteratively modify a candidate poisoning point $(\hat{x}, \hat{y})$ in the set $\mathcal{D}_p$ based on the test loss defined on $\hat{x}$ (keeping $\hat{y}$ fixed). This kind of attack was first studied on SVM models (Biggio et al., 2012; Demontis et al., 2019), and later extended to linear and logistic regression (Mei & Zhu, 2015b; Demontis et al., 2019), and recently to larger neural network models (Koh & Liang, 2017; Shafahi et al., 2018; Zhu et al., 2019; Huang et al., 2020). Jagielski et al. (2018) studied gradient attacks and principled defenses on linear regression tasks. In addition to classification and regression tasks, gradient-based poisoning attacks are also applied to topic modeling (Mei & Zhu, 2015a), collaborative filtering (Li et al., 2016) and algorithmic fairness (Solans et al., 2020).

Besides the gradient-based attacks, researchers also utilize generative adversarial networks to craft poisoning points efficiently for larger neural networks, but with limited effectiveness (Yang et al., 2017; Muñoz-González et al., 2019). The strongest attacks so far are the KKT attack (Koh et al., 2018) and the min-max attack (Steinhardt et al., 2017; Koh et al., 2018). However, the KKT attack cannot scale well for multi-class classification and is limited to margin-based losses (Koh et al., 2018). The min-max attack only works for indiscriminate attack setting, but additionally provides a certificate on worst-case test loss for a fixed number of poisoning points. We are also inspired by Steinhardt et al. (2017) to adopt online convex optimization to instantiate our model-targeted attack, but now dealing with a more general attack scenario. We also distinguish ourselves from the poisoning attack against online learning (Wang & Chaudhuri, 2018). The attack against online learning considers a setting where training data arrives in a streaming manner while we consider the offline setting with training data being fixed. Another line of work studies "targeted" poisoning attacks where an adversary guarantees to increase the probability of an arbitrary "bad" property (Mahloujifar et al., 2019a; 2018; 2019b), as long as that property has some non-negligible chance of naturally happening. These attacks cannot be applied in model-targeted setting as the

probability of naturally producing a specific target model is often negligible. Related to our Theorem 2, Ma et al. (2019) also derived a lower bound on the number of poisoning points (to induce a target model), but their lower bound only applies when differential privacy is deployed during the model training process (and hence hurts model utility), which is different from our problem setting.

4. Poisoning Attack with a Target Model

Our new poisoning attack determines a target model and selects poisoning points to achieve that target model. The target model generation is not our focus and we adopt the heuristic approach proposed by Koh et al. (2018). For the new poisoning attack, we first show how the algorithm generates the poisoning points (Section 4.1). Then, we prove that the generated poisoning points, once added to the clean data, can produce a classifier that asymptotically converges to the target classifier (Section 4.2).

4.1. Model-Targeted Poisoning with Online Learning

The main idea of our model-targeted poisoning attack, as outlined in Algorithm 1, is to sequentially add a point into the training set that has maximum loss-difference between the intermediate model obtained so far and the target model. By training models on the updated training set we actually minimize the gap in the loss of the intermediate classifier and the target classifier. Repeating the process then eventually generates classifiers that have similar loss distribution as the target classifier. We show in Section 4.2 why similar loss distribution implies convergence.

Algorithm 1 ModelTargetedPoisoning

Input: $\mathcal{D}_c$, the loss functions (L and l), θ_p
Output: $\mathcal{D}_p$

1: $\mathcal{D}_p = \emptyset$
2: **while** stop criteria not met **do**
3: $\quad \theta_t = \arg\min L(\theta; \mathcal{D}_c \cup \mathcal{D}_p)$
4: $\quad (x^*, y^*) = \arg\max_{\mathcal{X}\times\mathcal{Y}} l(\theta_t; x, y) - l(\theta_p; x, y)$
5: $\quad \mathcal{D}_p = \mathcal{D}_p \cup \{(x^*, y^*)\}$
6: **end while**
return $\mathcal{D}_p$

Algorithm 1 requires the input of clean training set $\mathcal{D}_c$, the Loss function (L for a set of points and l for individual point), and the target model θ_p. The output from Algorithm 1 will be the set of poisoning points $\mathcal{D}_p$. The algorithm is simple: first, adversaries train the intermediate model θ_t on the mixture of clean and poisoning points $\mathcal{D}_c \cup \mathcal{D}_p$ with $\mathcal{D}_p$ an empty set in the first iteration (Line 3). The adversary then searches for the point that maximizes the loss difference between θ_t and θ_p (Line 4). After the point of maximum loss difference is found, it is added to the poisoning set $\mathcal{D}_p$ (Line 5). The whole process repeats until the stop condition is satisfied in Line 2. The stop condition is flexible and it can take various forms: 1) adversary has a budget T on the number of poisoning points, and the algorithm halts when the algorithm runs for T iterations; 2) the intermediate classifier θ_t is closer to the target classifier (than a preset threshold ϵ) in terms of the maximum loss difference, and more details regarding this distance metric will be introduced in Section 4.2; 3) adversary has some requirement on the accuracy and the algorithm terminates when θ_t satisfies the accuracy requirement. Since we focus on producing a classifier close to the target model, we adopt the second stop criterion that measures the distance with respect to the maximum loss difference, and report results based on this criterion in Section 5.

A nice property of Algorithm 1 is that the classifier θ_{atk} trained on $\mathcal{D}_c \cup \mathcal{D}_p$ is close to the target model θ_p and asymptotically converges to θ_p. Details of the convergence will be shown in the next section. The algorithm may appear to be slow, particularly for larger models due to the requirement of repeatedly training a model in line 3. However, this is not an issue. First, as will be shown in the next section, the algorithm is an online optimization process and line 3 corresponds to solving the online optimization problem exactly. However, people often use the very efficient online gradient descent method to approximately solve the problem and its asymptotic performance is the same (Shalev-Shwartz, 2012). Second, if we solve the optimization problem exactly, we can add multiple copies of (x^*, y^*) into $\mathcal{D}_p$ each time. This reduces the overall iteration number, and hence reduces the number of times retraining models. The proof of convergence will be similar. For simplicity in interpreting the results, we do not use this in our experiments and add only one copy of (x^*, y^*) each iteration. However, we also tested the performance by adding two copies of (x^*, y^*) and find that the attack results are nearly the

same while the efficiency is improved significantly. For example, for experiments on MNIST 1–7 dataset, by adding 2 copies of points, with the same number of poisoning points, the attack success rate decreases at most by 0.7% while the execution time is reduced approximately by half.

4.2. Convergence of Our Poisoning Attack

Before proving the convergence of Algorithm 1, we need to measure the distance of the model θ_{atk} trained on $\mathcal{D}_c \cup \mathcal{D}_p$ to the target model θ_p. First, we define a general closeness measure based on their prediction performance which we will use to state our convergence theorem:

Definition 1 (Loss-based distance and ϵ-close)**.** For two models θ_1 and θ_2, a space $\mathcal{X} \times \mathcal{Y}$ and a loss $l(\theta; x, y)$, we define *loss-based distance* $D_{l,\mathcal{X},\mathcal{Y}} \colon \Theta \times \Theta \to R$ as

$$D_{l,\mathcal{X},\mathcal{Y}}(\theta_1, \theta_2) = \max_{(x,y)\in\mathcal{X}\times\mathcal{Y}} l(\theta_1; x, y) - l(\theta_2; x, y),$$

and we say model θ_1 is ϵ-*close* to model θ_2 when the loss-based distance from θ_1 to θ_2 is upper bounded by ϵ.

Measuring model distance We use loss-based distance to capture the "behavioral" distance between two models. Namely, if θ_1 is ϵ-close (as measured by loss-based distance) to θ_2 and vice versa, then θ_1 and θ_2 would have an almost equal loss on all the points, meaning that they have almost the same behavior across all the space. Note that our general definition of loss-based distance does not have the symmetry property of metrics and hence is not a metric. However, it has some other properties of metrics in the space of attainable[1] models. For example, if some model θ is attainable using ERM, no model could have a negative distance to it. To further show the value of this distance notion, in Appendix B we demonstrate an $O(\epsilon)$ upper bound on the ℓ_1-norm of difference between two models that are ϵ-close with respect to loss-based distance for the special case of Hinge loss. For Hinge loss, it also satisfies the *bi-directional closeness*, that is if θ_1 is ϵ-close to θ_2, then θ_2 is $O(\epsilon)$-close to θ (details can be found in Corollary 3), and the proof details can be found in Appendix B. In the rest of the paper, we will use the terms ϵ-close or ϵ-closeness to denote that a model is ϵ away from another model based on the loss-based distance.

Our convergence theorem uses the loss-based distance to establish that the attack of Algorithm 1 produces model that converges to the target classifier:

Theorem 1. After at most T steps, Algorithm 1 will produce the poisoning set $\mathcal{D}_p$ and the classifier trained on $\mathcal{D}_c \cup \mathcal{D}_p$ is ϵ-close to θ_p, with respect to loss-based distance, $D_{l,\mathcal{X},\mathcal{Y}}$, for

$$\epsilon = \frac{\alpha(T) + L(\theta_p; D_c) - L(\theta_c; D_c)}{T \cdot \gamma}$$

where, γ is a constant for a given θ_p and classification task, and $\alpha(T)$ is the regret of the online algorithm when the loss function used for training is convex.

Remark 1. Online learning algorithms with sublinear regret bound can be applied to show the convergence. Here, we adopt results from McMahan (2017). Specifically, $\alpha(T)$ is in the order of $O(\log T))$ and we have $\epsilon \leq O(\frac{\log T}{T})$ when the loss function is additionally Lipschitz continuous and the regularizer $R(\theta)$ is strongly convex, and $\epsilon \to 0$ when $T \to +\infty$. $\alpha(T)$ is also in the order of $O(\log T)$ when the loss function used for training is strongly convex and the regularizer is convex.

Proof idea. The full proof of Theorem 1 is in Appendix A. Here, we only summarize the high-level proof idea. The key idea is to frame the poisoning problem as an online learning problem. In this formulation, each step of the online learning problem corresponds to the i^{th} poison point (x_i, y_i). In particular, the loss function at iteration i of the online learning problem is set to $l(\cdot; x_i, y_i)$. Then, we show that by defining the parameters of the online learning problem carefully, the output of the follow-the-leader (FTL) algorithm (Shalev-Shwartz, 2012) at iteration i is a model that is identical to training a model on a dataset consisting of the clean points and the first $i-1$ poisoning points. On the other hand, the way the poisoning points are selected, we can show that at the i^{th} iteration the maximum loss difference between the target model and the best induced model so far would be smaller than the regret of the FTL algorithm divided by the number of poisoning points. The convergence bound of Theorem 1 boils down to regret analysis of the algorithm based on the loss function.

[1] Attainable models are models that can be obtained by training on some data from the input space. See formal definition in Appendix A.

Since we are assuming the loss function is convex with a strongly convex regularizer (or a strongly convex loss function with a convex regularizer), we can show that the regret is bounded by $O(\log T)$ and hence the loss distance between the induced model and the target model converges to 0.

Implications of Theorem 1 The theorem says that the loss-based distance of the model trained on $\mathcal{D}_c \cup \mathcal{D}_p$ to the target model correlates to the loss difference between the target model and the clean model θ_c (trained on $\mathcal{D}_c$) on $\mathcal{D}_c$, and correlates inversely with the number of poisoning points. Therefore, it implies 1) if the target classifier θ_p has a lower loss on $\mathcal{D}_c$, then it is easier to achieve the target model, and 2) with more poisoning points, we get closer to the target classifier and our attack will be more effective. The theorem also justifies the motivation behind the heuristic method in Koh et al. (2018) to select a target classifier with a lower loss on clean data. For the indiscriminate attack scenario, we also improve the heuristic approach by adaptively updating the model and producing target classifiers with a much lower loss on the clean set. This helps to empirically validate our theorem. Details of the original and improved heuristic approach and relevant experiments are in Appendix D.3.

4.3. Lower Bound on the Number of Poisoning Points

We first provide the lower bound on the number of poisoning points required for producing the target classifier in the addition-only setting (Theorem 2) and then explain how the lower bound estimation can be incorporated into Algorithm 1. The intuition behind the theorem below is, when the number of poisoning points added to the clean training set is smaller than the lower bound, there always exists a classifier θ with lower loss compared to θ_p and hence the target classifier cannot be attained. The full proof of the theorem can be found in Appendix A.

Theorem 2 (Lower Bound)**.** Given a target classifier θ_p, to reproduce θ_p by adding the poisoning set $\mathcal{D}_p$ into $\mathcal{D}_c$, the number of poisoning points $|\mathcal{D}_p|$ cannot be lower than

$$\sup_{\theta} z(\theta) = \frac{L(\theta_p; \mathcal{D}_c) - L(\theta; \mathcal{D}_c) + NC_R(R(\theta_p) - R(\theta))}{\sup_{x,y} \big(l(\theta; x, y) - l(\theta_p; x, y)\big) + C_R(R(\theta) - R(\theta_p))}.$$

Corollary 1. If we further assume bi-directional closeness in the loss-based distance, we can also derive the lower bound on number of poisoning points needed to induce models that are ϵ-close to the target model. More precisely, if θ_1 being ϵ-close to θ_2 implies that θ_2 is also $k \cdot \epsilon$ close to θ_1, then we have,

$$\sup_{\theta} z'(\theta) = \frac{L(\theta_p; \mathcal{D}_c) - L(\theta; \mathcal{D}_c) - NC_R \cdot R^* - Nk\epsilon}{\sup_{x,y} \big(l(\theta; x, y) - l(\theta_p; x, y)\big) + C_R \cdot R^* + k\epsilon}.$$

where R^* is an upper bound on the regularizer $R(\theta)$.

The formula for the lower bound in Theorem 2 (and also the lower bound in Corollary 1) can be easily incorporated into Algorithm 1 to obtain a tighter theoretical lower bound. We simply need to check all of the intermediate classifiers θ_t produced during the attack process and replace θ with θ_t, and the lower bound can be computed for the pair of θ_t and θ_p. Algorithm 1 then additionally returns the lower bound, which is the highest lower bound computed from our poisoning procedure.

5. Experiments

We present the experimental results by showing the convergence of Algorithm 1, the comparison of attack success rates to state-of-the-art model-targeted poisoning attack, and the theoretical lower bound for inducing a given target classifier and its gap to the number of poisoning points used by our attack. All of our evaluation code is available at: `https://github.com/suyeecav/model-targeted-poisoning`.

Datasets and Subpopulations. We experiment on both the practical subpopulation and the conventional indiscriminate attack scenarios. We selected datasets and models for our experiments based on evaluations of previous poisoning attacks (Biggio et al., 2012; Mei & Zhu, 2015a; Koh et al., 2018; Steinhardt et al., 2017; Koh & Liang, 2017; Jagielski et al., 2019). For the subpopulation attack experiments, we use the Adult dataset (Dua & Graff, 2017), which was used for evaluation by (Jagielski et al., 2019). We downsampled the Adult dataset to make it class-balanced and ended up with 15,682

training and 7,692 test examples. Each example has the dimension of 57 after one-hot encoding the categorical attributes. For the indiscriminate setting, we use the Dogfish (Koh & Liang, 2017) and MNIST 1–7 datasets (LeCun, 1998)[2]. The Dogfish dataset contains 1,800 training and 600 test samples. We use the same Inception-v3 features (Szegedy et al., 2016) as in Koh & Liang (2017); Steinhardt et al. (2017); Koh et al. (2018) and each image is represented by a 2,048-dimensional vector. The MNIST 1–7 dataset contains 13,007 training and 2,163 test samples, and each image is flattened to a 784-dimensional vector.

We identify the subpopulations for the Adult dataset using k-means clustering techniques (ClusterMatch in Jagielski et al. (2019)) to obtain different clusters ($k = 20$). For each cluster, we select instances with the label "$\leq$ 50K" to form the subpopulation (indicating all instances in the subpopulation are in the low-income group). This way of defining subpopulation is rather arbitrary (in contrast to a more likely attack goal which would select subpopulations based on demographic characteristics), but enables us to simplify analyses. From the 20 subpopulations obtained, we select three subpopulations with the highest test accuracy on the clean model. They all have 100% test accuracy, indicating all instances in these subpopulations are correctly classified as low income. This enables us to use "attack success rate" and "accuracy" without any ambiguity on the subpopulation—for each of our subpopulations, all instances are originally classified as low income, and the simulated attacker's goal is to have them classified as high income.

Models and Attacks. We conduct experiments on linear SVM and logistic regression (LR) models. Although our theoretical results do not apply to non-convex models, for curiosity we also tested our attack on deep neural networks and report results in Appendix F.

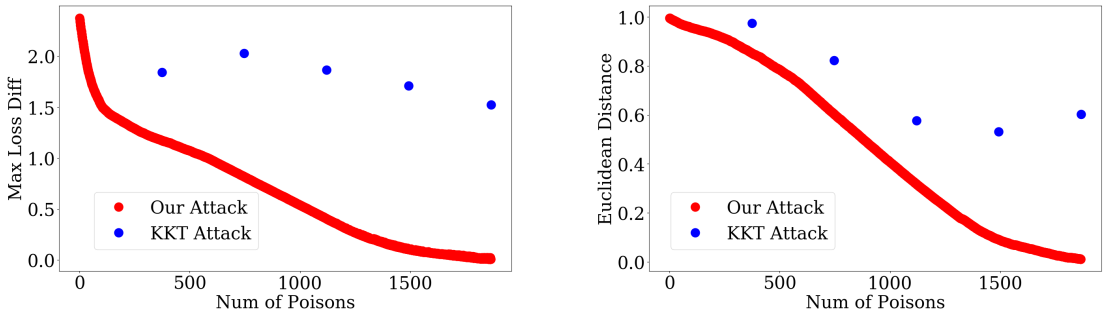
We use the heuristic approach from Koh et al. (2018) to generate target classifiers for both attack settings. In the subpopulation setting, for each subpopulation, we generate a target model that has 0% accuracy (100% attacker success) on the subpopulation, indicating that all subpopulation instances are now classified as high income. In the indiscriminate setting, for MNIST 1–7, we aim to generate three target classifiers with overall test errors of 5%, 10%, and 15%. For SVM, we obtained target models of test accuracies of 94.0%, 88.8%, and 83.3%, and for LR, the target models are of test accuracies of 94.7%, 89.0%, and 84.5%. For Dogfish, we aim to generate target models with overall test errors of 10%, 20%, and 30%. For SVM, we obtained target models of test accuracies of 89.3%, 78.3%, and 67.2% and for logistic regression, we obtained target models of test accuracies of 89.0% 79.5%, and 67.3%. The test accuracy of the clean SVM model is 78.5% on Adult, 98.9% on MNIST 1–7 and is 98.5% on Dogfish. The test accuracy of clean LR model is 79.9% on Adult, 99.1% on MNIST 1–7 and 98.5% on Dogfish.

We compare our model-targeted poisoning attack in Algorithm 1 to the state-of-the-art KKT attack (Koh et al., 2018). We do not include the model-targeted attack from Mei & Zhu (2015b) because there is no open source implementation and this attack is also reported to underperform the KKT attack (Koh et al., 2018). Our main focus here is on comparing to other model-targeted attacks in terms of achieving the target models, but we also do include experiments (in Appendix E) comparing our attack to existing objective-driven attacks, where the target model for our attack is selected to achieve that objective. With a carefully selected target model, our attack can also outperform the state-of-the-art objective-driven attack.

Both our attack and the KKT attack take as input a target model and the original training data, and output a set of poisoning points intended to induce a model as close as possible to the target model when the poisoning points are added to the original training data. We compare the effectiveness of the attacks by testing them using the same target model and measuring convergence of their induced models to the target model.

The KKT attack requires the number of poisoning points as an input, while our attack is more flexible and can produce poisoning points in priority order without a preset number. As a stopping condition for our experiments, we use either a target number of poisoning points or a threshold for ϵ-close distance to the target model. Since we do not know the number of poisoning points needed to reach some attacker goal in advance for the KKT attack, we first run our attack and produce a classifier that satisfies the selected ϵ-close distance threshold. The loss function is hinge loss for SVM and logistic loss for LR. For SVM model, we set ϵ as 0.01 on Adult, 0.1 on MNIST 1–7 and 2.0 on Dogfish dataset. For LR model, we set ϵ as 0.05 on Adult, 0.1 on MNIST 1–7 and 1.0 on Dogfish. Then, we use the size of the poisoning set returned from our attack (denoted by n_p) as the input to the KKT attack for the target number of poisons needed. We also compare the two attacks with varying numbers of poisoning points up to n_p. For the KKT attack, its entire optimization process must be rerun whenever the target number of poisoning points changes. Hence, it is infeasible to evaluate the KKT attack on many different poisoning set sizes. In our experiments, we run the KKT attack five poisoning set sizes: $0.2 \cdot n_p$, $0.4 \cdot n_p$, $0.6 \cdot n_p$,

[2]MNIST 1–7 dataset is a subset of the well-known MNIST dataset that only contains digit 1 and 7.



(a) Max Loss Difference (b) Euclidean Distance

Figure 1. Convergence to the target model. Results shown are for the first subpopulation (Cluster 0) and the model is SVM.

$0.8 \cdot n_p$, and n_p. For our attack, we simply run iterations up to the maximum number of poisoning points, collecting a data point for each iteration up to n_p. In Appendix D, we also plot the performance of our attack with respect to the number of poisoning points added across iterations.

Model/ Dataset	Target Model	n_p	Lower Bound	$0.2n_p$		$0.4n_p$		$0.6n_p$		$0.8n_p$		n_p	
				KKT	Ours	KKT	Ours	KKT	Ours	KKT	Ours	KKT	Ours
SVM/ Adult	Cluster 0	1,866	1,667±2.4	96.8	98.4	65.4	**51.6**	14.9	35.6	1.1	2.7	15.4	**0.5**
	Cluster 1	2,097	1,831.4±5.0	72.2	77.1	41.0	**23.6**	2.8	**0.7**	1.4	**0.7**	6.9	**0.0**
	Cluster 2	2,163.3±2.5	1,863.0±9.2	94.9 ± 0.7	**24.3± 0.3**	15.9	20.3	34.3 ± 0.2	**12.1**	21.6 ± 0.1	**0.3**	20.3 ± 0.7	**0.3**
LR/ Adult	Cluster 0	2,005	N/A	82.1 ± 1.0	**75.6**	71.7 ± 0.4	**42.2**	46.7	**15.9**	36.6 ± 2.1	**1.9**	24.3 ± 0.8	**0.4**
	Cluster 1	1,630±1.1	N/A	98.1	**94.9**	97.2	**79.0**	96.7	**34.1**	95.8	**6.1**	95.8	**0.5**
	Cluster 2	2,428	N/A	97.9	**94.5**	93.9	**45.8**	89.8 ± 0.7	**5.8**	79.2 ± 4.6	**0.6**	60.2 ± 1.7	**0.6**

Table 1. Subpopulation attack on Adult: comparison of test accuracies on subpopulations (%). Target models for Adult dataset consist of models with 0% accuracy on the selected subpopulations (Cluster 0 - Cluster 2). n_p denotes the maximum number of poisoning points used by our attack, and xn_p denotes comparing the two attacks at xn_p poisoning points. n_p is set by running our attack till the induced model becomes 0.01-close to the target model. All results are averaged over 4 runs and standard error is 0 (exact same number of misclassifications across the runs), except where reported. We do not show lower bound for LR because we can only compute an approximate maximum loss difference and the lower bound will no longer be valid.

Model/ Dataset	Target Model	n_p	Lower Bound	$0.2n_p$		$0.4n_p$		$0.6n_p$		$0.8n_p$		n_p	
				KKT	Ours	KKT	Ours	KKT	Ours	KKT	Ours	KKT	Ours
SVM/ MNIST 1-7	5% Error	1,737	874	97.3	**97.1**	96.4	**96.1**	95.7	95.7	94.9	94.9	94.3	94.6
	10% Error	5,458	3,850.4±0.8	95.8	**95.5**	93.4	**92.1**	92.7	**90.9**	91.1	**90.7**	90.2	90.2
	15% Error	6,192	4,904	98.3	**97.8**	96.3	98.1	97.2	97.3	98.3	**92.7**	82.7	85.9
SVM/ Dogfish	10% Error	32	15	97.0	**95.8**	94.0	**93.3**	92.2	**91.2**	90.7	**90.2**	90.3	**89.8**
	20% Error	89	45	95.5	95.7	92.5	**92.2**	90.3	**88.7**	84.7	84.7	82.3	**82.0**
	30% Error	169	83	95.5	95.7	93.5	**90.7**	88.3	**82.5**	78.3	**75.2**	71.8	**71.7**
LR/ MNIST 1-7	5% Error	756	N/A	97.5	**96.9**	97.4	**96.5**	97.2	**96.0**	96.9	**95.7**	96.9	**95.2**
	10% Error	2,113	N/A	97.0	**95.7**	96.9	**93.8**	96.8	**92.3**	96.2	**91.4**	96.4	**90.4**
	15% Error	3,907	N/A	96.9	**95.4**	97.0	**93.3**	97.1	**90.7**	97.1	**88.3**	97.1	**87.1**
LR/ Dogfish	10% Error	62	N/A	98.8	**93.0**	98.5 ± 0.1	**89.7 ± 0.3**	98.8	**89.2**	98.8	**89.2**	98.8	**89.0**
	20% Error	120	N/A	98.5	**93.2**	99.2	**88.2**	99.3	**85.3**	99.5	**83.0**	99.5	**80.7**
	30% Error	181	N/A	97.8	**92.3**	98.8	**85.7 ± 0.2**	99.2	**81.3 ± 0.3**	99.5	**75.7**	99.5	**72.5 ± 0.2**

Table 2. Indiscriminate attack on MNIST 1–7 and Dogfish: comparison of overall test accuracies (%). The target models are of certain overall test errors. n_p is set by running our attack till the induced model becomes ϵ-close to the target model and we set ϵ as 0.1 for MNIST 1–7 and 2.0 for Dogfish dataset. All results are averaged over 4 runs and standard error is 0 (exact same number of misclassifications across the runs), except where reported.

Convergence. Figure 1 shows the convergence of Algorithm 1 using both maximum loss difference and Euclidean distance to the target, and the result is reported on the first subpopulation (Cluster 0) of Adult and the model is SVM. The maximum

number of poisoning points (n_p) for the experiments is obtained when the classifier from Algorithm 1 is 0.01-close to the target classifier. Our attack steadily reduces the maximum loss difference and Euclidean distance to the target model, in contrast to the KKT attack which does not seem to converge towards the target model reliably. Concretely, at the maximum number of poisons in Figure 1, both the maximum loss difference and Euclidean distance of our attack (to the target) are less than 2% of the corresponding distances of the KKT attack. Similar observations are also observed for the indiscriminate and other subpopulation attack settings, see Appendix D.

We believe our attack outperforms the KKT attack in terms of convergence to the target model because it approaches the target classifier differently. The foundation of the KKT attack is that for binary classification, for any target classifier generated by training on a set $D_c \cup D_p$ with $|D_p| = n$, the (exact) same classifier can also be obtained by training on set $D_c \cup D'_p$ with $|D'_p| \leq n$ and this poisoning set D'_p only contains two distinct points, one from each class. In practice, the KKT attack often aims to induce the exact same classifier with much fewer poisoning points, which may not be feasible and leads the KKT attack to fail. In contrast, our attack does not try to obtain the exact target model but just selects each poisoning point in turn as the one with the best expected impact. Hence, our attack gets close to the target model with fewer poisoning points than the number of points used to exactly produce the target model.

Attack Success. Next, we compare the classifiers induced by the two attacks in terms of the attacker's goal. Table 1 summarize the results of the subpopulation attacks, where attack success is measured on the targeted cluster. At the maximum number of poisons, our attack is much more successful than the KKT attack, for both the SVM and LR models. For example, on Cluster 1 with LR, the induced classifier from our attack has 0.5% accuracy compared to the 95.8% accuracy of KKT. Table 2 shows the results of indiscriminate attacks on MNIST 1–7 and Dogfish, and the attack success is the overall test error. For the indiscriminate attack on SVM, both on MNIST 1–7 and Dogfish, the two attacks have similar performance while for LR, our attack is much better than the KKT attack. The reason of KKT failing on LR is, its objective function becomes highly non-convex and is very hard to optimize. More details about the formulation can be found in Koh et al. (2018). For logistic loss, our attack also needs to maximize a non-concave maximum loss difference[3] (Step 4 in Algorithm 1). However, this objective is much easier to optimize than that of the KKT attack and our attack is still very effective on LR models.

Optimality of Our Attack. To check the optimality of our attack, we calculate a lower bound on the number of poisoning points needed to induce the model that is induced by the poisoning points found by our attack. We calculate this lower bound on the number of poisons using Theorem 2 (details in Section 4.3). Note that Theorem 2 provides a valid lower bound based on any intermediate model. To get a lower bound on the number of poisoning points, we only need to use Theorem 2 on the encountered intermediate models and report the best one. We do this by running Algorithm 1 using the induced model (and not the previous target model) as the target model, terminating when the induced classifier is ϵ-close to the given target model. Note that for LR, maximizing the loss difference is not concave and therefore, we cannot obtain the actual maximum loss difference, which is required in the denominator in Theorem 2. Therefore, we only report results on SVM. For the subpopulation attack on Adult, we set $\epsilon = 0.01$ and for the indiscriminate attack on MNIST 1–7 and Dogfish, we set ϵ to 0.1 and 2.0 respectively. We then consider all the intermediate classifiers that the algorithm induced across the iterations. Our calculated lower bound in Table 1 (Column 3-4) shows that for the Adult dataset, the gap between the lower bound and the number of used poisoning points is relatively small. This means our attack is nearly optimal in terms of minimizing the number of poisoning points needed to induce the target classifier. However, for the MNIST 1–7 and Dogfish datasets in Table 2, there still exists some gap between the lower bound and the number of poisoning points used by our attack, indicating there might exist more efficient model-targeted poisoning attacks.

6. Conclusion

We propose a general poisoning framework with provable guarantees to approach any attainable target classifier, along with a lower and upper bound on the number of poisoning points needed. Our attack is a generic tool that first captures the adversary's goal as a target model and then focuses on the power of attacks to induce that model. This separation enables future work to explore the effectiveness of poisoning attacks corresponding to different adversarial goals. We have not considered defenses in this work, and it is an important and interesting direction to study the effectiveness of our attack against data poisoning defenses. Defenses may be designed to limit the search space of the points with maximum loss difference and hence increase the number of poisoning points needed. We also leave the investigation of the application of

[3] We use Adam optimizer (Kingma & Ba, 2014) with random restarts to solve this maximization problem approximately.

our model-targeted attacks in other attacker objectives, e.g. backdoor attacks and privacy attacks, for future work.

Acknowledgements

This work was partially funded by awards from the National Science Foundation (NSF) SaTC program (Center for Trustworthy Machine Learning, #1804603), NSF Office of Advanced Cyberinfrastructure (#2002985) and Amazon research award.

References

Biggio, B., Nelson, B., and Laskov, P. Support Vector Machines under adversarial label noise. In *Asian Conference on Machine Learning*, 2011.

Biggio, B., Nelson, B., and Laskov, P. Poisoning attacks against Support Vector Machines. *arXiv preprint arXiv:1206.6389*, 2012.

Dacrema, M. F., Cremonesi, P., and Jannach, D. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *13th ACM Conference on Recommender Systems*, 2019.

Demontis, A., Melis, M., Pintor, M., Jagielski, M., Biggio, B., Oprea, A., Nita-Rotaru, C., and Roli, F. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *28th USENIX Security Symposium*, 2019.

Diamond, S. and Boyd, S. CVXPY: A Python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research*, 17(1):2909–2913, 2016.

Dua, D. and Graff, C. UCI Machine Learning Repository, 2017. URL http://archive.ics.uci.edu/ml.

Geiping, J., Fowl, L., Huang, W. R., Czaja, W., Taylor, G., Moeller, M., and Goldstein, T. Witches' brew: Industrial scale data poisoning via gradient matching. *arXiv preprint arXiv:2009.02276*, 2020.

Gurobi Optimization, Inc. Gurobi optimizer reference manual, 2020. URL https://www.gurobi.com/documentation/9.1/refman/index.html.

Hong, S., Chandrasekaran, V., Kaya, Y., Dumitraş, T., and Papernot, N. On the effectiveness of mitigating data poisoning attacks with gradient shaping. *arXiv preprint arXiv:2002.11497*, 2020.

Huang, L., Joseph, A. D., Nelson, B., Rubinstein, B. I., and Tygar, J. D. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pp. 43–58, 2011.

Huang, W. R., Geiping, J., Fowl, L., Taylor, G., and Goldstein, T. Metapoison: Practical general-purpose clean-label data poisoning. *arXiv preprint arXiv:2004.00225*, 2020.

Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., and Li, B. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *IEEE Symposium on Security and Privacy*, 2018.

Jagielski, M., Hand, P., and Oprea, A. Subpopulation data poisoning attacks. In *NeurIPS 2019 Workshop on Robust AI in Financial Services*, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, 2017.

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

LeCun, Y. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 1998.

Li, B., Wang, Y., Singh, A., and Vorobeychik, Y. Data poisoning attacks on factorization-based collaborative filtering. In *NeurIPS*, 2016.

Ma, Y., Zhu, X., and Hsu, J. Data poisoning against differentially-private learners: Attacks and defenses. *arXiv preprint arXiv:1903.09860*, 2019.

Mahloujifar, S., Diochnos, D. I., and Mahmoody, M. Learning under p-tampering attacks. In *Algorithmic Learning Theory*, 2018.

Mahloujifar, S., Diochnos, D. I., and Mahmoody, M. The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure. In *AAAI Conference on Artificial Intelligence*, 2019a.

Mahloujifar, S., Mahmoody, M., and Mohammed, A. Universal multi-party poisoning attacks. In *International Conference on Machine Learning*, 2019b.

McMahan, H. B. A survey of algorithms and analysis for adaptive online learning. *The Journal of Machine Learning Research*, 18(1):3117–3166, 2017.

Mei, S. and Zhu, X. The security of Latent Dirichlet Allocation. In *Artificial Intelligence and Statistics*, pp. 681–689, 2015a.

Mei, S. and Zhu, X. Using machine teaching to identify optimal training-set attacks on machine learners. In *AAAI Conference on Artificial Intelligence*, 2015b.

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

Nelson, B., Barreno, M., Chi, F. J., Joseph, A. D., Rubinstein, B. I., Saini, U., Sutton, C. A., Tygar, J. D., and Xia, K. Exploiting machine learning to subvert your spam filter. *LEET*, 8:1–9, 2008.

Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *NeurIPS*, 2018.

Shalev-Shwartz, S. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2): 107–194, 2012.

Solans, D., Biggio, B., and Castillo, C. Poisoning attacks on algorithmic fairness. *arXiv preprint arXiv:2004.07401*, 2020.

Steinhardt, J., Koh, P. W. W., and Liang, P. S. Certified defenses for data poisoning attacks. In *NeurIPS*, 2017.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

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

Wang, Y. and Chaudhuri, K. Data poisoning attacks against online learning. *arXiv preprint arXiv:1808.08994*, 2018.

Xiao, H., Xiao, H., and Eckert, C. Adversarial label flips attack on Support Vector Machines. In *European Conference on Artificial Intelligence*, 2012.

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

Yang, C., Wu, Q., Li, H., and Chen, Y. Generative poisoning attack method against neural networks. *arXiv preprint arXiv:1703.01340*, 2017.

Zhu, C., Huang, W. R., Shafahi, A., Li, H., Taylor, G., Studer, C., and Goldstein, T. Transferable clean-label poisoning attacks on Deep Neural Nets. In *ICML*, 2019.

A. Proofs

In this section, we provide the proofs of the main theorems shown in this paper. For convenience, we restate all the theorems below while also referencing to the main paper.

Before proving the main theorem, we introduce two new definitions and several lemmas to assist with the proof.

Definition 2 (Attainable models)**.** We say θ is C_R-attainable with respect to loss function l and regularization function R if there exists a training set $\mathcal{D}$ such that

$$\theta = \arg\min_{\theta \in \Theta} \frac{1}{|D|} \cdot L(\theta; \mathcal{D}) + C_R \cdot R(\theta)$$

Lemma 1. Let θ_1 and θ_2 be two C_R-attainable parameters for some $C_R > 0$ such that $R(\theta_1) > R(\theta_2)$. Then,

$$\sup_{x,y} \big(l(\theta_2; x, y) - l(\theta_1; x, y)\big) / \big(R(\theta_1) - R(\theta_2)\big) > C_R.$$

Proof. Consider any attainable pairs of (θ_1, θ_2) such that $R(\theta_1) > R(\theta_2)$ and let $\mathcal{D}_1$ to be training set that the training algorithm produces the unique minimizer θ_1. Namely,

$$\theta_1 = \arg\min_{\theta} \frac{1}{|\mathcal{D}_1|} \cdot L(\theta; \mathcal{D}_1) + C_R \cdot R(\theta)$$

Since θ_1 minimizes the total loss on $\mathcal{D}_1$ uniquely, we have

$$\frac{1}{|\mathcal{D}_1|} L(\theta_2; \mathcal{D}_1) + C_R \cdot R(\theta_2) > \frac{1}{|\mathcal{D}_1|} L(\theta_1; \mathcal{D}_1) + C_R \cdot R(\theta_1)$$

By rearranging the above inequality and by an averaging argument, we have

$$\sup_{x,y} \big(l(\theta_2; x, y) - l(\theta_1; x, y)\big) \geq \frac{1}{|\mathcal{D}_1|} L(\theta_2; \mathcal{D}_1) - \frac{1}{|\mathcal{D}_1|} L(\theta_1; \mathcal{D}_1) > C_R \cdot \big(R(\theta_1) - R(\theta_2)\big).$$

Now since $R(\theta_1) > R(\theta_2)$ we have

$$\sup_{x,y} \big(l(\theta_2; x, y) - l(\theta_1; x, y)\big) / \big(R(\theta_1) - R(\theta_2)\big) > C_R.$$

□

Lemma 2. Let F be the family of all C_R-attainable models. For any $\theta_1 \in F$, there is a constant γ where for all $\theta_2 \in F$ we have

$$\sup_{x,y} \big(l(\theta_2; x, y) - l(\theta_1; x, y)\big) + C_R(R(\theta_2) - R(\theta_1)) > \gamma \cdot \sup_{x,y} \big(l(\theta_2; x, y) - l(\theta_1; x, y)\big)$$

where γ is a positive constant related to, θ_1, C_R and other model parameters (fixed for a given classification task).

Proof. We prove the lemma for $\gamma = 1 - C_R/C$ for

$$C = \left(\inf_{\substack{\theta_2 \in F \\ \text{s.t. } R(\theta_1) > R(\theta_2)}} \sup_{x,y} (l(\theta_2; x, y) - l(\theta_1; x, y)) / (R(\theta_1) - R(\theta_2)) \right).$$

First, note that by Lemma 1 we have

$$C > C_R \geq 0. \tag{2}$$

which implies γ is positive. Now we consider two subcases based on the sign of $R(\theta_2) - R(\theta_1)$:

Case 1: $R(\theta_2) - R(\theta_1) \geq 0$. In this case the inequality is straightforward:

$$\sup_{x,y}\big(l(\theta_2;x,y) - l(\theta_1;x,y)\big) + C_R \cdot (R(\theta_2) - R(\theta_1)) \geq \sup_{x,y}\big(l(\theta_2;x,y) - l(\theta_1;x,y)\big)$$
$$> (1 - C_R/C) \cdot \sup_{x,y}\big(l(\theta_2;x,y) - l(\theta_1;x,y)\big),$$

where the last inequality is based on equation 2.

Case 2: $R(\theta_2) - R(\theta_1) < 0$. From the definition of C we have

$$R(\theta_1) - R(\theta_2) \leq \frac{\sup_{x,y}\big(l(\theta_2;x,y) - l(\theta_1;x,y)\big)}{C}.$$

Equivalently, we can say

$$R(\theta_2) - R(\theta_1) \geq -\frac{\sup_{x,y}\big(l(\theta_2;x,y) - l(\theta_1;x,y)\big)}{C}.$$

Replacing $R(\theta_2) - R(\theta_1)$ with the lower bound above completes the proof, namely

$$\sup_{x,y}\big(l(\theta_2;x,y) - l(\theta_1;x,y)\big) + C_R(R(\theta_2) - R(\theta_1)) \geq (1 - C_R/C) \cdot \sup_{x,y}\big(l(\theta_2;x,y) - l(\theta_1;x,y)\big).$$

□

With Definition 2 and the lemmas, we are ready to prove Theorem 4.1 (restating Theorem 1, from Section 4.2):

Theorem 1. *After at most T steps, Algorithm 1 will produce the poisoning set $\mathcal{D}_p$ and the classifier trained on $\mathcal{D}_c \cup \mathcal{D}_p$ is ϵ-close to θ_p, with respect to loss-based distance, $D_{l,\mathcal{X},\mathcal{Y}}$, for*

$$\epsilon = \frac{\alpha(T) + L(\theta_p; D_c) - L(\theta_c; D_c)}{T \cdot \gamma}$$

where, γ is a constant for a given θ_p and classification task, and $\alpha(T)$ is the regret of the online algorithm when the loss function used for training is convex.

The goal of the adversary is to get ϵ-close to θ_p (in terms of the loss-based distance) by injecting (potentially few) number of poisoned training data. The algorithm is in essence an online learning problem and we transform Algorithm 1 into the form of standard online learning problem. Specifically, we adopt the *follow the leader* (FTL) framework to describe Algorithm 1 in the language of standard online learning problem. We first describe the online learning setting considered in this paper and the notion of the regret.

Definition 3. Let $\mathcal{L}$ be a class of loss functions, Θ set of possible models, $A\colon (\Theta \times \mathcal{L})^* \to \Theta$ an online learner and $S\colon (\Theta \times \mathcal{L})^* \times \Theta \to \mathcal{L}$ a strategy for picking loss functions in different rounds of online learning (adversarial environment in the context of online convex optimization). We use $\mathsf{Regret}(A, S, T)$ to denote the regret of A against S, in T rounds. Namely,

$$\mathsf{Regret}(A, S, T) = \sum_{j=0}^{T} l_j(\theta_j) - \min_{\theta \in \Theta} \sum_{j=0}^{T} l_j(\theta)$$

where

$$\theta_i = A\big((\theta_0, l_0), \ldots, (\theta_{i-1}, l_{i-1})\big) \quad \text{and} \quad l_i = S\big((\theta_0, l_0), \ldots, (\theta_{i-1}, l_{i-1}), \theta_i\big).$$

With the online learning problem set up, we proceed to the main proof which first describes Algorithm 1 in the FTL framework.

Proof of Theorem 1. The FTL framework proceeds by solving all the functions incurred during the previous online optimization steps, namely, $A_{\mathsf{FTL}}((\theta_0, l_0), \ldots, (\theta_i, l_i)) = \arg\min_{\theta\in\Theta} \sum_{j=0}^{i} l_i(\theta)$.

Next, we describe how we design the ith loss function l_i in each round of the online optimization. For the first choice, A_{FTL} chooses a random model $\theta_0 \in \Theta$. In the first round (round 0), S_{θ_p} uses the clean training set $\mathcal{D}_c$ and the loss is set as

$$S_{\theta_p}(\theta_0) = l_0(\theta) = L(\theta; \mathcal{D}_c) + N \cdot C_R \cdot R(\theta).$$

According to the FTL framework, A_{FTL} returns model that minimizes the loss on the clean training set $\mathcal{D}_c$ using the structural empirical risk minimization. For the subsequent iterations ($i \geq 1$), the loss functions is defined as, given the latest model θ_i, S_{θ_p} first finds (x_i^*, y_i^*) that maximizes the loss difference between θ_i and a target model θ_p. Namely,

$$(x_i^*, y_i^*) = \arg\max_{(x,y)} l(\theta_i; x, y) - l(\theta_p; x, y)$$

and then chooses the ith loss function as follows:

$$S_{\theta_p}\big((\theta_0, l_0), \ldots, (\theta_{i-1}, l_{i-1}), \theta_i\big) = l_i(\theta) = l(\theta; x_i^*, y_i^*) + C_R \cdot R(\theta).$$

Now we will see how FTL framework behaves when working on these loss functions at different iterations. We use D_p^i to denote the set $\{(x_1^*, y_1^*), \ldots, (x_i^*, y_i^*)\}$. We have

$$\begin{aligned}
\theta_i = A_{\mathsf{FTL}}((\theta_0, l_0), \ldots, (\theta_{i-1}, l_{i-1})) &= \arg\min_{\theta\in\Theta} \sum_{j=0}^{i-1} l_j(\theta) \\
&= \arg\min_{\theta\in\Theta} L(\theta; \mathcal{D}_c) + N \cdot C_R \cdot R(\theta) \\
&\quad + \sum_{j=1}^{i-1} l(\theta; x_i^*, y_i^*) + C_R \cdot R(\theta) \\
&= \arg\min_{\theta\in\Theta} L(\theta; \mathcal{D}_c \cup \mathcal{D}_p^{i-1}) + (N + i - 1) \cdot C_R \cdot R(\theta) \\
&= \arg\min_{\theta\in\Theta} \frac{1}{|\mathcal{D}_c \cup \mathcal{D}_p^{i-1}|} L(\theta; \mathcal{D}_c \cup \mathcal{D}_p^{i-1}) + C_R \cdot R(\theta)
\end{aligned}$$

This means that A_{FTL} algorithm, at each step, trains a new model over the combination of clean data and poison data so far ($i - 1$ number of poisons). Now we want to see what is the translation of the $\mathsf{Regret}(A_{\mathsf{FTL}}, S_{\theta_p}, T)$. If we can prove an upper bound on regret, namely if we show $\mathsf{Regret}(A_{\mathsf{FTL}}, S_{\theta_p}, T) \leq \alpha(T)$ for some function α, then we have

$$\sum_{j=0}^{T} l_j(\theta_j) - \sum_{j=0}^{T} l_j(\theta_p) \leq \sum_{j=0}^{T} l_j(\theta_j) - \min_{\theta\in\Theta} \sum_{j=0}^{T} l_j(\theta) \leq \alpha(T)$$

which implies

$$\begin{aligned}
\sum_{j=0}^{T} l_j(\theta_j) - \sum_{j=0}^{T} l_j(\theta_p) &= L(\theta_c; D_c) - L(\theta_p; D_c) + N \cdot C_R \cdot (R(\theta_c) - R(\theta_p)) \\
&\quad + \sum_{j=1}^{T} l_j(\theta_j) - \sum_{j=1}^{T} l_j(\theta_p) \\
&= L(\theta_c; D_c) - L(\theta_p; D_c) + N \cdot C_R \cdot (R(\theta_c) - R(\theta_p)) \\
&\quad + \sum_{j=1}^{T} \big[\max_{x,y} \big(l(\theta_j; x, y) - l(\theta_p; x, y) \big) + C_R \cdot (R(\theta_j) - R(\theta_p)) \big] \\
&\leq \alpha(T)
\end{aligned}$$

Therefore we have

$$\sum_{j=1}^{T}\left[\max_{x,y}\left(l(\theta_j;x,y)-l(\theta_p;x,y)\right)+C_R\cdot(R(\theta_j)-R(\theta_p))\right]\leq\alpha(T)+L(\theta_p;D_c)-L(\theta_c;D_c)$$
$$+N\cdot C_R\cdot(R(\theta_p)-R(\theta_c))$$

Based on Lemma 2, we further have

$$\sum_{j=1}^{T}\gamma\cdot\left(\max_{x,y}l(\theta_j;x,y)-l(\theta_p;x,y)\right)\leq\alpha(T)+L(\theta_p;D_c)-L(\theta_c;D_c)$$
$$+N\cdot C_R\cdot(R(\theta_p)-R(\theta_c))$$

Above inequality states that average of the maximum loss difference in all previous rounds is bounded from above. Therefore, we know that among the T iterations, there exist an iteration $j^*\in[T]$ (with lowest maximum loss difference) such that the maximum loss difference of θ_{j^*} is ϵ-close to θ_p with respect to the loss-based distance where

$$\epsilon=\frac{\alpha(T)+L(\theta_p;D_c)-L(\theta_c;D_c)+N\cdot C_R\cdot(R(\theta_p)-R(\theta_c))}{T\cdot\gamma}.$$

□

Theorem 1 characterizes the dependencies of ϵ on $\alpha(T)$ and the constant term $L(\theta_p;D_c)-L(\theta_c;D_c)+N\cdot C_R\cdot(R(\theta_p)-R(\theta_c))$. To show the convergence of Algorithm 1, we need to ensure $\epsilon\to 0$ when $T\to+\infty$, which implies we need to show $\alpha(T)\leq O(\sqrt{T})$. Following remark (restating Remark 1 in Section 4.2) and its proof shows the desired convergence.

Remark 1. *Online learning algorithms with sublinear regret bound can be applied to show the convergence. Here, we adopt the regret analysis from McMahan (2017). Specifically, $\alpha(T)$ is in the order of $O(\log T)$) and we have $\epsilon\leq O(\frac{\log T}{T})$ when the loss function is Lipschitz continuous and the regularizer $R(\theta)$ is strongly convex, and $\epsilon\to 0$ when $T\to+\infty$. $\alpha(T)$ is also in the order of $O(\log T)$ when the loss function used for training is strongly convex and the regularizer is convex.*

Our FTL framework formulation can utilize the existing logarithmic regret bound of adaptive FTL algorithm when the objective functions are strongly convex with respect to some norm $\|\cdot\|$, as illustrated in Section 3.6 in McMahan (2017). For clarity in presentation, we first restate their related results below.

Setting 1 (Setting 1 in McMahan (2017))**.** Given a sequence of objective loss functions $f_1,f_2,...,f_i$ and a sequence of incremental regularization functions $r_0,r_1,...,r_i$ we consider an algorithm that selects the response point based on

$$\theta_1=\arg\min_{\theta\in\mathbb{R}^d}r_0(\theta)$$
$$\theta_{i+1}=\arg\min_{\theta\in\mathbb{R}^d}\sum_{j=1}^{i}f_j(\theta)+r_j(\theta)+r_0(\theta),\text{for } i=1,2,...$$

We simplify the summation notation with $f_{1:i}(\theta)=\sum_{j=1}^{i}f_j(\theta)$. Assume that r_i is a convex function and satisfy $r_i(\theta)\geq 0$ for $i\in\{0,1,2,...\}$, against a sequence of convex loss functions $f_i:\mathbb{R}^d\to R\cup\{\infty\}$. Further, letting $h_{0:i}=r_{0:i}+f_{1:i}$ we assume dom $h_{0:i}$ is non-empty. Recalling $\theta_i=\arg\min_\theta h_{0:i-1}(\theta)$, we further assume $\partial f_i(\theta_i)$ is non-empty. We denote the dual norm of a norm $\|\cdot\|$ as $\|\cdot\|_*$.

Theorem 3 (Restatement of Theorem 1 in McMahan (2017))**.** Consider Setting 1, and suppose the r_i are chosen such that $r_{0:i}+f_{1:i+1}$ is 1-strongly-convex w.r.t. some norm $\|\cdot\|_{(i)}$.. If we define the regret of the algorithm with respect to a selected point θ^* as

$$\mathsf{Regret}_T(\theta^*,f_i)\equiv\sum_{i=1}^{T}f_i(\theta_i)-\sum_{i=1}^{T}f_i(\theta^*).$$

Then, for any $\theta^* \in \mathbb{R}^d$ and for any $T > 0$, with $g_i \in \partial f_i(\theta_i)$, we have

$$\mathsf{Regret}_T(\theta^*, f_i) \leq r_{0:T-1}(\theta^*) + \frac{1}{2}\|g_i\|^2_{(i-1),*}$$

Corollary 2 (Formalization of FTL result in Section 3.6 in McMahan (2017))**.** In the FTL framework (no individual regularizer is used in the optimization procedure), suppose each loss function f_i is 1-strongly convex w.r.t. a norm $\|\cdot\|$, then we have

$$\mathsf{Regret}_T(\theta^*, f_i) \leq \frac{1}{2}\sum_{i=1}^{T}\frac{1}{i}\|g_i\|^2_* \leq \frac{G^2}{2}(1+\log T)$$

with $\|g_i\|_* \leq G$.

Proof. The following proof is a restatement of the proof in Section 3.6 in McMahan (2017). The proof follows from Theorem 3. Since we are considering the FTL framework, let $r_i(\theta) = 0$ for all i and define $\|\theta\|_{(i)} = \sqrt{i}\|\theta\|$. Observe that $h_{0:i}$ (i.e., $f_{1:i}$) is 1-strongly convex with respect to $\|\theta\|_{(i)}$ (Lemma 3 in McMahan (2017)), and we have $\|\theta\|_{(i),*} = \frac{1}{\sqrt{i}}\|\theta\|_*$. Then by applying Theorem 3, we have

$$\mathsf{Regret}_T(\theta^*, f_i) \leq \frac{1}{2}\sum_{i=1}^{T}\|g_i\|^2_{(i),*} = \frac{1}{2}\sum_{i=1}^{T}\frac{1}{i}\|g_i\|^2_*$$

Based on the inequality of $\sum_{i=1}^{T} 1/i \leq 1 + \log T$ and if we further assume $\|g_i\|_* \leq G$, then we can have

$$\frac{1}{2}\sum_{i=1}^{T}\frac{1}{i}\|g_i\|^2_* \leq \frac{G^2}{2}(1+\log T)$$

□

Proof of Remark 1. We will prove the logarithmic regret bound in Remark 1 utilizing Corollary 2. First of all, our online learning process fits into Setting 1. Specifically, we set $r_i(\theta) = 0$ for all i. For $f_i(\theta)$, when $1 \leq i \leq N$, we set $f_i(\theta) = \frac{1}{N}L(\theta; \mathcal{D}_c) + C_R \cdot R(\theta)$ (evenly distributing the term $L(\theta; \mathcal{D}_c) + N \cdot C_R \cdot R(\theta)$ across N iterations) and when $i \geq N+1$, we set $f_i(\theta) = l_{i-N}(\theta)$. Details of l_i can be referred from the proof of Theorem 1. Therefore, f_i is 1-strongly convex with respect to a norm $\|\cdot\|$ (the norm is determined by the regularizer $R(\theta)$ and C_R). Further, $l_{0:i}(\theta) = f_{1:N+i}(\theta)$. In addition, the assumption that dom $h_{0:i}$ is non-empty in Setting 1 means when if we train a classifier on the poisoned data set, we can always return a model and hence the assumption is satisfied. The assumption of the existence of subgradient $\partial f_i(\theta_i)$ in Setting 1 is also satisfied by the poisoning attack scenario.

The logarithmic regret of $\mathsf{Regret}(A_{\mathsf{FTL}}, S_{\theta_p}, T)$ of our algorithm then follows from the result of $\mathsf{Regret}_T(\theta^*, f_i)$ in Corollary 2. Specifically, $l_{0:i}(\theta) = f_{1:N+i}(\theta)$ is 1-strongly convex to norm $\|\cdot\|_i = \sqrt{N+i}\|\cdot\|$ and since we assume the loss function is G-Lipschitz, we have $\|g_i\|_* \leq G$. Therefore, we have the logarithmic regret bound as:

$$\mathsf{Regret}(A_{\mathsf{FTL}}, S_{\theta_p}, T) \leq \alpha(T) = \frac{1}{2}\sum_{i=1}^{T}\frac{1}{i+N}\|g_i\|^2_* \leq \frac{1}{2}\sum_{i=1}^{T}\frac{1}{i}\|g_i\|^2_* \leq \frac{G^2}{2}(1+\log T) \leq O(\log T).$$

□

We next provide the proof of the certified lower bound (restating Theorem 2 from Section 4.3):

Theorem 2. *Given a target classifier θ_p, to reproduce θ_p by adding the poisoning set $\mathcal{D}_p$ into $\mathcal{D}_c$, the number of poisoning points $|\mathcal{D}_p|$ cannot be lower than*

$$\sup_{\theta} z(\theta) = \frac{L(\theta_p; \mathcal{D}_c) - L(\theta; \mathcal{D}_c) + NC_R(R(\theta_p) - R(\theta))}{\sup_{x,y}\big(l(\theta; x, y) - l(\theta_p; x, y)\big) + C_R(R(\theta) - R(\theta_p))}.$$

The main intuition behind the theorem is, when the the number of poisoning points added to the clean training set is lower than the certified lower bound, for structural empirical risk minimization problem (shown in equation 1 in the main paper), then target classifier will always have higher loss than another classifier and hence cannot be achieved.

Proof. We first show that for all models θ, we can derive a lower bound on the number of poison points required to get θ_p. Then since these lower bounds all hold, we can take the maximum over all of them and get a valid lower bound. We first show that for any model θ, the minimum number of poisoning points cannot be lower than

$$z(\theta) = \frac{L(\theta_p; \mathcal{D}_c) - L(\theta; \mathcal{D}_c) + NC_R(R(\theta_p) - R(\theta))}{\sup_{x,y} \big(l(\theta; x, y) - l(\theta_p; x, y)\big) + C_R(R(\theta) - R(\theta_p))}.$$

Let us denote the point corresponding to the supremum of the loss difference between θ and θ_p as (x^*, y^*) [4]. Namely, $l(\theta; x^*, y^*) - l(\theta_p; x^*, y^*) = \sup_{x,y} \big(l(\theta; x, y) - l(\theta_p; x, y)\big)$. Now suppose we can obtain θ_p with lower number of poisoning points $\underline{z} < z(\theta)$. Assume there is a poisoning set $\mathcal{D}_p$ with size $\underline{z}$ such that when added to $\mathcal{D}_c$ would result in θ_p. We have

$$\begin{aligned}\sup_{x,y} \big(l(\theta; x, y) - l(\theta_p; x, y)\big) \geq \frac{1}{|\mathcal{D}_c \cup \mathcal{D}_p|} L(\theta; \mathcal{D}_c \cup \mathcal{D}_p) - \frac{1}{|\mathcal{D}_c \cup \mathcal{D}_p|} L(\theta_p; \mathcal{D}_c \cup \mathcal{D}_p) \\ > C_R \cdot \big(R(\theta_p) - R(\theta)\big),\end{aligned}$$

implying $\sup_{x,y} \big(l(\theta; x, y) - l(\theta_p; x, y)\big) + C_R \cdot (R(\theta) - R(\theta_p)) > 0$. Based on the assumption that $\underline{z} < z(\theta)$, and the fact that $\sup_{x,y} \big(l(\theta; x, y) - l(\theta_p; x, y)\big) + C_R \cdot (R(\theta) - R(\theta_p)) > 0$, we have

$$\begin{aligned}\underline{z} \cdot \big(l(\theta; x^*, y^*) - l(\theta_p; x^*, y^*) + C_R(R(\theta) - R(\theta_p))\big) &< z(\theta) \cdot \big(l(\theta; x^*, y^*) - l(\theta_p; x^*, y^*) + C_R(R(\theta) - R(\theta_p))\big) \\ &= L(\theta_p; \mathcal{D}_c) - L(\theta; \mathcal{D}_c) + NC_R(R(\theta_p) - R(\theta)).\end{aligned}$$

where the equality is based on the definition of $z(\theta)$. On the other hand, by definition of (x^*, y^*) for any D_p of size $\underline{z}$, we have

$$L(\theta; D_p) - L(\theta_p, D_p) + \underline{z} \cdot (C_R \cdot R(\theta) - C_R \cdot R(\theta_p)) \leq \underline{z} \cdot \big(l(\theta; x^*, y^*) - l(\theta_p; x^*, y^*) + C_R(R(\theta) - R(\theta_p))\big).$$

The above two inequalities imply that for any set D_p with size $\underline{z}$ we have

$$\frac{1}{|\mathcal{D}_c \cup \mathcal{D}_p|} L(\theta; \mathcal{D}_c \cup \mathcal{D}_p) + C_R \cdot R(\theta) < \frac{1}{|\mathcal{D}_c \cup \mathcal{D}_p|} L(\theta_p; \mathcal{D}_c \cup \mathcal{D}_p) + C_R \cdot R(\theta_p).$$

which indicates that adding $\mathcal{D}_p$ poisoning points into the training set $\mathcal{D}_c$, the model θ has lower loss compared to θ_p, which is a contradiction to the assumption that θ_p has lowest loss on $\mathcal{D}_c \cup \mathcal{D}_p$ and can be achieved. Now, since θ_p needs to have lower loss on $\mathcal{D}_c \cup \mathcal{D}_p$ compared to any classifier $\theta \in \Theta$, the best lower bound is the supremum over all models in the model space Θ. □

Corollary 1. *If we further assume bi-directional closeness in the loss-based distance, we can also derive the lower bound on number of poisoning points needed to induce models that are ϵ-close to the target model. More precisely, if θ_1 being ϵ-close to θ_2 implies that θ_2 is also $k \cdot \epsilon$ close to θ_1, then we have,*

$$\sup_{\theta} z^{'}(\theta) = \frac{L(\theta_p; \mathcal{D}_c) - L(\theta; \mathcal{D}_c) - NC_R \cdot R^* - Nk\epsilon}{\sup_{x,y} \big(l(\theta; x, y) - l(\theta_p; x, y)\big) + C_R \cdot R^* + k\epsilon}.$$

where R^* is an upper bound on the nonnegative regularizer $R(\theta)$.

[4] In practice, the data space $\mathcal{X}$ is a closed convex set and hence, we can find (x^*, y^*) using convex optimization. In other words, as we saw in experiments, calculating the lower bound is possible in practical scenarios.

Proof of Corollary 4.2.1. The lower bound for all ϵ-close models to the target classifier is given exactly as follows:

$$\inf_{\|\theta'-\theta_p\|_{\mathcal{D}_l,\mathcal{X},\mathcal{Y}}\leq\epsilon} \sup_{\theta} \left(z(\theta,\theta') = \frac{L(\theta^{'};\mathcal{D}_c) - L(\theta;\mathcal{D}_c) + NC_R(R(\theta^{'}) - R(\theta))}{\sup_{x,y}\left(l(\theta;x,y) - l(\theta';x,y)\right) + C_R(R(\theta) - R(\theta'))} \right),$$

where $\inf_{\|\theta'-\theta_p\|_{\mathcal{D}_l,\mathcal{X},\mathcal{Y}}\leq\epsilon}$ denotes $\theta^{'}$ is ϵ-close to θ_p in the loss-based distance. However, the formulation above is a min-max optimization problem and hard to analytically compute the lower bound (by plugging the lower bound formula into Algorithm 1. Therefore, we need to make several relaxations such that the lower bound is computable. For any model $\theta^{'}$ that is ϵ-close to θ_p, based on the bi-directional assumption, then θ_p is $k\epsilon$-close to $\theta^{'}$. Therefore we have,

$$L(\theta^{'};\mathcal{D}_c) - L(\theta;\mathcal{D}_c) = L(\theta^{'};\mathcal{D}_c) - L(\theta_p;\mathcal{D}_c) + L(\theta_p;\mathcal{D}_c) - L(\theta;\mathcal{D}_c) \geq -Nk\epsilon + L(\theta_p;\mathcal{D}_c) - L(\theta;\mathcal{D}_c)$$

and

$$\begin{aligned}\sup_{x,y}\left(l(\theta;x,y) - l(\theta^{'},x,y)\right) &= \sup_{x,y}\left(l(\theta;x,y) - l(\theta_p,x,y)\right) + \sup_{x,y}\left(l(\theta_p,x,y) - l(\theta^{'};x,y)\right) \\ &\leq \sup_{x,y}\left(l(\theta;x,y) - l(\theta_p,x,y) + k\epsilon\right.\end{aligned}$$

and the inequalities are all based on the definition of θ_p being $k\epsilon$-close to $\theta^{'}$.

Plugging the above inequalities into the formula of $\sup_{\theta,\theta'}$ for model $\theta^{'}$, and with the assumption that $0 \leq R(\theta) \leq R^*, \forall\theta \in \Theta$, we immediately have

$$\begin{aligned}\sup_{\theta} z(\theta,\theta') &\geq \sup_{\theta} \frac{L(\theta_p;\mathcal{D}_c) - L(\theta;\mathcal{D}_c) - Nk\epsilon + NC_R(R(\theta^{'}) - R(\theta))}{\sup_{x,y}\left(l(\theta;x,y) - l(\theta_p;x,y)\right) - k\epsilon + C_R(R(\theta) - R(\theta'))} \\ &\geq \sup_{\theta}\left(\frac{L(\theta_p;\mathcal{D}_c) - L(\theta;\mathcal{D}_c) - Nk\epsilon - NC_R\cdot R^*}{\sup_{x,y}\left(l(\theta;x,y) - l(\theta_p;x,y)\right) - k\epsilon + C_R\cdot R^*} = z^{'}(\theta)\right).\end{aligned}$$

Since the inequality holds for any $\theta^{'}$, we have

$$\inf_{\|\theta'-\theta_p\|_{\mathcal{D}_l,\mathcal{X},\mathcal{Y}}\leq\epsilon} \sup_{\theta} z(\theta,\theta^{'}) \geq \sup_{\theta} z^{'}(\theta)$$

and hence $z^{'}(\theta)$ is a valid lower bound.

□

Remark 2 (Improving Results in Corollary 1)**.** Assuming $0 \leq R(\theta) \leq R^*$ is not a strong assumption and actually can be satisfied by many common convex models. For example, for SVM model with ℓ_2-regularizer (in fact, applies to any regularizer $R(\theta)$ with $R(\mathbf{0}) = 0$), we have $R(\theta) \leq \frac{1}{C_R}$ and hence $R^* \leq \frac{1}{C_R}$. Moreover, we can further tighten the lower bound by better bounding the term $R(\theta') - R(\theta)$. Specifically, $R(\theta') - R(\theta) = R(\theta') - R(\theta_p) + R(\theta_p) - R(\theta)$ and we only need to have a tighter upper and lower bounds on $R(\theta') - R(\theta_p)$ utilizing some special properties of the loss functions. For the constant k in the bi-directional closeness, we can also compute its value for some specific loss functions. For example, for Hinge loss, we can compute the value based on Corollary 3 in Appendix B.

B. Relating closeness of loss-based distance to closeness of parameters

In theorem below, we show how one can relate the notion of ϵ-closeness in Definition 1 in the main paper to closeness of parameters in the specific setting of hinge loss. We use this just as an example to show that our notion of ϵ-closeness can be tightly related to the closeness of the models.

Theorem 4. Consider the hinge loss function $l(\theta;x,y) = \max(1 - y\cdot\langle x,\theta\rangle, 0)$ for $\theta \in \mathbb{R}^d$ and $x \in \mathbb{R}^d$ and $y \in \{-1,+1\}$. For $\theta,\theta' \in \mathbb{R}^d$ such that $\|\theta\|_1 \leq r$ and $\|\theta'\|_1 \leq r$, if θ is ϵ-close to θ' in the loss-based distance, then, $\|\theta - \theta'\|_1 \leq r\cdot\epsilon$.

Remark 3. In Theorem 4 above with ℓ_2-regularizer, an upper bound on the ℓ_1-norm of θ and θ' is $\sqrt{d/C_R}$. however, the models that we care about in practice usually have smaller norms.

Remark 3 can be obtained by plugging $\mathbf{0} \in \mathbb{R}^d$ and compare the resulting (regularized) optimization loss to the model θ^* that minimizes the model loss.

Proof of Theorem 4. We construct a point x^* as follows:

$$x_i^* = \begin{cases} -\frac{1}{r}, \text{ if } \theta_i > \theta_i', i \in [d] \\ +\frac{1}{r} \text{ if } \theta_i \leq \theta_i', i \in [d] \end{cases}$$

Then we have

$$\langle \theta - \theta', x^* \rangle = \frac{1}{r} \cdot \|\theta - \theta'\|_1 \tag{3}$$

Since $\|\theta\|_1 \leq r$ we have

$$\langle x^*, \theta \rangle \geq -1 \tag{4}$$

and similarly since $\|\theta'\|_1 \leq r$ we have

$$\langle x^*, \theta' \rangle \geq -1. \tag{5}$$

Therefore by Inequalities equation 4 and equation 5 we have

$$l(\theta; x^*, -1) - l(\theta'; x^*, -1) = \max(1 + \langle x^*, \theta \rangle, 0) - \max(1 + \langle x^*, \theta' \rangle, 0) = \langle \theta - \theta', x^* \rangle$$

which by equation 3 implies

$$l(\theta; x^*, -1) - l(\theta'; x^*, -1) = \frac{1}{r} \cdot \|\theta - \theta'\|_1. \tag{6}$$

Now since we know that, $\forall x \in \mathbb{R}^d$, the loss difference between θ and θ' is bounded by ϵ, the bound should also hold for the point $(x^*, -1)$, meaning that

$$\frac{1}{r} \cdot \|\theta - \theta'\|_1 \leq \epsilon.$$

which completes the proof. □

Theorem 5. Consider the hinge loss function $l(\theta; x, y) = \max(1 - y \cdot \langle x, \theta \rangle, 0)$ for $\theta \in \mathbb{R}^d$ and $x \in \mathbb{R}^d$ and $y \in \{-1, +1\}$. For $\mathcal{X} = \{x \in \mathbb{R}^d \colon \|x\|_1 \leq q\}$ and $\mathcal{Y} = \{-1, +1\}$, For any two models $\theta, \theta^{'}$ if $\|\theta - \theta^{'}\|_1 \leq \epsilon$, then θ is $q \cdot \epsilon$-close to $\theta^{'}$ in the loss-based distance. Namely,

$$D_{\ell, \mathcal{X}, \mathcal{Y}}(\theta, \theta') \leq q \cdot \epsilon.$$

Proof. For any given θ and $\theta^{'}$, by triangle inequality for maximum, we have

$$l(\theta; x, y) - l(\theta^{'}, x, y) = \max(1 - y \cdot \langle x, \theta \rangle, 0) - \max(1 - y \cdot \langle x, \theta^{'} \rangle, 0) \leq \max(0, \langle yx, \theta^{'} - \theta \rangle).$$

Therefore, we have

$$\max_{(x,y) \in \mathcal{X} \times \mathcal{Y}} l(\theta; x, y) - l(\theta^{'}; x, y) \leq \max_{(x,y) \in \mathcal{X} \times \mathcal{Y}} \max(0, \langle yx, \theta^{'} - \theta \rangle).$$

Our goal is then to obtain an upper bound of $O(\epsilon)$ for $\max_{(x,y) \in \mathcal{X} \times \mathcal{Y}} \langle yx, \theta^{'} - \theta \rangle$ when $\|\theta - \theta^{'}\|_1 \leq \epsilon$. To maximize $\langle yx, \theta^{'} - \theta \rangle$ by choosing x and y, we only need to ensure that $\operatorname{sign} yx_i = \operatorname{sign} \theta_i, i \in [d]$. Therefore, based on the assumption that $\frac{1}{q}\|x\| \leq 1$ (i.e.,$\frac{1}{q}|x_i| \leq 1, i \in [d]$) we have

$$\max_{(x,y) \in \mathcal{X} \times \mathcal{Y}} \frac{1}{q} \langle yx, \theta^{'} - \theta \rangle = \sum_{i=1}^{d} \frac{1}{q} |x|_i |\theta_i - \theta_i^{'}| \leq \sum_{i=1}^{d} |\theta_i - \theta_i^{'}| = \|\theta - \theta^{'}\|_1 \leq \epsilon,$$

which concludes the proof. □

Corollary 3. For Hinge loss, with Theorem 4 and Theorem 5, if θ is ϵ-close to $\theta^{'}$, then $\theta^{'}$ is $r \cdot q \cdot \epsilon$-close to θ.

C. Instantiating Theorem 1 for the Case of SVM

Here we show how to instantiate Theorem 1 for SVM with exact constants instead of the asymptotic notations. We need to calculate the constant γ to get the exact constant. Imagine the feature domain is $\mathbb{R}^d$. Now we calculate the constant C as follows. Let $i^*_\theta = \arg\min_{i\in[d]} |\theta[i]/\theta_p[i]|$ and $\alpha_\theta = |\theta[i^*_\theta]/\theta_p[i^*_\theta]|$. Let $x^*_\theta \in \mathbb{R}^d$ be a point where is equal to 0 everywhere and is equal to $1/\theta_p[i^*_\theta]$ on the i^* coordinate. We have,

$$l(\theta, x^*_\theta, +1) - l(\theta_p, x^*_\theta, +1) = l(\theta, x^*_\theta, +1) \geq (1 - \alpha_\theta). \tag{7}$$

Now we can calculate C as follows

$$\begin{aligned}
C &= \left(\inf_{\substack{\theta\in F \\ \text{s.t. } R(\theta_p)>R(\theta)}} \sup_{x,y} (l(\theta; x, y) - l(\theta_p; x, y))/(R(\theta_p) - R(\theta)) \right) \\
&\geq \left(\inf_{\substack{\theta\in F \\ \text{s.t. } R(\theta_p)>R(\theta)}} (l(\theta, x^*_\theta, +1) - l(\theta_p, x^*_\theta, +1))/(R(\theta_p) - R(\theta)) \right) \\
\text{(By Inequality 7)} \quad &\geq \inf_{\substack{\theta\in F \\ \text{s.t. } R(\theta_p)>R(\theta)}} \sup_{x,y} \frac{1 - \alpha_\theta}{R(\theta_p) - R(\theta)} \\
\text{(By definition of } \alpha_\theta) \quad &\geq \inf_{\substack{\theta\in F \\ \text{s.t. } R(\theta_p)>R(\theta)}} \frac{1 - \alpha_\theta}{R(\theta_p)(1 - \alpha^2_\theta)} \\
&\geq \inf_{\substack{\theta\in F \\ \text{s.t. } R(\theta_p)>R(\theta)}} \frac{1 - \alpha_\theta}{R(\theta_p)(1 - \alpha^2_\theta)} \\
&\geq \frac{1}{2R(\theta_p)}
\end{aligned}$$

Therefore $\gamma \geq 1 - 2 \cdot C_R \cdot R(\theta_p)$. On the other hand, we can also calculate $\alpha(T)$ based on the exact form given in the proof of Theorem 1.

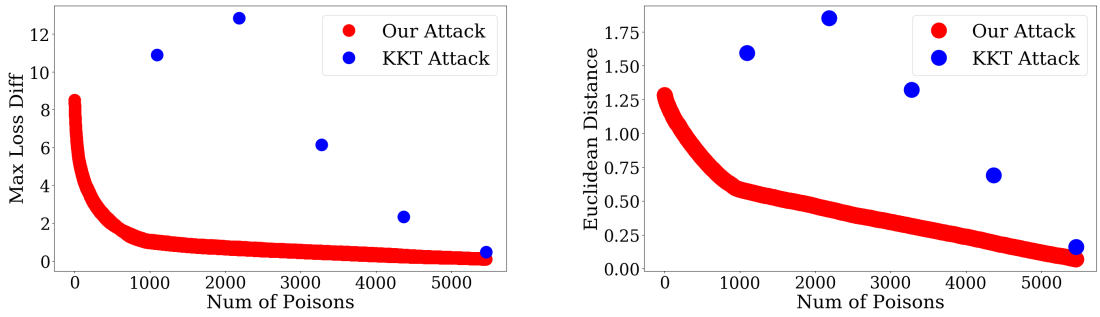
D. Additional Experimental Results

In this section, we provide more results in addition to the results in the main paper. In Section D.1, we show the additional results on SVM model and more results on logistic regression model are given in Section D.2. In Section D.3, we show results on improved target model generation process, which helps to validate the implications we made (below Theorem 1) in the main paper.

D.1. More Results on SVM model

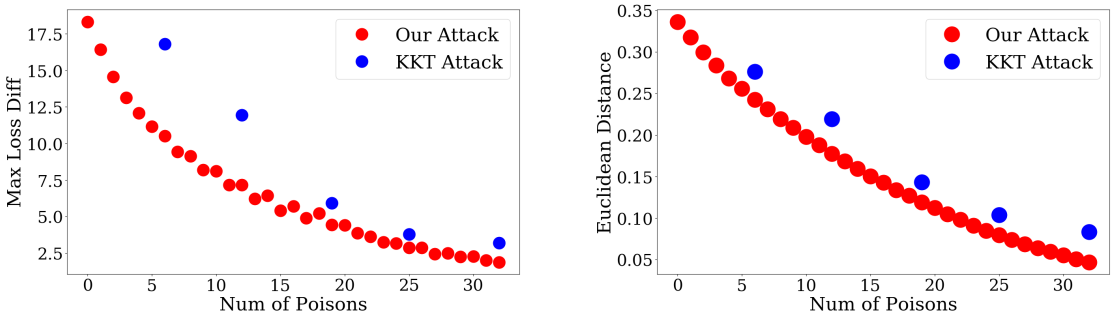
In this section, we first compare our attack to the KKT attack regarding the convergence to the target model. Then compare their attack success in achieving the attacker goals. Last, we provide the lower bound for inducing the model that are induced by our attack and the KKT attack. We use the exact same setup in Section 5 in the main paper regarding the datasets and related models.

Convergence. We show the convergence of Algorithm 1 by reporting the maximum loss difference and Euclidean distance between the classifier induced by the attack and the target classifier. Figures 2 summarizes the results on MNIST 1–7 dataset for the target classifier of 10% error rate. The maximum number of poisoning points in the figure is obtained when the classifier from Algorithm 1 is 0.1-close to the target classifier in the loss-based distance. Figure 3 shows the results on Dogfish dataset with the target classifier of 10% error rate and the maximum number of poisoning points is obtained when the induced classifier is 2.0-close to the target classifier. From the two figures, we observe that classifiers induced by our algorithm steadily converge to the target classifier both in the maximum loss difference and Euclidean distance, while the classifier induced by the KKT attack either cannot converge reliably (Figure 2) or converges slower than our attack (Figure 3). We observe similar observations for other indiscriminate attack settings, and omitted those results here for clarity in presentation.



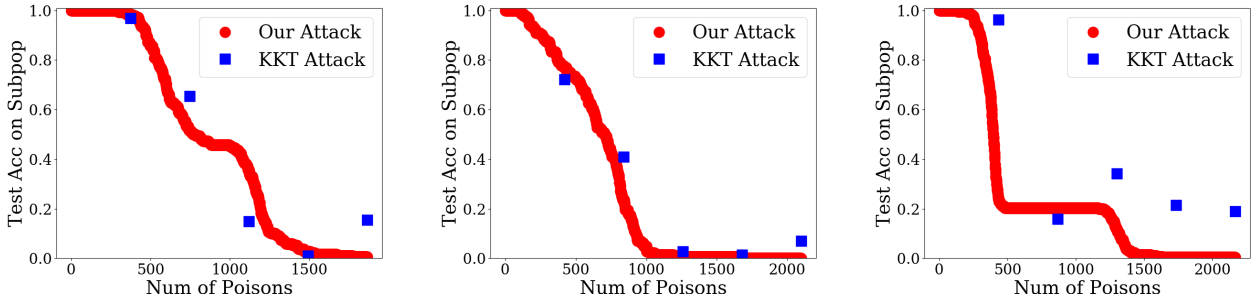
(a) Max Loss Difference (b) Euclidean Distance

Figure 2. SVM on MNIST 1–7 dataset: attack convergence (results shown are for the target classifier of error rate 10%). The maximum number of poisons is set using the 0.1-close threshold to target classifier



(a) Max Loss Difference (b) Euclidean Distance

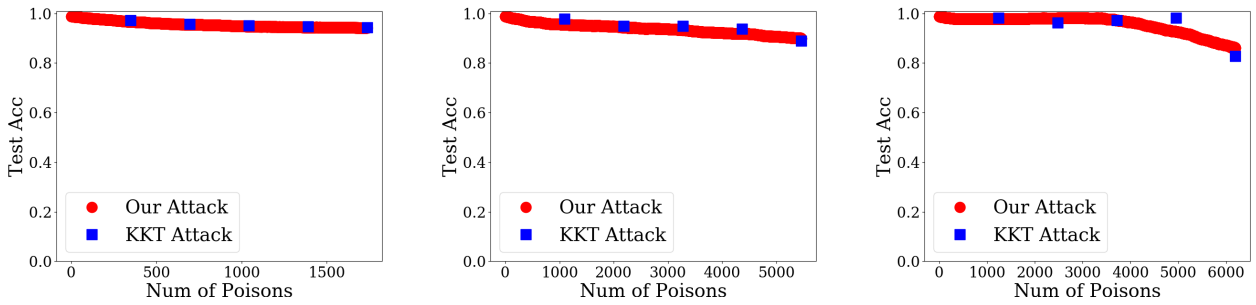
Figure 3. SVM on Dogfish dataset: attack convergence (results shown are for the target classifier of error rate 10%). The maximum number of poisons is set using the 2.0-close threshold to target classifier



(a) Cluster 0 (b) Cluster 1 (c) Cluster 2

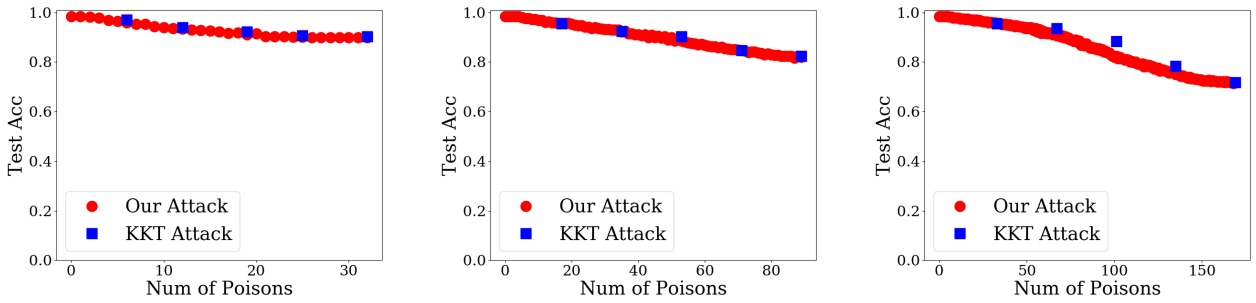
Figure 4. SVM on Adult dataset: test accuracy for each target model of given error rate with classifiers induced by poisoning points obtained from our attack and the KKT attack.

Attack Success. In Figure 4 - Figure 6, we show the attack success of our attack as the number of poisoning points gradually increases. These figures present Table 1 and Table 2 (in the main paper) in the form of figures. The main purpose of these figures is to highlight the online nature of our attack – in contrast to the KKT attack, our attack does not require the number of poisoning points in advance and the attack performance in each iteration can be easily tracked. Besides the online and incremental property, the conclusion from the figures is the same as the conclusion for SVM model in Table 1 and Table 2 – our attack has better attack success than the KKT attack in subpopulation setting and has comparable performance in the



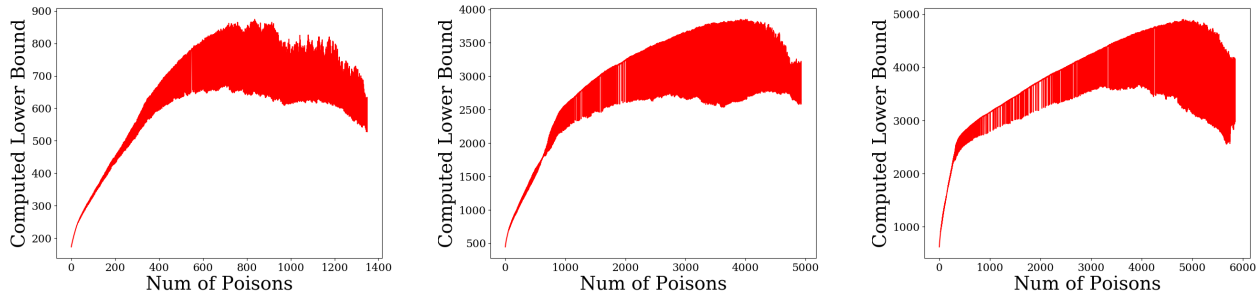
(a) 5% Error Rate (b) 10% Error Rate (c) 15% Error Rate

Figure 5. SVM on MNIST 1–7 dataset: test accuracy for each target model of given error rate with classifiers induced by poisoning points obtained from our attack and the KKT attack.



(a) 10% Error Rate (b) 20% Error Rate (c) 30% Error Rate

Figure 6. SVM on Dogfish dataset: test accuracy of each target model of given error rate with classifiers induced by poisoning points obtained from our attack and the KKT attack.

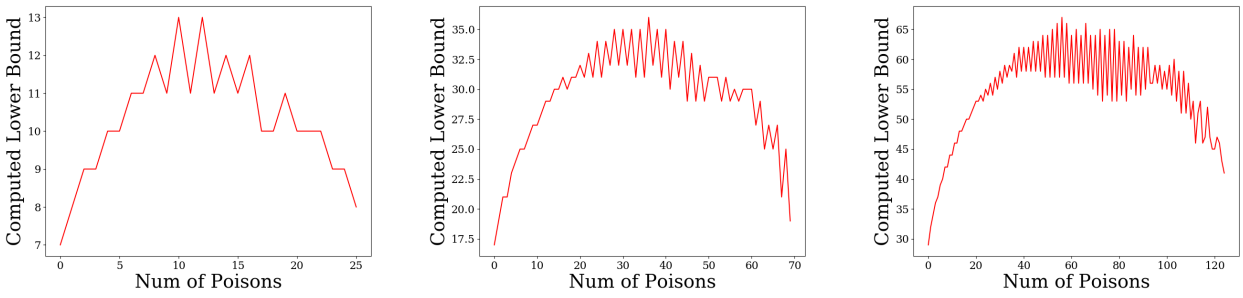


(a) 5% Error Rate (b) 5% Error Rate (c) 15% Error Rate

Figure 7. SVM on MNIST 1–7: lower bound computed in each iteration of running algorithm 1. The target classifier of the algorithm is the classifier induced from our Attack. The maximum number of poisons is obtained when the induced classifier is 0.1-close to the target classifier.

indiscriminate setting.

Lower Bound on Number of Poisons. The lower bounds for SVM in Table 1 and Table 2 in the main paper is obtained by running Algorithm 1 and using the intermediate classifier θ_t to compute the lower bound (with Theorem 2) in each iteration, and returning the highest lower bound computed across all iterations. In this section, we directly plot the computed lower bound in each iteration to show the trend of the lower bound as more number of poisoning points are added. Figure 7 and Figure 8 shows the results on MNIST 1–7 and Dogfish datasets. From the figures, we can easily observe that the peak value

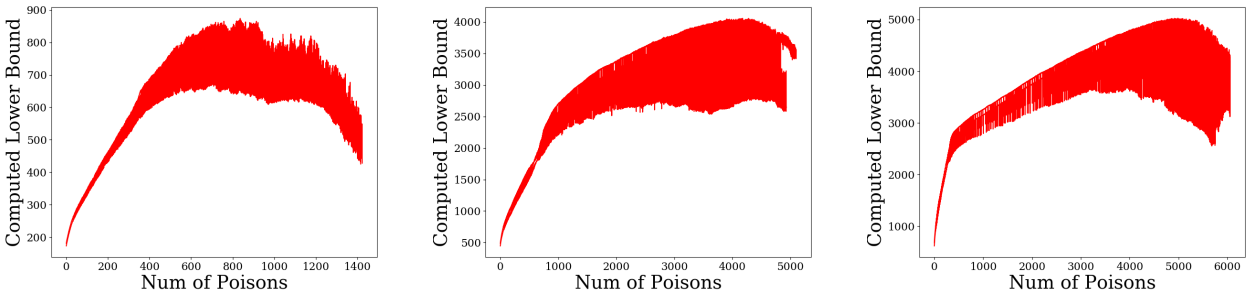


(a) 10% Error Rate (b) 20% Error Rate (c) 30% Error Rate

Figure 8. SVM on Dogfish: lower bound computed in each iteration when running algorithm 1. The target classifier of for the algorithm is the classifier induced from our Attack. The maximum number of poisons is obtained when the induced classifier is 2.0-close to the target classifier.

	5% Error	10% Error	15% Error
# of Poisons	1737	5458	6192
Lower Bound	856	4058.4±1.4	5031.4±4.8

Table 3. SVM on MNIST 1–7: poisoning points needed to achieve target classifiers induced from the KKT attack. Top row means number of poisoning points used by the KKT attack. Bottom row means the lower bound computed from Theorem 2 for the target classifier, which is the model induced by the KKT attack. All results are averaged over 4 runs, integer value in the cell means we get exactly same value for 4 runs and others are shown with the average and standard error.

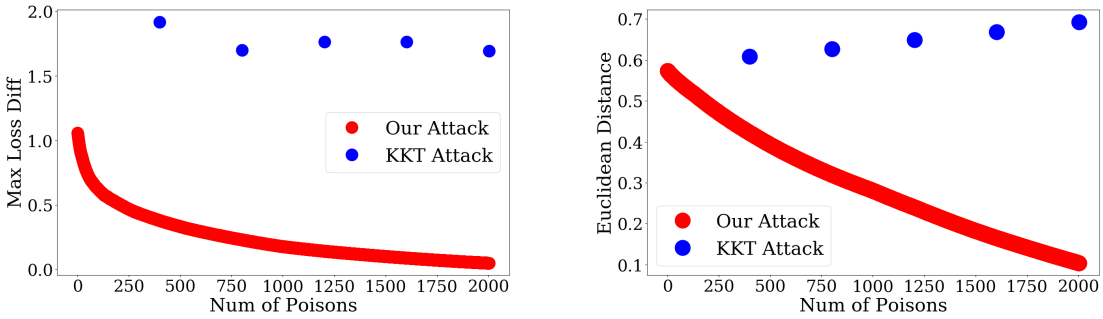


(a) 5% Error Rate (b) 5% Error Rate (c) 15% Error Rate

Figure 9. SVM on MNIST 1–7: lower bound computed in each iteration of running algorithm 1 when the target classifier of the algorithm is the classifier induced from the KKT Attack. The maximum number of poisons is set using the 0.1-close threshold to KKT induced classifier.

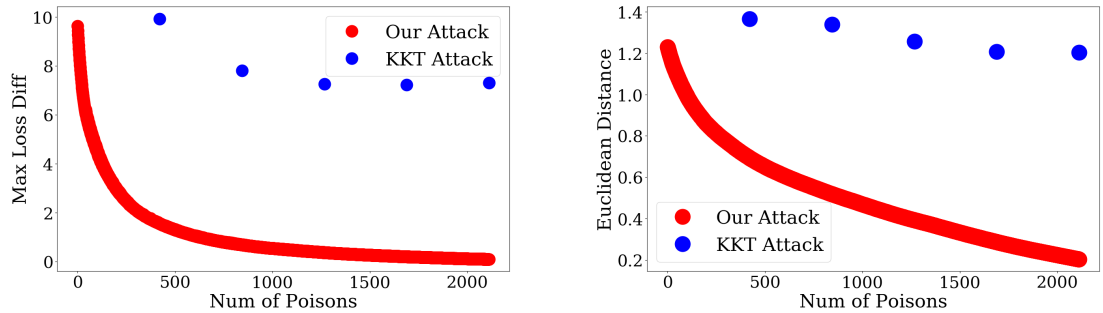
of the lower bound is obtained in the middle of the attack process. Therefore, it might be the case that the computed lower bound is already very tight, as we cannot improve the highest lower bound by running the attack for more iterations. This implies that, it is more likely that our attack is not very optimal on these two datasets and we should seek for more efficient data poisoning attacks. We did not show the curves for Adult dataset because the gap between the lower bound and the number of poisoning points used by our attack is small, indicating our attack is nearly optimal.

For completeness, we also repeat the same experiment, but now with the model induced from the KKT attack as the target model for our attack to compute its lower bound. In Table 3 and Figure 9, we report the lower bound results on MNIST 1–7 dataset. Table 3 shows the highest computed lower bound and Figure 9 plots the lower bound computed in each iteration. The conclusion is still the same as our attack – there still exists a large gap between the lower bound and the number of poisoning points used by the KKT attack, which indicates that the KKT attack is also not very efficient. We have similar observations on the Dogfish dataset using the KKT attack.



(a) Max Loss Difference (b) Euclidean Distance

Figure 10. Logistic regression model on Adult: attack convergence (results shown are for the first subpopulation, Cluster 0). The maximum number of poisons is set using the 0.05-close threshold to target classifier.



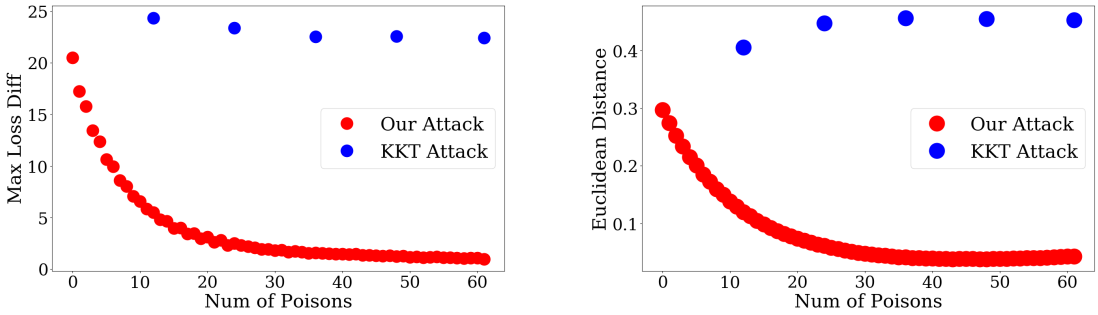
(a) Max Loss Difference (b) Euclidean Distance

Figure 11. Logistic regression model on MNIST 1–7 dataset: attack convergence (results shown are for the target classifier of error rate 10%). The maximum number of poisons is set using the 0.1-close threshold to target classifier.

D.2. More Results on Logistic Regression

In this section, we provide additional results on the logistic regression model. The experiment setup is as the same as in Section D.1. Compared to SVM, we do not report the lower bound results for logistic regression because the maximum loss difference found for logistic regression is an approximate solution and hence the lower bound can be invalid. In what follows, we first discuss the impact of approximate maximum loss difference and then show results on the attack convergence and attack success.

Approximate Maximum Loss Difference. The convergence guarantee in the paper also holds for logistic regression model (more generally, holds for any Lipschitz and convex function with strongly convex regularizer). However, for logistic regression, we may not be able to efficiently search for the globally optimal point with maximum loss difference (Line 4 in Algorithm 1) because the difference of two logistic losses is not concave. Therefore, we adopt gradient descent strategy, using the Adam optimizer (Kingma & Ba, 2014) to search for the point that (approximately) maximizes the loss difference. This is in contrast to the SVM model, where the difference of Hinge loss is piece-wise linear and we can deploy general (convex) solvers to search for the globally optimal point in each linear segment (Diamond & Boyd, 2016; Gurobi Optimization, Inc., 2020). However, as will be demonstrated next, poisoning points with approximate maximum loss difference can still be very effective. More formally, if the approximate maximum loss difference $\hat{l}$ found from local optimization techniques is within a constant factor from the globally optimal value l^* (i.e., $\hat{l} \geq \alpha l^*, 0 < \alpha < 1$), then we still enjoy similar convergence guarantees. A similar issue of global optimality also applies to the KKT attack (Koh et al., 2018), where the attack objective function is no longer convex for logistic regression models, and therefore, we also utilize gradient based technique to (approximately) solve the optimization problem and present the results below.

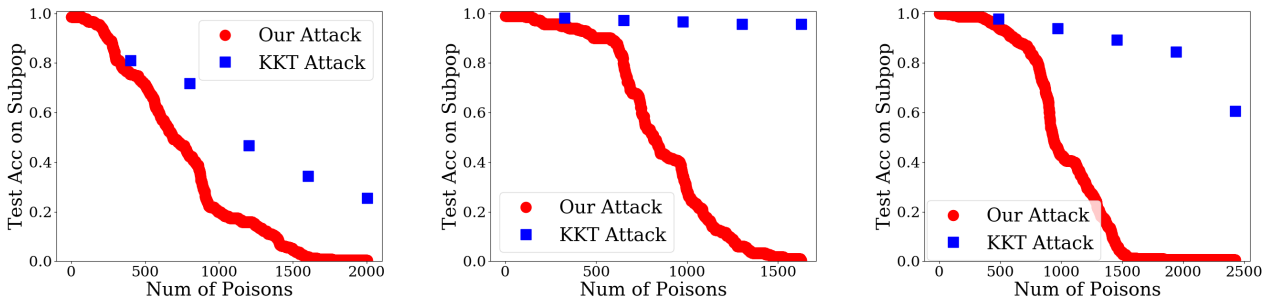


(a) Max Loss Difference (b) Euclidean Distance

Figure 12. Logistic regression model on Dogfish: attack convergence (results shown are for the target classifier of error rate 10%). The maximum number of poisons is set using the 1.0-close threshold to target classifier.

Convergence. The results results for logistic regression on Adult, MNIST 1–7 and Dogfish datasets are show in Figure 10, Figure 11 and Figure 12 respectively. For the Adult dataset, we show the convergence on the first subpopulation (cluster 0). For MNIST 1–7 and Dogfish, similar to Section D.1, we show the convergence on the target models of 10% error rates. All results show that, our attack steadily converges to the target model while the KKT attack fails to have a reliable convergence. Similar observations are also found in other settings (i.e., different clusters for the subpopulation setting and different target models in the indiscriminate settings).

Attack Success. The attack success results on Adult, MNIST 1–7 and Dogfish datasets are show in Figure 13, Figure 14 and Figure 15 respectively. These figures present the logistic regression results in Table 1 and Table 2 (in the main paper) in the form of figures. All the results show that our attack is much more effective than the KKT attack on logistic regression models, and in fact, the KKT attack cannot effectively poison the models in most cases. In addition, our attack runs in an online fashion and we can easily track the attack performance in each iteration.

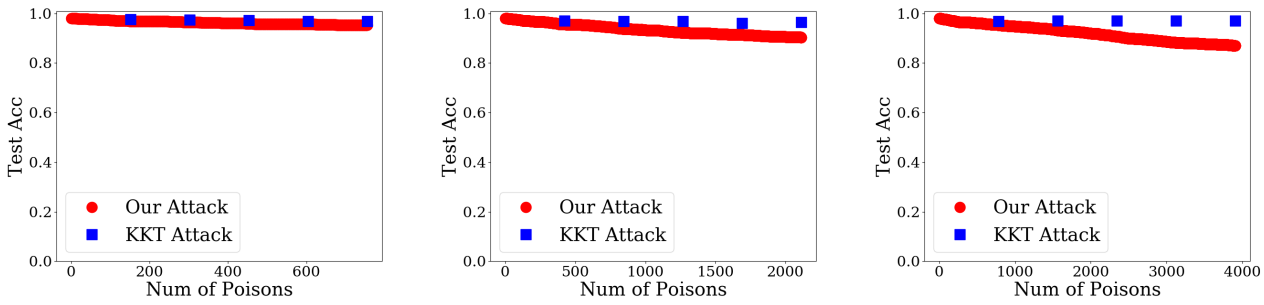


(a) Cluster 0 (b) Cluster 1 (c) Cluster 2

Figure 13. Logistic regression model on Adult: test accuracy for each subpopulation with classifiers induced by poisoning points obtained from our attack and the KKT attack.

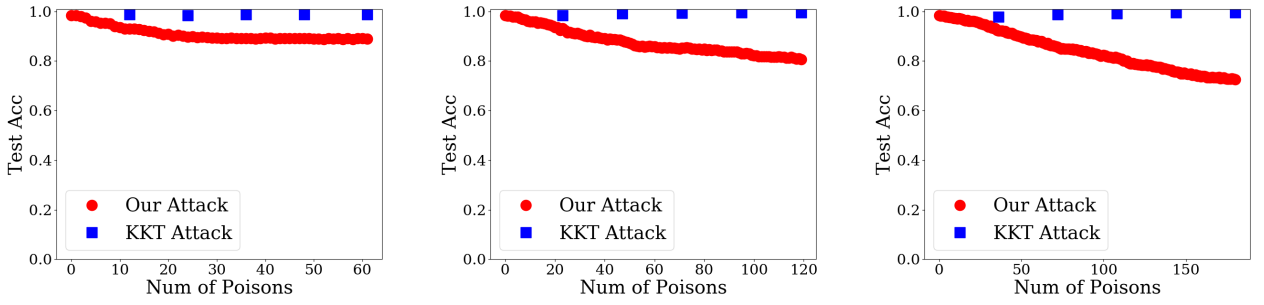
D.3. Improved Target Generation Process

The original heuristic approach in Koh et al. (2018) works by finding different quantiles of training points that have higher loss on the clean model, flipping their labels, repeating those points for multiple copies, and adding them to the clean training set. We find that, in the process of trying different quantiles and copies of high loss points, if we also adaptively update the model where the high loss points are found (instead of just always fixing it to be the clean model), we can generate a target classifier that still satisfies the attack objective but with much lower loss on the clean training. Such an improved generation process can significantly reduce the number of poisoning points needed to reach the same ϵ-closeness (with respect to the loss-based distance) to the target classifier, consistent with the claims in Theorem 1 in the main paper. In addition, we find



(a) 5% Error Rate (b) 10% Error Rate (c) 15% Error Rate

Figure 14. Logistic regression model on MNIST 1–7: test accuracy for each target model of given error rate with classifiers induced by poisoning points obtained from our attack and the KKT attack.



(a) 10% Error Rate (b) 20% Error Rate (c) 30% Error Rate

Figure 15. Logistic regression model on Dogfish: test accuracy of each target model of given error rate with classifiers induced by poisoning points obtained from our attack and the KKT attack.

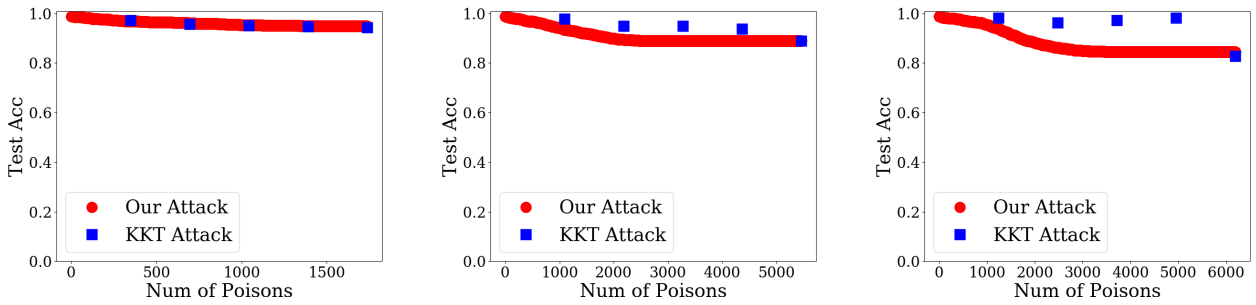
that, if we compare our attack with improved generation process to the KKT attack with the original generation process (Koh et al., 2018), we can also reach the desired target error rate much faster using our attack.

Implication of Theorem 1. We first empirically validate the implication of Theorem 1 in the main paper: to obtain the same ϵ-closeness in loss-based distance, a target classifier with lower loss on the clean training set $\mathcal{D}_c$ requires fewer poisoning points. Therefore, when adversaries have multiple target classifiers that satisfy the attack goal, the one with lower loss on clean training set is preferred.

We run experiments on the SVM and the MNIST 1–7 dataset. For both the original and improved target generation methods, we generate three target classifiers with error rates of 5%, 10% and 15%. The original target classifier generation method returns classifiers with test accuracy of 94.0%, 88.8% and 82.3% respectively (also used in the previous experiments on

Target Models	Test Acc (%)		Loss on Clean Set		# of Poisons	
	Original	Improved	Original	Improved	Original	Improved
5% Error	94.0	94.9	2254.6	1767.1	2170	1340
10% Error	88.8	88.9	4941.0	3233.1	5810	2432
15% Error	83.3	84.5	5428.4	4641.6	6762	3206

Table 4. SVM on MNIST 1–7: comparison of two target generation methods on number of poisoning points used to reach 0.1-closeness to the target. *Original* indicates the original target generation process from Koh et al. (2018). *Improved* denotes our improved target generation process with adaptive model updating.



(a) 5% Error Rate (b) 10% Error Rate (c) 15% Error Rate

Figure 16. SVM on MNIST 1–7: test accuracy with classifiers obtained from our attack and KKT attack. Target model for KKT attack is generated from the original generation process and target model for our attack is generated from the improved generation process. Maximum number of poisoning points is obtained by running our attack with target model generated from the original process and resultant classifier is 0.1-close to the target.

indiscriminate attack). The improved target generation process returns target classifiers with approximately the same test accuracy (94.9%, 88.9% and 84.5%). However, for classifiers of same error rate returned from the two target generation processes, the improved generation method produces classifiers with significantly lower loss compared to the original one.

Table 4 compares the two target generation approaches by showing the number of poisoning points needed to get 0.1-close to the corresponding target model of same error rate. For example, for target models of 15% error rate, the model from the original approach has a total clean loss of 5428.4 while our improved method reduces it to 4641.6. With the reduced clean loss, getting 0.1-close to the target model generated from our improved process only requires 3206 poisoning points, while reaching the same distance from the target model produced by the original method would require 6762 poisoning points, a more than 50% reduction.

End-to-End Comparison. Figure 16 compares the two attacks in an end-to-end manner in terms of their attack success (show as the overall test accuracy after poisoning). With the improved target generation process, our attack can achieve the desired error rate much faster than the KKT attack with the original process. For the KKT attack with target model generated from the original process, we determine the target number of poisoning points by running our attack with 0.1-closeness as the stopping criteria and the model generated from the original process as the target classifier. To run our attack with improved generation process, we terminate the algorithm when the size of the poisoning points is same as the number of poisoning points used by the KKT attack with original process. Such a termination criteria helps us to ensure that both attacks use same number of poisoning points and can be compared easily. We also evaluate the KKT attack on fractions of the maximum target number of poisoning points (0.2, 0.4, 0.6, and 0.8), as in the previous experiments. The accuracy plot shows that our attack (with improved target model) can achieve the desired error rate (e.g., 10% and 15%) much faster than the KKT attack (with original target model). For example, for the attacker objective of having 15% error rate, with target classifier of error rate of 15% error, our attack can achieve the attacker goal much faster than the KKT attack.

E. Comparison of Model-Targeted and Objective-Driven Attacks

Although model-targeted attacks work to induce the given target classifiers by generating poisoning points, the end goal is still to achieve the attacker objectives encoded in the target models. In terms of the comparison to the objective-driven attacks, we first demonstrate that objective-driven attacks can be used to generate a target model, which can then be used as the target for a model-targeted attack, resulting in an attack that achieves the desired attacker objective with fewer poisoning points. Then, we show that to have competitive performance against state-of-the-art objective-driven attacks (e.g., the min-max attack (Steinhardt et al., 2017)), the target classifiers should be generated carefully, such that the attacker objectives of the target classifiers can be achieved efficiently with model-targeted attacks using fewer poisoning points. Although the investigation of a systematic approach to generate such "desired" classifiers is out of the scope of this paper, in the indiscriminate setting, we have some empirical evidence. Specifically, we find that target classifiers with a lower loss on the clean training set and higher error rates (higher than what are desired in the attacker objectives) often require fewer poisoning points to achieve the attacker objectives. The following experiments are conducted on the MNIST 1–7 dataset.

Attacker Objectives	5% Error	10% Error	15% Error
Label-flipping Attack	6,510	8,648	10,825
Our Attack	1,737	5,458	6,192

Table 5. Generate target classifiers using objective-driven label-flipping attacks and achieve similar attacker objectives using our attack with fewer poisoning points. The attacker objectives are to increase the test error to certain amounts (i.e., 5%, 10% and 15%) and the target classifiers to our attack are generated by running the label-flipping attacks with given attacker objectives.

Target Models Generated from Objective-driven Label-Flipping Attacks. In our experiments, the target classifiers are generated from the label-flipping based objective-driven attacks that are effective but need too many poisoning points to achieve their objective. Then, our attacks are deployed to achieve the same objective with fewer poisoning points. Table 5 shows the number of poisoning points used by the label-flipping attack described in Koh et al. (2018) and our model-targeted attack, to achieve desired attack objectives of increasing the test error to a certain amount. We can see that using our attack, the number of poisoning points used by label-flipping attacks can be saved up to 73%.

Comparison to Objective-driven Attacks. Still using target classifiers generated from label-flipping attacks, we show that our attack can outperform existing objective-driven attacks (including the state-of-the-art min-max attack (Steinhardt et al., 2017)) at reducing the overall test accuracy, under the same amount of poisoning points. We still experiment on the SVM model and the MNIST 1–7 dataset. Since we aim to produce target classifiers with lower loss on clean training set and higher error rates, we adopt the improved target model generation process described in Section D.3 (helps to reduce the loss on clean training set) and generate a classifier of 15% error rate. With the target model, we terminate our attack when a fixed number of poisoning points are generated, and then compare the attack effectiveness to existing objective-driven attacks under same number of poisoning points. We compare the test accuracies of all attacks at poisoning ratios of 5%, 15% and 30%. We also modified the baseline objective-driven attacks slightly for a fair comparison:

1. The min-max attack (Steinhardt et al., 2017) and the gradient attack (Koh & Liang, 2017) consider evading defenses during the attack process, which degrades their effectiveness. We simply remove those defenses in our evaluation.
2. Since the generated poisoning points should be valid normalized images in [0,1] range (need not be semantically meaningful), we clip their generated poisoning points into the [0,1] range.
3. The attacks by Biggio et al. (2011); Demontis et al. (2019) use validation data to compute gradients. However, our splits only contain training and testing data, To avoid leaking test-data information or using gradients from data already used to train the model, we create a 70:30 train-validation split using the original training data: this new 70% of the training data is used while the adversary trains its models, and the remaining 30% is used as validation data for gradient computations. The victim then trains the model on the mixture of the original (100%) training data and the generated poisoning points.

We note that the gradient attacks in Biggio et al. (2011); Demontis et al. (2019) are extremely slow to run on MNIST 1–7 dataset (when we use the full training set) because the poisoning points are generated sequentially and the computational cost in each step of generation is very high. Therefore, we choose to improve the attack efficiency by repeating each generated poisoning point N times and produce the desired number of poisoning points faster. We set $N = 10$ for the attack on Logistic regression by Demontis et al. (2019) and $N = 100$ for the attack on SVM by Biggio et al. (2011) (still took 3 days to finish on the linear SVM model). For the attack by Demontis et al. (2019), we compared the setting of $N = 10$ to the default setting of $N = 1$ for poisoning ratios of 5% and 10% [5], and did not find a significant degradation in the attack effectiveness when $N = 10$ (the attack effectiveness drops by 0.3% at most). This might be explained by the size of the training dataset: the impact of just one data point in nearly 13,000 (and even more, once the poison data generation starts) might not vary significantly across iterations, which results in adding multiple copies of the same poison data to be a fair approximation while giving a significant speedup in the runtime. We did not repeat this comparison for the attack from Biggio et al. (2011) because running it for the case of $N = 1$ is simply infeasible.

The results are summarized in Table 6. From the table, we observe that, compared to the existing objective-driven attacks, our attack reduces more on the test accuracy under the same poisoning budget and the gap becomes larger when the poisoning

[5] We did not compare $N = 1$ and $N = 10$ for larger poisoning ratios because the $N = 1$ case will take too long to finish.

Attack/Model	5% Poison Ratio	15% Poison Ratio	30% Poison Ratio
Min-Max Attack (Steinhardt et al., 2017)/SVM	97.0%	93.9%	92.9%
Biggio et al. (2011)/SVM	98.7%	98.2%	96.8%
Koh & Liang (2017)/SVM	98.7%	98.0%	97.2%
Our Attack/SVM	**96.2%**	**88.6%**	**84.3%**
Demontis et al. (2019)/LR	98.2%	97.6%	95.7%
Our Attack /LR	**96.5%**	**89.1%**	**83.1%**

Table 6. Comparison of our attack to objective-driven attacks with different poisoning ratios. The target model of our attack is of 15% error rate. The poisoning ratio is with respect to the full training set size of 13,007. Each cell in the table denotes the test accuracy of the classifier after poisoning. The clean test accuracies of SVM and LR models are 98.9% and 99.1% respectively.

	SVM			Logistic Regression		
	Cluster 0	Cluster 1	Cluster 2	Cluster 0	Cluster 1	Cluster 2
Our Attack	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**
Label-Fipping	31.4%	2.8%	15.5%	15.9%	14.0%	19.1%

Table 7. Comparison of our attack to the label-flipping based subpopulation attack. The table compares the test accuracy on subpoplation of Adult dataset under same number of poisning points. The number of poisons are determined when our attack achieves 0% test accuracy on the subpopulation. Cluster 0-3 in the logistic regression and SVM models denote different clusters. For logistic regression, number of poisoning points for Cluster 0-3 are 1,575, 1,336 and 1,649 respectively. For SVM, number of poisoning points for Cluster 0-3 are 1,252, 1,268 and 1,179 respectively.

ratio increases. At 5% of poisoning ratio, our attack outperforms all baseline attacks by at least 0.8% in terms of the reduced test accuracy, and this gap increases to at least 8.6% at 30% of poisoning ratio.

Comparison to the Label-Flipping Subpopulation Attack. We also compare our attack to the label-flipping subpopulation attack from Jagielski et al. (2019). This attack works by randomly sampling fixed number (constrained by the poisoning budget) of instances from the training data of the subpopulation, flipping their labels and then injecting them into to the original training set. Although this attack is very simple, it shows relatively high attack success when the goal is to cause misclassification on the selected subpopulation (Jagielski et al., 2019).

To be consistent with our experiments in Section 5, we assume the attacker objectives are still to induce a model that has 0% accuracy on a selected subpopulation. For each of the SVM and logistic regression models, we selected the three subpopulations with highest test accuracy (all end up having have 100% accuracy). In indiscriminate setting, we already observed that models with lower loss on clean training set and larger overall error rates can achieve attacker objectives of smaller error rates faster. However, to leverage this observation into our subpopulation experiments, one challenge is the attacker objective is to have 100% test error on the subpopulation, but no classifiers can have test errors larger than 100%. To tackle this, we select models with larger loss on training samples from the subpopulation, with a hope that this process is "equivalent" to selecting target models with larger error rates (on subpopulation) than 100%. To this end, we heuristically select targeted models that satisfy the attacker objective, have larger loss on the training data from the subpopulation, and have relatively low loss on the entire clean training set. Empirically, this selection strategy works better than the original target generation process (as done in Section 5) in achieving the attacker objectives. A more detailed and systematic investigation of the target model search process is left as the future work.

To check the effectiveness of achieving the attacker objectives, we first run our attack and terminate when our attack achieves the attacker objective to have 0% accuracy on the selected subpopulation, and record the number of poisoning points used. Then, we run the random label-flipping attack with the same number of poisoning points. For both attacks, we report the final test accuracies of the resulting models on the subpopulations.

The attack comparisons on different subpopulation clusters and models are given in Table 7. Results in the table compare our attack and the label-flipping attack over the three distinct subpopulation clusters for the SVM and logistic regression models. Across all settings, our attack is considerably more successful. The number of poisoning points needed to reach the 0% accuracy goal is small compared to the entire training set size (e.g., the maximum poisoning ratio is only 10.5%). The

gap between our attack and the label-flipping attack is fairly small. For example, for Cluster 1 in the SVM experiment, the label-flipping attack is also quite successful and reduces the test accuracy to 2.8% (our attack achieves 0% accuracy). We believe the success of label-flipping attack is due to the following two reasons. First, label-flipping in the subpopulation setting can be successful because smaller subpopulations show some degree of locality and hence, injecting points (from the subpopulation) with flipped labels can have a strong impact on the selected subpopulation. This is confirmed by empirical evidence that increasing the subpopulation size (i.e., reducing its locality) gradually reduces the label-flipping effectiveness and the attack becomes much less effective in the indiscriminate setting (i.e., subpopulation is the entire population). Second, the Adult dataset only contains 57 features, where 53 of them are binary features with additional constraints. Therefore, the benefit from optimizing the feature values is less significant as the optimization search space of our attack is fairly limited.

F. Attacks on Deep Neural Networks

The theoretical guarantees of our proposed algorithm require convexity of the model loss. They do not hold for non-convex models such as deep neural networks (DNN). However, we hypothesize that our method of picking poisoning points incrementally might still perform well on non-convex models. Here, we report some preliminary results attacking DNNs.

Several poisoning attacks have been proposed for DNNs. However, all attacks with publicly available source code focus on causing misclassification for a given single instance (Shafahi et al., 2018; Zhu et al., 2019; Huang et al., 2020; Geiping et al., 2020), while we are more interested in the practical sub-population setting. Therefore, we use a random label-flipping attack as our baseline in the sub-population setting. The label-flipping attack selects a random image from the dataset in the targeted sub-population class without replacement and changes its label.

For these experiments, we use the MNIST 1–7 dataset and conduct sub-population poisoning attacks where the targeted sub-population is the class 1 (so, the adversary's goal is to have test images that would be correctly classified as 1 digits, classified as 7s). We compare the poisoning effectiveness in reducing the classification accuracy for the 1 class of our algorithm to the random label-flipping attack. We implement our attack for DNNs with the cross-entropy loss as our loss function. We conduct experiments poisoning a non-convex three-layer neural network with non-linear activation functions, a multilayer perceptron (MLP).[6]

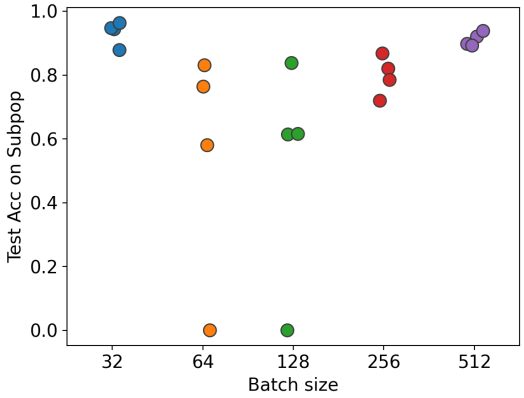
We observe that direct implementations of these poisoning attacks do not work. Via experiments to understand the kind of randomness introduced by varying hyper-parameters like batch-size and weight-initialization, we take steps to remove these sources of variation (Section F.1). Then, we describe modifications to our attack to make it work on DNNs (Section F.2). Finally, we show results with the modified attack and how it performs well when one of these assumptions is relaxed (Section F.3), giving us some hope for the possibility of relaxing other assumptions as well.

F.1. Simplifying the Setting

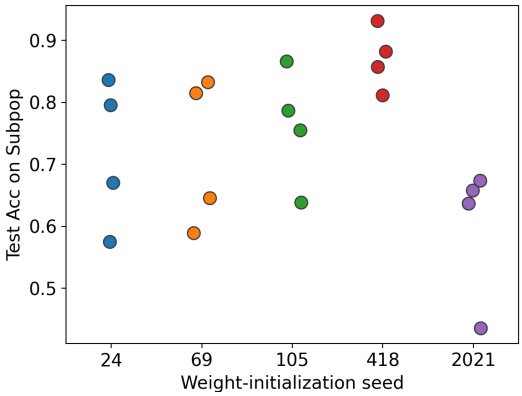
Both our attack and the label-flipping attack on DNNs are observed to be highly sensitive to hyper-parameters like batch-size, weight-initialization, and even randomness induced by the ordering of batches across epochs. As shown in Figure 17a, for the same initialization for model weights and batch-size, different runs of the label-flipping attack with the same poisoning ratio lead to wildly varying error rates. Figure 17b shows that, even when we only vary the model weight-initialization and keep other hyper-parameters fixed, attack effectiveness fluctuates significantly across different random weight initializations.

To better compare our attack with the label-flipping baseline reliably, we design our experiments by not batching the data (*i.e.* batch-size is the same as dataset size). The target model θ_p is trained with the label-flipping attack with fixed weight-initialization and no batching. Additionally, we ensure that the weight-initialization used to generate the intermediate model θ_t in each iteration of our attack is the same as the weight-initialization to train the target model θ_p. Using a different weight initialization in each round of retraining interferes with model convergence and leads to unstable results. This way, we can substantially eliminate randomness introduced by batching data and different model weight initializations.

[6]We also tested our attack on convolutional neural networks with all modifications to our attack (described below) that work well on the MLP models. However, its performance is unstable and exhibits erratic accuracy curves despite the smooth loss convergence. We leave exploring loss functions and tuning the attack to make it work for convolutional neural networks as part of future work.

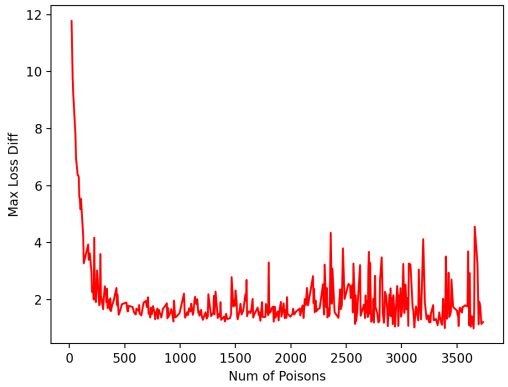


(a) Fixed Model Initialization, Varying Batch-sizes.

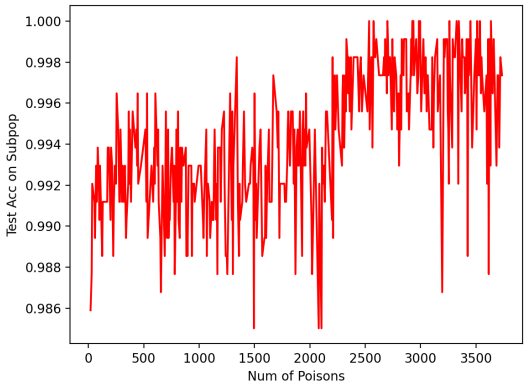


(b) Fixed Batch-size, Varying Model Initialization.

Figure 17. Variance of Poisoning Attacks. Each figure shows the test accuracy on sub-population for the label-flipping attack (same poisoning ratio: 0.5) on MNIST 1–7, with varying hyper-parameters of batch-size and model-weight initializations. Each setting is repeated four times, and each dot shows the result for one run. The attack is highly unstable, with large variations even when everything other than either the batch size or the weight initializations is changed.



(a) Maximum Loss Difference



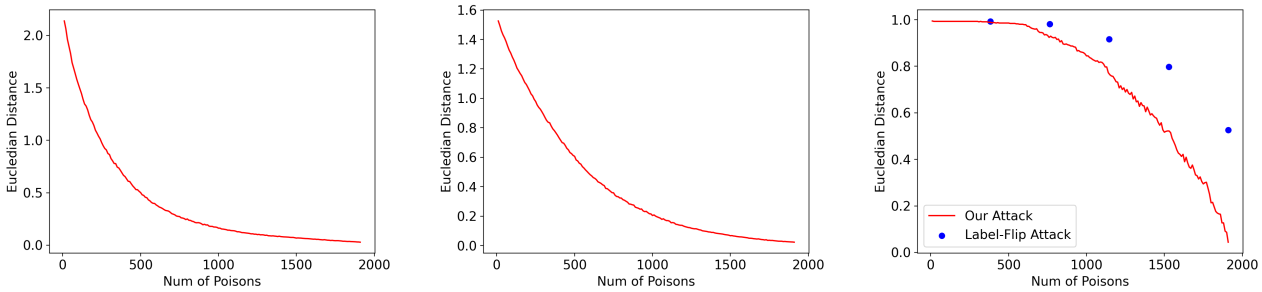
(b) Test Accuracy on Target Sub-population

Figure 18. Maximum loss difference and test accuracy on target sub-population across iterations for our algorithm. Data is not batched, and same weight-initializations for θ_t, θ_p are used. The loss drops sharply within the first few iterations, but the accuracy fluctuates within a very small window, even when $|\mathcal{D}_p| \sim 0.5|\mathcal{D}_c|$ is added.

F.2. Modifying Attack for DNNs

Despite removing batching and setting the weight-initialization for θ_p and θ_t to be the same, we observe that our attack still fails. Even though the loss difference seems to converge, the model preserves its accuracy on the target sub-population; dropping by less than 2% across the iterations even up to a poisoning rate of 0.55 (Figure 18). Although we do not understand what causes this behavior, we speculate that it is due to a disconnect in the attacker's objective and the loss-function used.

To mitigate this problem, we modify the algorithm to constrain the search space of possible poisoning points to a predefined set of candidates. By iterating over all the candidate points, the algorithm picks the most promising poisoning point (i.e., with maximum loss difference between θ_t and θ_p) from this candidate set. To define the candidate set, we construct two non-overlapping, equal-sized stratified splits of the dataset. The first one is used for training purposes ($\mathcal{D}_c$), while the second one is used as the candidate set for (x^*, y^*) optimization. We add an additional constraint on the candidate set that enforces the selection of points from the target sub-population but are assigned an incorrect label (i.e., the candidate set consists of digit 7, and the assigned labels are 1.).



(a) Maximum Loss Difference (b) Euclidean Distance (c) Test Accuracy on Target Sub-population

Figure 19. Maximum loss difference and test accuracy on target sub-population across iterations for our algorithm. The optimization process is constrained to select points from the candidate set. As visible, both the loss and Euclidean distance converges to zero smoothly. Additionally, our attack outperforms label-flip attack by a significant margin. We observed consistent results across several seeds.

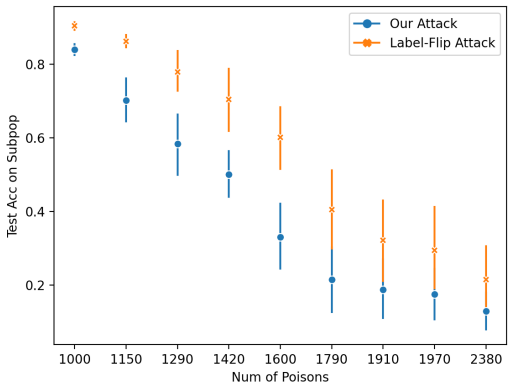


Figure 20. Poisoning attack effectiveness when the adversary does not know the victim's weight initializations (ten different seeds tried per experiment). The error bars show 68% confidence intervals (standard error).

F.3. Results

We start with the case where the weight-initialization used by the victim to train its models is known to the adversary. This setting is unrealistic, but shows how effective the attack could be when the adversary has full knowledge of everything about the victim's training process, including the random seeds used. With the constraints described in Section F.2, our attack consistently outperforms the label-flip attack by a large margin, as shown in Figure 19. For these experiments, we observe similar convergence and attack success rates between adding just one copy of (x^*, y^*) per iteration and adding as many as ten copies, and with at most ten copies, we can reduce attack execution time by nearly 90%.

Next, we evaluate the attack in a more realistic setting where the adversary does not know the weight-initializations used in training the victim model. As shown in Figure 20, we observe a large variation in the performance of trained models across different initial model weights, and the attack is not as effective as it can be when the initialization is known. The variance in attack performance is because these models are unstable to varying weight-initializations (Figure 17b) — some initial weights are biased towards having larger errors on the target sub-population, making it possible to poison these models with fewer points. Even in this setting, our model-targeted poisoning attack consistently outperforms the label-flipping attack.