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.

Lemma 1. Let θ_1 be a C_R-attainable parameter 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 an attainable model θ_1 and any model θ_2 such that $R(\theta_1) > R(\theta_2)$ and let $\mathcal{D}_1$ to be the training set that leads the training algorithm to produce 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. For $\delta > 0$, let F be the family of all $(C_R(1+\delta))$-attainable models. For any $\theta_1 \in F$ and for all θ_2 we have

$$\sup_{x,y} \big(l(\theta_2; x, y) - l(\theta_1; x, y)\big) + C_R(R(\theta_2) - R(\theta_1)) > \frac{\delta}{1+\delta} \cdot \sup_{x,y} \big(l(\theta_2; x, y) - l(\theta_1; x, y)\big).$$

Proof. By Lemma 1 we have

$$\sup_{x,y} \big(l(\theta_2; x, y) - l(\theta_1; x, y)\big) + C_R(1+\delta)(R(\theta_2) - R(\theta_1)) > 0.$$

Now by adding $\delta \sup_{x,y} \big(l(\theta_2; x, y) - l(\theta_1; x, y)\big)$ to both sides we have

$$(1+\delta)\Big(\sup_{x,y} \big(l(\theta_2; x, y) - l(\theta_1; x, y)\big) + C_R(R(\theta_2) - R(\theta_1))\Big) > \delta \sup_{x,y} \big(l(\theta_2; x, y) - l(\theta_1; x, y)\big)$$

which implies

$$\Big(\sup_{x,y} \big(l(\theta_2; x, y) - l(\theta_1; x, y)\big) + C_R(R(\theta_2) - R(\theta_1))\Big) > \frac{\delta}{1+\delta} \sup_{x,y} \big(l(\theta_2; x, y) - l(\theta_1; x, y)\big)$$

□

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

Theorem 1. *For $\delta > 0$, if θ_p is a $C_R(1+\delta)$-attainable model, after at most T steps Algorithm 1 will produce the poisoning set $\mathcal{D}_p$ so that the classifier trained on $\mathcal{D}_c \cup \mathcal{D}_p$ using Eq. (1) is ϵ-close to θ_p, with respect to loss-based distance, $D_{l,\mathcal{X},\mathcal{Y}}$, for*

$$\epsilon = \frac{\alpha(T) + L(\theta_p; D_c) - L(\theta_c; D_c)}{T \cdot {}^{\delta}/_{1+\delta}}$$

where $\alpha(T)$ is the regret of the Follow the Leader algorithm for a series of loss functions $\ell_i(\cdot) = \ell(\cdot, x_i, y_i) + C_R \cdot R(\cdot)$ and (x_i, y_i) is the ith poisoning point.

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 $\mathsf{Regret}(A_{\mathsf{FTL}}, S_{\theta_p}, T) \le \alpha(T)$ for some function α, then we have

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

which implies

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

Therefore we have

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

Based on Lemma 2, we further have

$$\sum_{j=1}^{T} \frac{\delta}{1+\delta} \cdot \big(\max_{x,y} l(\theta_j; x, y) - l(\theta_p; x, y) \big) \le \alpha(T) + L(\theta_p; D_c) - L(\theta_c; D_c) + N \cdot C_R \cdot (R(\theta_p) - R(\theta_c))$$

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

$$\epsilon = \frac{\alpha(T) + L(\theta_p; D_c) - L(\theta_c; D_c) + N \cdot C_R \cdot (R(\theta_p) - R(\theta_c))}{T \cdot {}^{\delta}/{}_{1+\delta}}.$$

□

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 $\alpha(T) \leq o(T)$. Following remark (restating Remark 1 in Section 4.2) and its proof shows the desired convergence.

Remark 1. *Sublinear regret bounds for FTL 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

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

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 infimum of the loss difference between θ and θ_p as (x^*, y^*) [3] 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

[3]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.

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 $R(\theta)$.

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

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

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

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

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

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

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

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 $\sup_\theta z'(\theta)$ 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 2 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

$$\langle \theta - \theta', x^* \rangle = \frac{1}{r} \cdot \|\theta - \theta'\|_1 \tag{2}$$

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

$$\langle x^*, \theta \rangle \geq -1 \tag{3}$$

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

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

Therefore by Inequalities equation 3 and equation 4 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 2 implies

$$l(\theta; x^*, -1) - l(\theta'; x^*, -1) = \frac{1}{r} \cdot \|\theta - \theta'\|_1. \tag{5}$$

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.

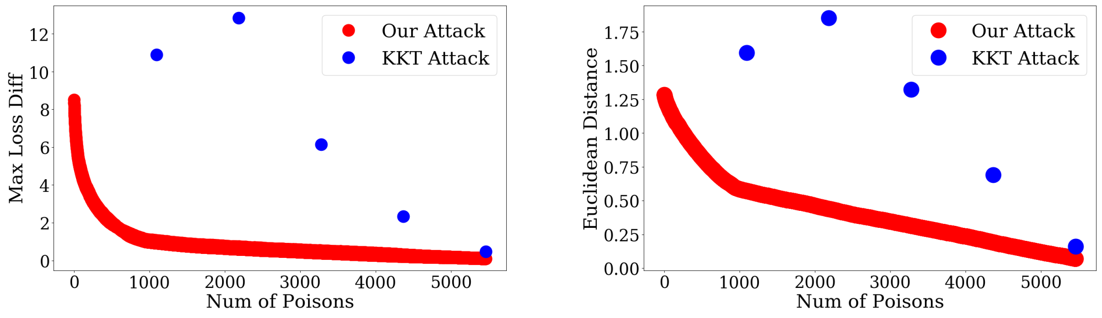
C. Additional Experimental Results

In this section, we provide more results in addition to the results in the main paper. In Section C.1, we show the additional results on SVM model and more results on logistic regression model are given in Section C.2. In Section C.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.

C.1. More Results on SVM model

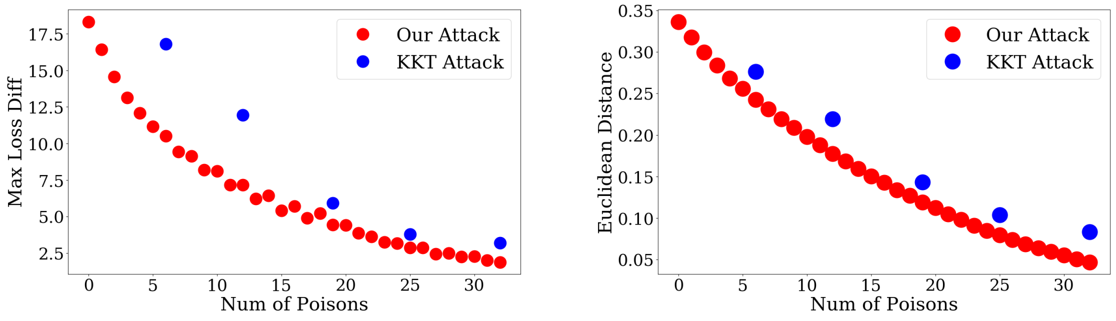
In this section, we first compare our attack to the KKT attack regarding the convergence to the target model. Then compare their attack success in achieving the attacker goals. Last, we provide the lower bound for inducing the model that are induced by our attack and the KKT attack. We use the exact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



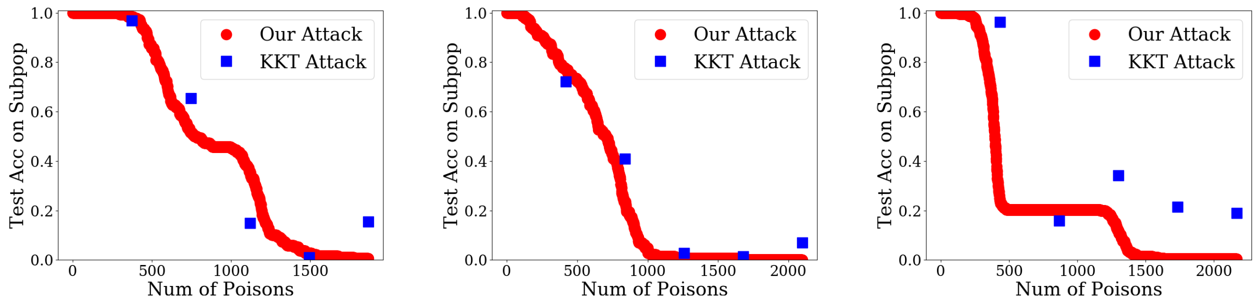
(a) Max Loss Difference (b) Euclidean Distance

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

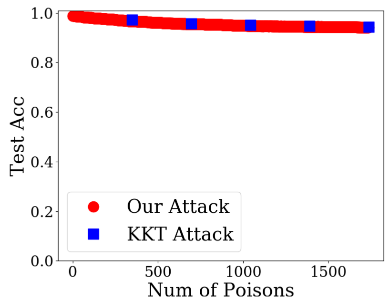
Figure 3. SVM on Dogfish dataset: attack convergence (results shown are for the target classifier of error rate 10%). The maximum number of poisons is set using the 2.0-close threshold to target classifier



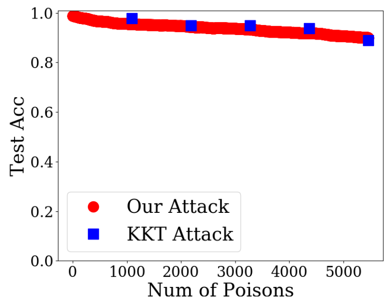
(a) Cluster 0 (b) Cluster 1 (c) Cluster 2

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.

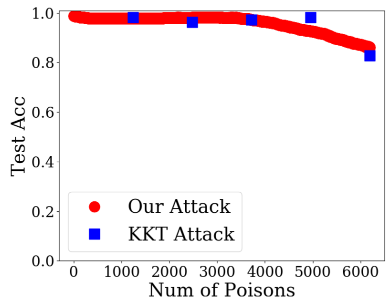
when the induced classifier is 2.0-close to the target classifier. From the two figures, we observe that classifiers induced by our algorithm steadily converge to the target classifier both in the maximum loss difference and Euclidean distance, while the classifier induced by the KKT attack either cannot 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) 5% Error Rate

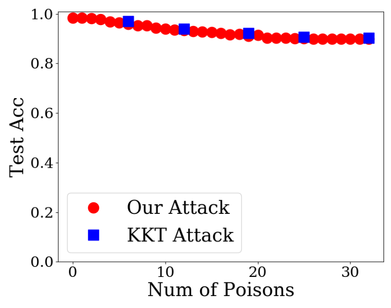


(b) 10% Error Rate

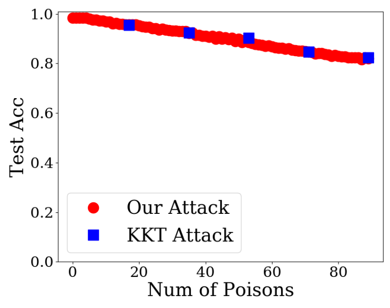


(c) 15% Error Rate

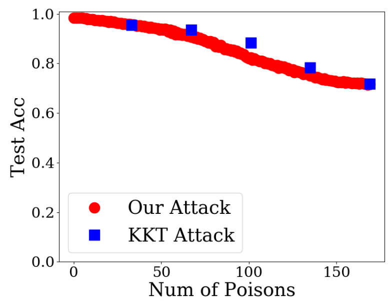
Figure 5. SVM on MNIST 1–7 dataset: test accuracy for each target model of given error rate with classifiers induced by poisoning points obtained from our attack and the KKT attack.



(a) 10% Error Rate

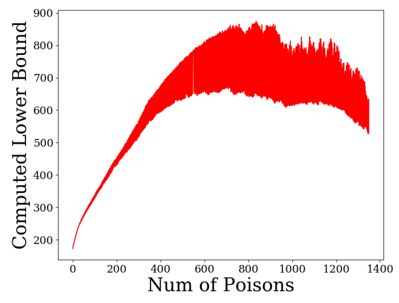


(b) 20% Error Rate

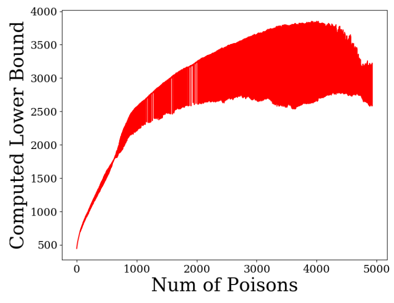


(c) 30% Error Rate

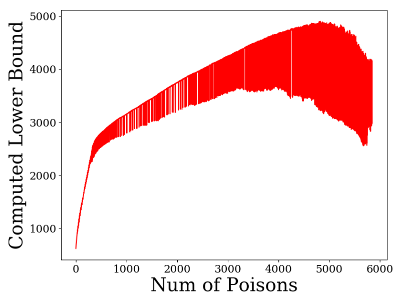
Figure 6. SVM on Dogfish dataset: test accuracy of each target model of given error rate with classifiers induced by poisoning points obtained from our attack and the KKT attack.



(a) 5% Error Rate



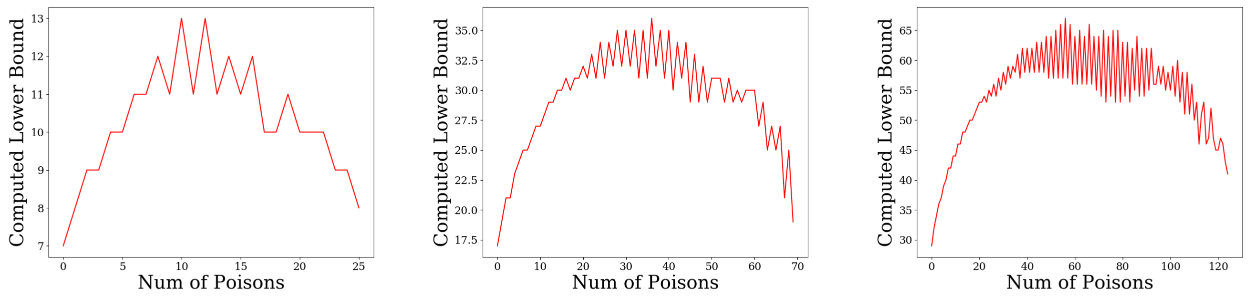
(b) 5% Error Rate



(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.

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 indiscriminate setting.



(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.

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 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.

C.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 C.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

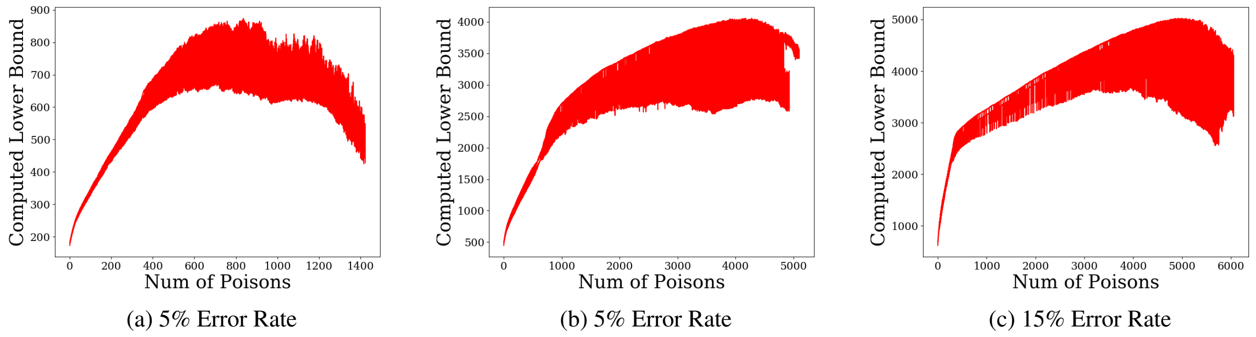


Figure 9. SVM on MNIST 1–7: lower bound computed in each iteration of running algorithm 1 when the target classifier of the algorithm is the classifier induced from the KKT Attack. The maximum number of poisons is set using the 0.1-close threshold to KKT induced classifier.

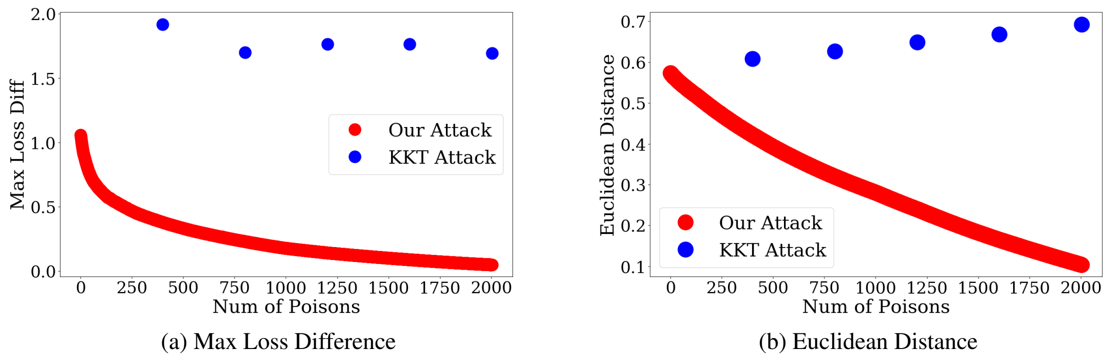
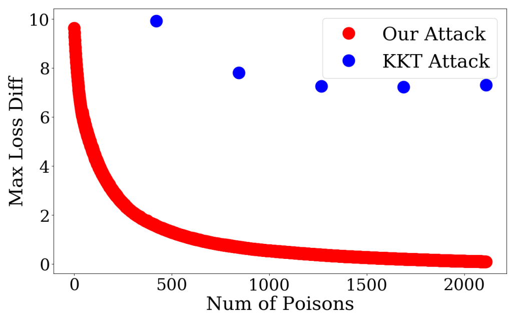


Figure 10. Logistic regression model on Adult: attack convergence (results shown are for the first subpopulation, Cluster 0). The maximum number of poisons is set using the 0.05-close threshold to target classifier.

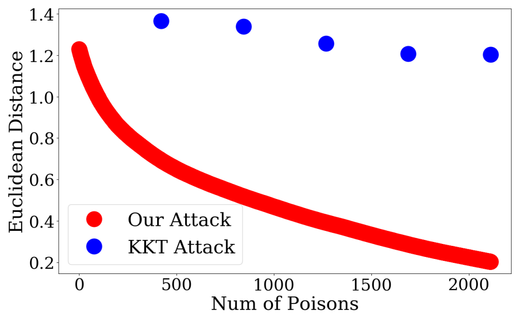
regression, we may not be able to efficiently search for the globally optimal point with maximum loss difference (Line 4 in Algorithm 1) because the difference of two logistic losses is not concave. Therefore, we adopt gradient descent strategy, using the Adam optimizer (Kingma & Ba, 2014) to search 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.

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 C.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

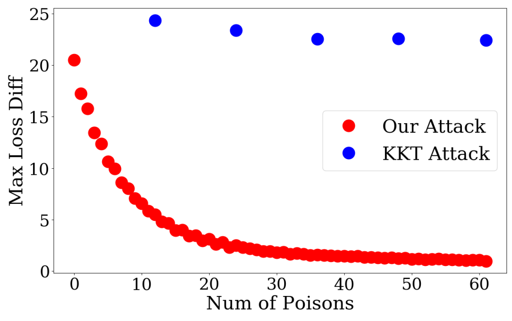


(a) Max Loss Difference

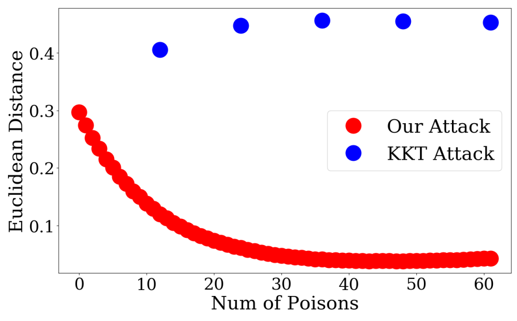


(b) Euclidean Distance

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.



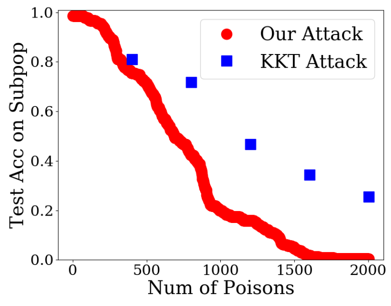
(a) Max Loss Difference



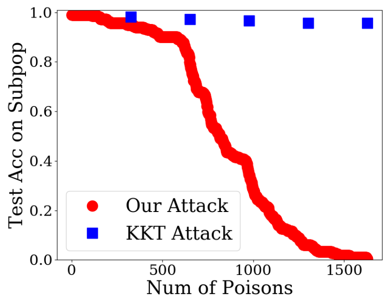
(b) Euclidean Distance

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.

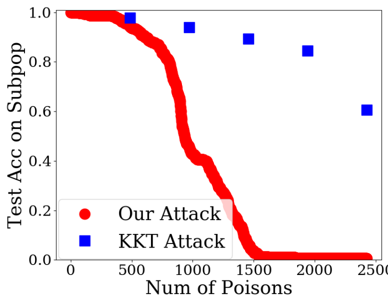
the form of figures. All the results show that our attack is much more effective than the KKT attack on logistic regression models, and in fact, the KKT attack cannot effectively poison the models in most cases. In addition, our attack runs in an online fashion and we can easily track the attack performance in each iteration.



(a) Cluster 0

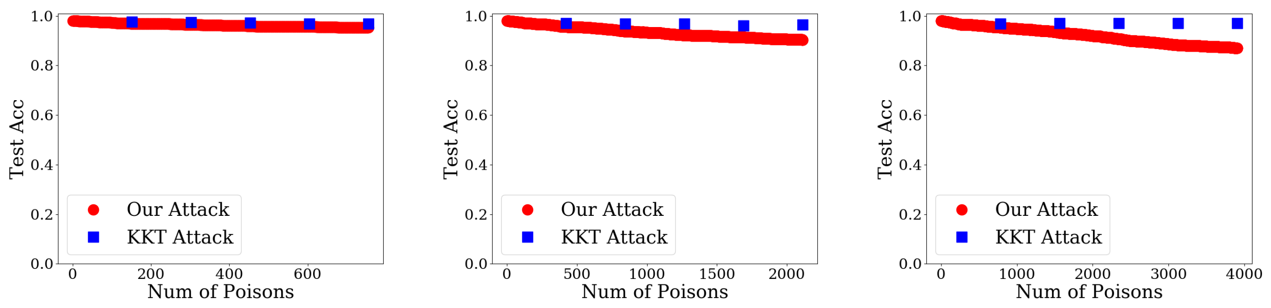


(b) Cluster 1



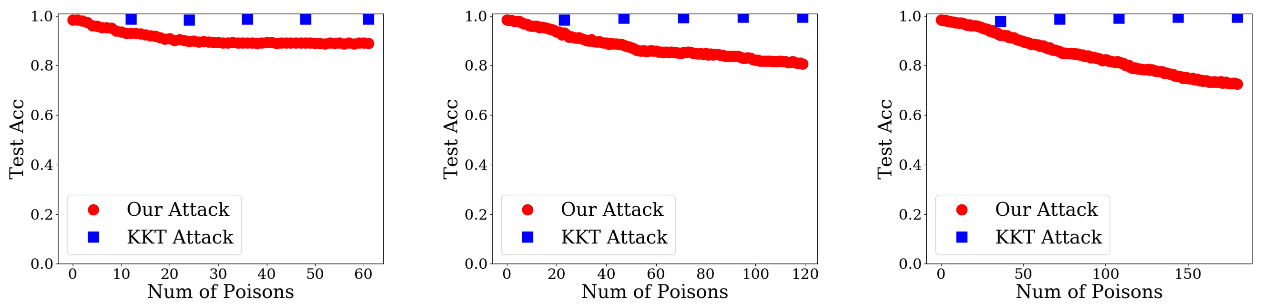
(c) Cluster 2

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.



(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.

C.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 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.

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.

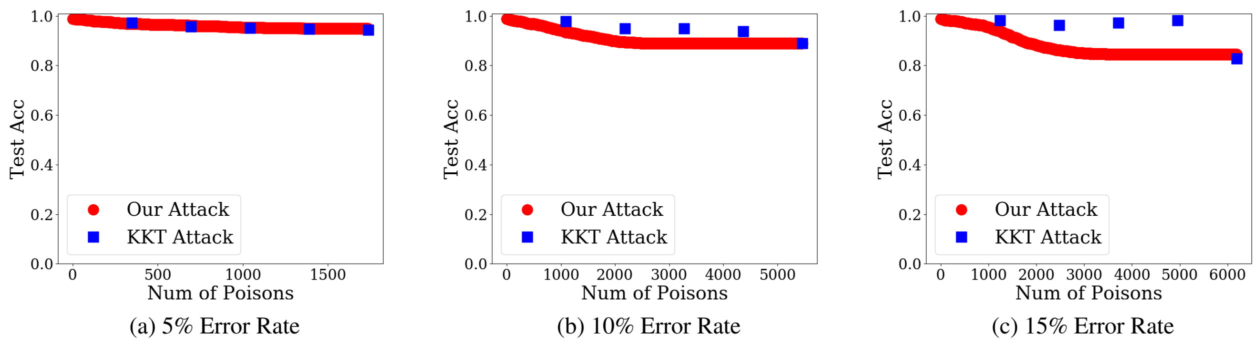


Figure 16. SVM on MNIST 1–7: test accuracy with classifiers obtained from our attack and KKT attack. Target model for KKT attack is generated from the original generation process and target model for our attack is generated from the improved generation process. Maximum number of poisoning points is obtained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

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 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.

D. 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 generate a target model, which can then be used as the target

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%). The target classifiers to our attack are generated by running the label-flipping attacks with given attacker objectives.

for a model-targeted attack,. This results 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. 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 this paper's scope, we have some empirical evidence in the indiscriminate setting. 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. We conduct the following experiments on the MNIST 1–7 dataset.

Target Models Generated from Objective-driven Label-Flipping Attacks. In our experiments, we generate the target classifiers from the label-flipping based objective-driven attacks. Although these label-flipping attacks are effective, they need too many poisoning points to achieve their objective. Then, we deploy our attacks 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 C.3. This process helps to reduce the loss on clean training set. Using this method, we 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 using 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 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. The adversary uses this new 70% of the training data to train its models, while the remaining 30% is reserved 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 for 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% [4], and did not find a significant

[4]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.

degradation in the attack effectiveness when $N = 10$ (the attack effectiveness drops by 0.3% at most). The size of the training dataset might explain this: 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. Thus, adding multiple copies of the same poison data is a fair approximation, 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 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. 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 a fixed number (constrained by the poisoning budget) of instances from the training data of the subpopulation, flipping their labels, and then injecting them into 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. We selected the three subpopulations with the highest test accuracy for each of the SVM and logistic regression models (all end up having 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, hoping 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 1) satisfy the attacker objective, 2) have larger loss on the training data from the subpopulation, and 3) 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 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%). Thus, the gap between our attack and the label-flipping attack is relatively 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. Hence, injecting points (from the subpopulation) with flipped labels can strongly impact 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. 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.

E. 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., 2021), 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).[5]

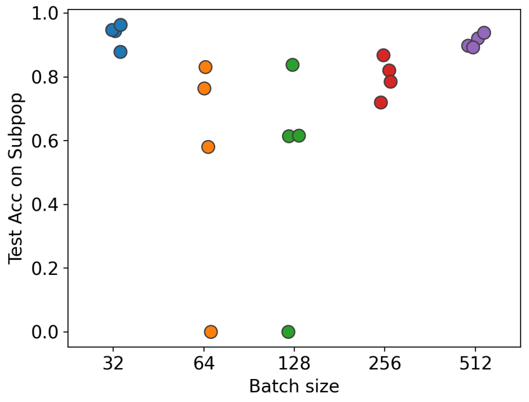
We observe that direct implementations of these poisoning attacks do not work. Via experiments to understand the kind of randomness introduced by varying hyper-parameters like batch-size and weight-initialization, we take steps to remove these sources of variation (Section E.1). Then, we describe modifications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

E.1. Simplifying the Setting

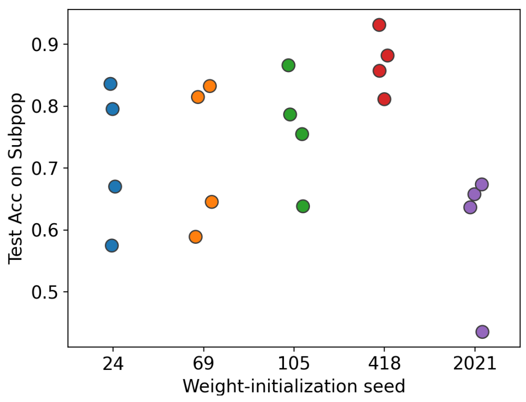
Both our attack and the label-flipping attack on DNNs are observed to be highly sensitive to hyper-parameters like batch-size, weight-initialization, and even randomness induced by the ordering of batches across epochs. As shown in Figure 17a, for the same initialization for model weights and batch-size, 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

[5]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.

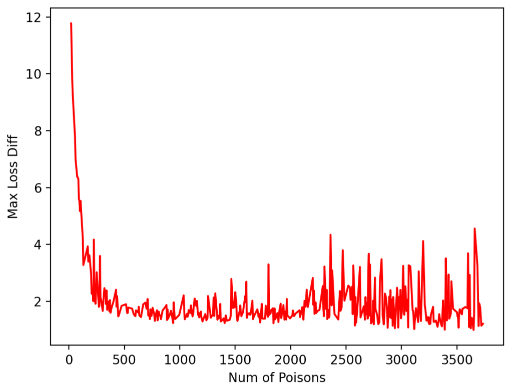


(a) Fixed Model Initialization, Varying Batch-sizes.

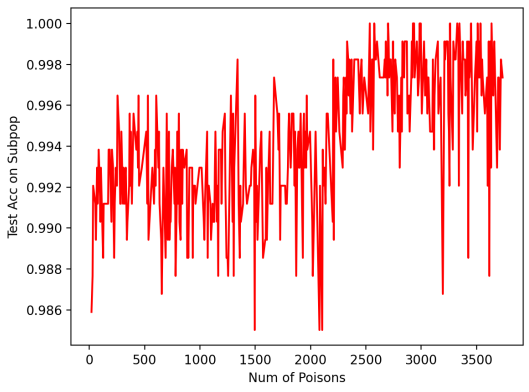


(b) Fixed Batch-size, Varying Model Initialization.

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

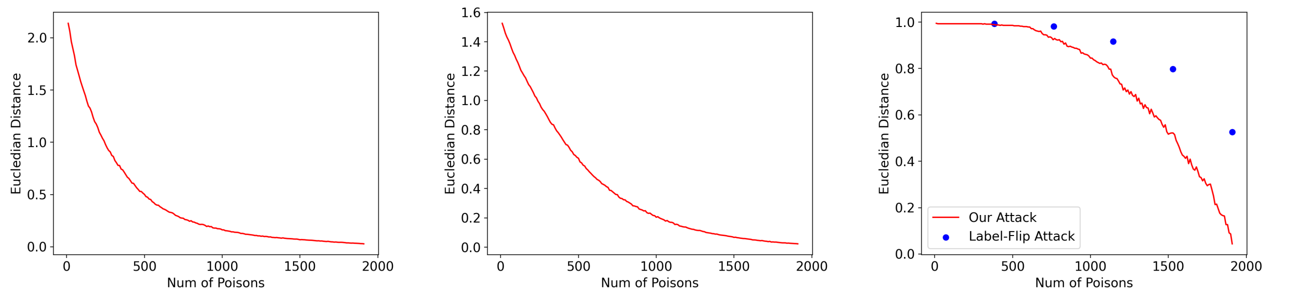
Figure 18. Maximum loss difference and test accuracy on target sub-population across iterations for our algorithm. Data is not batched, and same weight-initializations for θ_t, θ_p are used. The loss drops sharply within the first few iterations, but the accuracy fluctuates within a very small window,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

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.

E.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.,



(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.

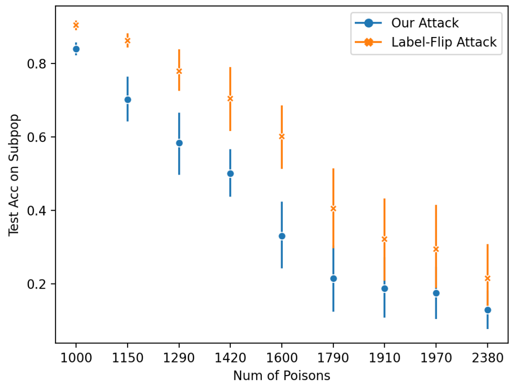


Figure 20. Poisoning attack effectiveness when the adversary does not know the victim's weight initializations (ten different seeds tried per experiment). The error bars show 68% confidence intervals (standard error).

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.).

E.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 E.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.