Geometric Algorithms for k-NN Poisoning

Diego Ihara Centurion
Department of Computer Science, University of Illinois at Chicago, USA

Karine Chubarian
Department of Computer Science, University of Illinois at Chicago, USA

Bohan Fan
Department of Computer Science, University of Illinois at Chicago, USA

Francesco Sgherzi
Department of Computer Science, University of Illinois at Chicago, USA

Thiruvenkadam Sivaprakasam Radhakrishnan
Department of Computer Science, University of Illinois at Chicago, USA

Anastasios Sidiropoulos
Department of Computer Science, University of Illinois at Chicago, USA

Angelo Straight
Department of Computer Science, University of Illinois at Chicago, USA

Abstract

We propose a label poisoning attack on geometric data sets against k-nearest neighbor classification. We provide an algorithm that can compute an εn-additive approximation of the optimal poisoning in $n \cdot 2^{2^{O(d+k/\varepsilon)}}$ time for a given data set $X \in \mathbb{R}^d$, where $|X| = n$. Our algorithm achieves its objectives through the application of multi-scale random partitions.

Keywords and phrases random partitions, algorithms, machine learning

Digital Object Identifier 10.57717/cgt.v4i2.55

Related Version https://arxiv.org/abs/2306.12377

Acknowledgements This work was partially supported by NSF grant 1815145.

1 Introduction

Recent developments in machine learning have spiked the interest in robustness, leading to several results in *adversarial machine learning* [1, 2, 11]. A central goal in this area is the design of algorithms that are able to impair the performance of traditional learning methods by adversarially perturbing the input [3, 17, 19]. Adversarial attacks can be exploratory, such as evasion attacks, or causative, poisoning the training data to affect the performance of a machine learning algorithm or attack the algorithm itself. Backdoor poisoning is a type of causative adversarial attack, in which the attacker has access to the whole or a portion of the training data that they can perturb. Clean-label poisoning attacks are a type of backdoor poisoning attack that perturb only the features of the training data leaving the labels untouched, so as to make the poison less detectable. In the other end of the spectrum are label poisoning attacks that perturb or flip the training data labels.

Why compute provably nearly-optimal poison attacks? A limitation with current poisoning methods is that it is not possible to adversarially perturb an input so that the performance of *any* algorithm is negatively affected. Moreover, it is generally not clear how to provably compare different poisoning methods. We seek to address these limitations of adversarial machine learning research using tools from computational geometry.

Specifically, we study the following optimization problem: Given some data set, X, compute a small perturbation of X, so that the performance of a specific classifier deteriorates as much as possible. An efficient solution to this optimal poisoning problem can be used to compare the performance of different classification algorithms, as follows. Suppose we want to compare the performance of a collection of classification algorithms, $A_1, ..., A_t$, on some fixed data set X, in the presence of a poisoning attack that produces a bounded perturbation, X', of X. Ideally, we would like to have provable worst-case guarantees on the robustness of $A_1, ..., A_t$. However, such results are often hard to prove rigorously, and thus many existing methods lack such guarantees. Since the poisoned data set X' is unknown, we cannot simply run A_1, ..., A_t on X' and compare the results. Instead, our method allows us to compute from X some poisoned data set, X'', which is provably a nearly-optimal poison against the specific classification task.

1.1 Robustness of k-nearest neighbors

We instantiate the above general optimization problem of computing nearly-optimal poison attacks to the specific task of k-nearest neighbor classification. Nearest-neighbor based algorithms are naturally robust due to the presence of an inherent majority voting mechanism. In [12], they are used to provide individual and joint certifications for test predictions in the presence of data poisoning and backdoor attacks. In [16], a defense algorithm is proposed using k-nearest neighbors against label-flipping attacks. However, computing provably nearly-optimal poisoning against such algorithms has not been studied prior to our work. We provide approximation algorithms that compute a nearly-optimal label flipping poisoning attack against k-nearest neighbors with provable guarantees.

1.2 Our results

We design and analyze poisoning algorithms against k-nearest neighbor classification (k-NN) in the setting of binary label classification. The k-NN classifier is arguably one of the most popular and well-studied methods used in machine learning and geometric data analysis [8]. The classifier works as follows: Given a set of labeled points, $X \subset \mathbb{R}^d$, and some unlabeled $p \in \mathbb{R}^d$, we can compute a label for p by taking the most frequently occurring label in the multiset of labels of the k nearest neighbors of p in X.

We formulate the poisoning problem against k-NN as follows. Given some set of points, X, with binary labels, and some $m \in \mathbb{N}$, the goal is to flip the labels of at most m points, so that we maximize the number of points in X for which their *predicted* label differs from their true label. We refer to the set of points with flipped labels as an m-poison and define the number of points for which their predicted label differs from the original label as the *corruption*. The following summarizes our results.

▶ **Theorem 1.** *On input $X \subset \mathbb{R}^d$, with $|X| = n$, and $m \in \mathbb{N}$, Algorithm Poison-k-NN computes a m-poison against k-NN, with expected corruption $\texttt{OPT}_m(X) - \varepsilon n$, in time $n \cdot 2^{2^{O(d+k/\varepsilon)}}$, where $\texttt{OPT}_m(X)$ denotes the maximum corruption of any m-poison.*

While the above problem formulation only involves a fully labeled set X, typical tasks involve a training set X_{train} and a test set X_{test}. In order to address this case, we modify the algorithm in Theorem 1 so that it computes a poison of the training set, so that the prediction error on the test set deteriorates as much as possible. This result is summarized in the following.

▶ **Theorem 2.** *On input $X_{train}, X_{test} \subset \mathbb{R}^d$, with $|X_{train}| = n_{train}$, $|X_{test}| = n_{test}$, and $m \in \mathbb{N}$, Algorithm Poison-k-NN' computes a m-poison against k-NN, with expected corruption $\mathit{OPT}_m(X_{train}, X_{test}) - \varepsilon n_{test}$, in time $(n_{train} + n_{test}) \cdot 2^{2^{O(d+k/\varepsilon)}}$, where $\mathit{OPT}_m(X_{train}, X_{test})$ denotes the maximum corruption incurred on X_{test} when all neighbors are chosen from X_{train}, of any m-poison on X_{train}.*

Algorithm Poison-k-NN' is an adaptation of Poison-k-NN, and has a similar analysis. Algorithm Poison-k-NN uses as a subroutine a procedure for computing a random partition of a metric space. The random partition has the property that the diameter of each cluster is upper bounded by some given Lipschitz function, while the probability of two points being separated is upper bounded by a multiple of their distance divided by the cluster diameter (see Section 2 for a formal definition). This is inspired by the multi-scale random partitions invented by Lee and Naor [13] in the context of the Lipschitz extension problem. We believe that our formulation could be of independent interest.

1.3 Related work

To the best of our knowledge, no prior work tackles the adversarial poisoning of geometric algorithms giving provable bounds. The most traditional poisoning method is the **fsgm** [9], which consists in adding, to each testing sample, noise with the same dimensionality of the input and proportional to the gradient of the cost function in that point. This approach is proven to be the most damaging adversarial example against linear models like logistic regression. However, it is less effective on deep neural networks, as they are able to approximate arbitrary functions [10]. **pgd** [15] improves upon **fsgm** by iteratively looking for better adversarial examples for a given input toward the direction of the increase of the cost function. However, although producing *better* adversarial samples, it still encounters the same drawbacks of **fsgm**.

There are a few existing algorithms that perform label flipping attacks. In [16] they use a greedy algorithm to flip the examples that maximize the error on a validation set, when the classifier is trained on the poisoned dataset, and use k-NN to reassign the label in the training set as the defense against this type of label flipping attacks. [20] model the label attacks as a bilevel optimization problem targeting linear classifiers and also experiment with the transferability of these attacks. Traditional poisoning methods, however, do not offer provable guarantees on the reduction in performance, thus yielding results that are not *problem* dependent but rather *implementation* or *model* dependent [5, 6, 7].

There have also been a few defenses proposed against label flipping attacks. In [18] they propose a pointwise certified defense against adversarially manipulated data up to some "radius of perturbation" through randomized smoothing. Specifically, each prediction is certified robust against a certain number of training label flips.

1.4 Notation

For any $k \in \mathbb{N}$, let $[k] = \{1, \ldots, k\}$. Let $M = (X, \rho)$ be a metric space. For any $X' \subseteq X$, we denote by $\mathsf{diam}_M(X')$ the diameter of X', i.e. $\mathsf{diam}_M(X') = \sup_{x,y \in X'} \rho(x, y)$; we also write $\mathsf{diam}(X')$ when M is clear from the context. For any $x \in X$, $Y \subseteq X$, we write $\delta(x, Y) = \inf_{y \in Y} \rho(x, y)$. For any metric space $M = (X, \rho)$, any finite $Y \subset X$, any $i \in \mathbb{N}$, and any $q \in X$, let $\mathsf{NN}_i(q, Y)$ denote the i-th nearest neighbor of q in Y.

1.5 Organization

The rest of the paper is organized as follows. Section 2 presents our result on random partitions of metric spaces. Section 3 presents our algorithm for poisoning against k-NN classifiers. We conclude and highlight some open problems in Section 4.

2 Random partitions of metric spaces

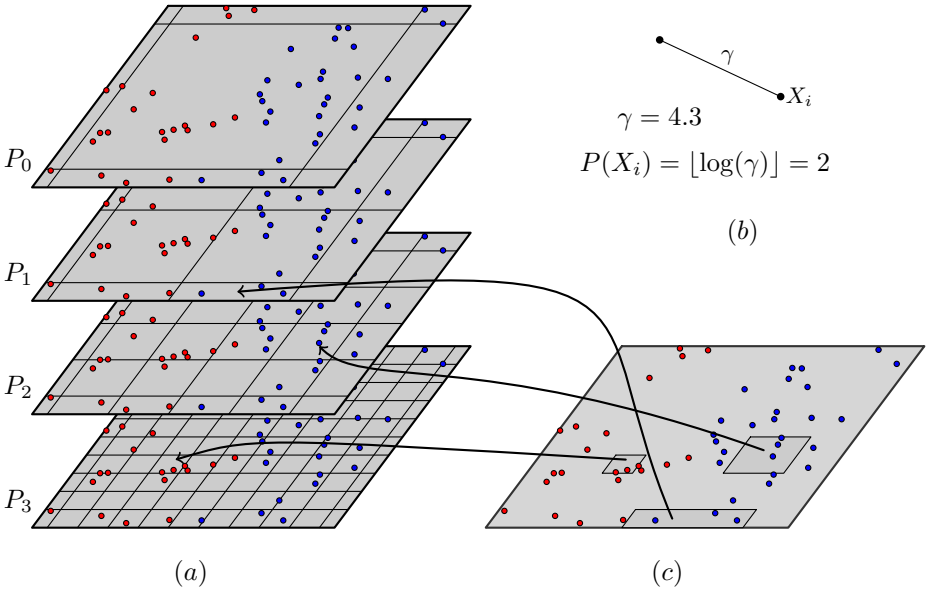

Figure 1 Illustration of the application of the multi-scale random partition approach to a set of points. (a) We begin with a random partition and refine to produce an additional level. (b) The selection of the level depends on the distance to the k-th neighbor. (c) The resulting partition is the union of cells originating at different levels of granularity.

In this section, we introduce a random metric space partitioning scheme. The main idea is to partition a given metric space so that the radius of each cluster is bounded by some Lipschitz function, while ensuring that only a small fraction of pairs end up in different clusters, in expectation. This partition is used by our algorithm for partitioning the problem into several sub-problems for each cluster.

For any partition P of a ground set Y, and for any $y \in Y$, we denote by $P(y)$ the unique cluster in P that contains y. Let $M = (X, \rho)$ be some metric space. Let P be a random partition of M. For any $\Delta > 0$, we say that P is Δ-bounded if, with probability 1, for all clusters $C \in P$, we have $\mathsf{diam}(C) \leq \Delta$. For any $\beta > 0$, we say that a Δ-bounded P is β-Lipschitz if for all $x, y \in X$,

$$\Pr[P(x) \neq P(y)] \leq \beta \frac{\rho(x, y)}{\Delta}.$$

The infimum $\beta > 0$ such that for all $\Delta > 0$, M admits a Δ-bounded β-Lipschitz random partition, is referred to as the *modulus of decomposability* of M, and is denoted by β_M.

▶ **Lemma 3** ([4]). *For any $d \in \mathbb{N}$, and for any subset of d-dimensional Euclidean space, M, we have that $\beta_M = O(\sqrt{d})$.*

The following uses ideas from [13] and [14].

▶ **Lemma 4** (Multi-scale random partition). *Let $C > 0$. Let $M = (X, \rho)$ be a metric space, and let $r : X \to \mathbb{R}_{\geq 0}$. Then, there exists a random partition P of M, satisfying the following conditions:*

(1) With probability 1, for any $p \in X$,

$$\mathsf{diam}(P(p)) \leq r(p)C^2$$

(2) For any $p, q \in X$,

$$\Pr_P[P(p) \neq P(q)] \leq \left(\frac{2\|r\|_{\mathsf{Lip}}}{\log C} + \beta_M\right)\frac{\rho(p,q)}{r(p)}$$

Moreover, given M as input, the random partition, P can be sampled in time polynomial in $|X|$.

Proof. Let $B = \|r\|_{\mathsf{Lip}}$.

For any $i \in \mathbb{Z}$, let P_i be a C^i-bounded β_M-Lipschitz random partition of M. Thus, each point $x \in X$ is mapped to some cluster in each P_i. We construct P by assigning each $x \in X$ to a single one of these clusters. We first sample $\alpha \in [0, 1]$, uniformly at random. Then, for each $x \in X$, we assign x to the cluster $P_{\lceil \alpha + \log_C(r(x))\rceil}(x)$. This concludes the construction of P. It remains to show that the assertion is satisfied.

For any $p \in X$, we have that $P(p) \subseteq P_i(p)$, where $i = \lceil \alpha + \log_C(r(x))\rceil \leq 2 + \log_C(r(x))$. Since P_i is C^i-bounded, it follows that

$$\begin{aligned}
\mathsf{diam}(P(p)) &\leq \mathsf{diam}(P_i(p)) && (P(p) \subseteq P_i(p)) \\
&\leq C^i && (P_i \text{ is } C^i\text{-bounded}) \\
&\leq C^{2+\log_C(r(p))} \\
&= r(p)C^2,
\end{aligned}$$

with probability 1, which establishes part (1) of the assertion.

It remains to establish part (2). Let $p, q \in X$. We may assume, without loss of generality, that $0 < r(p) \leq r(q)$. Let $\mathcal{E}_1$ be the event that $\lceil \alpha + \log_C(r(p))\rceil \neq \lceil \alpha + \log_C(r(q))\rceil$. We have

$$\begin{aligned}
\Pr[\mathcal{E}_1] &= \Pr[\lceil \alpha + \log_C(r(p))\rceil \neq \lceil \alpha + \log_C(r(q))\rceil] \\
&\leq |\log_C(r(p)) - \log_C(r(q))| \\
&= \log_C \frac{r(q)}{r(p)} \\
&= \left(\log \frac{r(q)}{r(p)}\right) / (\log C) \\
&\leq (1/\log C) \log \frac{r(p) + B \cdot \rho(p,q)}{r(p)} && (\|r\|_{\mathsf{Lip}} = B) \\
&= (1/\log C) \log \left(1 + \frac{B \cdot \rho(p,q)}{r(p)}\right) \\
&\leq (1/\log C) 2B \frac{\rho(p,q)}{r(p)}. && (1)
\end{aligned}$$

Conditioned on $\neg\mathcal{E}_1$, there exists $t \in \mathbb{Z}$, such that $t = \lceil \alpha + \log_C(r(p))\rceil = \lceil \alpha + \log_C(r(q))\rceil$; let $\mathcal{E}_2$ be the event that $P_t(p) \neq P_t(q)$. Since P_t is C^t-bounded β_M-Lipschitz, it follows that

$$\Pr[\mathcal{E}_2] \leq \beta_M \frac{\rho(p,q)}{C^t} \leq \beta_M \frac{\rho(p,q)}{C^{\log_C(r(p))}} = \beta_M \frac{\rho(p,q)}{r(p)}. \quad (2)$$

By (1) and (2) we obtain that

$$\Pr[P(p) \neq P(q)] \leq ((1/\log C)2B + \beta_M)\frac{\rho(p,q)}{r(p)},$$

which establishes part (2) of the assertion, and concludes the proof. ◀

Figure 1 illustrates the partitioning process.

3 Poisoning k-NN

In this section, we describe our poisoning algorithm, which is our main result.

Let $d \in \mathbb{N}$, and let $X \subset \mathbb{R}^d$. Let $\mathsf{label} : X \to \{1, 2\}$, and let $k \in \mathbb{N}$ be odd (to avoid ties), with $k \leq n$. For any $p \in \mathbb{R}^d$, for any $i \in [k]$, let $\Gamma_i(p)$ be an i-th nearest neighbor of p, breaking ties arbitrarily, and let

$$\gamma_i(p) = \|p - \Gamma_i(p)\|_2.$$

We write $\gamma(p) = \gamma_k(p)$.

▶ **Lemma 5.** *The function $\gamma : \mathbb{R}^d \to \mathbb{R}$ is 1-Lipschitz.*

Proof. WLOG, let $\gamma(p) \geq \gamma(q)$. It is sufficient to prove that $\gamma(p) \leq \|p - q\| + \gamma(q)$, which means $\Gamma_k(p)$ is within distance $\|p - q\| + \gamma(q)$ from p. Now consider two cases:

Case 1: $\Gamma_k(p)$ falls in $\mathsf{ball}(q, \gamma(q))$. By triangle inequality, $\gamma(p) \leq \|p - q\| + \|\Gamma_k(p) - q\| \leq \|p - q\| + \gamma(q)$.

Case 2: $\Gamma_k(p)$ falls outside of $\mathsf{ball}(q, \gamma(q))$. If $\gamma(p) > \|p - q\| + \gamma(q)$, then all the k-nearest neighbour of q are closer to p than $\Gamma_k(p)$, which is a contradiction.

This concludes the proof. ◀

Now we consider the result of Lemma 4 in Euclidean space:

▶ **Lemma 6** (Euclidean multi-scale random partition)**.** *Let $\varepsilon > 0$, there exists a random partition P of X, satisfying the following conditions:*

(1) The following statement holds with probability 1: For any $p \in X$,

$$\mathsf{diam}(P(p)) \leq \frac{2k}{\varepsilon}\gamma(p)2^{8k/\varepsilon}O(\sqrt{d})$$

(2) For any $p, q \in X$,

$$\Pr[P(p) \neq P(q)] \leq \frac{\varepsilon\|p - q\|_2}{k\gamma(p)}.$$

Moreover, P can be sampled in time polynomial in $|X|$.

Proof. Let $M = (X, \rho)$ be the metric space obtained by setting ρ to be the Euclidean metric. By Lemma 3 we have $\beta_M = O(\sqrt{d})$. Let P be the random partition of X obtained by applying Lemma 4, setting $\gamma : X \to \mathbb{R}_{\geq 0}$ where $r = B\gamma$, with $B = 2k\beta_M/\varepsilon$, and $C = 2^{4k/\varepsilon}$. By Lemma 5 we have that $\|\gamma\|_{\mathsf{Lip}} = 1$, and thus $\|r\|_{\mathsf{Lip}} = \|B\gamma\|_{\mathsf{Lip}} = B\|\gamma\|_{\mathsf{Lip}} = B$. The assertion now follows by straightforward substitution on Lemma 4. ◀

Now we bound the size of cluster with Lemma 7.

▶ **Lemma 7.** *Let $h > 0$, and let $A \subset \mathbb{R}^d$ be such that for all $p \in A$, we have $\mathsf{diam}(A) \leq h \cdot \gamma(p)$. Then, $|X \cap A| = k \cdot h^{d+O(1)}$.*

Proof. For any $p \in \mathbb{R}^d$, we have that $\gamma(p)$ is the distance between p and the k-th nearest neighbor of p in X. It follows that the interior of $\mathsf{ball}(p, \gamma(p))$ contains at most k points in X (it contains at most $k-1$ points in X if $p \notin X$). In particular, the (closed) ball $\mathsf{ball}(p, \gamma(p)/2)$ contains at most k points in X. Let

$$r^* = \inf_{p \in A} \gamma(p).$$

It follows that for all $p \in A$,

$$|X \cap \mathsf{ball}(p, r^*/2)| \leq k. \tag{3}$$

We have by the assumption that $\mathsf{diam}(A) \leq h \cdot r^*$, and thus $A \subseteq \mathsf{ball}(p^*, R^*)$, for some $p^* \in A$, and some $R^* = 2h \cdot r^*$. For any $0 < \alpha < \beta$, we have that any ball of radius β in $\mathbb{R}^d$ can be covered by at most $O(\beta/\alpha)^d = (\beta/\alpha)^{d+O(1)}$ balls of radius α. Therefore, A can be covered by a set of at most $(R^*/r^*)^{d+O(1)} = h^{d+O(1)}$ balls of radius $r^*/2$. Combining with (3) it follows that

$$|X \cap A| = k \cdot h^{d+O(1)},$$

which concludes the proof. ◀

3.1 The main poisoning algorithm

We are now ready to describe the main poisoning algorithm against k-NN. For any finite $Y \subset \mathbb{R}^d$, and for any integer $i \geq 0$, let $\mathtt{OPT}_i(X)$ be the maximum corruption that can be achieved for X with a poison set of size at most i. Let $\mathit{corruption}(X, Y)$ be the corruption of poisoning X by flipping labels of point set $Y \subset X$. Now we describe our poisoning algorithm.

Algorithm Poison-k-NN for k-NN Poisoning: The input consists of $X \subset \mathbb{R}^d$, $m \in \mathbb{N}$, and $\mathsf{label} : X \to \{1, 2\}$, with $|X| = n$.

Step 1. Sample the random partition P according to the algorithm in Lemma 6.

Step 2. For any cluster $C \subset X$ in P, by Lemma 7 we have that $|C| = k \cdot (\sqrt{d}(2k/\varepsilon)2^{8k/\varepsilon})^{d+O(1)} = (2k^2/\varepsilon) \cdot 2^{(d+O(1))8k/\varepsilon}$. For any $i \in \{1, \ldots, m\}$, we compute an optimal poisoning, $S_{C,i} \subseteq C$, for C with i poison points via brute-force enumeration. Each solution can be uniquely determined by selecting the i points for which we flip their label. Thus, the number of possible solutions is at most $2^{|C|} = 2^{(2k^2/\varepsilon) \cdot 2^{(d+O(1))8k/\varepsilon}}$. The enumeration can thus be done in time $2^{(2k^2/\varepsilon) \cdot 2^{(d+O(1))8k/\varepsilon}}$, for each cluster in P. Since there are at most n clusters, the total time is $n \cdot 2^{(2k^2/\varepsilon) \cdot 2^{(d+O(1))8k/\varepsilon}} = n \cdot 2^{2^{O(d+k/\varepsilon)}}$.

Step 3. We next combine the partial solutions computed in the previous step to obtain a solution for the whole pointset. This is done via dynamic programming, as follows. We order the clusters in P arbitrarily, as $P = \{C_1, \ldots, C_{|P|}\}$. For any $i \in \{0, \ldots, |P|\}$, $j \in \{1, \ldots, m\}$, let

$$A_{i,j} = \mathtt{OPT}_j(C_1 \cup \ldots \cup C_i).$$

We can compute $A_{i,j}$ via dynamic programming using the formula

$$A_{i,j} = \begin{cases} \max\limits_{t \in [j]} \left(A_{i-1,t} + \mathsf{corruption}(C_i, S_{C_i, j-t})\right) & \text{if } i > 0 \\ 0 & \text{otherwise} \end{cases}$$

The size of the dynamic programming table is $O(|P| \cdot m) = O(nm)$. The same recursion can also be used to compute an optimal k-poison, Y, for $C_1 \cup \ldots \cup C_{|P|}$. The algorithm terminates and outputs Y as the final poison for X.

Correctness of dynamic programming. By definition, $A_{i,j}$ is the optimal (maximum) corruption with j poison points on the first i clusters ($C_1 \cup \ldots \cup C_i$). $A_{i,0} = 0$ for all i. Suppose the solution $A_{i-1,j}$ is correct, then $A_{i,j}$ is the maximum of optimal corruption for poisoning the first $i-1$ clusters with t points, plus the corruption using the remaining $j-t$ points on i-th cluster. ◀

▶ **Lemma 8.** $E[\mathsf{corruption}(X,Y)] \geq \mathit{OPT}_m(X) - \varepsilon n$.

Proof. Let $Z \subseteq X$ be an optimal k-poison for X. Recall that P is the random partition sampled in Step 1.

For any $x \in X$, $i \in \mathbb{N}$, let $\mathcal{E}_{x,i}$ be the event that the cluster of P containing x, does not contain the i-nearest neighbor of x in X; i.e. $\mathsf{NN}_i(x,X) \notin P(x)$. Let also $\mathcal{E}_x$ be the event that the cluster of P containing x, does not contain all of the k-nearest neighbors of x in X; i.e.

$$\mathcal{E}_x = \mathcal{E}_{x,1} \vee \ldots \vee \mathcal{E}_{x,k}.$$

Thus

$$\begin{aligned}
\Pr[\mathcal{E}_x] &= \Pr[\mathcal{E}_{x,1} \vee \ldots \vee \mathcal{E}_{x,k}] \\
&\leq \sum_{i=1}^{k} \Pr[\mathcal{E}_{x,i}] && \text{(union bound)} \\
&= \sum_{i=1}^{k} \Pr[P(x) \neq P(\mathsf{NN}_i(x)] \\
&\leq \sum_{i=1}^{k} \frac{\varepsilon \|x - \mathsf{NN}_i(x)\|_2}{k\gamma(p)} && \text{(Lemma 6)} \\
&\leq k \frac{\varepsilon \|x - \mathsf{NN}_k(x)\|_2}{k\gamma(p)} \\
&= \varepsilon && (4)
\end{aligned}$$

Let

$$X' = \{x \in X : \{\mathsf{NN}_1(x), \ldots, \mathsf{NN}_k(x)\} \not\subseteq P(x)\}.$$

By (4) and the linearity of expectation, it follows that

$$E[|X'|] = \sum_{x \in |X|} Pr[\mathcal{E}_x] \leq \varepsilon |X| = \varepsilon n. \tag{5}$$

Let Y be the poison that the algorithm returns. Note that $X \setminus X' = C_1 \cup \ldots \cup C_{|P|}$. For any $x \in X \setminus X'$, if Y corrupts x in $X \setminus X'$, then it must also corrupt x in X (since, by the definition of X, all k-nearest neighbors of x are in $X \setminus X'$). Thus, $\mathtt{OPT}_m(X) \geq \mathtt{OPT}_m(X \setminus X') \geq \mathtt{OPT}_m(X) - |X'|$, and moreover,

$$\begin{aligned}
\mathsf{corruption}(X,Y) &\geq \mathsf{corruption}(X \setminus X', Y) \\
&= \mathtt{OPT}_m(X \setminus X') \\
&\quad \text{(by the dynamic program)} \\
&\geq \mathtt{OPT}_m(X) - |X'|.
\end{aligned}$$

Combining with (5) and the linearity of expectation we get $E[\mathsf{corruption}(X,Y)] \geq \mathtt{OPT}_m(X) - E[|X'|] \geq \mathtt{OPT}_m(X) - \varepsilon n$, which concludes the proof. ◀

Proof of Theorem 1. The bound on the corruption follows by Lemma 8. The running time is dominated by Step 2, which takes time $n \cdot 2^{2^{O(d+k/\varepsilon)}}$. ◀

3.2 Poisoning k-NN: with train and test datasets

In this section, we describe the poisoning algorithm that will be used to obtain an m-poison with the guarantees of Theorem 2. This follows the algorithm from 3.1 with three major differences:

- The $\gamma_i(p)$ function is only defined with respect to the points within X_{train}.
- The random partition in Step 1 is only on X_{train}.
- The corruption in Step 2 is measured only on X_{test} for the test points that fall within the same cluster.

For any finite $Y \subset \mathbb{R}^d$, and any integer $i \geq 0$, let $\texttt{OPT}_i(X_{train}, X_{test})$ be the maximum corruption that can be achieved for X_{train}, X_{test} with a poison set size of at most i. Let $corruption(X_{train}, X_{test}, Y)$ be the corruption of X_{test} by flipping the labels of $Y \subseteq X_{train}$.

Algorithm Poison-k-NN for k-NN Poisoning with Train-Test: The input consists of X_{train} and $X_{test} \subset \mathbb{R}^d$, with $|X_{train}| = n_{train}$, $|X_{test}| = n_{test}$ and a map $\mathsf{label} : X \to \{1, 2\}$ that maps X_{train} and X_{test} to their corresponding labels.

Step 1. Sample the random partition P of X_{train} according to the algorithm in Lemma 6.

Step 2. For any cluster $C \subset X_{train}$ in P, by Lemma 7 we have that

$$|C| = k \cdot (\sqrt{d}(2k/\varepsilon)2^{8k/\varepsilon})^{d+O(1)} = (2k^2/\varepsilon) \cdot 2^{(d+O(1))8k/\varepsilon}.$$

For any $i \in \{1, \ldots, m\}$, we compute an optimal poisoning, $S_{C,i} \subseteq C$, for C with i poison points via brute-force enumeration. Each solution can be uniquely determined by selecting the i points for which we flip their label. Thus, the number of possible solutions is at most $2^{|C|} = 2^{(2k^2/\varepsilon)\cdot 2^{(d+O(1))8k/\varepsilon}}$. The enumeration can thus be done in time $2^{(2k^2/\varepsilon)\cdot 2^{(d+O(1))8k/\varepsilon}}$, for each cluster in P_X. For each possible solution, we also measure the corruption of the poisoning on the points in test set that fall within the same cluster, which takes $O(n_{test})$ time. Since there are at most n_{train} clusters, the total time is $(n_{train} + n_{test}) \cdot 2^{(2k^2/\varepsilon)\cdot 2^{(d+O(1))8k/\varepsilon}} = (n_{train} + n_{test}) \cdot 2^{2^{O(d+k/\varepsilon)}}$.

Step 3. We next combine the partial solutions computed in the previous step to obtain a solution for the whole pointset. This is done via dynamic programming, as follows. We order the clusters in P arbitrarily, as $P = \{C_1, \ldots, C_{|P|}\}$. For any $i \in \{0, \ldots, |P|\}$, $j \in \{1, \ldots, m\}$, let

$$A_{i,j} = \texttt{OPT}_j(C_1 \cup \ldots \cup C_i).$$

We can compute $A_{i,j}$ via dynamic programming using the formula

$$A_{i,j} = \begin{cases} \max\limits_{t \in [j]} \left(A_{i-1,t} + \mathsf{corruption}(C_i, S_{C_i, j-t})\right) & \text{if } i > 0 \\ 0 & \text{otherwise} \end{cases}$$

The size of the dynamic programming table is $O(|P| \cdot m) = O(nm)$. The same recursion can also be used to compute an optimal k-poison, Y, for $C_1 \cup \ldots \cup C_{|P|}$. The algorithm terminates and outputs Y as the final poison for X.

▶ **Lemma 9.** $E[\mathsf{corruption}(X_{train}, X_{test}, Y)] \geq \mathit{OPT}_m(X_{train}, X_{test}) - \varepsilon n_{test}$.

Proof. Let $Z \subseteq X_{train}$ be an optimal k-poison for X_{test}. Recall that P is the random partition sampled at Step 1.

For any $x \in X_{test}$, $i \in \mathbb{N}$, let $\mathcal{E}_{x,i}$ be the event that the cluster of P containing x, does not contain the i_{th}-nearest neighbor of x in X_{train}; i.e. $\mathsf{NN}_i(x) \notin P(x)$. Let also $\mathcal{E}_x$ be the event that the cluster of P containing x, does not contain all of the k-nearest neighbors of x in X_{train}; i.e.

$$\mathcal{E}_x = \mathcal{E}_{x,1} \vee \ldots \vee \mathcal{E}_{x,k}.$$

Thus

$$\begin{aligned}
\Pr[\mathcal{E}_x] &= \Pr[\mathcal{E}_{x,1} \vee \ldots \vee \mathcal{E}_{x,k}] \\
&\leq \sum_{i=1}^{k} \Pr[\mathcal{E}_{x,i}] && \text{(union bound)} \\
&= \sum_{i=1}^{k} \Pr[P(x) \neq P(\mathsf{NN}_i(x)] \\
&\leq \sum_{i=1}^{k} \frac{\varepsilon \|x - \mathsf{NN}_i(x)\|_2}{k\gamma(p)} && \text{(Lemma 6)} \\
&\leq k \frac{\varepsilon \|x - \mathsf{NN}_k(x)\|_2}{k\gamma(p)} \\
&= \varepsilon && (6)
\end{aligned}$$

Let

$$X' = \{x \in X_{test} : \{\mathsf{NN}_1(x), \ldots, \mathsf{NN}_k(x)\} \not\subseteq P(x)\}.$$

By (6) and the linearity of expectation, it follows that

$$E[|X'|] = \sum_{x \in |X_{test}|} Pr[\mathcal{E}_x] \leq \varepsilon |X_{test}| = \varepsilon n_{test}. \tag{7}$$

From (7), it follows that,

$$\begin{aligned}
\mathsf{corruption}(X_{train}, X_{test}, Y) &\geq \mathsf{corruption}(X_{train}, X_{test} \setminus X', Y) \\
&= \mathtt{OPT}_m(X_{train}, X_{test} \setminus X') \\
&\quad \text{(by the dynamic program)} \\
&\geq \mathtt{OPT}_m(X_{train}, X_{test}) - |X'|.
\end{aligned}$$

Combining with (7) and by linearity of expectation we get $E[\mathsf{corruption}(X, Y)] \geq \mathtt{OPT}_m(X) - E[|X'|] \geq \mathtt{OPT}_m(X) - \varepsilon n_{test}$, which concludes the proof. ◀

Proof of Theorem 2. The bound on the corruption on the test set follows Lemma 9. The running time is dominated by Step 2 of 3.2 which takes time $(n_{train} + n_{test}) \cdot 2^{2^{O(d+k/\varepsilon)}}$. ◀

4 Conclusion

We have introduced an approximation algorithm along with provable guarantees for a label flipping poisoning attack against the geometric classification task of k-nearest neighbors. Our poisoning framework, specifically the application of approximation algorithms using random metric partitions could also be extended to propose similar defense algorithms.

References

1 Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. Recent advances in adversarial training for adversarial robustness. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4312–4321. ijcai.org, 2021.

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

3 Nicholas Carlini. Poisoning the unlabeled dataset of semi-supervised learning. In Michael Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 1577–1592. USENIX Association, 2021.

4 Moses Charikar, Chandra Chekuri, Ashish Goel, Sudipto Guha, and Serge Plotkin. Approximating a finite metric by a small number of tree metrics. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*, pages 379–388. IEEE, 1998.

5 Anshuman Chhabra, Abhishek Roy, and Prasant Mohapatra. Suspicion-free adversarial attacks on clustering algorithms. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, February 7-12, 2020*, pages 3625–3632. AAAI Press, 2020.

6 Anshuman Chhabra, Adish Singla, and Prasant Mohapatra. Fairness degrading adversarial attacks against clustering algorithms. *CoRR*, abs/2110.12020, 2021.

7 Antonio Emanuele Cinà, Alessandro Torcinovich, and Marcello Pelillo. A black-box adversarial attack for poisoning clustering. *Pattern Recognition*, 122:108306, 2022. URL: https://www.sciencedirect.com/science/article/pii/S0031320321004866, doi:10.1016/j.patcog.2021.108306.

8 Evelyn Fix and Joseph Lawson Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique*, 57(3):238–247, 1989.

9 Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL: http://arxiv.org/abs/1412.6572.

10 Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. URL: https://www.sciencedirect.com/science/article/pii/0893608089900208, doi:10.1016/0893-6080(89)90020-8.

11 Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and J Doug Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58, 2011.

12 Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Certified robustness of nearest neighbors against data poisoning attacks. *CoRR*, abs/2012.03765, 2020. URL: https://arxiv.org/abs/2012.03765, arXiv:2012.03765.

13 James R Lee and Assaf Naor. Extending lipschitz functions via random metric partitions. *Inventiones mathematicae*, 160(1):59–95, 2005.

14 James R Lee and Anastasios Sidiropoulos. On the geometry of graphs with a forbidden minor. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 245–254, 2009.

15 Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019. arXiv:1706.06083.

16 Andrea Paudice, Luis Muñoz-González, and Emil C Lupu. Label sanitization against label flipping poisoning attacks. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 5–15. Springer, 2018.

17 Nikolaos Pitropakis, Emmanouil Panaousis, Thanassis Giannetsos, Eleftherios Anastasiadis, and George Loukas. A taxonomy and survey of attacks against machine learning. *Comput. Sci. Rev.*, 34, 2019.

18 Elan Rosenfeld, Ezra Winston, Pradeep Ravikumar, and Zico Kolter. Certified robustness to label-flipping attacks via randomized smoothing. In *International Conference on Machine Learning*, pages 8230–8241. PMLR, 2020.

19 Koosha Sadeghi, Ayan Banerjee, and Sandeep K. S. Gupta. A system-driven taxonomy of attacks and defenses in adversarial machine learning. *IEEE Trans. Emerg. Top. Comput. Intell.*, 4(4):450–467, 2020.

20 Mengchen Zhao, Bo An, Wei Gao, and Teng Zhang. Efficient label contamination attacks against black-box learning models. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3945–3951, 2017. doi:10.24963/ijcai.2017/551.