Training-Time Attacks against k-Nearest Neighbors

Ara Vartanian[1], **Will Rosenbaum**[2], **Scott Alfeld**[2]

[1] University of Wisconsin–Madison
[2] Amherst College
aravart@cs.wisc.edu, {wrosenbaum, salfeld}@amherst.edu

Abstract

Nearest neighbor-based methods are commonly used for classification tasks and as subroutines of other data-analysis methods. An attacker with the capability of inserting their own data points into the training set can manipulate the inferred nearest neighbor structure. We distill this goal to the task of performing a training-set data insertion attack against k-Nearest Neighbor classification (kNN). We prove that computing an optimal training-time (a.k.a. poisoning) attack against kNN classification is NP-Hard, even when $k = 1$ and the attacker can insert only a single data point. We provide an anytime algorithm to perform such an attack, and a greedy algorithm for general k and attacker budget. We provide theoretical bounds and empirically demonstrate the effectiveness and practicality of our methods on synthetic and real-world datasets. Empirically, we find that kNN is vulnerable in practice and that dimensionality reduction is an effective defense. We conclude with a discussion of open problems illuminated by our analysis.

1 Introduction

Using machine learning (ML) and automated data analysis methods in practice introduces security vulnerabilities. An attacker can assert their limited influence over the input data to manipulate the output of the learning system. Here, we consider training-time (a.k.a. poisoning) attacks against nearest-neighbor based classification, where the attacker perturbs data prior to learning.

k-Nearest Neighbors (kNN) is a classical non-parametric ML algorithm, where a data point label is chosen by selecting the plurality class of its k closest neighbors in the training set. Outside of classification, many algorithms construct a nearest neighbor graph (connecting each data point to its closest neighbors) as a subroutine. Examples include ISOMAP (Tenenbaum, de Silva, and Langford (2000)), persistent homology (Zhu (2013); Zhu et al. (2016)), and various spectral clustering algorithms (Luxburg (2007)).

We study neighbor-based classification as the canonical process of constructing nearest-neighbor graphs. An attacker which can affect classifications can similarly alter the nearest-neighbor graph. We lay the foundation for a theoretical understanding of training-time attacks against kNN. The primary contributions are:

1. We prove that computing an optimal training-time attack against k-Nearest Neighbor classification is NP-Hard.
2. We present a dynamic programming anytime algorithm that, if run to completion, computes an optimal single-point attack against kNN. Using this procedure as a subroutine, we employ and provide bounds for a greedy approach to efficiently construct an effective attack of any size against kNN.
3. We perform an empirical investigation on a method of defense against our attack. For example, reducing the number of dimensions via PCA in the MNIST dataset decreased the attacker's average effectiveness by as much as 26% while increasing prediction error by less than 1%.

In Section 2, after defining the learner and threat models for attacking kNN, we translate the attacker's task into a geometric problem. In Section 3, we prove that computing the optimal attack is NP-Hard, even for an attacker who targets one class with one additional data point. In Section 4, we present a (worst-case exponential-time) algorithm for finding the optimal single-point attack against kNN that uses dynamic programming to find successively better attacks as it proceeds, allowing it to be used as an anytime algorithm. We then employ a greedy algorithm to construct a data-insertion attack of any size against kNN. We show a bound for our greedy algorithm which is dependent on k (but independent of the attacker's budget). We discuss the "plug-and-play" nature of our analysis, in that any future improvements to the single-point attacks algorithm automatically yield a better bound for our general attack. In Section 5, we empirically demonstrate the effectiveness and practicality of our attack on synthetic and real world data. We then turn our attention to defense, and empirically investigate defense strategies based on dimensionality reduction. We discuss related work in Section 6 and conclude in Section 7 with a discussion of open problems.

2 Problem Setup

We focus on the role of an attacker, inserting training data so as to manipulate what is learned. A summary of notation used in this work is presented in the appendix.

Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

2.1 Learner Model

We consider a learning algorithm $\mathcal{A}$ performing k-Nearest Neighbors (kNN) classification. Given a training set $\mathcal{D}$ of $|\mathcal{D}|$ points in d dimensions and query point x, the learner predicts

$$\mathcal{A}[x; \mathcal{D}] = \text{Plurality label of the } k \text{ points in } \mathcal{D} \text{ closest to } x$$

where "closest" is defined in terms of an ℓ_p norm for $p \in (1, \infty)$. The specific implementation is not important to the work presented here. For simplicity of exposition, we ignore cases involving ties (two or more points in $\mathcal{D}$ are exactly the same distance from the query point) as well as cases where no strict plurality exists. Our methods can handle such cases (using e.g., random tie breaking) with minor modifications.

2.2 Threat Model

kNN is a non-parametric learner. As such, we consider a score function for the attacker based on the number of prescribed points $\mathcal{A}$ classifies according to the attacker's wishes. This measure of the effectiveness – the success rate of an attack – is in contrast to most previous work on training-time attacks against parametric learners, where the effectiveness is defined in terms of a distance between (learned and target) models (Vorobeychik and Kantarcioglu (2018)). We consider an attacker aiming to control the predicted labels of a set of points important to them. For example they may have a collection of emails that she would like classified as ham. They accomplish their task by constructing examples and inserting them into the training set. We formalize this threat model as follows.

The attacker, ATKR, has full knowledge of the training set $\mathcal{D}$,the value of k, and p (the norm used) . ATKR has a budget of b and their capability is to insert a set $\mathbf{\Delta}$ of b points to $\mathcal{D}$, each with features of their choosing[1] each with label we denote as y^+.

In addition, ATKR has an *target pool* $\mathcal{T} = \{(\mathbf{x}_1^{\text{tar}}, y^+), \ldots, (\mathbf{x}_n^{\text{tar}}, y^+)\}$. The *score* of an attack is the number of points in $\mathcal{T}$ that $\mathcal{A}$ labels consistently with $\mathcal{T}$:

$$\text{score}(\mathbf{\Delta}, \mathcal{T}, \mathcal{D}, \mathcal{A}) = \textstyle\sum_{(\mathbf{x},y)\in\mathcal{T}} \mathbf{1}_{(\mathcal{A}[\mathbf{x};\mathcal{D}\cup\mathbf{\Delta}]=y)}. \quad (1)$$

When $\mathcal{T}, \mathcal{D}$ and $\mathcal{A}$ are clear from context, we denote this quantity by $\text{score}(\mathbf{\Delta})$. ATKR's goal is to find an attack $\mathbf{\Delta}$ that maximizes their score, i.e., to find $\arg\max_{\mathbf{\Delta}\in(\mathbf{R}^d)^b} \text{score}(\mathbf{\Delta}, \mathcal{T}, \mathcal{D}, \mathcal{A})$.

2.3 Geometric View

We rephrase the problem of finding an optimal single-point training-time attack against kNN as the geometric problem of computing the maximum intersection of d-dimensional balls in $\mathbf{R}^d$.

We define the *influencing region* (IR) of a point $(\mathbf{x}_i^{\text{tar}}, y_i^{\text{tar}})$ in the target pool to be the set of feature vectors $\mathbf{x}$ such that adding the element $(\mathbf{x}, y^+)$ to the training set with some multiplicity ℓ changes the prediction of $\mathbf{x}_i^{\text{tar}}$ to y^+:

$$\text{IR}(\mathbf{x},y,\mathcal{D}) = \left\{\mathbf{x}' : \exists \ell \mid y = \mathcal{A}\left[\mathbf{x};\mathcal{D}\cup\{(\mathbf{x}',y^+)^\ell\}\right] \neq \mathcal{A}[\mathbf{x};\mathcal{D}]\right\} \quad (2)$$

[1]Constraints on which feature values ATKR can select, should any exist, can be incorporated into the attack algorithms as discussed in Section 4.

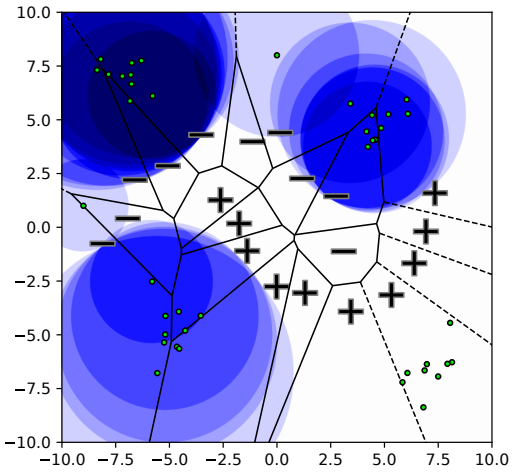

Figure 1: Synthetic Illustration. An illustration the balls (in this case, disks) induced by $\mathcal{D}$ (the training set) and $\mathcal{T}$ (the target pool) for 1NN in $d = 2$ dimensions. A training set from the well-known construction of two "moons" is shown as $+$'s and $-$'s, together with the decision boundary induced by it. We plot an example $\mathcal{T}$ as green dots. The color of each point in the plane indicates the total score increase (TSI) associated with adding a single $+$ element at that point in the plane: darker shades of blue indicate larger TSI values. Each colored region is an intersection of disks centered at the points in $\mathcal{T}$. The optimal single-point attack inserts a $+$ in one of the darkest blue regions.

Here, we use $\{(\mathbf{x}', y^+)^\ell\}$ to denote the multiset containing the element $(\mathbf{x}', y^+)$ with multiplicity ℓ. Observe that $\text{IR}(\mathbf{x}, y^+)$ is empty if $\mathcal{A}[\mathbf{x}; \mathcal{D}] = y^+$.

In the case of k-Nearest Neighbor classification using ℓ_p distance each $\text{IR}(\mathbf{x}, y^+, \mathcal{D})$ is a (possibly empty) ball[2] centered at $\mathbf{x}$. Specifically, $\text{IR}(\mathbf{x}, y^+, \mathcal{D})$ can be computed as follows. Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ be $\mathbf{x}$'s k nearest neighbors in (strictly) increasing order of distance from $\mathbf{x}$. Let j be the largest index such that setting the labels of $\mathbf{x}_j, \mathbf{x}_{j+1}, \ldots, \mathbf{x}_k$ all to y^+ changes $\mathcal{A}$'s prediction of $\mathbf{x}$ to y^+. Then $\text{IR}(\mathbf{x}, y^+, \mathcal{D})$ is the ball centered at $\mathbf{x}$ with radius $r = \|\mathbf{x} - \mathbf{x}_j\|_p$.

To see that this this procedure correctly computes $\text{IR}(\mathbf{x}, y^+, \mathcal{D})$, first observe that by adding $(\mathbf{x}', y^+)$ to $\mathcal{D}$ with multiplicity $\ell = k - j + 1$ and $\|\mathbf{x} - \mathbf{x}'\|_2 < r$, these new points replace $\mathbf{x}_j, \mathbf{x}_{j+1}, \ldots, \mathbf{x}_k$ among $\mathbf{x}$'s k nearest neighbors. Thus, $y^+ = \mathcal{A}\left[\mathbf{x}; \mathcal{D} \cup \{(\mathbf{x}', y^+)^\ell\}\right]$. Conversely, if $\|\mathbf{x} - \mathbf{x}'\|_p > r$, then adding $(\mathbf{x}', y^+)$ with any multiplicity cannot change the labels of $\mathbf{x}$'s j nearest neighbors. Therefore, by the maximality of j, adding these points does not change $\mathcal{A}$'s prediction of $\mathbf{x}$.

Given a point $(\mathbf{x}, y^+) \in \mathcal{T}$, we can associate a *cost* and *value* with $\text{IR}(\mathbf{x}, y^+, \mathcal{D})$. The *cost* $c = c(\mathbf{x}, y^+, \mathcal{D})$ is the minimum multiplicity of a point $(\mathbf{x}', y^+)$ with $\mathbf{x}' \in \text{IR}(\mathbf{x}, y^+, \mathcal{D})$ such that adding $(\mathbf{x}', y^+)$ with multiplicity c

[2]Recall that a ball B with center $\mathbf{c}$ and radius r is defined as $B = \left\{\mathbf{x} \in \mathbf{R}^d \,\middle|\, \|\mathbf{x} - \mathbf{c}\|_p \leq r\right\}$.

changes the classification of $\mathbf{x}$ to y^+. Note that for kNN, we always have $c \leq \lceil k/2 \rceil$.[3] The *value* v is determined by

$$v = \begin{cases} 1 & \text{if } y^+ \neq \mathcal{A}[\mathbf{x}; \mathcal{D}] \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Thus, v is the change score of an attack upon adding $(\mathbf{x}', y^+)$ to $\mathbf{\Delta}$ with multiplicity c, while ATKR's score is unchanged when adding $(\mathbf{x}', y^+)$ with any multiplicity less than c. Given $\mathcal{D}$ and $\mathcal{T}$, Algorithm 1 constructs a family $\mathcal{B}^{\text{lab}} = \{((\mathbf{x}, r), v, c)\}_i$ consisting of the influencing regions of points in $\mathcal{T}$, together with their associated values and costs.

We define the *total score increase* (TSI) of an attack point $(\mathbf{x}, y^+)$ with multiplicity ℓ to be the increase in ATKR's score if $(\mathbf{x}, y^+)$ is added to $\mathbf{\Delta}$ with multiplicity ℓ. TSI can be computed directly from $\mathcal{B}^{\text{lab}}$:

$$\text{TSI}(\mathbf{x}, y^+, \ell, \mathcal{B}^{\text{lab}}) = \sum_{((\mathbf{x}', r), v, c) \in \mathcal{B}^{\text{lab}}} v \cdot \mathbf{1}_{\|\mathbf{x} - \mathbf{x}'\|_2 < r} \cdot \mathbf{1}_{\ell \geq c} \tag{4}$$

This expression is central to the discussion of both our hardness result and our (greedy) algorithm for computing the full ($b > 1$) attack. Figure 1 shows an example for $p = 2$ of constructing hyperspheres and the induced TSI values.

3 Proof of Hardness

We define the problem ATK-KNN and show that it is NP-hard in general. An instance of ATK-KNN consists of a quadruple $(\mathcal{D}, \mathcal{T}, b, p)$, where $\mathcal{D}, \mathcal{T} \subseteq \mathbf{R}^d \times \mathcal{C}$, b is a positive integer (ATKR's *point budget*), and $p \geq 1$ is the ℓ_p norm used. Without loss of generality, we assume that all $(\mathbf{x}, y) \in \mathcal{D}$ are labeled the same fixed label y^-, and ATKR wishes for $\mathcal{A}$ to classify all $(\mathbf{x}, y^+) \in \mathcal{T}$ as $y^+ \neq y^-$. The desired output is the maximum score achieveable by adding any b element set $\mathbf{\Delta} \subseteq \mathbf{R}^d \times \mathcal{C}$ to $\mathcal{D}$:

[3] In the case where $y^+ = \mathcal{A}[\mathbf{x}; \mathcal{D}]$ (hence $\text{IR}(\mathbf{x}, y^+) = \varnothing$), we assign a cost of $c = 0$.

Algorithm 1: ConstructIRs

Input:
$\mathcal{D} = \{(\mathbf{x}^{\text{tr}}, y^{\text{tr}})\}_i, \mathcal{T} = \{(\mathbf{x}^{\text{tar}}, y^{\text{tar}})\}_i, y^+$
Output:
$\mathcal{B}^{\text{lab}} = \{((\mathbf{c}, r), v, c)\}_i$

$\mathcal{B}^{\text{lab}} \leftarrow [\,]$
for $(\mathbf{x}, y^{tar}) \in \mathcal{T}$ **do**
  nbr $\leftarrow [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k]$ where $\mathbf{x}_i$ is $\mathbf{x}$'s i^{th} nearest neighbor in $\mathcal{D}$
  lbl $\leftarrow [y_1, y_2, \ldots, y_k]$ where y_i is $\mathbf{x}_i$'s label in $\mathcal{D}$
  $v \leftarrow$ value according to (3)
  $j \leftarrow k$
  while plurality(lbl) $\neq y^+$ **do**
    lbl$[j] \leftarrow y^+$
    $j \leftarrow j - 1$
  Append $((\mathbf{x}, \|\mathbf{x} - \mathbf{x}_{j+1}\|_2), v, k - j + 1)$ to $\mathcal{B}^{\text{lab}}$
return $\mathcal{B}^{\text{lab}}$

$$\text{ATK-KNN}(\mathcal{D}, \mathcal{T}, b, p) = \max_{\mathbf{\Delta}: |\mathbf{\Delta}| = b} \text{score}(\mathbf{\Delta}, \mathcal{T}, \mathcal{D}, k\text{NN}).$$

Note that ATK-KNN is a special case of attacking kNN. Namely, $k = b = 1$, and all points in the target pool are the same label. In what follows we prove that ATK-KNN is NP-Hard. In the appendix, we show that the general case (higher k, b, and a heterogeneous target pool) is no easier.

Theorem 1. ATK-KNN *is NP-hard.*

When $b = |\mathbf{\Delta}| = 1$, ATK-KNN can be reduced to the problem of finding the maximum intersection of a family $\mathcal{B}$ of d-balls in $\mathbf{R}^d$ using Algorithm 1. For $b > 1$ and $k = 1$, ATK-KNN reduces to finding a set of b points in $\mathbf{R}^d$ intersecting the maximum number of d-balls $B \in \mathcal{B}$. We refer to the latter problem as the *maximum intersection problem*, or MAX-INT. We show that even for $b = 1$, MAX-INT is NP-hard. We then show that the instances of MAX-INT constructed in our reduction can be realized as instances of ATK-KNN, thereby establishing Theorem 1 for $k = 1$. We then give a simple modification of our construction that establishes the result for general k and b as well. We sketch the argument here; details appear in the appendix.

A d-dimensional instance of MAX-INT consists of a set $\mathcal{B} = \{B_1, B_2, \ldots, B_\ell\}$, where each B_i has a rational center and rational squared radius. The goal is to find the maximum number of mutually intersecting d-balls in $\mathcal{B}$, i.e., to find $\max\left\{\, |I| \mid \bigcap_{i \in I} B_i \neq \varnothing \right\}$.

We prove that MAX-INT is NP-hard via a reduction from the maximum independent set (MAX-IS) problem. Recall that given a graph $G = (V, E)$, an *independent set* $I \subseteq V$ is a set of vertices such that no pair of vertices in I share an edge. MAX-IS is to find the maximum cardinality among all independent sets in G. MAX-IS is known to be NP-complete (Garey and Johnson 1979).

Theorem 2. MAX-INT *is NP-hard.*

Towards proving Theorem 2, let $G = (V, E)$ be a graph on d vertices. We denote $V = \{1, 2, \ldots, d\}$ and edges in E by ij. The idea of the reduction is to associate a ball $B_i \subseteq \mathbf{R}^d$ with each vertex $i \in V$ and a ball B_{ij} with each edge $ij \in E$. Each triple of the form B_i, B_j, B_{ij} intersect pairwise, but the three do not mutally intersect (see Figure 2). The interpretation is that choosing a point $\mathbf{x} \in B_i$ corresponds to i being in an independent set I, while $\mathbf{x} \in B_{ij}$ corresponds to the edge ij being satisfied (i.e., i and j are not both in I). In our construction, a point $\mathbf{x}$ is contained in the largest number of balls in $\mathcal{B} = \{B_i\} \cup \{B_{ij}\}$ if and only if the set of vertices $I = \{i \mid \mathbf{x} \in B_i\}$ is a maximum independent set.

Formally, we define the reduction φ from MAX-IS to MAX-INT as follows. Given a graph G, we define a collection of balls $\varphi(G) = \mathcal{B} = \{B_i \mid i \in V\} \cup \{B_{ij} \mid ij \in E\}$ in $\mathbf{R}^d$, where B_i has center $r\mathbf{e}_i$ and radius r, while B_{ij} has center $-r(\mathbf{e}_i + \mathbf{e}_j)$ and radius $r\sqrt[p]{2 - \varepsilon}$. Here $\mathbf{e}_i$ denotes the i^{th} standard basis vector in $\mathbf{R}^d$. Values r and ε are positive numbers whose values will be determined later. That is,

$$B_i = \left\{ \mathbf{x} \in \mathbf{R}^d \,\middle|\, |x_i - r|^p + \textstyle\sum_{k \neq i} |x_k|^p \leq r^p \right\}, \tag{5}$$

$$B_{ij} = \left\{ \mathbf{x} \in \mathbf{R}^d \,\middle|\, |x_i + r|^p + |x_j + r|^p + \textstyle\sum_{k \neq i,j} |x_k|^p \leq 2|r - \varepsilon|^p \right\}. \tag{6}$$

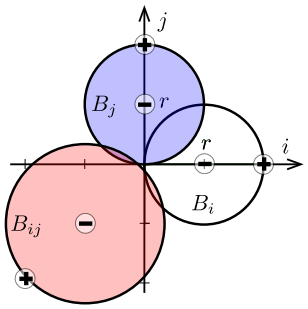

Figure 2: The configuration $\mathcal{B}$ for 2 dimensions. The disks intersect pair-wise, but not mutually. The instance of ATK-KNN where points in $\mathcal{D}$ are indicated with "+" signs, and points in $\mathcal{T}$ are indicated with "−" signs realizes $\mathcal{B}$.

The configuration $\mathcal{B}$ is depicted in Figure 2 for $d = p = 2$. We view $\mathcal{B}$ as a multiset where each B_i occurs with multiplicity 1, and each B_{ij} occurs with multiplicity d.

For any graph G, $\mathcal{B} = \varphi(G)$ can be computed in polynomial time. The G has an independent set I of size M if and only if $\mathcal{B}$ has a maximum intersection of size $|V|\,|E| + M$ (see appendix). Thus, φ is a polynomial-time reduction from MAX-IS to MAX-INT, and Theorem 2 follows.

Given our proof of Theorem 2, Theorem 1 follows by constructing instances of ATK-KNN realizing each instance of MAX-INT constructed in the proof of Theorem 2. To this end, we note that choosing

$$\mathcal{T}=\left\{(r\mathbf{e}_i,y^-)\,\middle|\,i\in V\right\}\cup\left\{(-r\mathbf{e}_i-r\mathbf{e}_j,y^-)\,\middle|\,ij\in E\right\} \quad (7)$$

$$\mathcal{D}=\left\{(2r\mathbf{e}_i,y^+)\,\middle|\,i\in V\right\}\cup\left\{(-(2r-\varepsilon)(\mathbf{e}_i+\mathbf{e}_j),y^+)\,\middle|\,ij\in E\right\} \quad (8)$$

suffices (see Figure 2). The proof appears in the appendix.

4 Proposed Methods

Having proven the hardness of ATKR's task, we now present two algorithms for attacking kNN. The first algorithm finds a one-point attack that maximizes the *coverage* of the attack point—i.e., the number of influencing regions in which it is contained. The second attack greedily performs one-point attacks to construct a full attack. Since finding the optimal single point to add to $\mathcal{D}$ is NP-Hard, we compute a one-point attack with an anytime algorithm that finds successively better solutions over time.

Our algorithms use Algorithm 1 as a preprocessing step to convert a training set and target pool to a labeled collection of balls.[4] The problem of mounting an optimal attack in which a single point is added to $\mathbf{\Delta}$ with multiplicity ℓ is completely determined by $\mathcal{B}^{\text{lab}}$ via Equation (4).

4.1 Constructing a One-Point Attack

We present an anytime algorithm which **C**onstructs a **H**ypergraph for **O**ne-**P**oint **P**oisoning **A**ttacks (CHOPPA). Consider the hypergraph $G = (V, E)$ where each vertex corresponds to one of the influencing regions and the hyper-edge $\mathbf{e} = \{v_i, \ldots, v_j\}$ exists if the intersection $v_i \cap \cdots \cap v_j$ is non-empty. Each hyper-edge represents a set of target points which can be influenced by a single training point with some multiplicity, ℓ. We say that a point $\mathbf{x}'$ *covers* $\mathbf{e}$ if $\mathbf{x}'$ is contained in the intersection of influencing regions in $\mathbf{e}$. We denote $\text{coverage}(\mathbf{x})$ the size of the maximal hyper-edge covered by $\mathbf{x}$. While $\mathbf{x}$ lies in the intersection of $\text{coverage}(\mathbf{x})$ influencing regions, adding $(\mathbf{x}, y^+)$ to $\mathcal{D}$ will only increase TSI by $\text{coverage}(\mathbf{x})$ if the point is added with sufficient multiplicity—i.e., the the maximum cost of any labeled ball in $\mathbf{e}$. Given a maximum allowable multiplicity ℓ, CHOPPA returns both an attack point $\mathbf{x}$ that covers some hyper-edge $\mathbf{e}$ and the minimum multiplicity $\ell_{\mathbf{x}} \leq \ell$ needed to change the classification of points in $\mathbf{e}$.

An optimal *coverage* single-point attack places a new training point in intersection with the largest coverage. In the case of attacking 1NN, an optimal coverage single point attack also maximizes the TSI of any single-point attack. For $k > 1$ adding any single point (with multiplicity 1) may have $\texttt{TSI} = 0$, though adding the point with multiplicity $\ell = k' = \lceil k/2 \rceil$ sufficies to achieve $\text{coverage}(\mathbf{x}) = \texttt{TSI}(\mathbf{x}, y)$. Thus, in general, we must have $\ell \geq k'$ in order to guarantee that a nontrivial (single point) attack exists.

Algorithm 2: CHOPPA

Input:
$\mathcal{D} = \{(\mathbf{x}^{\text{tr}}, y^{\text{tr}})\}_i, \mathcal{T} = \{(\mathbf{x}^{\text{tar}}, y^{\text{tar}})\}_i, y^+$
T, ℓ — Time budget, Point budget
Output:
$(\mathbf{x}, \ell_{\mathbf{x}})$ — Attack point and associated cost

```
H ← ConstructIRs(D, T, y+)
edges ← [∅]
done ← False
b ← ⊥, ℓ_best ← 0
while not done ∧ T not exhausted do
    newEdges ← ∅
    for e ∈ {e ∪ {h} | e ∈ edges, h ∈ H} do
        if all (|e| − 1)-sized subsets of e are in edges
        ∧ There exists a point of mutual overlap then
            newEdges ← newEdges ∪ {e}
            x ← any point in intersection
            if TSI(x, y+, ℓ, H) > TSI(b, y+, ℓ, H)
              then
                b ← x, ℓ_best ← max{c | (B, v, c) ∈ e}
    edges ← newEdges
    if edges = ∅ then
        done ← True
return (b, ℓ_best)
```

To compute G, we utilize the following observation: If an edge $\mathbf{e}$ of size m is not an element of E, then neither is any superset of $\mathbf{e}$. We use a dynamic-programming approach which first considers edges of size $m = 1$, then $m = 2$, etc. The run time is highly dependent on the structure of G. When all balls are identical there are an exponential number of edges to check, whereas if no two balls intersect, the algorithm terminates before checking for any size 3 sets. By

[4] A simple optimization is to remove from $\mathcal{B}^{\text{lab}}$ all balls with radius 0 and any for which $y_i^{\text{orig}} = y$ when $\mathcal{C} = \{y\}$. For ease of notation we ignore this optimization, maintaining $|\mathcal{B}^{\text{lab}}| = |\mathcal{T}|$.

Algorithm 3: GIT^{2}ACHOPPA

Input:
$\mathcal{D} = \{(x^{\text{tr}}, y^{\text{tr}})\}_i, \mathcal{T} = \{(x^{\text{tar}}, y^{\text{tar}})\}_i, y^+$
T, b Time budget, Point budget
Output:
$\mathbf{\Delta} = \{((\mathbf{x}_1, y_1), \ell_1), \ldots, ((\mathbf{x}_b, y_m), \ell_m)\}$ The attack

$\mathbf{\Delta} \leftarrow [\,], b_{\text{rem}} \leftarrow b$
while $b_{rem} > 0$ **do**
  $((\mathbf{x}, y), \ell) \leftarrow \texttt{CHOPPA}\left(\mathcal{D} \cup \mathbf{\Delta}, \mathcal{T}, y^+, \frac{T}{b}, b_{\text{rem}}\right)$
  if $\perp$ returned, then **break**
  $\mathbf{\Delta} \leftarrow \mathbf{\Delta} \cup \{(\mathbf{x}, y^+)\}$
  $b_{\text{rem}} \leftarrow b_{\text{rem}} - \ell$
return $\mathbf{\Delta}$

keeping track of the best point (a point with maximal TSI) as the algorithm considers larger cardinality collections of balls we allow for early termination given a time budget, making our solution deployable as an anytime algorithm.

4.2 Constructing a Full Attack

For the full attack, ATKR has a initial budget of b points to add to $\mathbf{\Delta}$. We adopt the following greedy strategy: invoke CHOPPA to find an optimal single-point attack with multiplicity ℓ set to the minimum of ATKR's remaining budget and $k' = \lceil k/2 \rceil$. Then add the point $\mathbf{x}$ returned by CHOPPA to $\mathbf{\Delta}$ with multiplicity $\ell_{\mathbf{x}} = \min\{c(\mathbf{x}') \mid \mathbf{x} \in \text{IR}(\mathbf{x}', y, \mathcal{D} \cup \mathbf{\Delta})\}$, and deduct $\ell_{\mathbf{x}}$ from the remaining budget.As CHOPPA is an anytime algorithm, we split a total time budget T evenly across the b calls.[5] We call this method **G**reedy **I**dentification of a **T**raining-**T**ime **A**ttack via CHOPPA (GIT^{2}ACHOPPA). It is a greedy anytime algorithm for the general $b > 1$ attack. The pseudocode is shown in Algorithm 3.

Since CHOPPA is an anytime algorithm, it may not always return an optimal single-point attack. Nonetheless, the following theorem asserts that if each call to CHOPPA runs to completion, and thus returns an optimal single-point attack, then the b-point attack found by GIT^{2}ACHOPPA is approximately optimal. In the appendix, we prove and generalize this result, and describe further practical optimizations.

Theorem 3. *Let $k' = \lceil k/2 \rceil$. Suppose each call to CHOPPA in GIT^{2}ACHOPPA returns an optimal single-point attack against kNN. Then the score of the $\mathbf{\Delta}$ returned by GIT^{2}ACHOPPA is a $\frac{1}{k'}(1 - 1/e)$-fraction of the optimal b-point attack's score.*

In practice, a user can detect when CHOPPA has run to completion (hence yielding an optimal result). Theorem 3 can then be used as a post-hoc bound. If each call to CHOPPA returns a solution that within a β factor of the optimal single-point attack, then the solution returned by GIT^{2}ACHOPPA is a $\frac{1}{k'}(1 - 1/e^{\beta})$-factor approximation to the optimal b-point attack. Details appear in the appendix.

We provide a full proof of Theorem 3 in the appendix, and provide a high level overview of the argument here. First, consider the case $k = 1$. The problem of mounting an optimal b-point attack can then be reduced to the *maximum coverage problem* (MCP) defined by (Hochbaum and Pathria 1998). An instance of the b-MCP consists of (1) a universal set U of elements where each $x \in U$ has an associated weight $w(x)$, and (2) a family $\mathcal{S}$ of subsets of U. The goal is to find a family of sets $\mathcal{F} \subseteq \mathcal{S}$ of size b that maximizes the quantity $\sum_x w(x)$, where the sum is taken over $\bigcup \mathcal{F}$.

For ATK-KNN, we have $U = \mathcal{T}$, and weights $w(\mathbf{x}', y')$ are ± 1 depending on if $y' = y^+$. Each $S \in \mathcal{S}$ consists of elements in $\mathcal{T}$ whose influencing regions mutually intersect.

In general, MCP is NP-hard, but Hochbaum and Pathria (1998) show that choosing sets greedily from $\mathcal{S}$ gives a $(1 - 1/e)$-factor approximation to the optimal solution. Under the assumptions that CHOPPA returns optimal single-point attacks and $k = 1$, GIT^{2}ACHOPPA simulates Hochbaum and Pathria's algorithm, whence Theorem 3 follows.

For $k > 1$, an optimal b-point attack is no longer equivalent to MCP. Indeed, a point $(\mathbf{x}', y') \in \mathcal{T}$ may need to be "covered" with multiplicity greater than 1 in order to have its prediction flipped, and it may be covered by multiple different sets in $\mathcal{S}$. Thus, the $k > 1$ case appears to be strictly more general than the MCP and the related "budgeted MCP" studied in (Khuller, Moss, and Naor 1999).

To obtain Theorem 3 for $k > 1$, let $\mathbf{\Delta}_{\text{opt}}$ be an optimal b-point attack—i.e., one that maximizes $\texttt{TSI}(\mathbf{\Delta})$. For any set $\mathbf{\Delta}$, let

$$\text{coverage}(\mathbf{\Delta}) = |\{(\mathbf{x}', y') \in \mathcal{T} \mid \mathbf{\Delta} \cap \text{IR}(\mathbf{x}', y, \mathcal{D}) \neq \varnothing\}|\,.$$

When $k = 1$, we have $\texttt{TSI}(\mathbf{\Delta}) = \text{coverage}(\mathbf{\Delta})$, and for all k, $\texttt{TSI}(\mathbf{\Delta}) \leq \text{coverage}(\mathbf{\Delta})$. Let $\mathbf{\Delta}_{\text{cov}}$ be a b'-point attack that maximizes $\text{coverage}(\mathbf{\Delta})$. Then we have

$$\texttt{TSI}(\mathbf{\Delta}_{\text{opt}}) \leq \text{coverage}(\mathbf{\Delta}_{\text{opt}}) \leq k' \text{coverage}(\mathbf{\Delta}_{\text{cov}}).$$

Now consider the set $\mathbf{\Delta}^*$ returned by GIT^{2}ACHOPPA in an execution in which each call to CHOPPA returns an optimal point. Since points are added to $\mathbf{\Delta}^*$ with multiplicities as large as k', it may contain as few as $b' = \lfloor b/k' \rfloor$ distinct points, and we have

$$\text{coverage}(\mathbf{\Delta}_{\text{cov}}) \leq (1 - 1/e)^{-1} \text{coverage}(\mathbf{\Delta}^*).$$

By construction—since each point $\mathbf{\Delta}^*$ is taken with sufficient multiplicity—we additionally have $\texttt{TSI}(\mathbf{\Delta}^*) = \text{coverage}(\mathbf{\Delta}^*)$. Combining the previous two expressions with (9) proves Theorem 3.

5 Experiments

Having laid the foundation for a theoretical understanding of attacking kNN at training time, we turn to the question of how well our attack works in practice with our ultimate application being defense. We explore the effectiveness of CHOPPA on synthetic data in scenarios small enough to compute the optimal one-point attack, then turn to real-world data and investigate GIT^{2}ACHOPPA on larger instances. Driven by the results on the synthetic data and with defense in mind, we look at the effect of dimensionality reduction on the attacks. Experiments were run on 128-core machines from AWS EC2. The QCLPs were solved using Mosek (ApS (2019)). Pre-processing, plotting, and data

[5] Examining more sophisticated scheduling is left as future work.

d:		2	4	8	16	32
Normal	ℓ_2	27.2	36.8	53.2	83.4	100.0
	ℓ_∞	27.0	37.2	51.0	73.8	88.6
Uniform	ℓ_2	27.0	34.2	54.4	83.6	100.0
	ℓ_∞	27.0	35.6	61.0	94.0	100.0

Table 1: ATKR's Scores on Synthetic Data. For dimensions $d = \{2, 4, 8, 16, 32\}$ and both ℓ_2 and ℓ_∞ norms, we report the ATKR score (as a percentage of successfully attacked points), averaged over trials and training set sizes. ATKR's effectiveness increases with dimension. All standard error of the means were less than 8.2%. Details are presented in Appendix D.

analysis were performed with pandas (The Pandas Development Team (2020)), scikit-learn (Pedregosa et al. (2011)) and matplotlib (Hunter (2007)).

In all experiments presented here we focus mostly on kNN using Euclidean distance, deferring some experiments using the ℓ_∞-norm to the appendix. Given a set of balls $\mathbf{e}$, we determine whether or not there exists a point in the intersection $\bigcap_i B_i$. For the ℓ_∞-norm, a simple interval intersection algorithm suffices. For the Euclidean norm, this is done by constructing the following quadratically constrained linear program (QCLP):[6]

$$\min_{\mathbf{x}} 1 \text{ s.t. } \|(\mathbf{x} - \mathbf{c}_1)\|_2^2 \leq r_1^2, \ldots, \|(\mathbf{x} - \mathbf{c}_{|\mathbf{e}|})\|_2^2 \leq r_{|\mathbf{e}|}^2. \quad (9)$$

The objective function of (9) is not important; if there exists any feasible point, then the balls have a point of mutual overlap. To reduce the number of times a QCLP solver is invoked, we make a further optimization. When considering a potential hyper-edge $\mathbf{e} = B_1, \ldots B_{|\mathbf{e}|}$ the following condition is necessary for $\mathbf{e}$ to be in E: All subsets of $\mathbf{e}$ of size $|\mathbf{e}| - 1$ must be in E. This check is computationally efficient because, when the check is performed, all edges of size $|\mathbf{e}| - 1$ have already been comptued. This is implemented on line 8 in Algorithm 2. If the first clause is false, the latter is not checked (the QCLP solver is not invoked).

5.1 Synthetic Experiments

We construct two families of synthetic experiments—Uniform and Normal—small enough to compute the optimal one-point attack. To create each instance, we sample m training points and 10 target points from the appropriate distribution, $\mathbf{X} \sim \text{Unif}([0, 1]^d)$ for Uniform, $\mathbf{X} \sim \text{Normal}(\mathbf{0}, \mathbf{I}_d)$ for Normal. For each family this procedure is repeated for the grid of values $m \in \{8, 16, 32, 64, 128\}$ and $d \in \{2, 4, 8, 16, 32\}$. For each (m, d) pair and family we perform 10 trials. Averages are reported in Table 1, with full details in Appendix D . Universally, ATKR is more successful in high dimenions. This suggests a natural defense strategy: perform dimension reduction prior to learning.

5.2 MNIST and HAPT

We consider two real-world datasets. MNIST (LeCun, Cortes, and Burges (2010)) consists of 28×28 greyscale images of handwritten digits ($d = 784$). Human Activity Recognition (HAPT) (Anguita et al. (2013)), is a dataset of sensor recordings of subjects performing a range of daily activities ($d = 561$). We use 6 of the labels from HAPT, omitting labels representing transitions between activities.

[6] If ATKR is constrained in her selection of $\mathbf{x}$, these constraints can be added to (9).

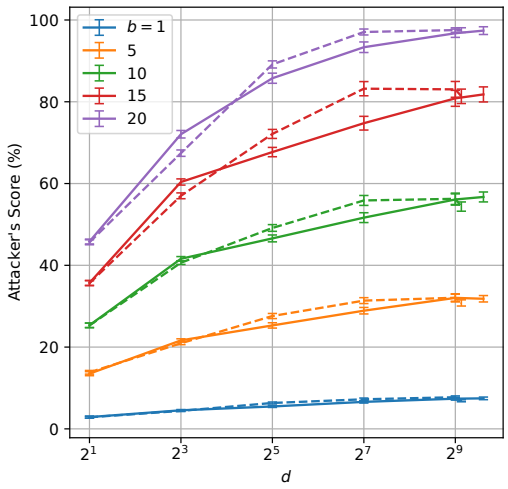

Figure 3: ATKR's Effectiveness as a Function of Dimension. For the original datasets and those reduced via PCA, we plot ATKR's score as a percentage of $|\mathcal{T}|$ averaged over 10 trials for each label. Error bars denote standard error of the mean. Attacks on MNIST (HAPT) datasets are shown as solid (dashed) lines. Universally, reducing the dimension is an effective defense.

For MNIST, we consider 10 attackers, one targeting each digit, and for HAPT we consider 6. Each attacker has an target pool of 50 points sampled uniformly at random from the original set that are originally classified as their label of interest. To evaluate dimensionality reduction as a form of defense we used Principle Component Analysis (PCA) (F.R.S. (1901)) on each dataset to reduce its dimension to $d = 2^1, 2^3, 2^5, 2^7, 2^9$. We train a 1NN classifier using 10,000 and 6,000 points from MNIST and HAPT, respectively, (1,000 from each class). Every attacker uses GIT^{2}ACHOPPA with a budget of $b = 1, 5, 10, 20$ and a time budget of b minutes. ATKR's effectiveness as a function of which class is attacked are omitted for brevity, as we noticed no definitive connection between classes that are difficult to classify and classes that are difficult to attack. Results are presented in Figure 3.

The attackers are very effective. Even with a modest budget of $b = 10$, ATKR is able to affect the predicted label of over half the target pool in both original datasets. Secondly, as in the synthetic experiments, the effectiveness of the attacks decreases as the data's dimension is reduced.

For dimensionality reduction to be a useful defense strategy, the defender would need to reduce the dimension while maintaining good performance on the underlying task. To understand the effects of reducing the dimension on the learning process, we report the percentage of variance explained (a measure independent of the choice of learner) and the zero-one loss of 1NN measured on a hold-out set

	MNIST		Hapt	
d	HOL	VarE	HOL	VarE
Original	0.102 (0.483)	1.000	0.087 (0.285)	1.000
512	0.126 (0.119)	0.999	0.093 (0.141)	1.000
128	0.111 (0.145)	0.936	0.081 (0.158)	0.983
32	0.103 (0.143)	0.745	0.111 (0.154)	0.891
8	0.201 (0.203)	0.438	0.217 (0.158)	0.774
2	0.613 (0.619)	0.169	0.431 (0.426)	0.661

Table 2: Loss and Variance Explained. We report the zero-one loss on a held out dataset (HOL) for ℓ_2-norm (ℓ_∞-norm) and the proportion of variance explained (VarE) for the each dataset (the original and those with dimension reduced by PCA). Reducing the dimension to 32 or 128 hardens the learner against attack while maintaining similar HOL.

of size 1,000 (randomly selected, disjoint from the training set) in Table 2. kNN continues to perform well even as the dimension is reduced (a well understood phenomenon) until the dimension is very low. ATKR's score, on the other hand, steadily decreases with the dimension. In the case of MNIST data, reducing to 32 dimensions decreased the attacker's effectiveness by between 26% ($b = 1$) and 12% ($b = 20$) while increasing prediction error by less than 1%. PCA served as a more costly defense for the HAPT data. At $d = 32$ the attacker's score decreased by between 11% and 8%, but the prediction loss increased by about 28%. Dimensionality reduction can be an effective defense for kNN against training-time attacks. Experiments on other real-world datasets showed qualitatively similar behavior and results are presented in Appendix D.

6 Related Work

Since its introduction (Fix and Hodges (1989)), kNN and related algorithms (e.g., rNN) have been extensively studied in contexts where no adversary is at play (Shakhnarovich, Darrell, and Indyk (2006); Bhatia et al. (2010)).

Training-time (a.k.a. poisoning or input manipulation) attacks were developed against various machine learning algorithms from deep networks (Koh and Liang (2017)) to support vector machines (Biggio, Nelson, and Laskov (2012)) to linear regression models (Jagielski et al. (2018); Alfeld et al. (2019)) to collaborative filtering (Li et al. (2016); Chen et al. (2020)) and online centroid anomaly detection (Kloft and Laskov (2010)). We complement this by furthering knowledge of another canonical learning method. We believe we are the first to investigate (theoretically or otherwise) training-time attacks against kNN.

Deployment-time attacks, often called "adversarial examples" or "adversarial perturbations", have been given a great deal of attention, especially against neural networks in recent years. With a focus on graph neural-networks Entezari *et al.* similarly observed that attackers are generally are less effective in low dimensions (Entezari et al. (2020)). We direct the reader to Biggio and Roli (2018); Vorobeychik and Kantarcioglu (2018); Joseph et al. (2018) for a general review and to Serban, Poll, and Visser (2020); Hazan, Papandreou, and Tarlow (2017) specifically for neural networks. Others have developed attacks against and analyzed the robustness of kNN and other non-parametric methods Wang, Jha, and Chaudhuri (2018); Bhattacharjee and Chaudhuri (2020)). In contrast to our work, they model an attacker acting at deployment-time, not training-time.

ATK-KNN is closely related to the "product positioning problem" in marketing theory (Albers and Brockhoff 1977), whose complexity is analyized in (Crama, Hansen, and Jaumard 1995). Their analysis gives an alternative proof of the special case of our Theorem 2 when $p = 2$ via a reduction from the maximum clique problem. Our proof of Theorem 2 is an adaptation of the argument in (Amaldi and Kann 1995), which shows that finding the maximum number of feasible linear constraints is NP-hard. The advantage of our proof of Theorem 2 is threefold: it is more general in that it covers ℓ_p norms for $p \in (1, \infty)$, it is both technically simpler than the argument of Crama, Hansen, and Jaumard (1995) and it is conceptually consistent with the remainder of our paper. Crama, Hansen, and Jaumard (1995) also describes an algorithm for MAX-INT that runs in time $O(n^{d+1})$, where n and d denote the number of points (i.e., size of our training set) and the dimension, respectively. The high-degree polynomial run-time of Crama et al.'s algorithm, however, is inherent to their approach, as they first test a set of $\Theta(n^d)$ potential single-point attacks before evaluating the scores of these attacks. As an anytime algorithm, CHOPPA finds good (albeit not necessarily optimal) solutions quickly.

7 Conclusions

We investigated the task of performing training-time attacks against nearest-neighbor based classifiers. We model an attacker adding points to the training set with the goal of affecting the classifications of points in an target pool. We proved that finding an optimal attack against kNN is NP-Hard, even when the attacker can add only a single data point to the training set. We introduced the anytime algorithm CHOPPA to efficiently compute an effective one-point attack and GIT^{2}ACHOPPA to compute a general attack. We provided a bound for GIT^{2}ACHOPPA when CHOPPA finds the optimal solution, and this bound holds in general if CHOPPA is replaced with a constant-approximation algorithm. With an eye toward defense strategies, we conducted an empirical investigation on synthetic and real-world data. Our experiments demonstrated both that the attack is highly effective in practice and that dimensionality reduction is often an effective defense strategy.

We close with open problems as potential future work: 1. Is there a polynomial-time, constant-factor approximation algorithm which solves the same problem as CHOPPA? Such an algorithm with approximation ratio α would yield a $(1 - e^{-\alpha})/k'$ approximation when used as a subroutine by GIT^{2}ACHOPPA. 2. Is there a bound for GIT^{2}ACHOPPA's performance that is independent of k? 3. Our defense reduces the dimensionality d. We observe empirically that this reduces the effectiveness of the optimal attack. However, the worst-case runtime of CHOPPA decreases with d. Are there effective defenses which act by increasing the computational burden of the attacker instead of (or in addition to) reducting the effectiveness of the optimal attack?

References

Albers, S.; and Brockhoff, K. 1977. A procedure for new product positioning in an attribute space. *European Journal of Operational Research*, 1(4): 230–238.

Alfeld, S.; Vartanian, A.; Newman-Johnson, L.; and Rubinstein, B. I. 2019. Attacking Data Transforming Learners at Training Time. In *AAAI*, volume 33, 3167–3174.

Amaldi, E.; and Kann, V. 1995. The complexity and approximability of finding maximum feasible subsystems of linear relations. *Theoretical Computer Science*, 147(1): 181–210.

Anguita, D.; Ghio, A.; Oneto, L.; Parra, X.; and Reyes-Ortiz, J. L. 2013. A public domain dataset for human activity recognition using smartphones. In *Esann*, volume 3, 3.

ApS, M. 2019. *MOSEK Optimizer API for Python 9.2.35*.

Bhatia, N.; et al. 2010. Survey of nearest neighbor techniques. *International Journal of Computer Science and Information Security*.

Bhattacharjee, R.; and Chaudhuri, K. 2020. When are Non-Parametric Methods Robust? In *ICML*, 832–841.

Biggio, B.; Nelson, B.; and Laskov, P. 2012. Poisoning Attacks against Support Vector Machines. In *ICML*, 1467–1474.

Biggio, B.; and Roli, F. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84: 317–331.

Chen, L.; Xu, Y.; Xie, F.; Huang, M.; and Zheng, Z. 2020. Data poisoning attacks on neighborhood-based recommender systems. *Transactions on Emerging Telecommunications Technologies*, e3872.

Crama, Y.; Hansen, P.; and Jaumard, B. 1995. Complexity of Product Positioning and Ball Intersection Problems. *Mathematics of Operations Research*, 20(4): 885–894.

Entezari, N.; Al-Sayouri, S. A.; Darvishzadeh, A.; and Papalexakis, E. E. 2020. All You Need Is Low (Rank) Defending Against Adversarial Attacks on Graphs. In *WSDM*, 169–177.

Fix, E.; and Hodges, J. L. 1989. Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties. *International Statistical Review / Revue Internationale de Statistique*, 57(3): 238–247.

F.R.S., K. P. 1901. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11): 559–572.

Garey, M. R.; and Johnson, D. S. 1979. *Computers and intractability*, volume 174. freeman San Francisco.

Guyon, I.; Gunn, S. R.; Ben-Hur, A.; and Dror, G. 2004. Result Analysis of the NIPS 2003 Feature Selection Challenge. In *NIPS*, volume 4, 545–552.

Hazan, T.; Papandreou, G.; and Tarlow, D. 2017. *Adversarial Perturbations of Deep Neural Networks*, 311–342. MIT Press.

Hochbaum, D. S.; and Pathria, A. 1998. Analysis of the greedy approach in problems of maximum k-coverage. *Naval Research Logistics (NRL)*, 45(6): 615–627.

Hunter, J. D. 2007. Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*, 9(3).

Jagielski, M.; Oprea, A.; Biggio, B.; Liu, C.; Nita-Rotaru, C.; and Li, B. 2018. Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, 19–35.

Joseph, A. D.; Nelson, B.; Rubinstein, B. I.; and Tygar, J. 2018. *Adversarial machine learning*. Cambridge University Press.

Khuller, S.; Moss, A.; and Naor, J. S. 1999. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1): 39–45.

Kloft, M.; and Laskov, P. 2010. Online Anomaly Detection under Adversarial Impact. In *AISTATS*, 405–412.

Koh, P. W.; and Liang, P. 2017. Understanding Black-box Predictions via Influence Functions. In *ICML*, 1885–1894.

LeCun, Y.; Cortes, C.; and Burges, C. 2010. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2.

Li, B.; Wang, Y.; Singh, A.; and Vorobeychik, Y. 2016. Data Poisoning Attacks on Factorization-Based Collaborative Filtering. In *Advances in Neural Information Processing Systems 29*, 1885–1893.

Luxburg, U. V. 2007. A Tutorial on Spectral Clustering. *Statistics and Computing*, 17(4): 395–416.

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

Sakar, B. E.; Isenkul, M. E.; Sakar, C. O.; Sertbas, A.; Gurgen, F.; Delil, S.; Apaydin, H.; and Kursun, O. 2013. Collection and analysis of a Parkinson speech dataset with multiple types of sound recordings. *IEEE Journal of Biomedical and Health Informatics*, 17(4): 828–834.

Serban, A.; Poll, E.; and Visser, J. 2020. Adversarial examples on object recognition: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 53(3): 1–38.

Shakhnarovich, G.; Darrell, T.; and Indyk, P. 2006. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. Neural Information Processing Series. MIT Press.

Tenenbaum, J. B.; de Silva, V.; and Langford, J. C. 2000. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290: 2319–2323.

The Pandas Development Team. 2020. pandas-dev/pandas: Pandas.

Vorobeychik, Y.; and Kantarcioglu, M. 2018. Adversarial machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3): 1–169.

Wang, Y.; Jha, S.; and Chaudhuri, K. 2018. Analyzing the robustness of nearest neighbors to adversarial examples. In *ICML*, 5133–5142. PMLR.

Zhu, X. 2013. Persistent homology: An introduction and a new text representation for natural language processing. In *IJCAI*, 1953–1959.

Zhu, X.; Vartanian, A.; Bansal, M.; Nguyen, D.; and Brandl, L. 2016. Stochastic Multiresolution Persistent Homology Kernel. In *IJCAI*, 2449–2457.