Near-Optimal Evasion of Convex-Inducing Classifiers

Blaine Nelson[1] **Benjamin I. P. Rubinstein**[1] **Ling Huang**[2] **Anthony D. Joseph**[1,2]
Shing-hon Lau[3] **Steven J. Lee**[1] **Satish Rao**[1] **Anthony Tran**[1] **J. D. Tygar**[1]
[1]Computer Science Division, UC Berkeley [2]Intel Labs Berkeley [3]School of Computer Science, CMU

Abstract

Classifiers are often used to detect miscreant activities. We study how an adversary can efficiently query a classifier to elicit information that allows the adversary to evade detection at near-minimal cost. We generalize results of Lowd and Meek (2005) to convex-inducing classifiers. We present algorithms that construct undetected instances of near-minimal cost using only polynomially many queries in the dimension of the space and without reverse engineering the decision boundary.

1 INTRODUCTION

Machine learning is often used to filter or detect miscreant activities in a variety of applications; *e.g.*, spam, intrusion, virus, and fraud detection. All known detection techniques have blind spots; *i.e.*, classes of miscreant activity that fail to be detected. While learning allows the detection algorithm to adapt over time, constraints on the learning algorithm also may allow an adversary to programmatically find these vulnerabilities. We consider how an adversary can systematically discover blind spots by querying the learner to find a low cost instance that the detector does not filter. Consider a spammer who wishes to minimally modify a spam message so it is not classified as a spam. By observing the responses of the spam detector, the spammer can search for a modification while using few queries.

The problem of near optimal evasion (*i.e.*, finding a low cost negative instance with few queries) was first posed by Lowd and Meek (2005). We continue this line of research by generalizing it to the family of convex-inducing classifiers—classifiers that partition their instance space into two sets: one of which is convex. Convex-inducing classifiers are a natural family to examine as they include linear classifiers, anomaly detection classifiers using bounded PCA (Lakhina et al., 2004), anomaly detection algorithms that use hypersphere boundaries (Bishop, 2006), and other more complicated bodies.

We also show that near-optimal evasion does not require reverse engineering the classifier. The algorithm of Lowd and Meek (2005) for evading linear classifiers reverse-engineers the decision boundary. Our algorithms for evading convex-inducing classifiers do not require fully estimating the classifier's boundary (which is hard in the general case; see Rademacher and Goyal, 2009) or reverse-engineering the classifier's state. Instead, we directly search for a minimal cost-evading instance. Our algorithms require only polynomial-many queries, with one algorithm solving the linear case with fewer queries than the previously-published reverse-engineering technique.

Related Work. Dalvi et al. (2004) uses a cost-sensitive game theoretic approach to patch a classifier's blind spots. They construct a modified classifier designed to detect optimally modified instances. This work is complementary to our own; we examine optimal evasion strategies while they have studied mechanisms for adapting the classifier. In this work we assume the classifier is not adapting during evasion.

A number of authors have studied evading intrusion detector systems (IDSs) (Tan et al., 2002; Wagner and Soto, 2002). In exploring *mimicry attacks* these authors demonstrated that real IDSs could be fooled by modifying exploits to mimic normal behaviors. These authors used offline analysis of the IDSs to construct their modifications; by contrast, our modifications are optimized by querying the classifier.

The field of active learning also studies a form of query based optimization (Schohn and Cohn, 2000). While both active learning and near-optimal evasion explore

Appearing in Proceedings of the 13^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2010, Chia Laguna Resort, Sardinia, Italy. Volume 9 of JMLR: W&CP 9.

optimal querying strategies, the objectives for these two settings are quite different (see Section 2.3).

2 PROBLEM SETUP

We begin by introducing our notation and assumptions. First, we assume that instances are represented in D-dimensional Euclidean space $\mathcal{X} = \mathbb{R}^D$. Each component of an instance $\mathbf{x} \in \mathcal{X}$ is a *feature* which we denote as $\mathbf{x}_d$. We denote each coordinate vector of the form $(0, \ldots, 1, \ldots, 0)$ with a 1 only at the d^{th} feature as δ_d. We assume that the feature space is known to the adversary and any point in $\mathcal{X}$ can be queried.

We further assume the target classifier f belongs to a family $\mathcal{F}$. Any classifier $f \in \mathcal{F}$ is a mapping from $\mathcal{X}$ to the labels '$-$' and '$+$'; *i.e.*, $f : \mathcal{X} \mapsto \{'-', '+'\}$. We assume the adversary's attack will be against a fixed f so the learning method and the training data used to select f are irrelevant. We assume the adversary does not know f but does know its family $\mathcal{F}$.

We assume $f \in \mathcal{F}$ is deterministic and so partitions $\mathcal{X}$ into a positive class $\mathcal{X}_f^+ = \{\mathbf{x} \in \mathcal{X} \mid f(\mathbf{x}) = '+'\}$ and a negative class $\mathcal{X}_f^- = \{\mathbf{x} \in \mathcal{X} \mid f(\mathbf{x}) = '-'\}$. We take the negative set to be *normal* instances. We assume the adversary is aware of at least one instance in each class, $\mathbf{x}^- \in \mathcal{X}_f^-$ and $\mathbf{x}^A \in \mathcal{X}_f^+$, and can observe $f(\mathbf{x})$ for any $\mathbf{x}$ by issuing a *membership query* (this last assumption does not always hold in practice, see Section 4 for a more detailed discussion).

2.1 Adversarial Cost

We assume the adversary has a notion of utility represented by a cost function $A : \mathcal{X} \mapsto \mathbb{R}^{0+}$. The adversary wishes to minimize A over the negative class, $\mathcal{X}_f^-$; *e.g.*, a spammer wants to send spam that will be classified as normal email ('$-$') rather than as spam ('$+$'). We assume this cost function is a distance to a positive target instance $\mathbf{x}^A \in \mathcal{X}_f^+$ that is most desirable to the adversary. As with Lowd and Meek, we focus on the class of weighted ℓ_1 cost functions

$$A(\mathbf{x}) = \sum_{d=1}^{D} c_d |\mathbf{x}_d - \mathbf{x}_d^A| \ , \tag{1}$$

where $0 < c_d < \infty$ is the cost the adversary associates with the d^{th} feature. The ℓ_1-norm is a natural measure of edit distance for email spam, while larger weights can model tokens that are more costly to remove (*e.g.*, a payload URL). We use $\mathcal{B}^C(\mathbf{x}^A)$ to denote the ball centered at $\mathbf{x}^A$ with cost no more than C. We use $\mathcal{B}_1^C(\mathbf{x})$ to refer specifically to a weighted ℓ_1 ball.

Lowd and Meek (2005) define *minimal adversarial cost (MAC)* of a classifier f to be the value

$$MAC(f, A) \triangleq \inf_{\mathbf{x} \in \mathcal{X}_f^-} [A(\mathbf{x})] \ .$$

They further define a data point to be an ϵ-approximate *instance of minimal adversarial cost (ϵ-IMAC)* if it is a negative instance with cost no more than a factor $(1+\epsilon)$ of the *MAC*; *i.e.*, every ϵ-*IMAC* is a member of the set[1]

$$\epsilon\text{-}IMAC(f, A) \triangleq \{\mathbf{x} \in \mathcal{X}_f^- \mid A(\mathbf{x}) \leq (1+\epsilon) \cdot \text{MAC}(f, A)\} \tag{2}$$

The adversary's goal is to find an ϵ-*IMAC* instance efficiently, while issuing as few queries as possible.

2.2 Search Terminology

An ϵ-*IMAC* instance is *multiplicatively optimal*; *i.e.*, it is within a *factor* of $(1+\epsilon)$ of the minimal cost. We also consider *additive optimality*; *i.e.*, requiring a η-*IMAC* to be no more than η *greater* than the minimal cost. The algorithms we present can achieve either criterion given initial bounds C^+ and C^- such that $C^+ \leq MAC \leq C^-$. If we can determine whether an intermediate cost establishes a new upper or lower bound on *MAC*, then binary search strategies can iteratively reduce the t^{th} gap between C_t^- and C_t^+. We now provide common terminology for the binary search and in Section 3 we use convexity to establish a new bound at each iteration.

In the t^{th} iteration of an additive binary search, $G_t^{(+)} = C_t^- - C_t^+$ is the additive gap between the t^{th} bounds. The search uses a proposal step of $C_t = \frac{C_t^- + C_t^+}{2}$, a stopping criterion of $G_t^{(+)} \leq \eta$ and terminates in

$$L^{(+)} = \left\lceil \log_2 \left[(C^- - C^+)/\eta \right] \right\rceil \tag{3}$$

steps. Binary search has the best worst-case query complexity for achieving η-additive optimality.

Binary search can be adapted for multiplicative optimality: by writing $C^- = 2^a$ and $C^+ = 2^b$, the multiplicative condition becomes $a - b \leq \log_2(1+\epsilon)$, an additive optimality condition. Thus, binary search on the exponent best achieves multiplicative optimality. The multiplicative gap of the t^{th} iteration is $G_t^{(*)} = C_t^-/C_t^+$. The t^{th} query is $C_t = \sqrt{C_t^- \cdot C_t^+}$, the stopping criterion is $G_t^{(*)} \leq 1+\epsilon$ and it stops in

$$L^{(*)} = \left\lceil \log_2 \left[\log_2 \left(C^-/C^+ \right) / \log_2(1+\epsilon) \right] \right\rceil \tag{4}$$

steps. Multiplicative optimality only makes sense when both C^- and C^+ are strictly positive.

[1] We use the term ϵ-*IMAC* to refer both to this set and members of it. The usage will be clear from the context.

For this paper, we only address multiplicative optimality and define $L = L^{(*)}$ and $G_t = G_t^{(*)}$, but note that our techniques also apply to additive optimality.

2.3 Near-Optimal Evasion

Lowd and Meek (2005) introduced the problem of *adversarial classifier reverse engineering (ACRE)* where a family of classifiers is called *ACRE ϵ-learnable* if there is an efficient query-based algorithm for finding an ϵ-*IMAC*. In generalizing their result, we slightly alter their definition of query complexity. First, to quantify query complexity we only use the dimension D and the number of steps L required by a univariate binary search. Second, we assume the adversary only has two initial points $\mathbf{x}^- \in \mathcal{X}_f^-$ and $\mathbf{x}^A \in \mathcal{X}_f^+$ (the original setting required a third $\mathbf{x}^+ \in \mathcal{X}_f^+$). Finally, our algorithms do not reverse engineer so ACRE would be a misnomer. Instead we call the overall problem *Near-Optimal Evasion* and replace *ACRE* ϵ-learnable with

> A family of classifiers $\mathcal{F}$ is ϵ-*IMAC searchable* under a family of cost functions $\mathcal{A}$ if for all $f \in \mathcal{F}$ and $A \in \mathcal{A}$, there is an algorithm that finds $\mathbf{x} \in \epsilon$-$IMAC(f, A)$ using polynomially many membership queries in D and L.

Reverse engineering is an expensive approach for near-optimal evasion in the general case. Efficient query-based reverse engineering for $f \in \mathcal{F}$ is sufficient for minimizing A over the estimated negative space. However, the requirements for finding an ϵ-*IMAC* differ from the objectives of reverse engineering approaches such as active learning. Both use queries to reduce the size of version space $\hat{\mathcal{F}} \subset \mathcal{F}$. However reverse engineering minimizes the expected number of disagreements between members of $\hat{\mathcal{F}}$. In contrast, to find an ϵ-*IMAC*, we only need to provide a single instance $\mathbf{x}^\dagger \in \epsilon$-$IMAC(f, A)$ for all $f \in \hat{\mathcal{F}}$, while leaving the classifier largely unspecified. We present algorithms for ϵ-*IMAC* search on a family of classifiers that generally cannot be efficiently reverse engineered—the queries we construct necessarily elicit an ϵ-*IMAC* only.

3 EVASION OF CONVEX CLASSES

We generalize ϵ-*IMAC* searchability to the family of *convex-inducing classifiers* $\mathcal{F}^{\text{convex}}$ that partition feature space $\mathcal{X}$ into a positive and negative class, one of which is convex. The convex-inducing classifiers include linear classifiers, one-class classifiers that predict anomalies by thresholding the log-likelihood of a log-concave (or uni-modal) density function, and quadratic classifiers of the form $\mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c \geq 0$ if $\mathbf{A}$ is semidefinite. The convex-inducing classifiers also include complicated families such as the set of all intersections of a countable number of halfspaces, cones, or balls.

We construct efficient algorithms for query-based optimization of the (weighted) ℓ_1 cost of Eq. (1) for convex-inducing classifiers. There appears to be an asymmetry depending on whether the positive or negative class is convex. When the positive set is convex, determining whether $\mathcal{B}_1^C(\mathbf{x}^A) \subset \mathcal{X}_f^+$ only requires querying the vertices of the ball. When the negative set is convex, determining whether $\mathcal{B}_1^C(\mathbf{x}^A) \cap \mathcal{X}_f^- = \emptyset$ is difficult since the intersection need not occur at a vertex. We present an efficient algorithm for optimizing an ℓ_1 cost when $\mathcal{X}_f^+$ is convex and a polynomial random algorithm for optimizing any convex cost when $\mathcal{X}_f^-$ is convex.

The algorithms we present achieve multiplicative optimality via binary search; we use L as the number of phases required by binary search, $C^- = A(\mathbf{x}^-)$ as an initial upper bound on the *MAC* and assume there is some $C^+ > 0$ that lower bounds the *MAC* (*i.e.*, $\mathbf{x}^A$ is in the interior of $\mathcal{X}_f^+$). This condition eliminates the degenerate case for which $\mathbf{x}^A$ is on the boundary of $\mathcal{X}_f^+$ where $MAC(f, A) = 0$ and ϵ-$IMAC(f, A) = \emptyset$.

3.1 ϵ-*IMAC* Search for a Convex $\mathcal{X}_f^+$

Solving the ϵ-*IMAC* search problem when $\mathcal{X}_f^+$ is convex is hard in the general case of convex cost $A(\cdot)$. We demonstrate algorithms for the (weighted) ℓ_1 cost that solve the problem as a binary search. Namely, given initial costs C^+ and C^- that bound the *MAC*, our algorithm can efficiently determine whether $\mathcal{B}_1^C(\mathbf{x}^A) \subset \mathcal{X}_f^+$ for any intermediate cost $C^+ < C < C^-$. If the ℓ_1 ball is contained in $\mathcal{X}_f^+$, then C becomes the new lower bound C^+. Otherwise C becomes the new upper bound C^-. Since our objective Eq. (2) is to obtain multiplicative optimality, our steps will be $C_t = \sqrt{C_{t-1}^+ \cdot C_{t-1}^-}$ (see Section 2.2). We now explain how we exploit the properties of the (weighted) ℓ_1 ball and convexity of $\mathcal{X}_f^+$ to efficiently determine whether $\mathcal{B}_1^C(\mathbf{x}^A) \subset \mathcal{X}_f^+$.

The existence of an efficient query algorithm relies on three facts: (1) $\mathbf{x}^A \in \mathcal{X}_f^+$; (2) every weighted ℓ_1 cost C-ball centered at $\mathbf{x}^A$ intersects $\mathcal{X}_f^-$ only if at least one of its vertices is in $\mathcal{X}_f^-$; and (3) C-balls only have $2 \cdot D$ vertices. We formalize the second fact as follows.

Lemma 3.1. *For all $C > 0$, if there exists some $\mathbf{x} \in \mathcal{X}_f^-$ that achieves a cost of $C = A(\mathbf{x})$, then there is*

some feature d such that a vertex of the form

$$\mathbf{x}^A \pm \tfrac{C}{c_d}\delta_d \tag{5}$$

is in $\mathcal{X}_f^-$ (and also achieves cost C by Eq. 1).

Proof. Suppose not; then there is some $\mathbf{x} \in \mathcal{X}_f^-$ such that $A(\mathbf{x}) = C$ and $\mathbf{x}$ has $M \geq 2$ features that differ from $\mathbf{x}^A$. Let $\{d_1, \ldots, d_M\}$ be the differing features and let $b_{d_i} = \operatorname{sign}\left(\mathbf{x}_{d_i} - \mathbf{x}^A_{d_i}\right)$ be the sign of the difference between $\mathbf{x}$ and $\mathbf{x}^A$ along the d_i-th feature. Let $\mathbf{e}_{d_i} = \mathbf{x}^A + \frac{C}{c_{d_i}} \cdot b_{d_i} \cdot \delta_{d_i}$ be a vertex of the form of Eq. (5) which has cost C (from Eq. 1). The M vertices $\mathbf{e}_{d_i}$ form a simplex of cost C on which $\mathbf{x}$ lies. If all $\mathbf{e}_{\mathbf{d_i}} \in \mathcal{X}_f^+$, then the convexity of $\mathcal{X}_f^+$ implies that $\mathbf{x} \in \mathcal{X}_f^+$ which violates our premise. Thus, if any instance in $\mathcal{X}_f^-$ achieves cost C, there is always a vertex of the form Eq. (5) in $\mathcal{X}_f^-$ that also achieves cost C. □

As a consequence, if all vertices of any C ball $\mathcal{B}_1^C\left(\mathbf{x}^A\right)$ are positive, then all $\mathbf{x}$ with $A(\mathbf{x}) \leq C$ are positive thus establishing C as a lower bound on the *MAC*. Conversely, if any of the vertices of $\mathcal{B}_1^C\left(\mathbf{x}^A\right)$ are negative, then C is an upper bound. Thus, by querying all $2 \cdot D$ vertices of $\mathcal{B}_1^C\left(\mathbf{x}^A\right)$, we either establish C as a new lower or upper bound on the *MAC*. By performing a binary search on C we iteratively halve the multiplicative gap between our bounds until it is within a factor of $1+\epsilon$. This yields an ϵ-*IMAC* of the form of Eq. (5).

A general form of this multiline search procedure is presented as Algorithm 3.2 which simultaneously searches along all unit-cost directions in the set $\mathcal{W}$. At each step, MultiLineSearch issues at most $|\mathcal{W}|$ queries to determine whether $\mathcal{B}_1^C\left(\mathbf{x}^A\right) \subset \mathcal{X}_f^+$. Once a negative instance is found at cost C, we cease further queries at cost C since a single negative instance is sufficient to establish a lower bound. We call this policy *lazy querying*. Further, when an upper bound is established for a cost C, our algorithm also prunes all directions that were positive at cost C. This pruning is sound; by the convexity assumption we know that the pruned direction is positive for all costs less than our new upper bound C. Applying MultiLineSearch to the $2 \cdot D$ axis-aligned directions yields an ϵ-*IMAC* for any (weighted) ℓ_1 cost with no more than $2 \cdot DL$ queries but at least $D + L$ queries. Thus the algorithm is $\mathcal{O}(DL)$.

3.1.1 K-step Multi-Line Search

The MultiLineSearch algorithm is $2 \cdot D$ simultaneous binary searches (breadth-first). Instead we could search sequentially (depth-first) and obtain a best case of $\mathcal{O}(D + L)$ and worst case of $\mathcal{O}(D \cdot L)$ but for exactly the opposite convex bodies. We therefore propose an algorithm that mixes these strategies. At each phase, the K-step MultiLineSearch (Algorithm 3.3) chooses a single direction $\mathbf{e}$ and queries it for K steps to generate candidate bounds B^- and B^+ on the *MAC*. The algorithm makes substantial progress without querying other directions. It then iteratively queries all remaining directions at the candidate lower bound B^+. Again we use lazy querying and stop as soon as a negative instance is found. We show that for $K = \lceil\sqrt{L}\rceil$, the algorithm achieves a delicate balance between breadth-first and depth-first approaches to attain a better worst-case complexity.

Algorithm 3.2. Multi-line Search

```
MLS(W, x^A, x^-, C^+, C^-, ε)
x* ← x^-
while C^-/C^+ > 1 + ε do begin
  C ← sqrt(C^+ · C^-)
  for all e ∈ W do begin
    Query classifier: f_e^C ← f(x^A + Ce)
    if f_e^C = '−' then begin
      x* ← x^A + Ce
      Prune i from W if f_i^C = '+'
      break for-loop
    end if
  end for
  if ∀e ∈ W f_e^C = '+' then C^+ ← C
  else C^- ← C
end while
return: x*
```

To analyze the worst case of K-step MultiLineSearch, we consider a *defender* that maximizes the number of queries. We refer to the querier as the *adversary*.

Theorem 3.4. *Algorithm 3.3 will find an ϵ-IMAC with at most $\mathcal{O}\left(L + \sqrt{L}|\mathcal{W}|\right)$ queries for $K = \lceil\sqrt{L}\rceil$.*

Proof. During the K steps of binary search, regardless of how the defender responds, the candidate gap along $\mathbf{e}$ will shrink by an exponent of 2^{-K}; *i.e.*,

$$B^-/B^+ = \left(C^-/C^+\right)^{2^{-K}} . \tag{6}$$

The primary decision for the defender occurs when the adversary begins querying other directions than $\mathbf{e}$. At iteration t, it has 2 options:

Case 1 ($t \in \mathcal{C}_1$): Respond with '+' for all remaining directions. Here the bounds B^+ and B^- are verified and thus the gap is reduced by an exponent of 2^{-K}.

Case 2 ($t \in \mathcal{C}_2$): Choose at least 1 direction to respond with '−'. Here the defender

Algorithm 3.3. K-Step Multi-line Search

$KMLS\left(\mathcal{W}, \mathbf{x}^A, \mathbf{x}^-, C^+, C^-, \epsilon, K\right)$
$\mathbf{x}^* \leftarrow \mathbf{x}^-$
while $C^-/C^+ > 1+\epsilon$ **do begin**
  Choose a direction $\mathbf{e} \in \mathcal{W}$
  $B^+ \leftarrow C^+$
  $B^- \leftarrow C^-$
  for K steps **do begin**
    $B \leftarrow \sqrt{B^+ \cdot B^-}$
    Query classifier: $f_{\mathbf{e}} \leftarrow f\left(\mathbf{x}^A + B\mathbf{e}\right)$
    if $f_{\mathbf{e}} =$ '+' **then** $B^+ \leftarrow B$
    else $B^- \leftarrow B$ **and** $\mathbf{x}^* \leftarrow \mathbf{x}^A + B\mathbf{e}$
  end for
  for all $\mathbf{i} \neq \mathbf{e} \in \mathcal{W}$ **do begin**
    Query classifier: $f_{\mathbf{i}} \leftarrow f\left(\mathbf{x}^A + (B^+)\mathbf{i}\right)$
    if $f_{\mathbf{i}} =$ '−' **then begin**
      $\mathbf{x}^* \leftarrow \mathbf{x}^A + (B^+)\mathbf{i}$
      Prune $\mathbf{k}$ from $\mathcal{W}$ if $f_{\mathbf{k}} =$ '+'
      break for-loop
    end if
  end for
  $C^- \leftarrow B^-$
  if $\forall \mathbf{i} \in \mathcal{W}\ f_{\mathbf{i}} =$ '+' **then** $C^+ \leftarrow B^+$
  else $C^- \leftarrow B^+$
end while
return: $\mathbf{x}^*$

can make the gap decrease negligible but also must choose some number $E_t \geq 1$ of eliminated directions.

By conservatively assuming the gap only decreases in case 1, the total number of queries is bounded regardless of the order in which the cases are applied. Thus if $t \in \mathcal{C}_1$ we have $G_t = G_{t-1}^{2^{-K}}$; otherwise we have $G_t = G_{t-1}$. Thus

$$|\mathcal{C}_1| \leq \left\lceil \tfrac{L}{K} \right\rceil \ , \tag{7}$$

since we need a total of L binary search steps and each case 1 iteration does K of them.

Every case 1 iteration makes exactly $K + |\mathcal{W}_t| - 1$ queries. The size of $\mathcal{W}_t$ is controlled by the defender, but we can bound it by $|\mathcal{W}|$. This and Eq. (7) bound the number of queries used in case 1 (Q_1) by

$$Q_1 = \sum_{t \in C_1} \left(K + |\mathcal{W}_t| - 1\right) \leq L + K + \left\lceil \tfrac{L}{K} \right\rceil \cdot \left(|\mathcal{W}| - 1\right)$$

Each case 2 iteration uses exactly $K + E_t$ queries and eliminates $E_t \geq 1$ directions. Since a case 2 iteration eliminates at least 1 direction, $|\mathcal{C}_2| \leq |\mathcal{W}| - 1$ and moreover, $\sum_{t \in \mathcal{C}_2} E_t \leq |\mathcal{W}| - 1$ since each direction can only be eliminated once. Thus

$$Q_2 = \sum_{i \in \mathcal{C}_2} \left(K + E_t\right) \leq \left(|\mathcal{W}| - 1\right)\left(K + 1\right) \ ,$$

and so the total queries used by Algorithm 3.3 is

$$Q = Q_1 + Q_2 < L + \left(\left\lceil \tfrac{L}{K} \right\rceil + K + 1\right)|\mathcal{W}| \ ,$$

which is minimized by $K = \lceil \sqrt{L} \rceil$. Substituting this for K and using $L/\lceil \sqrt{L} \rceil \leq \sqrt{L}$ we have

$$Q < L + \left(2\lceil \sqrt{L} \rceil + 1\right)|\mathcal{W}| \ . \qquad \square$$

As a consequence of Theorem 3.4, finding an ϵ-*IMAC* with Algorithm 3.3 for a (weighted) ℓ_1 cost requires $\mathcal{O}\left(L + \sqrt{L}D\right)$ queries. Moreover, linear classifiers are a special case of convex-inducing classifiers for our K-step MultiLineSearch algorithm. Thus K-step MultiLineSearch improves on the reverse-engineering technique's $\mathcal{O}\left(LD\right)$ queries and applies to a broader family.

3.1.2 Lower Bound

Here we find lower bounds on the number of queries required by any algorithm to find an ϵ-*IMAC* when $\mathcal{X}_f^+$ is convex. Notably, since an ϵ-*IMAC* uses multiplicative optimality, we incorporate a lower bound $r > 0$ on the *MAC* into our statement.

Theorem 3.5. *Consider any $D > 0$, $\mathbf{x}^A \in \mathbb{R}^D$, $\mathbf{x}^- \in \mathbb{R}^D$, $0 < r < R = A\left(\mathbf{x}^-\right)$ and $\epsilon \in \left(0, \frac{R}{r} - 1\right)$. For all query algorithms submitting $N < \max\{D, L^{(*)}\}$ queries, there exist two classifiers inducing convex positive classes in $\mathbb{R}^D$ such that*

1. *Both positive classes properly contain $\mathcal{B}^r\left(\mathbf{x}^A\right)$;*
2. *Neither positive class contains $\mathbf{x}^-$;*
3. *The classifiers return the same responses on the algorithm's N queries; and*
4. *The classifiers have no common ϵ-IMAC.*

That is, in the worst-case all query algorithms for convex positive classes must submit at least $\max\{D, L^{(*)}\}$ membership queries in order to be multiplicative ϵ-optimal.

Proof. Suppose some query-based algorithm submits N membership queries $\mathbf{x}^1, \ldots, \mathbf{x}^N$ to the classifier. For the algorithm to be ϵ-optimal, these queries must constrain all consistent positive convex sets to have a common point among their ϵ-*IMAC* sets.

First we consider the case that $N \geq L$. Then by assumption $N < D$. Suppose classifier f responds as

$$f\left(\mathbf{x}\right) = \begin{cases} +1\,, & \text{if } A\left(\mathbf{x}\right) < R \\ -1\,, & \text{otherwise} \end{cases} \ .$$

For this classifier, $\mathcal{X}_f^+$ is convex, $\mathcal{B}^r\left(\mathbf{x}^A\right) \subset \mathcal{X}_f^+$, and $\mathbf{x}^- \notin \mathcal{X}_f^+$. Moreover, since $\mathcal{X}_f^+$ is the open ball of cost R, $MAC(f, A) = R$.

Consider an alternative classifier g that responds identically to f for $\mathbf{x}^1, \ldots, \mathbf{x}^N$ but has a different convex positive set $\mathcal{X}_g^+$. Without loss of generality, suppose the first $M \leq N$ queries are positive and the remaining are negative. Let $\mathcal{G} = conv\left(\mathbf{x}^1, \ldots, \mathbf{x}^M\right)$; that is, the convex hull of the M positive queries. Now let $\mathcal{X}_g^+$ be the convex hull of the union of $\mathcal{G}$ and the r-ball around $\mathbf{x}^A$: $\mathcal{X}_g^+ = conv\left(\mathcal{G} \cup \mathcal{B}^r\left(\mathbf{x}^A\right)\right)$. Since $\mathcal{G}$ contains all positive queries and $r < R$, the convex set $\mathcal{X}_g^+$ is consistent with the responses from f, $\mathcal{B}^r\left(\mathbf{x}^A\right) \subset \mathcal{X}_f^+$, and $\mathbf{x}^- \notin \mathcal{X}_f^+$. Further, since $M \leq N < D$, $\mathcal{G}$ is contained in a proper subspace of $\mathbb{R}^D$ whereas $\mathcal{B}^r\left(\mathbf{x}^A\right)$ is not. Hence, $MAC(g, A) = r$. Since the accuracy ϵ is less than $\frac{R}{r} - 1$, any ϵ-*IMAC* of g must have cost less than R whereas any ϵ-*IMAC* of f must have cost greater than or equal to R. Thus we have constructed two convex-inducing classifiers f and g with consistent query responses but with no common ϵ-*IMAC*.

Second, we consider the case that $N < L$. First, recall our definitions: $C_0^- = R$ is the initial upper bound on the *MAC*, $C_0^+ = r$ is the initial lower bound on the *MAC*, and $G_t^{(*)} = C_t^- / C_t^+$ is the gap between the upper bound and lower bound at iteration t. Here the defender f responds with

$$f\left(\mathbf{x}^t\right) = \begin{cases} +1\,, & \text{if } A\left(\mathbf{x}^t\right) \leq \sqrt{C_{t-1}^- \cdot C_{t-1}^+} \\ -1\,, & \text{otherwise} \end{cases}.$$

This strategy ensures that at each iteration $G_t \geq \sqrt{G_{t-1}}$ and since the algorithm can not terminate until $G_N \leq 1 + \epsilon$, we have $N \geq L^{(*)}$ from Eq. (4). As in the $N \geq L$ case we have constructed two convex-inducing classifiers with consistent query responses but with no common ϵ-*IMAC*. The first classifier's positive set is the smallest cost-ball enclosing all positive queries, while the second classifier's positive set is the largest cost-ball enclosing all positive queries but no negatives. The *MAC* values of these sets differ by more than a factor of $(1 + \epsilon)$ if $N < L^{(*)}$ so they have no common ϵ-*IMAC*. □

This theorem shows that ϵ-multiplicative optimality requires $\Omega\left(D + L\right)$ queries. Hence K-step MultiLineSearch (Algorithm 3.3) has close to the optimal query complexity.

3.2 ϵ-*IMAC* Learning for a Convex $\mathcal{X}_f^-$

In this section we consider minimizing a convex cost function A (we focus on weighted ℓ_1 costs in Eq. 1) when the feasible set $\mathcal{X}_f^-$ is convex. Any convex function can be efficiently minimized within a known convex set *e.g.*, using the Ellipsoid or Interior Point methods (Boyd and Vandenberghe, 2004). However, in our problem the convex set is only accessible through queries. We use a randomized polynomial algorithm of Bertsimas and Vempala (2004) to minimize the cost given an initial $\mathbf{x}^- \in \mathcal{X}_f^-$. For any fixed cost C^t we use their algorithm to determine (with high probability) whether $\mathcal{X}_f^-$ intersects with $\mathcal{B}^{C^t}\left(\mathbf{x}^A\right)$; *i.e.*, whether or not C^t is a new lower or upper bound on the *MAC*. With high probability, we find an ϵ-*IMAC* in no more than L repetitions using binary search.

3.2.1 Intersection of Convex Sets

We now outline Bertsimas and Vempala's query-based algorithm for determining whether two convex sets intersect using a randomized Ellipsoid method. In particular $\mathcal{P}$ is only accessible through membership queries and $\mathcal{B}$ provides a separating hyperplane for any point outside it. They use efficient query-based approaches to uniformly sample from $\mathcal{P}$ to produce sufficiently many samples such that cutting $\mathcal{P}$ through the centroid of these samples with a separating hyperplane from $\mathcal{B}$ significantly reduces the volume of $\mathcal{P}$ with high probability. Their algorithm thus constructs a sequence of progressively smaller feasible sets $\mathcal{P}^s \subset \mathcal{P}^{s-1}$ until either the algorithm finds a point in $\mathcal{P} \cap \mathcal{Q}$ or it is highly unlikely that the sets intersect.

Our problem reduces to finding the intersection between $\mathcal{X}_f^-$ and $\mathcal{B}_1^{C^t}\left(\mathbf{x}^A\right)$. Though $\mathcal{X}_f^-$ may be unbounded, we can instead use $\mathcal{P}^0 = \mathcal{X}_f^- \cap \mathcal{B}_1^{2R}\left(\mathbf{x}^-\right)$ (where $R = 2A\left(\mathbf{x}^-\right)$) is a subset of $\mathcal{X}_f^-$ that envelops all of $\mathcal{B}_1^{C^t}\left(\mathbf{x}^A\right)$ since $C^t < A\left(\mathbf{x}^-\right)$. We also assume there is some $r > 0$ such that an r-ball centered at $\mathbf{x}^-$ is contained in $\mathcal{X}_f^-$. We now detail this IntersectSearch procedure (Algorithm 3.6).

The backbone of the algorithm is uniform sampling from a bounded convex body by means of the hit-and-run random walk technique introduced by Smith (1996) (Algorithm 3.7). Given an instance $\mathbf{x}^j \in \mathcal{P}^{s-1}$, hit-and-run selects a random direction $\mathbf{v}$ through $\mathbf{x}^j$ (we return to the selection of $\mathbf{v}$ in Section 3.2.2). Since $\mathcal{P}^{s-1}$ is a bounded convex set, the set $\Omega = \{\omega \mid \mathbf{x}^j + \omega\mathbf{v} \in \mathcal{P}^{s-1}\}$ is a bounded interval representing all points in $\mathcal{P}^{s-1}$ along direction $\mathbf{v}$. Sampling ω uniformly from Ω yields the next step of the walk; $\mathbf{x}^j + \omega\mathbf{v}$. Under the appropriate conditions (see Section 3.2.2), hit-and-run generates a sample uniformly from the convex body after $\mathcal{O}^*\left(D^3\right)$ steps[2] (Lovász and Vempala, 2004).

Using hit-and-run we obtain $2N$ samples $\{\mathbf{x}^j\}$ from $\mathcal{P}^{s-1}$ and check if any satisfy $A\left(\mathbf{x}^j\right) \leq C^t$. If so, $\mathbf{x}^j$ is in the intersection of $\mathcal{X}_f^-$ and $\mathcal{B}_1^{C^t}\left(\mathbf{x}^A\right)$. Otherwise, we want to significantly reduce the size of $\mathcal{P}^{s-1}$ without

[2] $\mathcal{O}^*\left(\cdot\right)$ denotes $\mathcal{O}\left(\cdot\right)$ without logarithmic terms.

Algorithm 3.6. Intersect Search

IntersectSearch $\left(\mathcal{P}^0, \mathcal{Q} = \{\mathbf{x}^j \in \mathcal{P}^0\}, C\right)$
for all $s = 1 \ldots T$ **do begin**
(1) Generate $2N$ samples $\{\mathbf{x}^j\}_{j=1}^{2N}$
Choose $\mathbf{x}$ from $\mathcal{Q}$
$\mathbf{x}^j \leftarrow$ *HitRun* $\left(\mathcal{P}^{s-1}, \mathcal{Q}, \mathbf{x}^j\right)$
(2) If any $\mathbf{x}^j$, $A\left(\mathbf{x}^j\right) \leq C$ terminate the for-loop
(3) Put samples into 2 sets of size N
$\mathcal{R} \leftarrow \{\mathbf{x}^j\}_{j=1}^{N}$ and $\mathcal{S} \leftarrow \{\mathbf{x}^j\}_{j=2N+1}^{2N}$
(4) $\mathbf{z}^s \leftarrow \frac{1}{N}\sum_{\mathbf{x}^j \in \mathcal{R}} \mathbf{x}^j$
(5) Compute $\mathcal{H}_{\mathbf{z}^s}$ using Eq. (9)
(6) $\mathcal{P}^s \leftarrow \mathcal{P}^{s-1} \cap \mathcal{H}_{\mathbf{z}^s}$
(7) Keep samples in $\mathcal{P}^s$
$\mathcal{Q} \leftarrow \{\mathbf{x} \in \mathcal{S} \wedge \mathbf{x} \in \mathcal{P}^s\}$
end for
Return: the found $[\mathbf{x}_j, \mathcal{P}^s, \mathcal{Q}]$; or No Intersect

Algorithm 3.7. Hit-and-Run Sampling

HitRun $\left(\mathcal{P}, \{\mathbf{y}^j\}, \mathbf{x}^0\right)$
for all $i = 1 \ldots K$ **do begin**
Pick a random direction:
$\nu_j \sim \mathrm{N}(0,1)$
$\mathbf{v} \leftarrow \sum_j \nu_j \mathbf{y}^j$
Find ω_1 and ω_2 s.t.
$\mathbf{x}^{i-1} - \omega_1 \mathbf{v} \notin \mathcal{P}$ and $\mathbf{x}^{i-1} + \omega_2 \mathbf{v} \notin \mathcal{P}$
repeat
$\omega \sim$ *Unif* $(-\omega_1, \omega_2)$
$\mathbf{x}^i \leftarrow \mathbf{x}^{i-1} + \omega \mathbf{v}$
if $\omega < 0$ **then** $\omega_1 \leftarrow -\omega$
else $\omega_2 \leftarrow \omega$
until $\mathbf{x}^i \in \mathcal{P}$
end for
Return: $\mathbf{x}^K$

excluding any of $\mathcal{B}_1^{C^t}\left(\mathbf{x}^A\right)$ so that sampling concentrates towards the intersection (if it exists)—for this we need a separating hyperplane of $\mathcal{B}_1^{C^t}\left(\mathbf{x}^A\right)$. For any $\mathbf{y} \notin \mathcal{B}_1^{C^t}\left(\mathbf{x}^A\right)$, the (sub)gradient of the weighted ℓ_1 cost given by

$$\mathbf{h}_f^{\mathbf{y}} = c_f \operatorname{sign}\left(\mathbf{y}_f - \mathbf{x}_f^A\right) \tag{8}$$

separates $\mathbf{y}$ and $\mathcal{B}_1^{C^t}\left(\mathbf{x}^A\right)$.

To achieve efficiency, we choose a point $\mathbf{z} \in \mathcal{P}^{s-1}$ so that cutting $\mathcal{P}^{s-1}$ through $\mathbf{z}$ with the hyperplane $\mathbf{h}^{\mathbf{z}}$ eliminates a significant fraction of $\mathcal{P}^{s-1}$. To do so, $\mathbf{z}$ must be centrally located within $\mathcal{P}^{s-1}$. We use the empirical centroid of half of the samples $\mathbf{z} = \frac{1}{N}\sum_{\mathbf{x} \in \mathcal{R}} \mathbf{x}$ (the other half will be used in Section 3.2.2). We cut $\mathcal{P}^{s-1}$ with the hyperplane $\mathbf{h}^{\mathbf{z}}$ through $\mathbf{z}$; *i.e.*, $\mathcal{P}^s = \mathcal{P}^{s-1} \cap \mathcal{H}_{\mathbf{z}}$ where $\mathcal{H}_{\mathbf{z}}$ is the halfspace

$$\mathcal{H}_{\mathbf{z}} = \{\mathbf{x} \mid \mathbf{x}^\top \mathbf{h}^{\mathbf{z}} \leq \mathbf{z}^\top \mathbf{h}^{\mathbf{z}}\} \ . \tag{9}$$

As shown by Bertsimas and Vempala, this cut achieves $vol\left(\mathcal{P}^s\right) \leq \frac{2}{3} vol\left(\mathcal{P}^{s-1}\right)$ with high probability if $N = \mathcal{O}^*(D)$ and $\mathcal{P}^{s-1}$ is near-isotropic (see Section 3.2.2). Since the ratio of volumes between the initial circumscribing and inscribing balls of the feasible set is $\left(\frac{R}{r}\right)^D$, the algorithm can terminate after $T = \mathcal{O}\left(D \log \frac{R}{r}\right)$ unsuccessful iterations with a high probability that the intersection is empty.

Because every iteration in Algorithm 3.6 requires $N = \mathcal{O}^*(D)$ samples, each of which need $K = \mathcal{O}^*\left(D^3\right)$ random walk steps, and there are $\mathcal{O}^*(D)$ iterations, Algorithm 3.6 requires $\mathcal{O}^*\left(D^5\right)$ queries.

3.2.2 Sampling from a Convex Body

Until this point, we assumed the HIT-AND-RUN random walk efficiently produces uniformly random samples from any bounded convex body $\mathcal{P}$ accessible through membership queries. However, if the body is severely elongated, randomly selected directions will rarely align with the long axis of the body and our random walk will take small steps (relative to the long axis) and mix slowly. For the sampler to mix effectively, we need the convex body $\mathcal{P}$ to be *near-isotropic*; *i.e.*, for any unit vector $\mathbf{v}$, $\mathbb{E}_{\mathbf{x}\sim\mathcal{P}}\left[\left(\mathbf{v}^\top\left(\mathbf{x} - \mathbb{E}_{\mathbf{x}\sim\mathcal{P}}[\mathbf{x}]\right)\right)^2\right]$ is bounded between $1/2$ and $3/2$ of $vol(\mathcal{P})$.

If the body is not near-isotropic, we can rescale $\mathcal{X}$ with an appropriate affine transformation $\mathbf{T}$. With sufficiently many samples from $\mathcal{P}$ we can estimate $\mathbf{T}$ as their empirical covariance matrix. Instead, we rescale $\mathcal{X}$ implicitly using a technique described by Bertsimas and Vempala (2004). We maintain a set $\mathcal{Q}$ of sufficiently many uniform samples from the body $\mathcal{P}^s$ and in HIT-AND-RUN we sample directions based on this set. Because the samples are distributed uniformly in $\mathcal{P}^s$, the directions we sample based on the points in $\mathcal{Q}$ implicitly reflect the covariance structure of $\mathcal{P}^s$.

We must ensure $\mathcal{Q}$ is a set of sufficiently many samples from $\mathcal{P}^s$ after each cut: $\mathcal{P}^s \leftarrow \mathcal{P}^{s-1} \cap \mathcal{H}_{\mathbf{z}^s}$. To do so, we resample $2N$ points from $\mathcal{P}^{s-1}$ using HIT-AND-RUN—half of these, $\mathcal{R}$, are used to estimate the centroid $\mathbf{z}^s$ for the cut and the other half, $\mathcal{S}$, are used to repopulate $\mathcal{Q}$ after the cut. Because $\mathcal{S}$ contains independent uniform samples from $\mathcal{P}^{s-1}$, those in $\mathcal{P}^s$ after the cut constitute independent uniform samples from $\mathcal{P}^s$ (rejection sampling). By choosing N sufficiently large, we will have sufficiently many points to repopulate $\mathcal{Q}$.

Finally, we also need an initial set $\mathcal{Q}$ of uniform samples from $\mathcal{P}^0$ but we only have a single point $\mathbf{x}^- \in \mathcal{X}_f^-$. The ROUNDINGBODY algorithm described by Lovász and Vempala (2003) uses $\mathcal{O}^*\left(D^4\right)$ membership queries to make the convex body near-isotropic. We use this as a preprocessing step; that is, given $\mathcal{X}_f^-$ and $\mathbf{x}^- \in \mathcal{X}_f^-$ we make $\mathcal{P}^0 = \mathcal{X}_f^- \cap \mathcal{B}_1^{2R}(\mathbf{x}^-)$ and use the ROUNDINGBODY algorithm to produce $\mathcal{Q} = \{\mathbf{x}^j \in \mathcal{P}^0\}$ for Algorithm 3.6.

Algorithm 3.8. Convex $\mathcal{X}_f^-$ Set Search

SetSearch $(\mathcal{P}, \mathcal{Q} = \{\mathbf{x}^j \in \mathcal{P}\}, C^-, C^+, \epsilon)$
while $C^-/C^+ > 1 + \epsilon$ **do begin**
  $C \leftarrow \sqrt{C^- \cdot C^+}$
  $[\mathbf{x}^*, \mathcal{P}', \mathcal{Q}'] \leftarrow$ *IntersectSearch* $(\mathcal{P}, \mathcal{Q}, C)$
  if intersection found **then begin**
    Let $C^- \leftarrow A(\mathbf{x}^*)$
    $\mathcal{P} \leftarrow \mathcal{P}'$ and $\mathcal{Q} \leftarrow \mathcal{Q}'$
  else
    $C^+ \leftarrow C$
  end if
end while
Return: $\mathbf{x}^*$

3.2.3 Optimization over ℓ_1 Balls

Here we suggest improvements for ℓ_1 minimization using iterative IntersectSearch and present them as SetSearch in Algorithm 3.8.

First, since $\mathbf{x}^A$, $\mathbf{x}^-$ and $\mathcal{Q}$ are the same for every iteration of the optimization procedure, we only run the RoundingBody procedure once as a preprocessing step. The set of samples $\{\mathbf{x}^j \in \mathcal{P}^0\}$ it produces are sufficient to initialize IntersectSearch at each stage of the binary search. Second, the separating hyperplane $\mathbf{h}_f^{\mathbf{y}}$ for point $\mathbf{y}$ given by Eq. (8) is valid for all weighted ℓ_1-balls of cost $C < A(\mathbf{y})$. Thus, the final state from a successful call to IntersectSearch can be used as the starting state for the subsequent call to IntersectSearch.

4 CONCLUSIONS & FUTURE WORK

The analysis of our algorithms shows that $\mathcal{F}^{\text{convex}}$ is ϵ-*IMAC* searchable for weighted ℓ_1 costs. When the positive class is convex we give efficient techniques that outperform previous reverse-engineering approaches for linear classifiers. When the negative class is convex, we apply a randomized Ellipsoid method to achieve efficient ϵ-*IMAC* search. If the adversary is unaware of which set is convex, they can trivially run both searches to discover an ϵ-*IMAC* with a combined polynomial query complexity.

Exploring near-optimal evasion is important for understanding how an adversary may circumvent learners in security-sensitive settings. As described here, our algorithms may not always directly apply in practice since various real-world obstacles persist. Queries may be only partially observable or noisy and the feature set may be only partially known. Moreover, an adversary may not be able to query all $\mathbf{x} \in \mathcal{X}$. Queries must be objects (such as email) that are mapped into $\mathcal{X}$. A real-world adversary must invert the feature-mapping—a generally difficult task. These limitations necessitate further research on the impact of partial observability and approximate querying on ϵ-*IMAC* search, and to design more secure filters. Broader open problems include: is ϵ-*IMAC* search possible on other classes of learners such as SVMs (linear in a large possibly infinite feature space)? Is ϵ-*IMAC* search feasible against an online learner that adapts as it is queried? Can learners be made resilient to these threats and how does this impact learning performance?

References

Dimitris Bertsimas and Santosh Vempala. Solving convex programs by random walks. *J. ACM*, 51(4):540–556, 2004.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *Proc. KDD'04*, pages 99–108, 2004.

Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing network-wide traffic anomalies. In *Proc. SIGCOMM'04*, pages 219–230, 2004.

László Lovász and Santosh Vempala. Hit-and-run from a corner. In *Proc. STOC'04*, pages 310–314, 2004.

László Lovász and Santosh Vempala. Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm. In *Proc. FOCS'03*, 2003.

Daniel Lowd and Christopher Meek. Adversarial learning. In *Proc. KDD'05*, pages 641–647, 2005.

Luis Rademacher and Navin Goyal. Learning convex bodies is hard. In *Proc. COLT'09*, pages 303–308, 2009.

Greg Schohn and David Cohn. Less is more: Active learning with support vector machines. In *Proc. ICML'00*, 2000.

Robert L. Smith. The hit-and-run sampler: a globally reaching Markov chain sampler for generating arbitrary multivariate distributions. In *Proc. WSC'96*, pages 260–264, 1996.

Kymie M. C. Tan, Kevin S. Killourhy, and Roy A. Maxion. Undermining an anomaly-based intrusion detection system using common exploits. In *Proc. RAID'02*, pages 54–73, 2002.

David Wagner and Paolo Soto. Mimicry attacks on host-based intrusion detection systems. In *Proc. CCS'02*, pages 255–264, 2002.