Query Strategies for Evading Convex-Inducing Classifiers

Blaine Nelson   NELSONB@CS.BERKELEY.EDU
Computer Science Division
University of California
Berkeley, CA 94720-1776, USA

Benjamin I. P. Rubinstein   BENR@CS.BERKELEY.EDU
Microsoft Research
Mountain View, CA 94043, USA

Ling Huang   LING.HUANG@INTEL.COM
Intel Labs Berkeley
Berkeley, CA 94709, USA

Anthony D. Joseph   ADJ@CS.BERKELEY.EDU
Computer Science Division
University of California
Berkeley, CA 94720-1776, USA

Steven J. Lee   STEVENJLEE@BERKELEY.EDU
Satish Rao   SATISHR@CS.BERKELEY.EDU
J. D. Tygar   TYGAR@CS.BERKELEY.EDU
Computer Science Division
University of California
Berkeley, CA 94720-1776, USA

Abstract

Classifiers are often used to detect miscreant activities. We study how an adversary can systematically query a classifier to elicit information that allows the adversary to evade detection while incurring a near-minimal cost of modifying their intended malfeasance. We generalize the theory of Lowd and Meek (2005) to the family of convex-inducing classifiers that partition input space into two sets one of which is convex. We present query algorithms for this family that construct undetected instances of approximately minimal cost using only polynomially-many queries in the dimension of the space and in the level of approximation. Our results demonstrate that near-optimal evasion can be accomplished without reverse-engineering the classifier's decision boundary. We also consider general ℓ_p costs and show that near-optimal evasion on the family of convex-inducing classifiers is generally efficient for both positive and negative convexity for all levels of approximation if $p = 1$.

Keywords: Query Algorithms, Evasion, Reverse Engineering, Adversarial Learning

1. Introduction

A number of systems and security engineers have proposed the use of machine learning techniques 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: classes of miscreant activity that fail to be detected. While learning algorithms allow the detection algorithm to adapt over time, real-world constraints on the learner typically allow an adversary to programmatically find vulnerabilities. We consider how an adversary can systematically discover blind spots by querying a fixed or learning-based detector to find a low cost (for some cost function) instance that the detector does not filter. As a motivating example, consider a spammer who wishes to minimally modify a spam message so it is not classified as a spam (here cost is a measure of how much the spam must be modified). There are a variety of domain specific mechanisms an adversary can use to observe the classifier's response to a query; *e.g.*, the spam filter of a public email system can be observed by creating a dummy account on that system and sending the queries to that account. We assume the attacker has access to a membership oracle for the filter. By observing the responses of the spam detector, the spammer can search for a modification while using as few queries as possible.

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 their investigation by generalizing their results to the family of convex-inducing classifiers—classifiers that partition their instance space into two sets one of which is convex. The family of convex-inducing classifiers is a particularly important and natural class to examine, as it includes the family of linear classifiers studied by Lowd and Meek as well as anomaly detection classifiers using bounded PCA (Lakhina et al., 2004), anomaly detection algorithms that use hyper-sphere boundaries (Bishop, 2006), 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, to name a few. Furthermore, the family of convex-inducing classifiers also includes more complicated bodies such as the countable intersection of halfspaces, cones, or balls.

We also show that near-optimal evasion does not require reverse engineering the classifier's decision boundary, which is the approach taken by Lowd and Meek (2005) for evading linear classifiers. Our algorithms for evading convex-inducing classifiers do not require fully estimating the classifier's boundary (which is hard in the general convex case; see Rademacher and Goyal, 2009) or otherwise 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 better query complexity than the previously-published reverse-engineering technique.

This paper is organized as follows. We overview past work related to near-optimal evasion in the remainder of this section. In Section 2 we formalize the near-optimal evasion problem, and review Lowd and Meek's definitions and results. We present algorithms for evasion that are near-optimal under ℓ_1 cost in Section 3 and we consider minimizing general ℓ_p costs in Section 4. We conclude the paper by discussing future directions for near-optimal evasion of classifiers in Section 5.

1.1 Related Work

Lowd and Meek (2005) first explored near-optimal evasion, and developed a method that reverse-engineered linear classifiers. Our approach generalizes their result and improves upon it in three significant ways.

- We consider a more general family of classifiers: the family of convex-inducing classifiers that partition the space of instances into two sets one of which is convex. This family subsumes the family of linear classifiers considered by Lowd and Meek.

- Our approach does not fully estimate the classifier's decision boundary (which is generally hard; see Rademacher and Goyal 2009) or reverse-engineer the classifier's state; instead, we directly search for an instance that the classifier recognizes as negative that is close to the desired attack instance (an evading instance of near-minimal cost).

- Even though our algorithms find solutions for a more general family of classifiers, our algorithms still only use a limited number of queries: they require only a number of queries polynomial in the dimension of the instance space. Moreover, our K-STEP MULTILINESEARCH (Algorithm 4) solves the linear case with fewer queries than the previously-published reverse-engineering technique.

Dalvi et al. (2004) use a cost-sensitive game theoretic approach to preemptively patch a classifier's blind spots (Dalvi et al., 2004). 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 paper we assume the classifier is not adapting during evasion.

A number of authors have studied evading sequence-based intrusion detector systems (IDSs) (Tan et al., 2002; Wagner and Soto, 2002). In exploring *mimicry attacks* these authors demonstrated that real IDSs can 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 active learning and near-optimal evasion are similar in their exploration of 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 our assumptions. First, we assume that instances are represented in a feature space $\mathcal{X}$ which is D-dimensional Euclidean space[1] $\mathcal{X} = \mathbb{R}^D$. Each component of an instance $\mathbf{x} \in \mathcal{X}$ is a *feature* which we denote as x_d. We denote each coordinate vector of the form $(0, \ldots, 1, \ldots, 0)$ with a 1 only at the dth feature as $\boldsymbol{\delta}_d$. We assume that the feature space representation is known to the adversary and there are no restrictions on the adversary's queries; *i.e.*, any point in feature space $\mathcal{X}$ can be queried by the adversary. These assumptions may not be true in every real-world setting, but they

---

1. Lowd and Meek also consider integer and Boolean-valued instance spaces and derive results for several classes of Boolean-valued learners.

allow us to investigate strategies taken by a worst-case adversary. We revisit this assumption in Section 5.

We further assume the target classifier f belongs to a family of classifiers $\mathcal{F}$. Any classifier $f \in \mathcal{F}$ is a mapping from feature space $\mathcal{X}$ to its response space $\mathcal{Y}$; *i.e.*, $f : \mathcal{X} \rightarrow \mathcal{Y}$. 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 knows its family $\mathcal{F}$. We also restrict our attention to binary classifiers and use $\mathcal{Y} = \{'-', '+'\}$.

We assume $f \in \mathcal{F}$ is deterministic and so it partitions $\mathcal{X}$ into 2 sets—the positive class $\mathcal{X}_f^+ = \{\mathbf{x} \in \mathcal{X} \mid f(\mathbf{x}) = '+'\}$ and the 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 that 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* (see Section **??** for a more detailed discussion of this assumption).

2.1 Adversarial Cost

We assume the adversary has a notion of utility over the instance space which we quantify with a cost function $A : \mathcal{X} \rightarrow \mathbb{R}^{0+}$; *e.g.*, for a spammer this could be edit distance on email messages. The adversary wishes to optimize A over the negative class, $\mathcal{X}_f^-$; *e.g.*, the 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 some instance $\mathbf{x}^A \in \mathcal{X}_f^+$ that is most desirable to the adversary. We focus on the general class of weighted ℓ_p ($0 < p \leq \infty$) cost functions:

$$A_p^{(\mathbf{c})}(\mathbf{x}) = \left(\sum_{d=1}^{D} c_d \left| x_d - x_d^A \right|^p \right)^{1/p} \quad , \tag{1}$$

where $0 < c_d < \infty$ is the relative cost the adversary associates with the dth feature. We also consider the cases when some features have $c_d = 0$ (adversary doesn't care about the dth feature) or $c_d = \infty$ (adversary requires the dth feature to match x_d^A). Weighted ℓ_1 costs are particularly appropriate for many adversarial problems since costs are assessed based on the degree to which a feature is altered and the adversary typically is interested in some features more than others. Unless stated otherwise, we take "ℓ_1 cost" to mean a weighted ℓ_1 cost in the sequel. 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). As with Lowd and Meek, we focus primarily on ℓ_1 costs in Section 3 before exploring general ℓ_p costs in Section 4. We use $\mathcal{B}^C(A) = \{\mathbf{x} \in \mathcal{X} \mid A(\mathbf{x}) \leq C\}$ to denote the cost-ball (or sublevel set) with cost no more than C. For instance, $\mathcal{B}^C(A_1)$ is the set of instances that do not exceed an ℓ_1 cost of C from the target $\mathbf{x}^A$.

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

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

i.e., the greatest lower bound on the cost obtained by any negative instance. 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 a cost no more than a factor $(1 + \epsilon)$ of the MAC; *i.e.*, every

ϵ-*IMAC* is a member of the set[2]

$$\epsilon\text{-}IMAC\,(f, A) \triangleq \left\{ \mathbf{x} \in \mathcal{X}_f^- \;\middle|\; A\,(\mathbf{x}) \leq (1 + \epsilon) \cdot \mathrm{MAC}\,(f, A) \right\} \;. \tag{2}$$

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

2.2 Search Terminology

The notion of near-optimality introduced in Eq. (2) is that of *multiplicative optimality*; *i.e.*, an ϵ-*IMAC* must have a cost within a factor of $(1 + \epsilon)$ of the *MAC*. However, the results of this paper can also be immediately adopted for *additive optimality* in which we seek instances with cost no more than $\eta > 0$ *greater* than the *MAC*. To differentiate between these notions of optimality, we will use the notation ϵ-*IMAC*$^{(*)}$ to refer to the set in Eq. (2) and define an analagous set η-*IMAC*$^{(+)}$ for addative optimality as

$$\eta\text{-}IMAC^{(+)}\,(f, A) \triangleq \left\{ \mathbf{x} \in \mathcal{X}_f^- \;\middle|\; A\,(\mathbf{x}) \leq \eta + \mathrm{MAC}\,(f, A) \right\} \;. \tag{3}$$

We use the terms ϵ-*IMAC*$^{(*)}$ and η-*IMAC*$^{(+)}$ to refer both to the sets defined in Eq. (2) and (3) as well as the members of them—the usage will be clear from the context.

Either notion of optimality allows us to efficiently use bounds on the *MAC* to find an ϵ-*IMAC*$^{(*)}$ or an η-*IMAC*$^{(+)}$. Suppose there is a negative instance, $\mathbf{x}$, with cost C^- and all instances with cost no more than C^+ are positive; *i.e.*, C^- is an upper bound and C^+ is a lower bound on the *MAC*: $C^+ \leq MAC\,(f, A) \leq C^-$. Then the negative instance $\mathbf{x}$ is ϵ-multiplicatively optimal if $C_0^- / C_0^+ \leq (1 + \epsilon)$ whereas it is η-additively optimal if $C_0^- - C_0^+ \leq \eta$. In the sequel, we will consider algorithms that can achieve either additive or multiplicative optimality. These algorithms employ binary search strategies to iteratively reduce the gap between any C^- and C^+. Namely, if we can determine whether an intermediate cost establishes a new upper or lower bound on *MAC*, then our binary search strategies can iteratively reduce the tth 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.

Lemma 1 *If an algorithm can provide bounds $C^+ \leq MAC\,(f, A) \leq C^-$, then this algorithm has achieved (1) $(C^- - C^+)$-additive optimality and (2) $(\frac{C^-}{C^+} - 1)$-multiplicative optimality.*

In the tth iteration of an additive binary search, the *additive gap* between the tth bounds is given by $G_t^{(+)} = C_t^- - C_t^+$ with $G_0^{(+)}$ defined accordingly by the initial bounds C_0^- and C_0^+. The search uses a proposal step of $C_t = (C_t^- + C_t^+)/2$, a stopping criterion of $G_t^{(+)} \leq \eta$ and achieves η-additive optimality in

$$L_\eta^{(+)} = \left\lceil \log_2 \left[\frac{G_0^{(+)}}{\eta} \right] \right\rceil \tag{4}$$

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

---

2. We use 'ϵ-*IMAC*' to refer both to this set and its members. The meaning will be clear from the context.

Binary search can also be used for multiplicative optimality by searching in exponential space. By rewriting our upper and lower bounds as $C^- = 2^a$ and $C^+ = 2^b$, the multiplicative optimality condition becomes $a - b \leq \log_2(1 + \epsilon)$, an additive optimality condition. Thus, binary search on the exponent achieves ϵ-multiplicative optimality and does so with the fewest queries. The *multiplicative gap* of the tth iteration is $G_t^{(*)} = C_t^- / C_t^+$ with $G_0^{(*)}$ defined accordingly by the initial bounds C_0^- and C_0^+. The tth query is $C_t = \sqrt{C_t^- \cdot C_t^+}$, the stopping criterion is $G_t^{(*)} \leq 1 + \epsilon$ and achieves ϵ-multiplicative optimality in

$$L_\epsilon^{(*)} = \left\lceil \log_2 \left[\frac{\log_2 \left(G_0^{(*)} \right)}{\log_2(1 + \epsilon)} \right] \right\rceil \tag{5}$$

steps. Multiplicative optimality only makes sense when both C_0^- and C_0^+ are strictly positive.

Binary searches for additive and multiplicative optimality differ in their proposal step and their stopping criterion. For additive optimality, the proposal is the arithmetic mean $C_t = (C_t^- + C_t^+)/2$ and search stops when $G_t^{(+)} \leq \eta$, whereas for multiplicative optimality, the proposal is the geometric mean $C_t = \sqrt{C_t^- \cdot C_t^+}$ and search stops when $G_t^{(*)} \leq 1 + \epsilon$. For the remainder of this paper, we will address ϵ-multiplicative optimality for an ϵ-IMAC (except where explicitly noted) and define $L_\epsilon = L_\epsilon^{(*)}$ and $G_t = G_t^{(*)}$. Nonetheless, our algorithms are immediately adapted to additive optimality by simply changing the proposal step, stopping condition, and the definitions of $L_\epsilon^{(*)}$ and G_t.

2.3 Near-Optimal Evasion

Lowd and Meek (2005) introduce the concept of *adversarial classifier reverse engineering (ACRE) learnability* to quantify the difficulty of finding an ϵ-IMAC instance for a particular family of classifiers $\mathcal{F}$, and a family of adversarial costs $\mathcal{A}$. Using our notation, their definition of *ACRE* ϵ-learnable is

> A set of classifiers $\mathcal{F}$ is *ACRE* ϵ-learnable under a set of cost functions $\mathcal{A}$ if an algorithm exists such that for all $f \in \mathcal{F}$ and $A \in \mathcal{A}$, it can find a $\mathbf{x} \in \epsilon$-*IMAC* (f, A) using only polynomially many membership queries in D, the encoded size of f, and the encoded size of $\mathbf{x}^+$ and $\mathbf{x}^-$.

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 $L_\epsilon^{(*)}$ required by a univariate binary search to narrow the gap between initial bounds C_0^+ and C_0^- to less than $(1 + \epsilon)$.[3] 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 $\mathbf{x}^+ \in \mathcal{X}_f^+$): we restrict our setting to

---

3. Using the encoded sizes of f, $\mathbf{x}^+$, and $\mathbf{x}^-$ in defining ϵ-*IMAC* searchable is problematic. For our purposes, it is clear that the encoded size of both $\mathbf{x}^+$ and $\mathbf{x}^-$ is D so it is unnecessary to include additional terms for their size. Further we allow for families of non-parametric classifiers for which the notion of *encoding size* is ill-defined but is also unnecessary for the algorithms we present. In extending beyond linear and parametric family of classifiers, it is not straightforward to define the encoding size of our classifier f. One could use notions such as the *VC-dimension* of $\mathcal{F}$ or its *covering number* (Anthony and Bartlett,

the case of $\mathbf{x}^A \in \mathcal{X}_f^+$, yielding simpler search procedures.[4] Finally, our algorithms do not reverse engineer the decision boundary, so "ACRE" would be a misnomer here. Instead we refer to the overall problem as *Near-Optimal Evasion* and replace *ACRE* ϵ-learnable with the following definition of ϵ-*IMAC* searchable.

> 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_ϵ. We will refer to such an algorithm as *efficient*.

Unlike Lowd and Meek's approach, our algorithms construct queries to provably find an ϵ-*IMAC* without reverse engineering the classifier's decision boundary. Efficient query-based reverse engineering for $f \in \mathcal{F}$ is sufficient for minimizing A over the estimated negative space. However, generally reverse engineering (*active learning*) is an expensive approach for near-optimal evasion, requiring query complexity that is exponential in the feature space dimension for general convex classes (Rademacher and Goyal, 2009), while finding an ϵ-*IMAC* need not be—the requirements for finding an ϵ-*IMAC* differ significantly from the objectives of reverse engineering approaches such as active learning. Both approaches use queries to reduce the size of version space $\hat{\mathcal{F}} \subset \mathcal{F}$, the set of classifiers consistent with the adversary's membership queries. However reverse engineering approaches minimize 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; *i.e.*,

$$\bigcap_{f \in \hat{\mathcal{F}}} \epsilon\text{-}IMAC\,(f, A) \neq \emptyset \ .$$

This objective allows the classifier to be unspecified in much of $\mathcal{X}$. 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; the classifier itself will be underspecified in large regions of $\mathcal{X}$ so our techniques do not reverse engineer the classifier.

2.4 Multiplicative vs. Additive Optimality

Additive and multiplicative optimality are intrinsically related by the fact that the optimality condition for multiplicative optimality $C_t^- / C_t^+ \leq 1 + \epsilon$ can be rewritten as additive optimality condition $\log_2 C_t^- - \log_2 C_t^+ \leq \log_2(1 + \epsilon)$. From this equilence we can take $\eta = \log_2(1 + \epsilon)$ and use the additive optimality criterion on the logarithm of the cost. However, this equivalence also leads to two differnces between these notions of optimality.

1999) but it is unclear why size of the classifier is important in quantifying the complexity of ϵ-*IMAC* search. Moreover, as we demonstrate in this paper, there are non-parametric families of classifiers for which ϵ-*IMAC* search is polynomial in D alone.

4. However, as is apparent in the algorithms we demonstrate, using $\mathbf{x}^+ = \mathbf{x}^A$ makes the attacker less covert since it is significantly easier to infer the attacker's intentions based on their queries. (Covertness is not an explicit goal in ϵ-*IMAC* search but it would be a requirement of many real-world attackers.) However, since our goal is not to design real attacks but rather analyze the best possible attack so as to understand our classifier's vulnerabilities, covertness can be ignored.

First, multiplicative optimality only makes sense when C_0^+ is strictly positive (we will need this assumption for our algorithms) whereas additive optimality can still be achieved if $C_0^+ = 0$. In this special case, $\mathbf{x}^A$ is on the boundary of $\mathcal{X}_f^+$ and there is no $\epsilon\text{-}IMAC^{(*)}$ for any $\epsilon > 0$. Practically speaking though, this is a minor hinderance—as we demonstrate in Section 3.1.3, there is an algorithm that can efficiently establish any lower bound C_0^+ if such a lower bound exists.

Second, the additive optimality criterion is not *scale invariant* (*i.e.*, any instance $\mathbf{x}^\dagger$ that satisfies the optimality criterion for cost A also satisfies it for $A'(\mathbf{x}) = s \cdot A(\mathbf{x})$ for any $s > 0$) whereas multiplictative optimality is scale invariant. Additive optimility is, however, *shift invariant* (*i.e.*, any instance $\mathbf{x}^\dagger$ that satisfies the optimality criterion for cost A also satisfies it for $A'(\mathbf{x}) = s + A(\mathbf{x})$ for any $s \geq 0$) whereas multiplicative optimality is not. Scale invariance is typically more salient because if the cost function is also scale invariant (all proper norms are) then the optimality condition is invariant to a rescaling of the underlying feature space; *e.g.*, a change in units for all features. Thus, multiplicative optimality is a unitless notion of optimality whereas additive optimality is not. The following result is a consequence of additive optimality's lack of scale invariance.

Theorem 2 *If for some hypothesis space $\mathcal{F}$, cost function A, and any initial bounds $0 < C_0^+ < C_0^-$ on the $MAC(f, A)$ for some $f \in \mathcal{F}$, there exists some $\bar{\epsilon} > 0$ such that no efficient query-based algorithm can find an $\epsilon\text{-}IMAC^{(*)}$ for any $0 < \epsilon \leq \bar{\epsilon}$, then there is no efficient query-based algorithm that can find a $\eta\text{-}IMAC^{(+)}$ for any $0 < \eta \leq \bar{\epsilon} \cdot C_0^-$.*

Proof We will proceed by contraposition. If there is an efficient query-based algorithm that can find a $\mathbf{x} \in \eta\text{-}IMAC^{(+)}$ for some $0 < \eta \leq \bar{\epsilon} \cdot C_0^-$, then, by definition of $\eta\text{-}IMAC^{(+)}$, $A(\mathbf{x}) \leq \eta + MAC(f, A)$. Taking $\eta = \epsilon \cdot MAC(f, A)$ for some $\epsilon > 0$, we have equivalently achieved $A(\mathbf{x}) \leq (1 + \epsilon)MAC(f, A)$; *i.e.*, $\mathbf{x} \in \epsilon\text{-}IMAC^{(*)}$. Moreover, since $MAC(f, A) \leq C_0^-$, this efficient algorithm is able to find a $\epsilon\text{-}IMAC^{(*)}$ for some $\epsilon \leq \bar{\epsilon}$. ∎

Corollary 3 *If for some hypothesis space $\mathcal{F}$, cost function A, there exists some $\bar{\epsilon} > 0$ such that no efficient query-based algorithm can find an $\epsilon\text{-}IMAC^{(*)}$ for any $0 < \epsilon \leq \bar{\epsilon}$, then there is no efficient query-based algorithm that can find a $\eta\text{-}IMAC^{(+)}$ for any η.*

Proof This follows from Theorem 2 since C_0^- may be arbitrarily large and $\bar{\epsilon} > 0$. ∎

This corollary demonstrates that the lack of scale invariance in the additive optimality condition allows for the feature space to be arbitrarily rescaled until any fixed level of additive optimality can no longer be achieved; *i.e.*, the units of the cost determine whether a particular level of additive accuracy can be achieved whereas multiplicative costs are unitless.

3. Evasion of Convex Classes

We generalize $\epsilon\text{-}IMAC$ searchability to the family of *convex-inducing classifiers* $\mathcal{F}^{\text{convex}}$ that partition the feature space $\mathcal{X}$ into a positive and negative class, one of which is convex. The
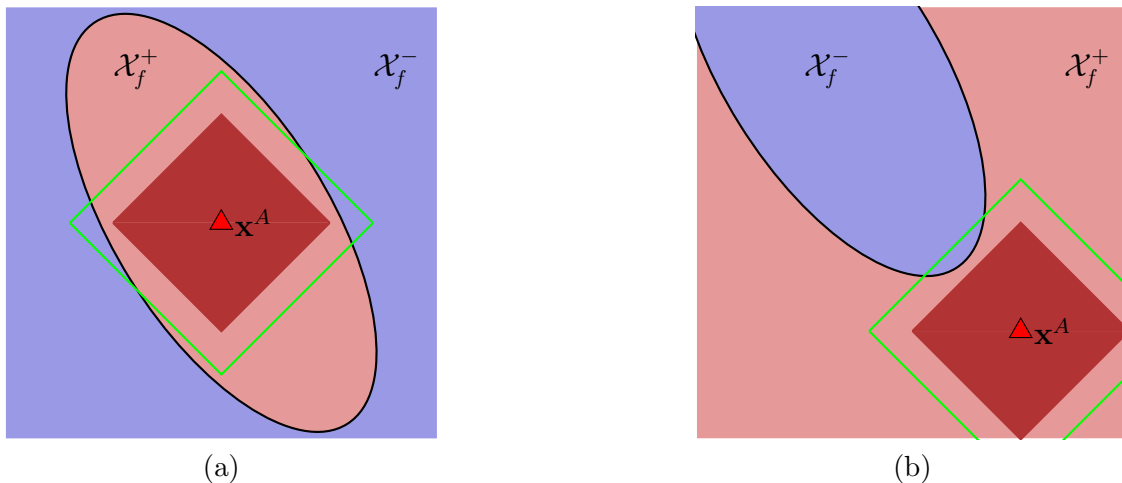
Figure 1: Geometry of convex sets and ℓ_1 balls. (a) If the positive set $\mathcal{X}_f^+$ is convex, finding an ℓ_1 ball contained within $\mathcal{X}_f^+$ establishes a lower bound on the cost, otherwise at least one of the ℓ_1 ball's corners witnesses an upper bound. (b) If the negative set $\mathcal{X}_f^-$ is convex, we can establish upper and lower bounds on the cost by determining whether or not an ℓ_1 ball intersects with $\mathcal{X}_f^-$, but this intersection need not include any corner of the ball.

convex-inducing classifiers include the linear classifiers studied by Lowd and Meek (2005), anomaly detectors using bounded PCA (Lakhina et al., 2004) and that use hyper-sphere boundaries (Bishop, 2006), 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 bodies such as any intersections of a countable number of halfspaces, cones, or balls.

Restricting $\mathcal{F}$ to be the family of convex-inducing classifiers simplifies ϵ-IMAC search. When the negative class $\mathcal{X}_f^-$ is convex, the problem reduces to minimizing a (convex) function A constrained to a convex set—if $\mathcal{X}_f^-$ were known to the adversary, this simply corresponds to solving a convex program. When the positive class $\mathcal{X}_f^+$ is convex, however, our task is to minimize the (convex) function A outside of a convex set; this is generally a hard problem (*cf.* Section 4.1.4 where we show that minimizing ℓ_2 cost can require exponential query complexity). Nonetheless for certain cost functions A, it is easy to determine whether a particular cost ball $\mathcal{B}^C(A)$ is completely contained within a convex set. This leads to efficient approximation algorithms.

We construct efficient algorithms for query-based optimization of the ℓ_1 cost of Eq. (1) for the convex-inducing classifiers. There appears to be an asymmetry depending on whether the positive or negative class is convex as illustrated in Figure 1. When the positive set is convex, determining whether an ℓ_1 ball $\mathcal{B}^C(A_1^{(\mathbf{c})}) \subset \mathcal{X}_f^+$ only requires querying the vertices of the ball as depicted in Figure 1(a). When the negative set is convex, determining whether or not $\mathcal{B}^C(A_1^{(\mathbf{c})}) \cap \mathcal{X}_f^- = \emptyset$ is non-trivial since the intersection need not occur at a vertex as

depicted in Figure 1(b). We present an efficient algorithm for the optimizing a 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 Eq. (5) to define L_ϵ as the number of phases required by our binary search to reduce the multiplicative gap to less than $1 + \epsilon$. We also use $C_0^- = A_1^{(\mathbf{c})}(\mathbf{x}^-)$ as an initial upper bound on the MAC and assume there is some $C_0^+ > 0$ that lower bounds the MAC (*i.e.*, $\mathbf{x}^A$ is in the interior of $\mathcal{X}_f^+$). This condition eliminates the case where $\mathbf{x}^A$ is on the boundary of $\mathcal{X}_f^+$ where $MAC(f, A) = 0$ and $\epsilon\text{-}IMAC(f, A) = \emptyset$—in this degenerate case, no algorithm can find an $\epsilon\text{-}IMAC$ since there are negative instances arbitrarily close to $\mathbf{x}^A$.

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

Solving the ϵ-*IMAC* Search problem when $\mathcal{X}_f^+$ is hard in the general case of convex cost $A(\cdot)$. We demonstrate algorithms for the ℓ_1 cost of Eq. (1) that solve the problem as a binary search. Namely, given initial costs C_0^+ and C_0^- that bound the MAC, our algorithm can efficiently determine whether $\mathcal{B}^C(A_1) \subset \mathcal{X}_f^+$ for any intermediate cost $C_t^+ < C_t < C_t^-$. If the ℓ_1 ball is contained in $\mathcal{X}_f^+$, then C_t becomes the new lower bound C_{t+1}^+. Otherwise C_t becomes the new upper bound C_{t+1}^-. Since our objective Eq. (2) is to obtain multiplicative optimality, our steps will be $C_t = \sqrt{C_t^+ \cdot C_t^-}$. We now explain how we exploit the properties of the ℓ_1 ball and convexity of $\mathcal{X}_f^+$ to efficiently determine whether $\mathcal{B}^C(A_1) \subset \mathcal{X}_f^+$ for any C. We also discuss practical aspects of our algorithm and extensions to other ℓ_p cost functions.

The existence of an efficient query algorithm relies on three facts: (1) $\mathbf{x}^A \in \mathcal{X}_f^+$; (2) every ℓ_1 cost C-ball centered at $\mathbf{x}^A$ intersects with $\mathcal{X}_f^-$ only if at least one of its vertices is in $\mathcal{X}_f^-$; and (3) C-balls of ℓ_1 costs only have $2 \cdot D$ vertices. The vertices of the ℓ_1 ball $\mathcal{B}^C(A_1)$ are axis-aligned instances differing from $\mathbf{x}^A$ in exactly one feature (*e.g.*, the dth feature) and can be expressed in the form

$$\mathbf{x}^A \pm \frac{C}{c_d}\boldsymbol{\delta}_d \ , \tag{6}$$

which belongs to the C-ball of our ℓ_1 cost (the coefficient $\frac{C}{c_d}$ normalizes for the weight c_d on the dth feature). We now formalize the second fact as follows.

Lemma 4 *For all $C > 0$, if there exists some $\mathbf{x} \in \mathcal{X}_f^-$ that achieves a cost of $C = A_1^{(\mathbf{c})}(\mathbf{x})$, then there is some feature d such that a vertex of the form of Eq. (6) 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 $A_1^{(\mathbf{c})}(\mathbf{x}) = C$ and $\mathbf{x}$ has $M \geq 2$ features that differ from $\mathbf{x}^A$ (if $\mathbf{x}$ only differs in 1 feature it would be of the form of Eq. 6). Let $\{d_1, \ldots, d_M\}$ be the differing features and let $b_{d_i} = \text{sign}\left(x_{d_i} - x_{d_i}^A\right)$ be the sign of the difference between $\mathbf{x}$ and $\mathbf{x}^A$ along the d_i-th feature. For each d_i, let $\mathbf{e}_{d_i} = \mathbf{x}^A + \frac{C}{c_{d_i}} \cdot b_{d_i} \cdot \boldsymbol{\delta}_{d_i}$ be a vertex of the form of Eq. (6) which has a cost C (from Eq. 1). The M vertices $\mathbf{e}_{d_i}$ form an M-dimensional equi-cost simplex of cost C on which $\mathbf{x}$ lies; *i.e.*, $\mathbf{x} = \sum_{i=1}^{M} \alpha_i \mathbf{e}_{d_i}$

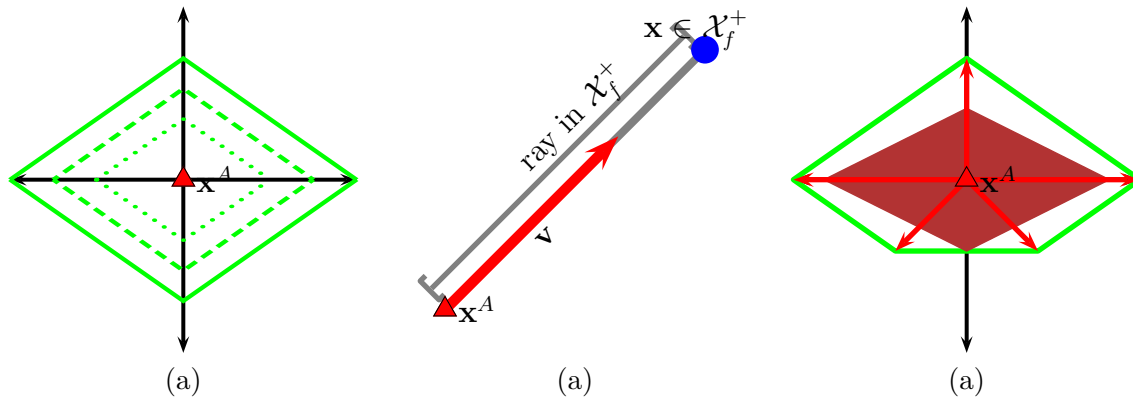(a)   (a)   (a)

Figure 2: The geometry of search. (a) Weighted ℓ_1 balls are centered around the target $\mathbf{x}^A$ and have 2^D vertices; 3.1 Search directions in multi-line search radiate from $\mathbf{x}^A$ to probe specific costs; 3.1 In general, we leverage convexity of the cost function when searching to evade. By probing all search directions at a specific cost, the convex hull of the positive queries bounds the ℓ_1 cost ball contained within it.

for some $0 \leq \alpha_i \leq 1$. If all $\mathbf{e_{d_i}} \in \mathcal{X}_f^+$, then the convexity of $\mathcal{X}_f^+$ implies that all points in their simplex are in $\mathcal{X}_f^+$ and so $\mathbf{x} \in \mathcal{X}_f^+$ which violates our premise. Thus, if any instance in $\mathcal{X}_f^-$ achieves cost C, there is always a vertex of the form Eq. (6) in $\mathcal{X}_f^-$ that also achieves cost C.   ∎

As a consequence, if all such vertices of any C ball $\mathcal{B}^C(A_1)$ are positive, then all $\mathbf{x}$ with $A_1^{(\mathbf{c})} \mathbf{x} \leq C$ are positive thus establishing C as a lower bound on the MAC. Conversely, if any of the vertices of $\mathcal{B}^C(A_1)$ are negative, then C is an upper bound on MAC. Thus, by simultaneously querying all $2 \cdot D$ equi-cost vertices of $\mathcal{B}^C(A_1)$, 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. (6).

A general form of this multiline search procedure is presented as Algorithm 1 and depicted in Figure 2. MultiLineSearch simultaneously searches along the directions in a set $\mathcal{W}$ of search directions that radiate from their origin at $\mathbf{x}^A$ and that are unit vectors for their cost; *i.e.*, $A(\mathbf{w}) = 1$ for any $\mathbf{w} \in \mathcal{W}$. (We transform a given set of non-normalized search vectors $\{\mathbf{v}\}$ into unit search vectors by simply applying a normalization constant of $A(\mathbf{v})^{-1}$ to each vector.) At each step of MultiLineSearch, at most $|\mathcal{W}|$ queries are issued in order to construct a bounding shell (*i.e.*, the convex hull of these queries will either form an upper or lower bound on the MAC) to determine whether $\mathcal{B}^C(A) \subset \mathcal{X}_f^+$. 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*[5]. Further, when an upper bound is established for a cost C (a negative vertex is

---

5. We could continue querying at any distance B^- where there is a known negative instance as it may allow us to prune other search directions quickly. However, once the classifier reveals a negative instance at

11

found), our algorithm prunes all directions that were positive at cost C. This pruning is sound; by the convexity assumption these pruned directions are positive for all costs less than the new upper bound C on the MAC. Finally, by performing a binary search on the cost, MULTILINESEARCH finds a ϵ-$IMAC$ with no more than $|\mathcal{W}| \cdot L_\epsilon$ queries but at least $|\mathcal{W}| + L_\epsilon$ queries. Thus, this algorithm is $\mathcal{O}\left(|\mathcal{W}| \cdot L_\epsilon\right)$ for ℓ_1 costs.

It is worth noting that, in its present form, MULTILINESEARCH has two implicit assumptions. First, we assume all search directions radiate from a common origin, $\mathbf{x}^A$, and $A\left(\mathbf{x}^A\right) = 0$. Without this assumption, the ray-constrained cost function $A\left(\mathbf{x}^A + s \cdot \mathbf{w}\right)$ is still convex in $s \geq 0$ but not necessarily monotonic as required for binary search. Second, we assume the cost function A is a *positive homogeneous function* along an ray from $\mathbf{x}^A$; *i.e.*, $A\left(\mathbf{x}^A + s \cdot \mathbf{w}\right) = |s| \cdot A\left(\mathbf{x}^A + \mathbf{w}\right)$. This assumption allows MULTILINESEARCH to scale its unit search vectors to achieve the same scaling of their cost. Although the algorithm could be adapted to eliminate these assumptions, the cost functions in Eq. (1) satisfy both assumptions since they are norms centered at $\mathbf{x}^A$.

Algorithm 2 uses MULTILINESEARCH for ℓ_1 costs by making $\mathcal{W}$ be the vertices of the unit-cost ℓ_1 ball centered at $\mathbf{x}^A$. In this case, the search issues at most $2 \cdot D$ queries to determine whether $\mathcal{B}^C\left(A_1\right) \subset \mathcal{X}_f^+$ and so Algorithm 2 is $\mathcal{O}\left(L_\epsilon \cdot D\right)$. However, MULTILINE-SEARCH does not rely on its directions being vertices of the ℓ_1 ball although those vertices are sufficient to span the ℓ_1 ball. Generally, MULTILINESEARCH is agnostic to the configuration of its search directions and can be adapted for any set of directions that can provide a bound on the cost using the convexity of $\mathcal{X}_f^+$. However, as we show in Section 4, the number of search directions required to bound an ℓ_p for $p > 1$ can be exponential in D.

3.1.1 K-STEP MULTI-LINE SEARCH

Here we present a variant of the multi-line search algorithm that better exploits pruning to reduce the query complexity of Algorithm 1—we call this variant K-STEP MULTILINESEARCH. The MULTILINESEARCH algorithm is $2 \cdot |\mathcal{W}|$ simultaneous binary searches (breadth-first). This strategy prunes directions most effectively when the convex body is assymetrically elongated relative to $\mathbf{x}^A$ but fails to prune for symmetrically rounded bodies. Instead we could search each direction sequentially (depth-first) and still obtain a worst case of $\mathcal{O}\left(L_\epsilon \cdot D\right)$ queries. In contrast, this strategy reduces queries used to shrink the cost gap on symmetrically rounded bodies but is unable to do so for assymetrically elongated bodies. We therefore propose an algorithm that mixes these strategies.

At each phase, the K-STEP MULTILINESEARCH (Algorithm 4) 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 towards reducing G_t without querying other directions (depth-first). It then iteratively queries all remaining directions at the candidate lower bound B^+ (breadth-first). Again we use lazy querying and stop as soon as a negative instance is found since B^+ is then no longer a viable lower bound. In this case, although the candidate bound is invalidated, we can still prune all directions that were positive at

---

distance B^-, the classifier would be foolish to subsequently reveal that another direction has a '+' at the same distance since it freely allows the adversary to prune a search direction. Hence, a malicious classifier will always respond with '$-$' for any cost where a negative instance has already been revealed. Thus, our algorithm uses lazy querying and only queries at costs below our upper bound C_t^- on the MAC.

Algorithm 1 MULTI-LINE SEARCH

$MLS\left(\mathcal{W}, \mathbf{x}^A, \mathbf{x}^-, C_0^+, C_0^-, \epsilon\right)$
$\mathbf{x}^* \leftarrow \mathbf{x}^-$
$t \leftarrow 0$
while $C_t^- / C_t^+ > 1 + \epsilon$ **do begin**
$\quad C_t \leftarrow \sqrt{C_t^+ \cdot C_t^-}$
$\quad$**for all** $\mathbf{e} \in \mathcal{W}$ **do begin**
$\quad\quad$**Query**: $f_{\mathbf{e}}^t \leftarrow f\left(\mathbf{x}^A + C_t \cdot \mathbf{e}\right)$
$\quad\quad$**if** $f_{\mathbf{e}}^t = '-'$ **then begin**
$\quad\quad\quad \mathbf{x}^* \leftarrow \mathbf{x}^A + C_t \cdot \mathbf{e}$
$\quad\quad\quad$Prune $\mathbf{i}$ from $\mathcal{W}$ if $f_{\mathbf{i}}^t = '+'$
$\quad\quad\quad$**break for-loop**
$\quad\quad$**end if**
$\quad$**end for**
$\quad C_{t+1}^+ \leftarrow C_t^+$ and $C_{t+1}^- \leftarrow C_t^-$
$\quad$**if** $\forall \mathbf{e} \in \mathcal{W}\ f_{\mathbf{e}}^t = '+'$ **then** $C_{t+1}^+ \leftarrow C_t$
$\quad$**else** $C_{t+1}^- \leftarrow C_t$
$\quad t \leftarrow t + 1$
end while
return: $\mathbf{x}^*$

Algorithm 2 CONVEX $\mathcal{X}_f^+$ SET SEARCH

$ConvexSearch\left(\mathcal{W}, \mathbf{x}^A, \mathbf{x}^-, \epsilon, C^+\right)$
$C^- \leftarrow A\left(\mathbf{x}^-\right)$
$\mathcal{W} \leftarrow \emptyset$
for all $i \in 1 \ldots D$ **do begin**
$\quad \mathbf{e}^i \leftarrow \frac{1}{c_i} \cdot \boldsymbol{\delta}_i$
$\quad \mathcal{W} \leftarrow \mathcal{W} \cup \left\{\pm \mathbf{e}^i\right\}$
end for
return: $MLS\left(\mathcal{W}, \mathbf{x}^A, \mathbf{x}^-, C^+, C^-, \epsilon\right)$

Algorithm 3 LINEAR $\mathcal{X}_f^+$ SET SEARCH

$LinearSearch\left(\mathcal{W}, \mathbf{x}^A, \mathbf{x}^-, \epsilon, C^+\right)$
$C^- \leftarrow A\left(\mathbf{x}^-\right)$
$\mathcal{W} \leftarrow \emptyset$
for all $i \in 1 \ldots D$ **do begin**
$\quad \mathbf{e}^i \leftarrow \frac{1}{c_i} \cdot \boldsymbol{\delta}_i$
$\quad b_i \leftarrow \text{sign}\left(x_i^- - x_i^A\right)$
$\quad$**if** $b_i = 0$ **then** $\mathcal{W} \leftarrow \mathcal{W} \cup \left\{b_i \mathbf{e}^i\right\}$
$\quad$**else** $\mathcal{W} \leftarrow \mathcal{W} \cup \left\{\pm \mathbf{e}^i\right\}$
end for
return: $MLS\left(\mathcal{W}, \mathbf{x}^A, \mathbf{x}^-, C^+, C^-, \epsilon\right)$

B^+. Thus, in every iteration, either the gap is decreased or at least one search direction is pruned. We show that for $K = \lceil\sqrt{L_\epsilon}\rceil$, the algorithm achieves a delicate balance between breadth-first and depth-first approaches to attain a better worst-case complexity than either.

Theorem 5 *Algorithm 4 will find an ϵ-IMAC with at most $\mathcal{O}\left(L_\epsilon + \sqrt{L_\epsilon}|\mathcal{W}|\right)$ queries when $K = \lceil\sqrt{L_\epsilon}\rceil$.*

The proof of this theorem appears in Appendix A. As a consequence of Theorem 5, finding a ϵ-IMAC with Algorithm 4 for a ℓ_1 cost requires $\mathcal{O}\left(L_\epsilon + \sqrt{L_\epsilon}D\right)$ queries. Further, both Algorithms 2 and 3 can incorporate K-STEP MULTILINESEARCH directly by replacing their function call to MLS to KLMS and using $K = \lceil\sqrt{L_\epsilon}\rceil$.

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 for any convex cost function (*e.g.*, Eq. 1 for $p \geq 1$). Below we present two theorems, one for both additive and multiplicative optimality. Notably, since an ϵ-IMAC uses multiplicative optimality, we incorporate a lower bound $C_0^+ > 0$ on the *MAC* into our statement.

Algorithm 4 K-Step Multi-line Search

$KMLS\left(\mathcal{W}, \mathbf{x}^A, \mathbf{x}^-, C_0^+, C_0^-, \epsilon, K\right)$

$\mathbf{x}^* \leftarrow \mathbf{x}^-$

$t \leftarrow 0$

while $C_t^-/C_t^+ > 1 + \epsilon$ **do begin**

    Choose a direction $\mathbf{e} \in \mathcal{W}$

    $B^+ \leftarrow C_t^+$

    $B^- \leftarrow C_t^-$

    for K steps **do begin**

       $B \leftarrow \sqrt{B^+ \cdot B^-}$

       Query: $f_\mathbf{e} \leftarrow f\left(\mathbf{x}^A + B \cdot \mathbf{e}\right)$

       if $f_\mathbf{e} =$ '+' **then** $B^+ \leftarrow B$

       else $B^- \leftarrow B$ **and** $\mathbf{x}^* \leftarrow \mathbf{x}^A + B \cdot \mathbf{e}$

    end for

    for all $\mathbf{i} \neq \mathbf{e} \in \mathcal{W}$ **do begin**

       Query: $f_\mathbf{i}^t \leftarrow f\left(\mathbf{x}^A + (B^+) \cdot \mathbf{i}\right)$

       if $f_\mathbf{i}^t =$ '−' **then begin**

          $\mathbf{x}^* \leftarrow \mathbf{x}^A + (B^+) \cdot \mathbf{i}$

          Prune $\mathbf{k}$ from $\mathcal{W}$ if $f_\mathbf{k}^t =$ '+'

          break for-loop

       end if

    end for

    $C_{t+1}^- \leftarrow B^-$

    if $\forall \mathbf{i} \in \mathcal{W} \ f_\mathbf{i}^t =$ '+' **then** $C_{t+1}^+ \leftarrow B^+$

    else $C_{t+1}^- \leftarrow B^+$

    $t \leftarrow t + 1$

end while

return: $\mathbf{x}^*$

Theorem 6 *For any $D > 0$, any positive convex function $A : \mathbb{R}^D \to \mathbb{R}^+$, any initial bounds $0 \leq C_0^+ < C_0^-$ on the MAC, and $0 < \eta < C_0^- - C_0^+$, all algorithms must submit at least $\max\{D, L_\eta^{(+)}\}$ membership queries in the worst case to be η-additive optimal on $\mathcal{F}^{\mathrm{convex},\,'+'}$.*

Theorem 7 *For any $D > 0$, any positive convex function $A : \mathbb{R}^D \to \mathbb{R}^+$, any initial bounds $0 < C_0^+ < C_0^-$ on the MAC, and $0 < \epsilon < \frac{C_0^-}{C_0^+} - 1$, all algorithms must submit at least $\max\{D, L_\epsilon^{(*)}\}$ membership queries in the worst case to be ϵ-multiplicatively optimal on $\mathcal{F}^{\mathrm{convex},\,'+'}$.*

The proof of both of these theorems is in Appendix B. In these theorems, we restrict η and ϵ to the intervals $\left(0, C_0^- - C_0^+\right)$ and $\left(0, \frac{C_0^-}{C_0^+} - 1\right)$ respectively. In fact, outside of these intervals the query strategies are trivial. For either $\eta = 0$ or $\epsilon = 0$ no approximation

algorithm will terminate and for $\eta \geq C_0^- - C_0^+$ or $\epsilon \geq \frac{C_0^-}{C_0^+} - 1$, $\mathbf{x}^-$ is an *IMAC*, so no queries are required.

Theorem 6 and 7 show that one needs that η-additive and ϵ-multiplicative optimality require $\Omega(L_\eta^{(+)} + D)$ and $\Omega(L_\epsilon^{(*)} + D)$ queries respectively. Thus, we see that our K-STEP MULTILINESEARCH algorithm (Algorithm 4) has close to the optimal query complexity for ℓ_1-costs with its $\mathcal{O}(L_\epsilon + \sqrt{L_\epsilon}D)$ queries. These results also hold for arbitrary ℓ_p ($p \geq 1$) costs but we show lower bounds in Section 4 for $p > 1$ that substantially exceed these results.

3.1.3 SPECIAL CASES

Here we present a number of special cases that require minor modifications to Algorithms 1 and 4 primarily as preprocessing steps.

Revisiting Linear Classifiers Lowd and Meek originally developed a method for reverse engineering linear classifiers for a ℓ_1 cost. First their method isolates a sequence of points from $\mathbf{x}^-$ to $\mathbf{x}^A$ that cross the classifier's boundary and then it estimates the hyperplane's parameters using D line searches. However, as a consequence of the ability to efficiently minimize our objective when $\mathcal{X}_f^+$ is convex, we immediately have an alternative method for linear classifiers (*i.e.*, half-spaces). In fact, for this special case, as many as half of the search directions can be eliminated using the initial orientation of the hyperplane separating $\mathbf{x}^A$ and $\mathbf{x}^-$. Intuitively, the minimizer in the negative halfspace can only occur along one of the axes of the orthants that contain $\mathbf{x}^-$. This algorithm is presented as Algorithm 3. Moreover, because linear classifiers are a special case of convex-inducing classifiers, our K-STEP MULTILINESEARCH algorithm improves on the reverse-engineering technique's $\mathcal{O}(L_\epsilon \cdot D)$ queries and applies to a broader family.

Extending MultiLineSearch algorithms to $c_d = \infty$ or $c_d = 0$ In Algorthms 2 and 3, we reweighted the dth axis-aligned directions by a factor $\frac{1}{c_d}$ to make unit cost vectors but implictly assuming $c_d \in (0, \infty)$. The case where $c_d = \infty$ (*e.g.*, immutable features) is dealt with simply removing those features from the set of search directions $\mathcal{W}$ used in the MULTILINESEARCH. In the case when $c_d = 0$ (*e.g.*, useless features), MULTILINESEARCH-like algorithms no longer ensure near-optimality because they implicitly assume that cost balls are bounded sets. If $c_d = 0$, $\mathcal{B}^0(A)$ is no longer bounded and a 0-cost could be achieved if $\mathcal{X}_f^-$ anywhere intersects the subspace spanned by the 0-cost features—this makes near-optimality unachievable unless a negative 0-cost instance can be found. In the worst case, such an instance could be arbitrarily far in any direction within the 0-cost subspace making search for such an instance intractable. Nonetheless, one possible search strategy is to assign all 0-cost features a non-zero weight that decays quickly toward 0 (*e.g.*, $c_d = 2^-t$ in the tth iteration) as we repeatedly rerun an MULTILINESEARCH on the altered objective for T iterations. We will either find a negative instance that only alters 0-cost features (and hence is a 0-*IMAC*), or we will terminate assuming no such instance exists. This algorithm does not ensure near-optimality but may find a suitable instance with only T runs of a MULTILINESEARCH.

Lack of an Initial Lower Bound Thus far, to find a ϵ-*IMAC* our algorithms have searched between initial bounds C_0^+ and C_0^-, but, in general, C_0^+ may not be known to a real-world adversary. We now present an algorithm we call SPIRALSEARCH that can

efficiently establish a lower bound on the MAC if one exists. This algorithm performs a halving search on the exponent along a single direction to find a positive example, then queries the remaining directions at that cost. Either the lower bound is verified or directions that were positive can be pruned for the remainder of the search.

Algorithm 5   SPIRAL SEARCH

$spiral\left(\mathcal{W}, \mathbf{x}^A, \mathbf{x}^-, C_0^-, \epsilon\right)$
$t \leftarrow 0$ and $\mathcal{V} \leftarrow \emptyset$
repeat
   Choose a direction $\mathbf{e} \in \mathcal{W}$
   Remove $\mathbf{e}$ from $\mathcal{W}$ and $\mathcal{V} \leftarrow \mathcal{V} \cup \{\mathbf{e}\}$
   Query: $f_{\mathbf{e}} \leftarrow f\left(\mathbf{x}^A + (C_0^-)2^{-2^t}\mathbf{e}\right)$
   if $f_{\mathbf{i}} =$ '$-$' **then begin**
     $\mathcal{W} \leftarrow \mathcal{W} \cup \{\mathbf{e}\}$ and $\mathcal{V} \leftarrow \emptyset$
     $t \leftarrow t + 1$
   end if
  until $\mathcal{W} = \emptyset$
  $C_0^+ \leftarrow C_0^- \cdot 2^{-2^t}$
  return: $(\mathcal{V}, C_0^+, C_0^-)$

At the tth iteration of SPIRALSEARCH a direction is selected and queried at the current lower bound of $(C_0^-)2^{-2^t}$. If the query is positive, that direction is added to the set V of directions consistent with the lower bound. Otherwise, all directions in V are discarded and the lower bound is lowered with an exponentially decreasing exponent. Thus, given that some lower bound $C_0^+ > 0$ does exist, one will be found in $\mathcal{O}\left(L_\epsilon + D\right)$ queries and this algorithm can be used as a precursor to any of the previous searches[6]. Further, the search directions pruned by SPRIALSEARCH are also invalid for the subsequent MULTILINESEARCH so the set $\mathcal{V}$ returned by SPRIALSEARCH will be used as the set $\mathcal{W}$ for the subsequent search.

Lack of a Negative Example   Our algorithms can also naturally be adapted to the case when the adversary has no negative example $\mathbf{x}^-$. This is accomplished by querying ℓ_1 balls of doubly exponentially increasing cost until a negative instance is found. During the tth iteration, we probe along every search direction at a cost $(C_0^+)2^{2^t}$; either all probes are positive (and we have a new lower bound) or at least one is negative and we can terminate the search. Once a negative example is located (having probed for T iterations), we must have $(C_0^+)2^{2^{T-1}} < MAC\left(f, A\right) \le (C_0^+)2^{2^T}$; thus, $T = \left\lceil \log_2 \log_2 \frac{MAC(f,A)}{C_0^+} \right\rceil$. We can subsequently perform MULTILINESEARCH with $C_0^+ = 2^{2^{T-1}}$ and $C_0^- = 2^{2^T}$; $i.e.$, $\log_2 G_0 = 2^{T-1}$. This precursor step requires at most $|\mathcal{W}| \cdot T$ queries to initialize the MULTILINESEARCH algorithm with a gap such that $L_\epsilon = \left\lceil (T-1) + \log_2 \frac{1}{\log_2(1+\epsilon)} \right\rceil$ according to Eq. (5).

If there is neither an initial upper bound or lower bound, we proceed by probing each search direction at cost 1 using an additional $|\{W\}|$ queries—we will subsequently have either an upper or lower bound and can proceed accordingly.

---

6. If no lower bound on the cost exists, no algorithm can find a ϵ-$IMAC$. As presented, this algorithm would not terminate, but in practice the search would be terminated after sufficiently many iterations.

Algorithm 6   INTERSECT SEARCH

$IntersectSearch\left(\mathcal{P}^0, \mathcal{Q} = \left\{\mathbf{x}^j \in \mathcal{P}^0\right\}, C\right)$
for all $s = 1 \ldots T$ **do begin**
  (1) Generate $2N$ samples $\left\{\mathbf{x}^j\right\}_{j=1}^{2N}$
    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 $\mathbf{x}^j$, $A\left(\mathbf{x}^j\right) \le C$ terminate the for-loop
  (3) Put samples into 2 sets of size N
    $\mathcal{R} \leftarrow \left\{\mathbf{x}^j\right\}_{j=1}^{N}$ and $\mathcal{S} \leftarrow \left\{\mathbf{x}^j\right\}_{j=2N+1}^{2N}$
  (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. (8)
  (6) $\mathcal{P}^s \leftarrow \mathcal{P}^{s-1} \cap \mathcal{H}_{\mathbf{z}^s}$
  (7) Keep samples in $\mathcal{P}^s$
    $\mathcal{Q} \leftarrow \left\{\mathbf{x} \in \mathcal{S} \wedge \mathbf{x} \in \mathcal{P}^s\right\}$
end for
Return: the found $[\mathbf{x}_j, \mathcal{P}^s, \mathcal{Q}]$; or No Intersect

---

Algorithm 7   HIT-AND-RUN

$HitRun\left(\mathcal{P}, \left\{\mathbf{y}^j\right\}, \mathbf{x}^0\right)$
for all $i = 1 \ldots K$ **do begin**
  (1) Choose a random direction:
    $\nu_j \sim \mathrm{N}\left(0, 1\right)$
    $\mathbf{v} \leftarrow \sum_j \nu_j \cdot \mathbf{y}^j$
  (2) Sample uniformly along $\mathbf{v}$ using rejection sampling:
  Choose Ω s.t. $\mathbf{x}^{i-1} + \Omega \cdot \mathbf{v} \notin \mathcal{P}$
  repeat
    $\omega \sim Unif\left(0, \Omega\right)$
    $\mathbf{x}^i \leftarrow \mathbf{x}^{i-1} + \omega \cdot \mathbf{v}$
    $\Omega \leftarrow \omega$
  until $\mathbf{x}^i \in \mathcal{P}$
end for
Return: $\mathbf{x}^K$

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 Method and Interior Point methods; see Boyd and Vandenberghe 2004). However, in our problem the convex set is only accessible via membership queries. We use a randomized polynomial algorithm of Bertsimas and Vempala (2004) to minimize the cost function A given an initial point $\mathbf{x}^- \in \mathcal{X}_f^-$. For any fixed cost C^t we use their algorithm to determine (with high probability) whether $\mathcal{X}_f^-$ intersects with $\mathcal{B}^{C^t}(A)$; *i.e.*, whether 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_ϵ repetitions using binary search. We now focus only on weighted ℓ_1 costs (Eq. 1) and return to more general cases in Section 4.2.

3.2.1 INTERSECTION OF CONVEX SETS

We now outline Bertsimas and Vempala's query-based procedure for determining whether two convex sets (e.g., $\mathcal{X}_f^-$ and $\mathcal{B}^{C^t}(A_1)$) intersect. Their INTERSECTSEARCH procedure (which we present as Algorithm 6) is a randomized Ellipsoid method for determining whether there is an intersection between two bounded convex sets: $\mathcal{P}$ 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 obtain sufficiently many samples such that cutting $\mathcal{P}$ through the centroid of these samples with a separating hyperplane from $\mathcal{B}$ will significantly reduce the volume of $\mathcal{P}$ with high probability. Their technique 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 intersection is non-empty.

Our problem reduces to finding the intersection between $\mathcal{X}_f^-$ and $\mathcal{B}^{C^t}(A_1)$. Though $\mathcal{X}_f^-$ may be unbounded, we are minimizing a cost with bounded equi-cost balls, so we can instead use the set $\mathcal{P}^0 = \mathcal{X}_f^- \cap \mathcal{B}^{2R}(A_1)$ (where $R = A(\mathbf{x}^-) > C^t$) is a (convex) bounded subset of $\mathcal{X}_f^-$ that envelops all of $\mathcal{B}^{C^t}(A_1)$ and thus the intersection $\mathcal{X}_f^- \cap \mathcal{B}^{C^t}(A_1)$ if it exists. We also assume that there is some $r > 0$ such that there is an r-ball contained in the convex set $\mathcal{X}_f^-$; *i.e.*, there exists $\mathbf{y} \in \mathcal{X}_f^-$ such that $\mathcal{B}^r(A_1; \mathbf{y}) \subset \mathcal{X}_f^-$. We now detail this INTERSECTSEARCH procedure (Algorithm 6).

The backbone of the algorithm is the capability to sample uniformly from an unknown but bounded convex body by means of the HIT-AND-RUN random walk technique introduced by Smith (1996) (Algorithm 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 $\Omega = \left\{ \omega > 0 \mid \mathbf{x}^j + \omega \mathbf{v} \in \mathcal{P}^{s-1} \right\}$ is a bounded interval indexing all feasible points along direction $\mathbf{v}$ through $\mathbf{x}^j$. Sampling ω uniformly from Ω (using rejection sampling) yields the next step of the random walk; $\mathbf{x}^j + \omega \mathbf{v}$. Under the appropriate conditions (see Section 3.2.2), the HIT-AND-RUN random walk generates a sample uniformly from the convex body after $\mathcal{O}^*(D^3)$ steps[7] (Lovász and Vempala, 2004).

Randomized Ellipsoid Algorithm: We use HIT-AND-RUN to obtain $2N$ samples $\{\mathbf{x}^j\}$ from $\mathcal{P}^{s-1} \subset \mathcal{X}_f^-$ for a single phase of the randomized ellipsoid algorithm. If any sample $\mathbf{x}^j$ satisfies $A_1(\mathbf{x}^j) \leq C^t$, then $\mathbf{x}^j$ is in the intersection of $\mathcal{X}_f^-$ and $\mathcal{B}^{C^t}(A_1)$ and the procedure is complete. Otherwise, we want to significantly reduce the size of $\mathcal{P}^{s-1}$ without excluding any of $\mathcal{B}^{C^t}(A_1)$ so that sampling concentrates toward the intersection (if it exists)—for this we need a separating hyperplane for $\mathcal{B}^{C^t}(A_1)$. For any point $\mathbf{y} \notin \mathcal{B}^{C^t}(A_1)$, the (sub)gradient of the ℓ_1 cost given by

$$h_d^{\mathbf{y}} = c_d \operatorname{sign}\left(y_d - x_d^A\right) \quad , \tag{7}$$

and is a separating hyperplane for $\mathbf{y}$ and $\mathcal{B}^{C^t}(A_1)$.

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 the half of our samples in $\mathcal{R}$: $\mathbf{z} = N^{-1} \sum_{\mathbf{x} \in \mathcal{R}} \mathbf{x}$ (the other half we 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

$$\mathcal{H}_{\mathbf{z}} = \left\{ \mathbf{x} \mid \mathbf{x}^\top \mathbf{h}^{\mathbf{z}} \leq \mathbf{z}^\top \mathbf{h}^{\mathbf{z}} \right\} \quad . \tag{8}$$

As shown by Bertsimas and Vempala, this cut achieves $vol(\mathcal{P}^s) \leq \frac{2}{3} vol(\mathcal{P}^{s-1})$ 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 $(R/r)^D$, the algorithm can terminate after $T = \mathcal{O}(D \log(R/r))$ unsuccessful iterations with a high probability that the intersection is empty.

Because every iteration in Algorithm 6 requires $N = \mathcal{O}^*(D)$ samples, each of which need $K = \mathcal{O}^*(D^3)$ random walk steps, and there are $\mathcal{O}^*(D)$ iterations, the total number of membership queries required by Algorithm 6 is $\mathcal{O}^*(D^5)$.

---

7. $\mathcal{O}^*(\cdot)$ denotes the standard complexity notation $\mathcal{O}(\cdot)$ without logarithmic terms.

3.2.2 SAMPLING FROM A QUERIABLE CONVEX BODY

In the randomized Ellipsoid algorithm, random samples are used for two purposes: estimating the convex body's centroid and maintaining the conditions required for the HIT-AND-RUN sampler to efficiently generate points uniformly from a sequence of shrinking convex bodies. 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 sufficiently round, or more formally *near-isotropic*; *i.e.*, for any unit vector $\mathbf{v}$, $\mathbb{E}_{\mathbf{x} \sim \mathcal{P}}\left[\left(\mathbf{v}^\top \left(\mathbf{x} - \mathbb{E}_{\mathbf{x} \sim \mathcal{P}}\left[\mathbf{x}\right]\right)\right)^2\right]$ is bounded between $1/2$ and $3/2$ of $vol\left(\mathcal{P}\right)$.

If the body is not near-isotropic, we must rescale $\mathcal{X}$ with an appropriate affine transformation $\mathbf{T}$ so the resulting body $\mathcal{P}'$ is near-isotropic. 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 the HIT-AND-RUN algorithm (Algorithm 7) we sample the direction $\mathbf{v}$ based on this set. Intuitively, because the samples in $\mathcal{Q}$ are distributed uniformly in $\mathcal{P}^s$, the directions we sample based on the points in $\mathcal{Q}$ implicitly reflect the covariance structure of $\mathcal{P}^s$. This is equivalent to sampling the direction $\mathbf{v}$ from a normal distribution with zero mean the covariance of $\mathcal{P}$.

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 initially 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 $\mathcal{P}^s$ (*i.e.*, rejection sampling). By choosing N sufficiently large, our cut will be sufficiently deep and we will have sufficiently many points to resample $\mathcal{P}^s$ after the cut.

Finally, we also need an initial set $\mathcal{Q}$ of uniform samples from $\mathcal{P}^0$ but, in our problem, we only have a single point $\mathbf{x}^- \in \mathcal{X}_f^-$. Fortunately, there is an iterative procedure for putting the initial convex set $\mathcal{P}^0$ into a near-isotropic position from which we obtain $\mathcal{Q}$. The ROUNDINGBODY algorithm described by Lovász and Vempala (2003) uses $\mathcal{O}^*\left(D^4\right)$ membership queries to transforms the convex body into a near-isotropic position. We use this as a preprocessing step for Algorithms 6 and 8; that is, given $\mathcal{X}_f^-$ and $\mathbf{x}^- \in \mathcal{X}_f^-$ we make $\mathcal{P}^0 = \mathcal{X}_f^- \cap \mathcal{B}^{2R}\left(A_1; \mathbf{x}^-\right)$ and then use the ROUNDINGBODY algorithm to produce an initial uniform sample $\mathcal{Q} = \left\{\mathbf{x}^j \in \mathcal{P}^0\right\}$. These sets are then the inputs to our search algorithms.

3.2.3 OPTIMIZATION OVER ℓ_1 BALLS

We now revisit the outermost optimization loop (for searching the minimum feasible cost) of the algorithm and suggest improvements. First, since $\mathbf{x}^A$, $\mathbf{x}^-$ and $\mathcal{Q}$ are the same for every iteration of the optimization procedure, we only need to run the ROUNDINGBODY procedure once as a preprocessing step rather than running it as a preprocessing step every time INTERSECTSEARCH is invoked. The set of samples $\left\{\mathbf{x}^j \in \mathcal{P}^0\right\}$ produced by ROUNDINGBODY

Algorithm 8   Convex $\mathcal{X}_f^-$ Set Search

$SetSearch \left(\mathcal{P}, \mathcal{Q} = \left\{ \mathbf{x}^j \in \mathcal{P} \right\}, C_0^-, C_0^+, \epsilon \right)$

$\mathbf{x}^* \leftarrow \mathbf{x}^-$ and $t \leftarrow 0$

while $C_t^- / C_t^+ > 1 + \epsilon$ **do begin**

   $C_t \leftarrow \sqrt{C_t^- \cdot C_t^+}$

   $[\mathbf{x}^*, \mathcal{P}', \mathcal{Q}'] \leftarrow IntersectSearch \left(\mathcal{P}, \mathcal{Q}, C \right)$

   if intersection found **then begin**

     $C_{t+1}^- \leftarrow A \left(\mathbf{x}^* \right)$ and $C_{t+1}^+ \leftarrow C_t^+$

     $\mathcal{P} \leftarrow \mathcal{P}'$ and $\mathcal{Q} \leftarrow \mathcal{Q}'$

   else

     $C_{t+1}^- \leftarrow C_t^-$ and $C_{t+1}^+ \leftarrow C_t$

   end if

   $t \leftarrow t + 1$

end while

Return: $\mathbf{x}^*$

---

are sufficient to initialize the IntersectSearch at each stage of the binary search over C^t. Second, the separating hyperplane $\mathbf{h}_f^{\mathbf{y}}$ given by Eq. (7) does not depend on the target cost C^t but only on $\mathbf{x}^A$, the common center of all the ℓ_1 balls. In fact, the separating hyperplane at point $\mathbf{y}$ is valid for all ℓ_1-balls of cost $C < A \left(\mathbf{y} \right)$. Further, if $C < C^t$, we have $\mathcal{B}^C \left(A_1 \right) \subset \mathcal{B}^{C^t} \left(A_1 \right)$. Thus, the final state from a successful call to IntersectSearch for the C^t-ball as the starting state for any subsequent call to IntersectSearch for all $C < C^t$. These improvements are reflected in our final procedure SetSearch in Algorithm 8—the total number of queries required is also $\mathcal{O}^* \left(D^5 \right)$.

4. General ℓ_p Costs

Here we further extend ϵ-$IMAC$ searchability over the family of convex-inducing classifiers to the full family of ℓ_p costs for any $0 < p < \infty$. As we demonstrate in this section, many ℓ_p costs are not generally ϵ-$IMAC$ searchable for all $\epsilon > 0$ over the family of convex-inducing classifiers(*i.e.*, we show that finding an ϵ-$IMAC$ for this family can require exponentially many queries in D and ϵ). In fact, only the weighted ℓ_1 costs are known to have (randomized) polynomial query strategies when either the positive or negative set is convex.

4.1 Convex Positive Set

Here we explore the ability of MultiLineSearch and K-step MultiLineSearch algorithms presented in Section 3.1 to find solutions to the near-optimal evasion problem for ℓ_p cost functions with $p \neq 1$. Particularly for $p > 1$ we will be exploring the consequences of using the MultiLineSearch algorithms using more search directions than just the $2 \cdot D$ axis-aligned directions. Figure 3 demonstrates how queries can be used to construct upper and lower bounds on general ℓ_p costs. The following Lemma also summarizes well known bounds on general ℓ_p costs based on an ℓ_1 cost.
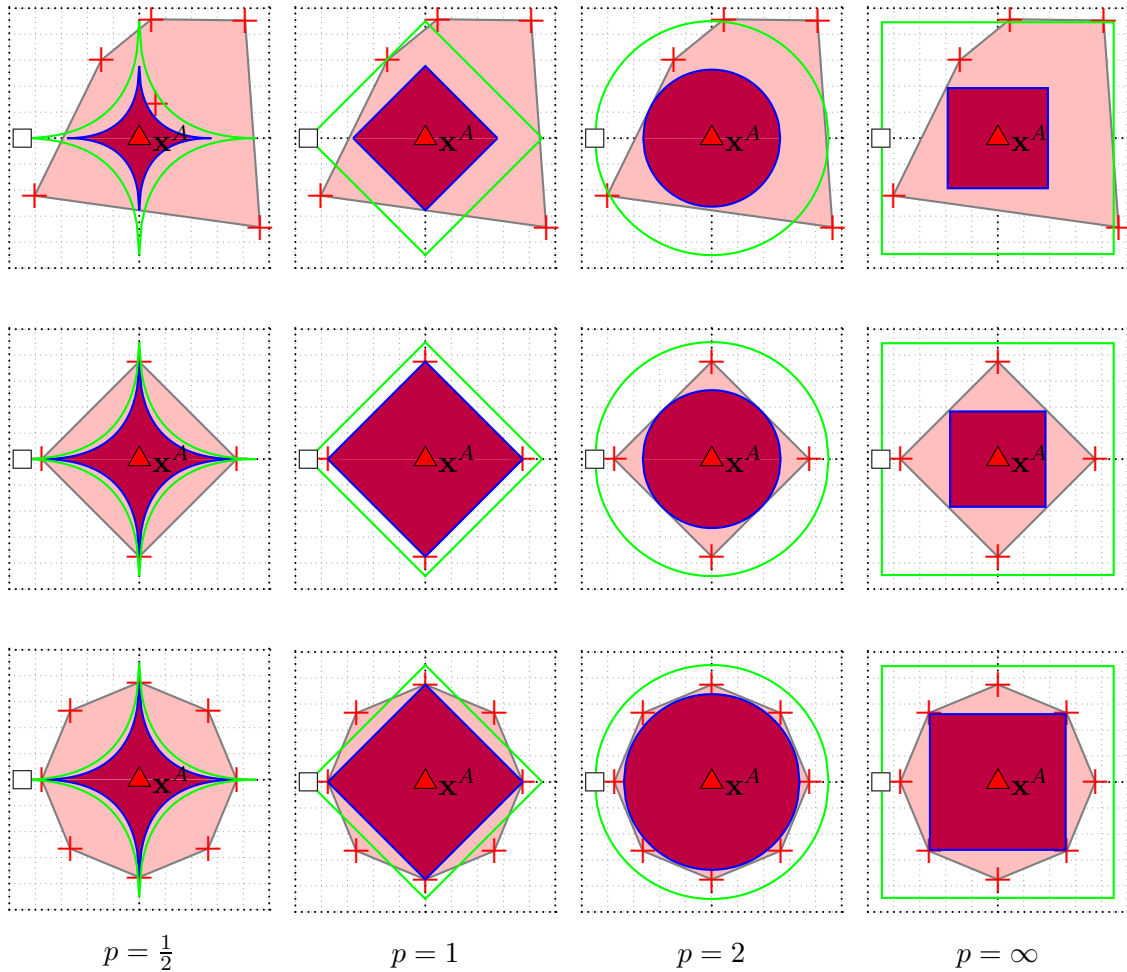
Figure 3: Convex hull for a set of queries and the resulting bounding balls for several ℓ_p costs. Each row represents a unique set of positive (red '+' points) and negative (green '−' points) queries and each column shows the implied upper bound (in green) and lower bound (in blue) for a different ℓ_p cost. In the first row, the body is defined by a random set of 7 queries, in the second, the queries are along the coordinate axes, and in the third, the queries are around a circle.

Lemma 8 *The largest ℓ_p ($p > 1$) ball enclosed within an ℓ_1 ball has a radius (cost) of $D^{\frac{1-p}{p}}$ and for $p = \infty$ the radius is D^{-1}.*

4.1.1 BOUNDING ℓ_p BALLS

In general, suppose we probe along some set of M unit directions and at some point we have at least one negative point supporting an upper bound of C_0^- and M positive points supporting at a cost of C_0^+. However, the lower bound provided by those M positive points is the cost of the largest ℓ_p cost ball that fits entirely within their convex hull; let's say this

cost is $C^\dagger < C_0^+$. In order to achieve ϵ-multiplicative optimality, we need

$$\frac{C_0^-}{C^\dagger} \leq 1 + \epsilon \ .$$

Expanding this, we need

$$\left(\frac{C_0^-}{C_0^+}\right)\left(\frac{C_0^+}{C^\dagger}\right) \leq 1 + \epsilon \ .$$

This allows us to break the problem into two parts. The first factor C_0^-/C_0^+ is only in terms of parameters controlled by the multiline search algorithm whereas the second factor $C_0^+/C^\dagger$ depends only on the shape of the ℓ_p ball as it captures how well the ball is approximated by the convex hull of the search directions. These two factors separate our task into choosing M and L_ϵ sufficiently so that their product is less than $1 + \epsilon$. First we choose factors $\alpha \geq 0$ and $\beta \geq 0$ so that $(1 + \alpha)(1 + \beta) \leq 1 + \epsilon$. Then we chose M so that

$$\frac{C_0^+}{C^\dagger} = 1 + \beta$$

and a parameter $\epsilon' = \alpha$ so that multiline search with M directions will achieve

$$\frac{C_0^-}{C_0^+} = 1 + \alpha \ .$$

In doing so, we create a generalized multiline search that is able to achieve ϵ-multiplicative optimality.

For example in the case of $p = 1$, we previously saw that choosing $M = 2 \cdot D$ allows us to exactly reconstruct the ℓ_1 ball so that $C_0^+/C^\dagger = 1$ (*i.e.*, $\beta = 0$). Thus we can just make $\alpha = \epsilon$ and we recover our original multiline search method exactly.

Objective: Below we present a number of results that deal with cases when $\beta > 0$. In this case, what we want to show is that a ratio of $\frac{C_0^+}{C^\dagger} = 1 + \beta$ can be achieved with a polynomial number of search directions when $\beta \leq \epsilon$; otherwise, $(1 + \alpha)(1 + \beta) > 1 + \epsilon$. Thus, we will be trying to find how many search directions are required for to achieve

$$\frac{C_0^+}{C^\dagger} \leq 1 + \epsilon \ ,$$

since this is the highest we can allow this ratio to be. Moreover, since this problem scales linearly with C_0^+ we will simply examine the values of $C^\dagger$ that can be achieved for the unit cost ball (*i.e.*, w.l.o.g. we make $C_0^+ = 1$ and rescale). Thus we will be looking at how many points are required to achieve:

$$C^\dagger \geq \frac{1}{1 + \epsilon} \ . \tag{9}$$

We will try to show that only polynomially many are required for at least some values of ϵ.

Lemma 9 *If there exists a configuration of M unit search directions with a convex hull that yields a bound $C^\dagger$ for the cost function A then multi-line search algorithms can use those search directions to achieve ϵ-multiplicative optimality with a query complexity that is polynomial in M and $L_\epsilon^{(*)}$ for any*

$$\epsilon > \frac{1}{C^\dagger} - 1 \ .$$

Corollary 10 *If there exists a configuration of M unit search directions with a convex hull that yields a bound $C^\dagger = 1$ for the cost function A then multi-line search algorithms then multi-line search algorithms can use those search directions to achieve ϵ-multiplicative optimality with a query complexity that is polynomial in M and $L_\epsilon^{(*)}$ for any $\epsilon > 0$.*

As this corollary reaffirms, for $p = 1$ using the $M = 2 \cdot D$ coordinate directions allows multi-line search algorithms to achieve ϵ-multiplicative optimality for any $\epsilon > 0$ with a query complexity that is polynomial in M and $L_\epsilon^{(*)}$.

4.1.2 MULTILINE SEARCH FOR $p < 1$

A simple result holds here. Namely, since the unit ℓ_1 ball bounds any unit ℓ_p balls with $p < 1$ we can achieve $C_0^+ / C^\dagger = 1$ using only the $2 \cdot D$ corners of the hyperoctahedron as search directions. Thus we can efficiently search for $p < 1$ for any value of $\epsilon > 0$. Whether or not the ℓ_p ($p < 1$) cost functions can be efficiently searched with fewer search directions is an open question.

4.1.3 MULTILINE SEARCH FOR $p > 1$

For this case, we can trivially use the ℓ_1 bound on ℓ_p balls as summarized by the following corollary:

Corollary 11 *For $1 < p < \infty$ and $\epsilon \in \left(D^{\frac{p-1}{p}} - 1, \infty \right)$ any multi-line search algorithm can achieve ϵ-multiplicative optimality on A_p using $M = 2 \cdot D$ search directions. Similarly for $p = \infty$ and $\epsilon \in (D - 1, \infty)$ any multi-line search algorithm can achieve ϵ-multiplicative optimality on A_∞.*

Proof From Lemma 8, the largest co-centered ℓ_p ball contained within the unit ℓ_1 ball has radius (cost) $D^{\frac{1-p}{p}}$ (or D for $p = \infty$). The bounds on ϵ then follows from Lemma 9. ∎

Unfortunately, this result only applies for a range of ϵ that grows with D, which is insufficient for ϵ-IMAC searchability. In fact, for some fixed values of ϵ, there is no query-based strategy that can bound ℓ_p costs using polynomially-many queries in Das the following result formalizes.

Theorem 12 *For $p > 1$, $D > 0$, any initial bounds $0 < C_0^+ < C_0^-$ on the MAC, and $0 < \epsilon < 2^{\frac{p-1}{p}} - 1$ (or $0 < \epsilon < 1$ for $p = \infty$), all algorithms must submit at least $\alpha_{p,\epsilon}^D$ membership queries (for some constant $\alpha_{p,\epsilon} > 1$) in the worst case to be ϵ-multiplicatively optimal on $\mathcal{F}^{\text{convex},'+'}$ for ℓ_p costs.*

The proof of this theorem is in Appendix C. A consequence of this theorem is that there is no query-based algorithm that can efficiently find an ϵ-IMAC of any ℓ_p cost ($p > 1$) for any $0 < \epsilon < 2^{\frac{p-1}{p}}$ (or $0 < \epsilon < 1$ for $p = \infty$) on the family $\mathcal{F}^{\text{convex},'+'}$. However, from Theorem 11 and Lemma 9, multiline-search type algorithms efficiently find the ϵ-IMAC of any ℓ_p cost ($p > 1$) for any $\epsilon \in \left(D^{\frac{p-1}{p}} - 1, \infty \right)$ (or $D - 1 < \epsilon < \infty$ for $p = \infty$). It is generally unclear if efficient algorithms exist for any values of ϵ between these intervals, but in the following section we derive a stronger bound for the case of $p = 2$.

4.1.4 Multiline Search for $p = 2$

Theorem 13 *For any $D > 1$, any initial bounds $0 < C_0^+ < C_0^-$ on the MAC, and $0 < \epsilon < \frac{C_0^-}{C_0^+} - 1$, all algorithms must submit at least $\alpha_\epsilon^{\frac{D-2}{2}}$ membership queries (where $\alpha_\epsilon = \frac{(1+\epsilon)^2}{(1+\epsilon)^2-1} > 1$) in the worst case to be ϵ-multiplicatively optimal on $\mathcal{F}^{\mathrm{convex},'+'}$ for ℓ_2 costs.*

The proof of this result is in Appendix D.

This result says that no algorithm can achieve ϵ-multiplicative optimality for ℓ_2 costs for any fixed $\epsilon > 0$ using only polynomially-many queries in D. However, for a fixed D, the bound provided by Theorem 13 suggests that reasonable approximations may be achievable $\alpha_\epsilon \to 1$.

It may appear that Theorem 13 contradicts Corollary 11. However, in Corollary 11 only applies for a range of ϵ that depends on D; *i.e.*, $\epsilon > \sqrt{D} - 1$. Interestingly, substituting this lower bound on ϵ into the bound given by Theorem 13, we get that the number of required queries for $\epsilon > \sqrt{D} - 1$ need only be

$$M \quad \geq \quad \left(\frac{(1+\epsilon)^2}{(1+\epsilon)^2 - 1} \right)^{\frac{D-2}{2}} \quad = \quad \left(\frac{D}{D-1} \right)^{\frac{D-2}{2}}$$

which is a monotonically increasing function in D that asymptotes at $\sqrt{e} \approx 1.64$. Thus, Theorem 13 and Corollary 11 are in agreement since for $\epsilon > \sqrt{D} - 1$, the former only requires that we need at least 2 queries.

4.2 Convex Negative Set

Algorithm 8 generalizes immediately to all weighted ℓ_p costs ($p \geq 1$) centered at $\mathbf{x}^A$ since these costs are convex. For these costs an equivalent separating hyperplane for $\mathbf{y}$ can be used in place of Eq. (7). These are given by the equivalent (sub)-gradients for ℓ_p cost-balls:

$$h_{p,d}^{\mathbf{y}} \quad = \quad c_d \operatorname{sign}\left(y_d - x_d^A\right) \cdot \left(\frac{|y_d - x_d^A|}{A_p^{(\mathbf{c})}(\mathbf{y})} \right)^{p-1}$$

$$h_{\infty,d}^{\mathbf{y}} \quad = \quad c_d \operatorname{sign}\left(y_d - x_d^A\right) \cdot \mathbb{I}\left\{ |y_d - x_d^A| = A_p^{(\mathbf{c})}(\mathbf{y}) \right\} \ .$$

By only changing the cost function A and the separating hyperplane $\mathbf{h}^{\mathbf{y}}$ used for the half-space cut in Algorithms 6 and 8, the randomize ellipsoid search can be applied for any weighted ℓ_p cost $A_p^{(\mathbf{c})}$.

For more general convex costs A, we still have that the set of all points $\mathbf{x}$ with $A(\mathbf{x}) \leq C$ (the *sublevel set* of cost C) is a subset of the sublevel set of cost D for all $D > C$; thus, the separating hyperplanes for the sublevel set at cost D will also be separating hyperplanes for the sublevel set at cost C. The SetSearch procedure therefore is applicable for any convex cost function A so long as we can compute the separating hyperplanes of any sublevel set of A for any point $\mathbf{y}$ not in sublevel set[8].

---

8. The sublevel set of any convex function is a convex set (see Boyd and Vandenberghe, 2004) so such a separating hyperplane always exists but may not be simple to compute.

For non-convex costs A such as weighted ℓ_p costs with $p < 1$, minimizing on a convex set $\mathcal{X}_f^-$ is generally a hard problem. However, there may be special cases when minimizing such a cost can be accomplished efficiently.

5. Conclusions and Future Work

In this paper we study ϵ-$IMAC$ searchability of convex-inducing classifiers. We present membership query algorithms that efficiently accomplish ϵ-$IMAC$ search on this family. When the positive class is convex we demonstrate very efficient techniques that outperform the 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. We also show our algorithms can be efficiently extended to cope with a number of special circumstances. Most importantly, we demonstrate that these algorithms can succeed without reverse engineering the classifier. Instead, these algorithms systematically eliminate inconsistent hypotheses and progressively concentrate their efforts in an ever-shrinking neighborhood of a MAC instance. By doing so, these algorithms only require polynomially-many queries in spite of the size of the family of all convex-inducing classifiers.

We also consider general ℓ_p costs and show that $\mathcal{F}^{convex}$ is only ϵ-$IMAC$ searchable for both positive and negative convexity for any $\epsilon > 0$ if $p = 1$. For $0 < p < 1$, the MULTILINESEARCH algorithms of Section 3.1 achieve identical results when the positive set is convex, but the non-convexity of these ℓ_p costs precludes the use of our randomized Ellipsoid method. The Ellipsoid method does provide an efficient solution for convex negative sets when $p > 1$ (since these costs are convex). However, for convex positive sets, our results show that for $p > 1$ there is no algorithm that can efficiently find an ϵ-$IMAC$ for all $\epsilon > 0$. Moreover, for $p = 2$ we prove that there is no efficient algorithm for finding an ϵ-$IMAC$ for any fixed value of ϵ.

By studying ϵ-$IMAC$ searchability, we provide a broader picture of how machine learning techniques are vulnerable to query-based evasion attacks. Exploring near-optimal evasion is important for understanding how an adversary may circumvent learners in security-sensitive settings. In such an environment, system developers are hesitant to trust procedures that may create vulnerabilities. The algorithms we demonstrate are invaluable tools not for an adversary to develop better attacks but rather for analysts to better understand the vulnerabilities of their filters. Our algorithms may not necessarily be easily used by an adversary since various real-world obstacles would first need to be overcome. Queries may only be partially observable or noisy and the feature set may only be partially known. Moreover, an adversary may not be able to query all $\mathbf{x} \in \mathcal{X}$; instead their queries must be legitimate 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?

Acknowledgements

References

Martin Anthony and Peter Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

Keith Ball. An elementary introduction to modern convex geometry. In *in Flavors of Geometry*, pages 1–58. University Press, 1997.

Dimitris Bertsimas and Santosh Vempala. Solving convex programs by random walks. *Journal of the 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 *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*, pages 99–108, 2004.

Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing network-wide traffic anomalies. In *Proceedings of the ACM SIGCOMM 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 219–230, 2004.

László Lovász and Santosh Vempala. Hit-and-run from a corner. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing (STOC '04)*, pages 310–314, 2004.

László Lovász and Santosh Vempala. Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS '03)*, pages 650–659, 2003.

Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD '05)*, pages 641–647, 2005.

Luis Rademacher and Navin Goyal. Learning convex bodies is hard. In *Proceedings of the 22nd Annual Conference on Learning Theory (COLT 2009)*, pages 303–308, 2009.

Greg Schohn and David Cohn. Less is more: Active learning with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pages 839–846, 2000.

Robert L. Smith. The hit-and-run sampler: A globally reaching Markov chain sampler for generating arbitrary multivariate distributions. In *Proceedings of the 28th Conference on Winter Simulation (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 *Proceedings of the 5th International Conference on Recent Advances in Intrusion Detection (RAID'02)*, pages 54–73, 2002.

David Wagner and Paolo Soto. Mimicry attacks on host-based intrusion detection systems. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 255–264, 2002.

Aaron D. Wyner. Capabilities of bounded discrepancy decoding. *The Bell System Technical Journal*, 44:1061–1122, Jul/Aug 1965.

Appendix A. Proof of Theorems for MultiLineSearch Algorithms

To analyze the worst case of K-STEP MULTILINESEARCH (Algorithm 4), we consider a *malicious classifier* that maximizes the number of queries. We refer to the agent that queries the classifier as the *adversary*.

Proof of Theorem 5 At each each iteration of Algorithm 4, the adversary choses some direction, $\mathbf{e}$ not yet eliminated from $\mathcal{W}$. Every direction in $\mathcal{W}$ is feasible (*i.e.*, could yield an ϵ-*IMAC*) and the malicious classifier, by definition, will make this choice as costly as possible. During the K steps of binary search along this direction, regardless of which direction $\mathbf{e}$ is selected or how the malicious classifier responds, the candidate multiplicative gap (see Section 2.2) along $\mathbf{e}$ will shrink by an exponent of 2^{-K}; *i.e.*,

$$\frac{B^-}{B^+} = \left(\frac{C^-}{C^+}\right)^{2^{-K}} \tag{10}$$

$$\log(G'_{t+1}) = \log(G_t) \cdot 2^{-K} \tag{11}$$

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

> Case 1 ($t \in \mathcal{C}_1$): Respond with '+' for all remaining directions. Here the bounds candidates B^+ and B^- are verified and thus the new gap is reduced by an exponent of 2^{-K}; however, no directions are eliminated from the search.

Case 2 ($t \in \mathcal{C}_2$): Choose at least 1 direction to respond with '$-$'. Here since only the value of C^- changes, the malicious classifier can chose to respond to the first K queries so that the gap decreases by a neglibile amount (by always responding with '$+$' during the first K queries along $\mathbf{e}$, the gap only decreases by an exponent of $(1 - 2^{-K})$). However, the malicious classifier must chose some number $E_t \geq 1$ of directions that will be eliminated.

We conservatively assume that the gap only decreases for case 1, which decouples the analysis of the queries for $\mathcal{C}_1$ and $\mathcal{C}_2$ and allows us to upper bound the total number of queries made by the algorithm. By this assumption, if $t \in \mathcal{C}_1$ we have $G_t = G_{t-1}^{2^{-K}}$ whereas if $t \in \mathcal{C}_2$, we have $G_t = G_{t-1}$. By analyzing the gap before and after the final iteration T, it can be shown that

$$|\mathcal{C}_1| = \left\lceil \frac{L_\epsilon}{K} \right\rceil \tag{12}$$

since, for the algorithm to terminate, there must be a total of at least L_ϵ binary search steps made during the case 1 iterations and each case 1 iteration takes exactly K steps.

At every case 1 iteration, the adversary make exactly $K + |\mathcal{W}_t| - 1$ queries where $\mathcal{W}_t$ is the set of feasible directions remaining at the tth iteration. While $\mathcal{W}_t$ is controlled by the malicious classifier, we can apply the bound $|\mathcal{W}_t| \leq |\mathcal{W}|$. Using this and the relation from Eq. (12), we can bound the number of queries Q_1 used in case 1 by

$$
\begin{aligned}
Q_1 &\leq \sum_{t \in C_1} (K + |\mathcal{W}| - 1) \\
&= \left\lceil \frac{L}{K} \right\rceil \cdot (K + |\mathcal{W}| - 1) \\
&\leq \left(\frac{L}{K} + 1 \right) \cdot K + \left\lceil \frac{L}{K} \right\rceil \cdot (|\mathcal{W}| - 1) \\
&= L + K + \left\lceil \frac{L}{K} \right\rceil \cdot (|\mathcal{W}| - 1) \ .
\end{aligned}
$$

For each case 2 iteration, we make exactly $K + E_t$ queries and this causes the elimination of $E_t \geq 1$ directions; hence, $|\mathcal{W}_{t+1}| = |\mathcal{W}_t| - E_t$. A malicious classifier will always make $E_t = 1$ whenever they use case 2 since that maximally limits how much the adversary gains. Nevertheless, since case 2 requires the elimination of at least 1 direction, we have $|\mathcal{C}_2| \leq |\mathcal{W}| - 1$ and moreover, regardless of the choice of E_t we have $\sum_{t \in \mathcal{C}_2} E_t \leq |\mathcal{W}| - 1$ since each direction can be eliminated no more than once. Thus,

$$
\begin{aligned}
Q_2 &= \sum_{i \in \mathcal{C}_2} (K + E_t) \\
&\leq |\mathcal{C}_2| \cdot K + |\mathcal{W}| - 1 \\
&\leq (|\mathcal{W}| - 1)(K + 1) \ .
\end{aligned}
$$

28

The total number of queries used by Algorithm 4

$$Q = Q_1 + Q_2 \quad \leq \quad L + K + \left\lceil \frac{L}{K} \right\rceil \cdot (|\mathcal{W}| - 1) + (|\mathcal{W}| - 1)(K + 1)$$

$$= \quad L + \left\lceil \frac{L}{K} \right\rceil \cdot |\mathcal{W}| + K \cdot |\mathcal{W}| + |\mathcal{W}| - \left\lceil \frac{L}{K} \right\rceil - 1$$

$$= \quad L + \left(\left\lceil \frac{L}{K} \right\rceil + K + 1 \right) |\mathcal{W}| .$$

Finally, choosing $K = \lceil \sqrt{L} \rceil$ minimizes this expression and using $L/\lceil \sqrt{L} \rceil \leq \sqrt{L}$ and substituting K into Q's bound, we have

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

■

Appendix B. Proof of Lower Bounds

Here we give proofs for the lower bound theorems in Section 3.1.2 first giving the proof for the more complictated multiplicative case followed by a similar proof sketch for the additive case. For these lower bounds, D is the dimension of the space, $A : \mathbb{R}^D \to \mathbb{R}^+$ is any positive convex function, $0 < C_0^+ < C_0^-$ are initial upper and lower bounds on the MAC, and $\hat{\mathcal{F}}^{\text{convex},'+'} \subset \mathcal{F}^{\text{convex},'+'}$ is the set of classifiers consistent with the constraints on the MAC; i.e., for $f \in \hat{\mathcal{F}}^{\text{convex},'+'}$ we have $\mathcal{X}_f^+$ is convex, $\mathcal{B}^{C_0^+}(A) \subset \mathcal{X}_f^+$, and $\mathcal{B}^{C_0^-}(A) \not\subset \mathcal{X}_f^+$.
Proof of Theorems 6 and 7 Suppose a query-based algorithm submits $N < D + 1$ membership queries $\mathbf{x}^1, \ldots, \mathbf{x}^N \in \mathbb{R}^D$ to the classifier. For the algorithm to be ϵ-optimal, these queries must constrain all consistent classifiers $\hat{\mathcal{F}}^{\text{convex},'+'}$ to have a common point among their ϵ-$IMAC$ sets. Suppose that the responses to the queries are consistent with the classifier f defined as:

$$f(\mathbf{x}) = \begin{cases} +1, & \text{if } A(\mathbf{x}) < C_0^- \\ -1, & \text{otherwise} \end{cases} . \tag{13}$$

For this classifier, $\mathcal{X}_f^+$ is convex since A is a convex function, $\mathcal{B}^{C_0^+}(A) \subset \mathcal{X}_f^+$ since $C_0^+ < C_0^-$, and $\mathcal{B}^{C_0^-}(A) \not\subset \mathcal{X}_f^+$ since $\mathcal{X}_f^+$ is the open C_0^--ball whereas $\mathcal{B}^{C_0^-}(A)$ is the closed C_0^--ball. Moreover, since $\mathcal{X}_f^+$ is the open C_0^--ball, $\nexists \mathbf{x} \in \mathcal{X}_f^-$ s.t. $A(\mathbf{x}) < C_0^-$ therefore $MAC(f, A) = C_0^-$, and any ϵ-optimal points $\mathbf{x}' \in \epsilon\text{-}IMAC^{(*)}(f, A)$ must satisfy $C_0^- \leq A(\mathbf{x}') \leq (1 + \epsilon)C_0^-$. Similarly, any η-optimal points $\mathbf{x}' \in \eta\text{-}IMAC^{(+)}(f, A)$ must satisfy $C_0^- \leq A(\mathbf{x}') \leq C_0^- + \eta$.

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 $\mathcal{G} = conv(\mathbf{x}^1, \ldots, \mathbf{x}^M)$; that is, the convex hull of the M positive queries. Now let $\mathcal{X}_g^+$ be the convex hull of $\mathcal{G}$ and the C_0^+-ball of A: $\mathcal{X}_g^+ = conv\left(\mathcal{G} \cup \mathcal{B}^{C_0^+}(A) \right)$. Since $\mathcal{G}$ contains all positive queries and

29

$C_0^+ < C_0^-$, the convex set $\mathcal{X}_g^+$ is consistent with the observed responses, $\mathcal{B}^{C_0^+}(A) \subset \mathcal{X}_g^+$ by definition, and $\mathcal{B}^{C_0^-}(A) \not\subset \mathcal{X}_g^+$ since the positive queries are all inside the open C_0^--sublevel set. Further, since $M \leq N < D+1$, $\mathcal{G}$ is contained in a proper linear subspace of $\mathbb{R}^D$ and hence $int(\mathcal{G}) = \emptyset$. Hence, there is always some point from $\mathcal{B}^{C_0^+}(A)$ that is on the boundary of $\mathcal{X}_g^+$; i.e., $\mathcal{B}^{C_0^+}(A) \not\subset int(\mathcal{G})$ because $int(\mathcal{G}) = \emptyset$ and $\mathcal{B}^{C_0^+}(A) \neq \emptyset$. Hence, there must be at least one point from $\mathcal{B}^{C_0^+}(A)$ on the boundary of the convex hull of $\mathcal{B}^{C_0^+}(A)$ and $\mathcal{G}$. Hence, $MAC(g, A) = \inf_{\mathbf{x} \in \mathcal{X}_g^-} [A(\mathbf{x})] = C_0^+$. Since the accuracy $\epsilon < \frac{C_0^-}{C_0^+} - 1$, any $\mathbf{x} \in \epsilon\text{-}IMAC^{(*)}(g, A)$ must have

$$A(\mathbf{x}) \leq (1+\epsilon)C_0^+ < \frac{C_0^-}{C_0^+}C_0^+ = C_0^- \;\;,$$

whereas any $\mathbf{y} \in \epsilon\text{-}IMAC^{(*)}(f, A)$ must have $A(\mathbf{y}) \geq C_0^-$. Thus, $\epsilon\text{-}IMAC^{(*)}(f, A) \cap \epsilon\text{-}IMAC^{(*)}(g, A) = \emptyset$ and we have constructed two convex-inducing classifiers f and g both consistent with the query responses with no common $\epsilon\text{-}IMAC^{(*)}$. Similarly, since $\eta < C_0^- - C_0^+$, any $\mathbf{x} \in \eta\text{-}IMAC^{(+)}(g, A)$ must have

$$A(\mathbf{x}) \leq \eta + C_0^+ < C_0^- - C_0^+ + C_0^+ = C_0^- \;\;,$$

whereas any $\mathbf{y} \in \eta\text{-}IMAC^{(+)}(f, A)$ must have $A(\mathbf{y}) \geq C_0^-$. Thus, $\eta\text{-}IMAC^{(+)}(f, A) \cap \eta\text{-}IMAC^{(+)}(g, A) = \emptyset$ and so the two convex-inducing classifiers f and g also have no common $\eta\text{-}IMAC^{(+)}$.

Suppose instead that a query-based algorithm submits $N < L_\epsilon^{(*)}$ membership queries (or $N < L_\eta^{(+)}$ for the additive case). Recall our definitions: C_0^- is the initial upper bound on the MAC, C_0^+ 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 ($G_t^{(+)} = C_t^- - C_t^+$ for the additive case). Here, the malicious classifier f responds with

$$f(\mathbf{x}^t) = \begin{cases} +1, & \text{if } A(\mathbf{x}^t) \leq \sqrt{C_{t-1}^- \cdot C_{t-1}^+} \\ -1, & \text{otherwise} \end{cases} \;\;. \tag{14}$$

When the classifier responds with '+', C_t^+ increases to no more than $\sqrt{C_{t-1}^- \cdot C_{t-1}^+}$ and so $G_t \geq \sqrt{G_{t-1}}$. Similarly when this classifier responds with '−', C_t^- decreases to no less than $\sqrt{C_{t-1}^- \cdot C_{t-1}^+}$ and so again $G_t \geq \sqrt{G_{t-1}}$. Thus, these responses ensure 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 $N \geq L_\epsilon^{(*)}$ from Eq. (5) (or in the additive case $N \geq L_\eta^{(+)}$ from Eq. 4). Again we have constructed two convex-inducing classifiers with consistent query responses but with no common $\epsilon\text{-}IMAC$. 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 $N < L_\epsilon^{(*)}$ (or, for the additive case, by a difference of more than η if $N < L_\eta^{(+)}$), so they have no common $\epsilon\text{-}IMAC$. ∎

Appendix C. Proof of Theorem 12

First we introduce the following lemma for the D-dimensional *hypercube graphs*—a collection of 2^D nodes of the form $(\pm 1, \pm 1, \ldots, \pm 1)$ where each node has an edge to every other node that is Hamming distance 1 from it.

Lemma 14 *For any $0 < \delta < 1/2$, to cover a D-dimensional hypercube graph so that every vertex has a Hamming distance of at most $\lfloor \delta D \rfloor$ to some vertex in the covering, the number of vertices in the covering must be*

$$Q(D, h) \geq 2^{D(1 - H(\delta))} \ ,$$

where $H(\delta) = -\delta \log_2 \delta - (1 - \delta) \log(1 - \delta)$ is the entropy *of δ.*

Proof There are 2^D vertices in the D-dimensional hypercube graph. Each vertex in the covering is within a Hamming distance of at most h for exactly $\sum_{k=0}^{h} \binom{D}{k}$ vertices. Thus, one needs at least $2^D / \left(\sum_{k=0}^{h} \binom{D}{k} \right)$ to cover the hypercube graph. Now we apply the bound

$$\sum_{k=0}^{\lfloor \delta D \rfloor} \binom{D}{k} \leq 2^{H(\delta)D}$$

to the denominator, which is valid for any $0 < \delta < 1/2$. ∎

Lemma 15 *The minimizer of the ℓ_p cost function A_p to any target $\mathbf{x}^A$ on the halfspace $\mathcal{H}_{\mathbf{w},\mathbf{b}} = \left\{ \mathbf{x} \mid \mathbf{x}^\top \mathbf{w} \geq \mathbf{b}^\top \mathbf{w} \right\}$ can be expressed in terms of the equilavent hyperplane $\mathbf{x}^\top \mathbf{w} \geq d$ parameterized by a normal vector $\mathbf{w}$ and displacement $d = \left(\mathbf{b} - \mathbf{x}^A \right)^\top \mathbf{w}$ as*

$$\begin{cases} d \cdot \|\mathbf{w}\|_{\frac{p}{p-1}}^{-1} \ , & \text{if } d > 0 \\ 0 \ , & \text{otherwise} \end{cases} \tag{15}$$

for all $1 < p < \infty$ and is

$$\begin{cases} d \cdot \|\mathbf{w}\|_1^{-1} \ , & \text{if } d > 0 \\ 0 \ , & \text{otherwise} \end{cases} \tag{16}$$

for $p = \infty$.

Proof For $1 < p < \infty$, minimizing A_p on the halfspace $\mathcal{H}_{\mathbf{w},\mathbf{b}}$ is equivalent to finding a minimizer for

$$\min_{\mathbf{x}} \frac{1}{p} \sum_{i=1}^{D} |x_i|^p \quad \text{s.t.} \quad \mathbf{x}^\top \mathbf{w} \leq d \ .$$

Clearly, if $d \leq 0$ then the vector $\mathbf{0}$ (corresponding to $\mathbf{x}^A$ in the transformed space) trivially satisfies the constraint and minimizes the cost function with cost 0 which yields the second case of Eq. (15). For the case $d > 0$, we construct the Lagrangian

$$\mathcal{L}(\mathbf{x}, \lambda) \triangleq \frac{1}{p} \sum_{i=1}^{D} |x_i|^p - \lambda \left(\mathbf{x}^\top \mathbf{w} - d \right) \ .$$

Differentiating this with respect to $\mathbf{x}$ and setting that partial derivative equal to zero yields

$$x_i^* = \text{sign}(w_i) \left(\lambda |w_i|\right)^{\frac{1}{p-1}} \ .$$

Plugging this back into the Lagrangian yields

$$\mathcal{L}\left(\mathbf{x}^*, \lambda\right) = \frac{1-p}{p} \lambda^{\frac{p}{p-1}} \sum_{i=1}^{D} |w_i|^{\frac{p}{p-1}} + \lambda d \ ,$$

which we now differentiate with respect to λ and set the derivative equal to zero to yield

$$\lambda^* = \left(\frac{d}{\sum_{i=1}^{D} |w_i|^{\frac{p}{p-1}}}\right)^{p-1} \ .$$

Plugging this solution into the formula for $\mathbf{x}^*$ yields the solution

$$x_i^* = \text{sign}(w_i) \left(\frac{d}{\sum_{i=1}^{D} |w_i|^{\frac{p}{p-1}}}\right) |w_i|^{\frac{1}{p-1}} \ .$$

The ℓ_p cost of this optimal solution is given by

$$A_p\left(\mathbf{x}^*\right) = d \cdot \|\mathbf{w}\|_{\frac{p}{p-1}}^{-1} \ ,$$

which is the first case of Eq. (15).

For $p = \infty$, once again if $d \leq 0$ then the vector $\mathbf{0}$ trivially satisfies the constraint and minimizes the cost function with cost 0 which yields the second case of Eq. (16). For the case $d > 0$, we use the geometry of hypercubes (the equi-cost balls of a ℓ_∞ cost function) to derive the second case of Eq. (16). For any optimal solution must occur at a point where the hyperplane given by $\mathbf{x}^\top \mathbf{w} = \mathbf{b}^\top \mathbf{w}$ is tangent to a hypercube about $\mathbf{x}^A$—this can either occur along a side (face) of the hypercube or at a corner. However, if the plane is tangent along a side (face) it is also tangent at a corner of the hypercube. Hence, there is always an optimal solution at some corner of optimal cost hypercube.

At a corner of the hypercube, we have the following property:

$$|x_1^*| = |x_2^*| = \ldots = |x_D^*| \ ;$$

that is, the magnitude of all coordiates of this optimal solution is the same value. Further, the sign of the optimal solution's i^{th} coordinate must agree with the sign of the hyperplane's i^{th} coordinate, $\mathbf{w}_i$. These constraints, along with the hyperplane constraint, lead to the following formula for an optimal solution:

$$x_i = d \cdot \text{sign}(w_i) \|\mathbf{w}\|_1^{-1} \ .$$

The ℓ_∞ cost of these solutions is simply

$$d \cdot \|\mathbf{w}\|_1^{-1} \ .$$

$\blacksquare$

For the proof of Theorem 12, we use the orthants (centered at $\mathbf{x}^A$)—an *orthant* is the D-dimensional generalization of a quadrant in 2-dimensions. There are 2^D orthants in a D-dimensional space. We represent each orthant by it's *canonical representation* which is a vector of D positive or negative ones; *i.e.*, the orthant represented by $\mathbf{a} = (\pm 1, \pm 1, \ldots, \pm 1)$ contains the point $\mathbf{x}^A + \mathbf{a}$ and is the set of all points $\mathbf{x}$ satisfying:

$$\mathbf{x}_i \in \begin{cases} [0, +\infty], & \text{if } \mathbf{a} = +1 \\ [-\infty, 0], & \text{if } \mathbf{a} = -1 \end{cases} .$$

Proof of Theorem 12 Suppose a query-based algorithm submits N membership queries $\mathbf{x}^1, \ldots, \mathbf{x}^N \in \mathbb{R}^D$ to the classifier. Again, for the algorithm to be ϵ-optimal, these queries must constrain all consistent classifiers $\hat{\mathcal{F}}^{\text{convex},'+'}$ to have a common point among their ϵ-$IMAC$ sets. The responses described above are consistent with the classifier f defined as

$$f(\mathbf{x}) = \begin{cases} +1, & \text{if } A_p(\mathbf{x}) < C_0^- \\ -1, & \text{otherwise} \end{cases} ; \tag{17}$$

For this classifier, $\mathcal{X}_f^+$ is convex since A_p is a convex function for $p \geq 1$, $\mathcal{B}^{C_0^+}(A_p) \subset \mathcal{X}_f^+$ since $C_0^+ < C_0^-$, and $\mathcal{B}^{C_0^-}(A_p) \not\subset \mathcal{X}_f^+$ since $\mathcal{X}_f^+$ is the open C_0^--ball whereas $\mathcal{B}^{C_0^-}(A_p)$ is the closed C_0^--ball. Moreover, since $\mathcal{X}_f^+$ is the open C_0^--ball, $\nexists\, \mathbf{x} \in \mathcal{X}_f^-$ s.t. $A_p(\mathbf{x}) < C_0^-$ therefore $MAC(f, A_p) = C_0^-$, and any ϵ-optimal points $\mathbf{x}' \in \epsilon\text{-}IMAC^{(*)}(f, A_p)$ must satisfy $C_0^- \leq A_p(\mathbf{x}') \leq (1 + \epsilon)C_0^-$.

Now 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. Here we consider a set which is a convex hull of the orthants of all M positive queries; that is,

$$\mathcal{G} = conv\left(orth\left(\mathbf{x}^1\right) \cap \mathcal{X}_f^+, orth\left(\mathbf{x}^2\right) \cap \mathcal{X}_f^+, \ldots, orth\left(\mathbf{x}^M\right) \cap \mathcal{X}_f^+ \right)$$

where $orth(\mathbf{x})$ is some orthant that $\mathbf{x}$ lies with in relative to $\mathbf{x}^A$ (a data point may lie within more than one orthant but we need only select any orthant that contains it in order to cover it). By intersecting each data point's orthant with the set $\mathcal{X}_f^+$ and taking the convex hull of these regions, $\mathcal{G}$ is convex, contains $\mathbf{x}^A$ and is a subset of $\mathcal{X}_f^+$ that is also consistent with all the query responses of f; *i.e.*, each of the M positive queries are in $\mathcal{X}_g^+$ and all the negative queries are in $\mathcal{X}_g^-$. Moreover, $\mathcal{G}$ is a superset of the convex hull of the M positive queries. Thus, by finding the largest enclosed ℓ_p ball within the $\mathcal{G}$, we upper bound $MAC(g, A_p)$.

We now represent each orthant as a vertex in a D-dimensional hypercube graph—the Hamming distance between any pair of orthants is the number of different coordinates in their canonical representations and two orthants are adjacent in the graph if and only if they have Hamming distance of 1. Using this notion of Hamming distance, we will seek a K-covering of the hypercube. We refer to the orthants used in $\mathcal{G}$ to cover the M positive queries as *covering orthants* and their corresponding vertices form a covering of the hypercube. Suppose the M covering orthants are sufficient for a K covering but not $K - 1$ covering; then there must be at least one vertex not in the covering that has at least a K

Hamming distance to every vertex in the covering. This vertex corresponds to an empty orthant that differs from all covered orthants in at least K coordinates of their canonical vertices. Without loss of generality, suppose this uncovered orthant has the canonical vertex of all postitive ones which we scale to $C_0^-(+1, +1, \ldots, +1)$. Consider the hyperplane with normal vector $\mathbf{w} = (+1, +1, \ldots, +1)$ and displacement

$$d = \begin{cases} C_0^- (D - K)^{\frac{p-1}{p}} & \text{if } 1 < p < \infty \\ C_0^- (D - K) & \text{if } p = \infty \end{cases}$$

that specifies the function $s(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} - d = \sum_{i=1}^D \mathbf{x}_i - d$. For this hyperplane, the vertex $C_0^-(+1, +1, \ldots, +1)$ yields

$$s\left(C_0^-(+1, +1, \ldots, +1)\right) = C_0^- D - d > 0 .$$

Also for any orthant $\mathbf{a}$ with Hamming distance at least K from this uncovered orthant, we have that for any $\mathbf{x} \in orth(\mathbf{a}) \cap \mathcal{X}_f^+$, by definition of the orthant and $\mathcal{X}_f^+$, the function s yields

$$s(\mathbf{x}) = \sum_{i=1}^D x_i - d$$
$$= \sum_{\{i \mid \mathbf{a_i}=+1\}} \underbrace{x_i}_{\geq 0} + \sum_{\{i \mid \mathbf{a_i}=-1\}} \underbrace{x_i}_{\leq 0} - d .$$

Since all the terms in the second summation are non-postive, the second sum is at most 0. Further, by maximizing the first summation, we upper bound $s(\mathbf{x})$. The summation $\sum_{\{i \mid \mathbf{a_i}=+1\}} x_i$ (with the constraint that $\|\mathbf{x}\|_p < C_0^-$) has at most $D - K$ terms and is maximized by $x_i = C_0^- (D - K)^{-1/p}$ (or $x_i = C_0^-$ for $p = \infty$) for which the first summation is upper bounded by $C_0^- (D - K)^{\frac{p-1}{p}}$ or $C_0^- (D - K)$ for $p = \infty$; i.e., it is upper bounded by d. Thus we see that

$$s(\mathbf{x}) \leq 0 .$$

Thus, this hyperplane seperates the scaled vertex $C_0^-(+1, +1, \ldots, +1)$ from each set $orth(\mathbf{a}) \cap \mathcal{X}_f^+$ where $\mathbf{a}$ is the canonical representation of any orthant with a Hamming distance of at least K. Thus, this hyperplane also seperates the scaled vertex from $\mathcal{G}$ by the properties of the convex hull. Since the displacement $C_0^- (D - K) > 0$, by applying Lemma 15, this separating hyperplane upper bounds the cost of the largest ℓ_p ball enclosed in $\mathcal{G}$ as

$$MAC(g, A_p) \leq C_0^- (D - K)^{\frac{p-1}{p}} \cdot \|\mathbf{w}\|_{\frac{p}{p-1}}^{-1} = C_0^- \left(\frac{D - K}{D}\right)^{\frac{p-1}{p}}$$

for $1 < p < \infty$ and

$$MAC(g, A_p) \leq C_0^- (D - K) \cdot \|\mathbf{1}\|_1^{-1} = C_0^- \frac{D - K}{D}$$

for $p = \infty$. Since we have an upper bound on the MAC of g and the MAC of f is C_0^-, in order to have a common ϵ-$IMAC$ between these classifiers, we must have

$$
(1 + \epsilon) \geq \begin{cases} \left(\frac{D}{D-K}\right)^{\frac{p-1}{p}}, & \text{if } 1 < p < \infty \\ \frac{D}{D-K}, & \text{if } p = \infty \end{cases} .
$$

Solving for the value of K required to achieve a desired accuracy of $1 + \epsilon$ we have

$$
K \leq \begin{cases} \frac{(1+\epsilon)^{\frac{p}{p-1}} - 1}{(1+\epsilon)^{\frac{p}{p-1}}} D, & \text{if } 1 < p < \infty \\ \frac{\epsilon}{1+\epsilon} D, & \text{if } p = \infty \end{cases} ,
$$

which bounds the size of the covering required to achieve the desired accuracy.

For the case $1 < p < \infty$, by Lemma 14, there must be

$$
M \geq \exp\left\{ \ln(2) \cdot D\left(1 - H\left(\frac{(1+\epsilon)^{\frac{p}{p-1}} - 1}{(1+\epsilon)^{\frac{p}{p-1}}}\right)\right)\right\}
$$

vertices of the hypercube in the covering to achieve any desired accuracy $0 < \epsilon < 2^{\frac{p-1}{p}} - 1$, for which

$$
\frac{(1+\epsilon)^{\frac{p}{p-1}} - 1}{(1+\epsilon)^{\frac{p}{p-1}}} < \frac{1}{2}
$$

as required by the Lemma. Moreover, since $0 < H(\delta) < 1$ for any $0 < \delta < 1$,

$$
\alpha_{p,\epsilon} = \exp\left\{ \ln(2)\left(1 - H\left(\frac{(1+\epsilon)^{\frac{p}{p-1}} - 1}{(1+\epsilon)^{\frac{p}{p-1}}}\right)\right)\right\} > 1
$$

and we have

$$
M > \alpha_{p,\epsilon}^D .
$$

Similarly for $p = \infty$, Lemma 14 can be applied yielding

$$
M \geq 2^{D\left(1 - H\left(\frac{\epsilon}{1+\epsilon}\right)\right)}
$$

to achieve any desired accuracy $0 < \epsilon < 1$ (for which $\epsilon/(1 + \epsilon) < 1/2$ as required by the Lemma). Again, by the properties of entropy the constant $\alpha_{\infty,\epsilon} = 2^{\left(1 - H\left(\frac{\epsilon}{1+\epsilon}\right)\right)} > 1$ for $0 < \epsilon < 1$ and we have

$$
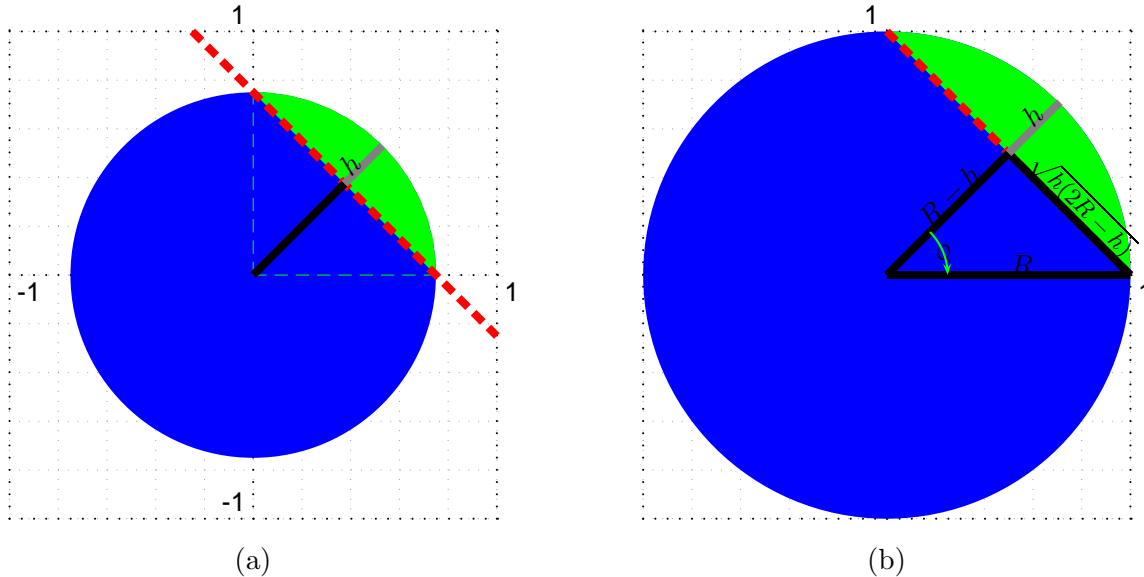M > \alpha_{\infty,\epsilon}^D .
$$

$\blacksquare$

Figure 4: This figure depictions the geometry of spherical caps. (a) A spherical cap of height h is shown that is created by a plane passing through the sphere. The green region represents the area of the cap. (b) We see the geometry of the spherical cap. Notice that the intersecting hyperplane forms a right triangle with the centroid of the hypershere. The length of the first side of that triangle is $R - h$, it's hypotenuse is length R, and its other side is length $\sqrt{h(2R - h)}$. The half angle ϕ of the right circular cone can also be used to parameterize the cap.

Appendix D. Proof of Theorem 13

For this proof, we build on previous results for covering hyperspheres. The proof is based on the following covering number result by Wyner and Shannon which bounds the minimum number of spherical caps required to cover a hypersphere. A D-dimensional *spherical cap* is the region formed by the intersection of a halfspace and a hypersphere facing away from the center of the hypersphere as depicted in Figure 4. This cap is parameterized by the hypersphere's radius R and the half-angle ϕ about a central radius (through the peak of the cap) as in the right-most diagram of Figure 4.

Based on these formula, we now derive a bound on the number of spherical caps of half-angle ϕ required to cover the sphere, mirroring the result due to Wyner (1965).

**Lemma 16 *(Result based on Wyner 1965)* ** *Covering the surface of D-dimensional hypersphere of radius R requires at least*

$$\left(\frac{1}{\sin \phi} \right)^{D-2}$$

spherical caps of half-angle ϕ.

Proof In *Capabilities of Bounded Discrepancy Decoding*, Wyner showed that the minimal number, M, of spherical caps of half-angle ϕ required to cover D-dimensional hypersphere

36

of radius R is given by

$$M \geq \frac{D\sqrt{\pi}\Gamma\left(\frac{D+1}{2}\right)}{(D-1)\Gamma\left(1+\frac{D}{2}\right)} \left[\int_0^\phi \sin^{D-2}(t)dt\right]^{-1} .$$

This result follows directly from computing the surface area of the hypersphere and the spherical caps.

We continue by lower bounding the above integral for a looser but more interpretable bound. Integrals of the form $\int_0^\phi \sin^D(t)dt$ also arise in computing the volume of a spherical cap. This volume (and thus the integral) can be bounded by enclosing the cap within a hypersphere; *cf.* Ball (1997). This yields the following bound:

$$\int_0^\phi \sin^D(t)dt \leq \frac{\sqrt{\pi}\Gamma\left(\frac{D+1}{2}\right)}{\Gamma\left(1+\frac{D}{2}\right)} \cdot \sin^D \phi .$$

Using this bound on the integral, our bound on the size of the covering is

$$M \geq \frac{D\sqrt{\pi}\Gamma\left(\frac{D+1}{2}\right)}{(D-1)\Gamma\left(1+\frac{D}{2}\right)} \left[\frac{\sqrt{\pi}\Gamma\left(\frac{D-1}{2}\right)}{\Gamma\left(\frac{D}{2}\right)} \cdot \sin^{D-2} \phi\right]^{-1} .$$

Now using properties of the gamma function, it can be shown that $\frac{\Gamma\left(\frac{D+1}{2}\right)\Gamma\left(\frac{D}{2}\right)}{\Gamma\left(1+\frac{D}{2}\right)\Gamma\left(\frac{D-1}{2}\right)} = \frac{D-1}{D}$ so that after canceling terms we arrive at our result:

$$M \geq \left(\frac{1}{\sin \phi}\right)^{D-2} .$$

■

Proof of Theorem 13 Suppose a query-based algorithm submits $N < D+1$ membership queries $\mathbf{x}^1, \ldots, \mathbf{x}^N \in \mathbb{R}^D$ to the classifier. For the algorithm to be ϵ-optimal, these queries must constrain all consistent classifiers $\hat{\mathcal{F}}^{\text{convex},'+'}$ to have a common point among their ϵ-*IMAC* sets. Suppose that all the responses are consistent with the classifier f defined as

$$f(\mathbf{x}) = \begin{cases} +1, & \text{if } A_2(\mathbf{x}) < C_0^- \\ -1, & \text{otherwise} \end{cases} ; \tag{18}$$

For this classifier, $\mathcal{X}_f^+$ is convex since A_2 is a convex function, $\mathcal{B}^{C_0^+}(A_2) \subset \mathcal{X}_f^+$ since $C_0^+ < C_0^-$, and $\mathcal{B}^{C_0^-}(A_2) \not\subset \mathcal{X}_f^+$ since $\mathcal{X}_f^+$ is the open C_0^--ball whereas $\mathcal{B}^{C_0^-}(A_2)$ is the closed C_0^--ball. Moreover, since $\mathcal{X}_f^+$ is the open C_0^--ball, $\nexists \mathbf{x} \in \mathcal{X}_f^-$ s.t. $A_2(\mathbf{x}) < C_0^-$ therefore $MAC(f, A_2) = C_0^-$, and any ϵ-optimal points $\mathbf{x}' \in \epsilon\text{-}IMAC^{(*)}(f, A_2)$ must satisfy $C_0^- \leq A_2(\mathbf{x}') \leq (1+\epsilon)C_0^-$.

Now 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 $\mathcal{G} = conv\left(\mathbf{x}^1, \ldots, \mathbf{x}^M\right)$;

37

that is, the convex hull of the M positive queries. We will assume $\mathbf{x}^A \in \mathcal{G}$ since if it is not, then we constuct the set $\mathcal{X}_g^+$ as in the proof for Theorems 7 and 6 above and achieve $MAC(f, A_2) = C_0^+$ thereby showing our desired result. Now consider the points $\mathbf{z}^i = C_0^- \frac{\mathbf{x}^i}{A_2(\mathbf{x}^i)}$; $i.e.,$, the projection of each of the positive queries onto the surface of the ℓ_2 ball $\mathcal{B}^{C_0^-}(A_2)$. Since each positive query lies along the line between $\mathbf{x}^A$ and its projection $\mathbf{z}^i$, by convexity and the fact that $\mathbf{x}^A \in \mathcal{G}$, we have $\mathcal{G} \subset conv\left(\mathbf{z}^1, \mathbf{z}^2, \dots, \mathbf{z}^M\right)$. We will call this enlarged hull $\hat{\mathcal{G}}$. These M projected points $\{\mathbf{z}^i\}$ must form a covering of the C_0^--hypersphere as the locii of caps of half-angle $\phi^* = \arccos\left((1+\epsilon)^{-1}\right)$. If not, then there exists some point on the surface of this hypersphere that is at least an angle ϕ^* from all $\mathbf{z}^i$ points and the resulting ϕ^*-cap centered at this uncovered point is not in $\hat{\mathcal{G}}$ (since a cap is defined as the intersection of the hypersphere and a halfspace). Moreover, by definition of the ϕ^*-cap, it achieves a minimal ℓ_2 cost of $C_0^- \cos \phi^*$. Thus, if we fail to achieve a ϕ^*-covering of the C_0^--hypersphere, the alternative classifier g has $MAC(g, A_2) < C_0^- \cos \phi^* = C_0^-/(1+\epsilon)$ and any $\mathbf{x} \in \epsilon\text{-}IMAC^{(*)}(g, A_2)$ must have

$$A_2(\mathbf{x}) \leq (1+\epsilon)MAC < (1+\epsilon)\frac{C_0^-}{1+\epsilon} = C_0^- \quad ,$$

whereas any $\mathbf{y} \in \epsilon\text{-}IMAC^{(*)}(f, A)$ must have $A(\mathbf{y}) \geq C_0^-$. Thus, we would have $\epsilon\text{-}IMAC^{(*)}(f, A) \cap \epsilon\text{-}IMAC^{(*)}(g, A) = \emptyset$ and thus fail to achieve ϵ-multiplicative optimality. Thus, we have shown that an ϕ^*-covering is necessary for ϵ-multiplicative optimality. However, from Lemma 16, to have a ϕ^*-covering we must have

$$M \geq \left(\frac{1}{\sin \phi^*}\right)^{D-2} \quad .$$

Using the trigonometric identity $\sin(\arccos(x)) = \sqrt{1-x^2}$ we can substitute for ϕ^* and find

$$
\begin{aligned}
M &\geq \left(\frac{1}{\sin\left(\arccos\left(\frac{1}{1+\epsilon}\right)\right)}\right)^{D-2} \\
&\geq \left(\frac{(1+\epsilon)^2}{(1+\epsilon)^2 - 1}\right)^{\frac{D-2}{2}} \quad .
\end{aligned}
$$

$\blacksquare$