Journal of Machine Learning Research 13 (2012) 1293-1332 Submitted 7/10; Revised 1/12; Published 5/12

Query Strategies for Evading Convex-Inducing Classifiers

Blaine Nelson BLAINE.NELSON@WSII.UNI-TUEBINGEN.DE
Wilhelm Schickard Institute for Computer Science
University of Tübingen
Sand 1
72076 Tübingen, Germany

Benjamin I. P. Rubinstein BEN.RUBINSTEIN@MICROSOFT.COM
Microsoft Research
1288 Pear Avenue
Mountain View, CA 94043, USA

Ling Huang LING.HUANG@INTEL.COM
ISTC on Secure Computing at UC Berkeley
711 Soda Hall
Berkeley, CA 94720-1776, USA

Anthony D. Joseph ADJ@CS.BERKELEY.EDU
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 94705-1545, USA

Editor: Tong Zhang

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 attacker 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 their feature 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 for this family 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 to detect miscreant activities in a variety of applications; for example, spam, intrusion, virus, and fraud detection. However, all known detection techniques have blind spots: classes of miscreant activity

that fail to be detected. While learning allows 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). As a second example, consider the design of an exploit that must avoid intrusion detection systems (here, cost may be a measure of the exploit's severity). There are a variety of domain-specific mechanisms an adversary can use to observe the classifier's response to a query, or in other words, to query a membership oracle of the filter; for example, 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. By observing the responses of the detector, the adversary can search for a modification while using as few queries as possible.

The idealized theoretical problem of near-optimal evasion was first posed by Lowd and Meek (2005). We continue their investigation by generalizing their results to convex-inducing classifiers—classifiers that partition feature space into two sets, one of which is convex. The family of convex-inducing classifiers is a particularly natural set 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 with a decision function of the form $\mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c \geq 0$ if $\mathbf{A}$ is semidefinite (see Boyd and Vandenberghe, 2004, Chapter 3), 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 in a continuous domain. Our algorithms for evading convex-inducing classifiers do not require fully estimating the classifier's boundary. Instead, we directly search for a minimal-cost evading instance. Since our algorithms require only polynomially-many queries, while reverse engineering the general convex case is hard (see Rademacher and Goyal, 2009), our algorithms witness a separation between the complexities of reverse engineering and evasion. In the special case of linear classifiers, our algorithms achieve better query complexity than the previously-published reverse-engineering technique. Finally, we also extend near-optimal evasion to general ℓ_p costs. For these costs, we show that our algorithms can also be used for near-optimal evasion, but are generally not efficient. However, in the cases when our algorithms are not efficient, we show that there is no efficient query-based algorithm.

A preliminary version of this paper was previously published as the report (Nelson et al., 2010b) extending our earlier work (Nelson et al., 2010a). 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 weighted ℓ_1 costs 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 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 in a continuous domain. Our approach generalizes that 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 labels as negative and is close to the desired attack instance (an evading instance of near-minimal cost). Lowd and Meek previously demonstrated a direct search technique for linear classifiers in Boolean spaces, but that technique is not applicable to the classifiers we consider.
- Even though our algorithms find solutions for a more general family of classifiers, our algorithms still use only polynomially-many queries in the dimension of the feature space and the accuracy of the desired approximation. Moreover, our K-STEP MULTILINESEARCH (Algorithm 3) solves the linear case with asymptotically fewer queries than the previously-published reverse-engineering technique.

Dalvi et al. (2004) use a game-theoretic approach to preemptively patch a cost-sensitive naive Bayes classifier's blind spots. They construct a modified classifier designed to detect optimally modified instances. Brückner and Scheffer (2009) and Kantarcioglu et al. (2009) have extended this setting to larger families of classifiers and developed techniques to solve for equilibrium strategies to their game. This prior research is complementary to query-based evasion; the near-optimal evasion problem studies how an adversary can use queries to find blind spots of a classifier that is unknown but queryable whereas their game-theoretic approaches assume the adversary knows the classifier and can optimize their evasion accordingly at each step of an iterated game.

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). As summarized by Settles (2009), the three primary approaches to active learning are membership query synthesis, stream-based selective sampling and pool-based sampling. Our work is most closely related to the membership query synthesis subfield introduced by Angluin (1988) in which the learner can request the label for any instance in feature space rather than for unlabeled instances drawn from a distribution. However, while active learning and near-optimal evasion are similar in their exploration of query strategies, the objectives for these two settings are quite different—evasion approaches search for low-cost negative instances within a factor $1+\varepsilon$ of an optimal cost whereas active learning algorithms seek to obtain hypotheses with low generalization error often in a PAC-setting (see Section 2.3 for a discussion on reverse-engineering approaches to evasion and active learning). It is interesting to note, nonetheless, that results in active learning settings (e.g., Dasgupta et al., 2009; Feldman, 2009) have also achieved polynomial query complexities

in specific cases. However, we focus solely on the evasion objective, and we leave the exploration of relationships between our results and those in active learning to future work.

Another class of related techniques that use query-based optimization are non-gradient global optimization methods often referred to as direct search. Simple examples of these techniques include bisection and golden-section search methods, for finding roots and extrema of univariate functions, and derivative approximation approaches such as the secant method and interpolation methods (e.g., Burden and Faires, 2000). Combinations of these approaches include Dekker's and Brent's algorithms (e.g., Brent, 1973), which exhibit superlinear convergence under certain conditions on the query function; that is, the number of queries is inversely quadratic in the desired error tolerance. However, while these approaches can be adapted to multiple dimensions, their query complexity grows exponentially with the dimension. Other approaches include the simplex method of Nelder and Mead (1965) and the DIRECT search algorithm introduced by Jones et al. (1993) (refer to Jones, 2001 and Kolda et al., 2003 for surveys of direct search methods), however, we are unaware of query bounds for these methods. While any direct search methods can be adapted for near-optimal evasion, these methods were designed to optimize an irregular function in a regular domain with few dimensions whereas the near-optimal evasion problem involves optimizing regular known functions (the cost function) over an unknown, possibly irregular, and high-dimensional domain (the points labeled as negative by the classifier). The methods we present specifically exploit the regular structure of ℓ_p costs and of the convex-inducing classifiers to achieve near-optimality with only polynomially-many queries.

2. Problem Setup

We begin by introducing our notation and assumptions. First, we assume that instances are represented in D-dimensional Euclidean *feature space*[1] $\mathcal{X} = \Re^D$ such as for some intrusion detection systems (e.g., Wang and Stolfo, 2004). Each component of an instance $\mathbf{x} \in \mathcal{X}$ is a *feature* which we denote as x_d. We use δ_d to denote each coordinate vector of the form $(0,\ldots,1,\ldots,0)$ with a 1 only at the d^{th} feature. We assume the feature space representation is known by the adversary and there are no restrictions on the adversary's queries; that is, any point $\mathbf{x}$ in feature space $\mathcal{X}$ can be queried by the adversary to learn the classifier's prediction at that point. These assumptions may not be true in every real-world setting (for instance, spam detectors are often defined with discrete features and designers often attempt to hide or randomize their feature set; for example, see Wang et al., 2006), but they allow us to investigate strategies taken by a worst-case adversary. We revisit these assumptions 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 $f : \mathcal{X} \to \mathcal{Y}$ from feature space $\mathcal{X}$ to its *response space* $\mathcal{Y}$. 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 with $\mathcal{Y} = \{'-','+'\}$.

We assume $f \in \mathcal{F}$ is deterministic and so it partitions $\mathcal{X}$ into two 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,

1. Lowd and Meek also consider integer and Boolean-valued feature spaces and derive results for several classes of learners in these discrete-valued spaces.

$\mathbf{x}^- \in \mathcal{X}_f^-$ 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 5 for a more detailed discussion).

2.1 Adversarial Cost

We assume the adversary has a notion of utility over the feature space, which we quantify with a cost function $A : \mathcal{X} \to \Re^{0+}$ (the non-negative reals); for example, for a spammer, this could be a string edit distance on email messages. The adversary wishes to optimize A over the negative class, $\mathcal{X}_f^-$; for example, 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 target 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 relative to the target $\mathbf{x}^A$ given by

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

where $0 < c_d < \infty$ is the relative cost the adversary associates with altering the d^{th} feature. When the relative costs are uniform, $c_d = 1$ for all d, we use the simplified notation A_p to refer to the cost function. Similarly, when referring to a generic weighted cost function with weights $\mathbf{c}$, we use the notation $A^{(\mathbf{c})}$. We also consider the cases when some features have $c_d = 0$ (adversary doesn't care about the d^{th} feature) or $c_d = \infty$ (adversary requires the d^{th} feature to match x_d^A). We use $\mathcal{B}^C(A;\mathbf{y}) = \{\mathbf{x} \in \mathcal{X} \mid A(\mathbf{x} - \mathbf{y}) \leq C\}$ to denote the cost ball (or sublevel set) centered at $\mathbf{y}$ with cost no more than the threshold C. For instance, $\mathcal{B}^C\left(A_1;\mathbf{x}^A\right)$ is the set of instances that do not exceed an ℓ_1 cost of C from the target $\mathbf{x}^A$. For convenience, we also use $\mathcal{B}^C(A) \triangleq \mathcal{B}^C\left(A;\mathbf{x}^A\right)$ to denote the C-cost-ball of A re-centered at the adversary's target, $\mathbf{x}^A$, since we focus on costs relative to this instance. Unless stated otherwise, we take "ℓ_1 cost" to mean a weighted ℓ_1 cost in the sequel.

Unfortunately, ℓ_p costs do not include many interesting costs such as string edit distances for spam and other real-world settings, such as the intrusion detection example from above where there may be no natural notion of distance between points. Nevertheless, the objective of this paper is not to provide practical evasion algorithms but rather to understand the theoretic capabilities of an adversary on the analytically tractable, albeit practically restrictive, family of ℓ_p costs. Weighted ℓ_1 costs are particularly appropriate for adversarial problems in which the adversary is interested in some features more than others and his cost is assessed based on the degree to which a feature is altered. Moreover, the ℓ_1-norm is a natural measure for a word-level edit distance for email spam, where larger weights model tokens that are more costly to remove (e.g., a payload URL). In Section 3, we focus on the weighted ℓ_1 costs studied by Lowd and Meek before exploring general ℓ_p costs in Section 4. In the latter case, our discussion focuses on uniform weights for ease of exposition, but the results also extend to the cost-sensitive case as presented for weighted ℓ_1 costs.

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

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

that is, 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 $(1+\varepsilon)$ of the *MAC*; that is, every ε-*IMAC* is a member of

the set

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

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 Equation (2) is that of *multiplicative optimality*; that is, an ε-*IMAC* must have a cost within a factor of $(1+\varepsilon)$ of the *MAC*. However, the results of this paper can also be immediately adapted 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 Equation (2) and define an analogous set η-$IMAC^{(+)}$ for additive optimality as

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

We use the terms ε-$IMAC^{(*)}$ and η-$IMAC^{(+)}$ to refer both to the sets defined in Equation (2) and (3) as well as the members of these sets—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 there is a $C^+ > 0$ such that all instances with cost no more than C^+ are positive; that is, $C^+ \leq MAC(f,A) \leq C^-$. Then the negative instance $\mathbf{x}^-$ is ε-multiplicatively optimal if $C^-/C^+ \leq (1+\varepsilon)$ whereas it is η-additively optimal if $C^- - C^+ \leq \eta$. In the sequel, we will consider algorithms that can achieve either additive or multiplicative optimality via binary search. Namely, if the adversary can determine whether an intermediate cost establishes a new upper or lower bound on the *MAC*, then binary search strategies can iteratively reduce the t^{th} gap between any bounds C_t^- and C_t^+ with the fewest steps. 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* $0 < C^+ \leq MAC(f,A) \leq C^-$*, then this algorithm has achieved* $(C^- - C^+)$*-additive optimality and* $(\frac{C^-}{C^+} - 1)$*-multiplicative optimality.*

In the t^{th} iteration of an additive binary search, the *additive gap* between the t^{th} bounds, C_t^- and C_t^+, is given by $G_t^{(+)} = C_t^- - C_t^+$ with $G_0^{(+)}$ defined accordingly by the initial bounds $C_0^- = C^-$ and $C_0^+ = C^+$. 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

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

steps. In fact, binary search has the best worst-case query complexity for achieving η-additive optimality.

Binary search can also be used for multiplicative optimality by searching in exponential space. Assuming that $C^- \geq C^+ > 0$, we can rewrite our upper and lower bounds as $C^- = 2^a$ and $C^+ = 2^b$, and thus the multiplicative optimality condition becomes $a - b \leq \log_2(1+\varepsilon)$; that is, an additive optimality condition. Thus, binary search on the exponent achieves ε-multiplicative optimality and

does so with the best worst-case query complexity. The *multiplicative gap* of the t^{th} iteration is $G_t^{(*)} = C_t^- / C_t^+$ with $G_0^{(*)}$ defined accordingly by the initial bounds C_0^- and C_0^+. The t^{th} query is $C_t = \sqrt{C_t^- \cdot C_t^+}$, the stopping criterion is $G_t^{(*)} \leq 1+\varepsilon$ and the search achieves ε-multiplicative optimality in

$$L_\varepsilon^{(*)} = \left\lceil \log_2 \left[\frac{\log_2 \left(G_0^{(*)} \right)}{\log_2 (1+\varepsilon)} \right] \right\rceil \tag{4}$$

steps. Although both additive and multiplicative criteria are related, there are two differences between these notions of optimality.

First, multiplicative optimality only makes sense when C_0^+ is strictly positive whereas additive optimality can still be achieved if $C_0^+ = 0$. Taking $C_0^+ > 0$ is equivalent to assuming that $\mathbf{x}^A$ is in the interior of $\mathcal{X}_f^+$ (a requirement for our algorithms to achieve multiplicative optimality). Otherwise, when $\mathbf{x}^A$ is on the boundary of $\mathcal{X}_f^+$, there is no ε-*IMAC*$^{(*)}$ for any $\varepsilon > 0$ unless there is some point $\mathbf{x}^* \in \mathcal{X}_f^-$ with 0 cost. Practically though, the need for a lower bound is a minor hindrance—as we demonstrate in Section 3.1.3, there is an algorithm that can efficiently establish a lower bound C_0^+ for any ℓ_p cost 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 multiplicative optimality is scale invariant. Additive optimality 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 more salient in near-optimal evasion 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; for example, a change in units for all features. Thus, multiplicative optimality is a unitless notion of optimality whereas additive optimality is not.

The following result states that additive optimality's lack of scale invariance allows for the feature space to be arbitrarily rescaled until any fixed level of additive optimality can no longer be achieved; that is, the units of the cost determine whether a particular level of additive accuracy can be achieved. By contrast multiplicative costs are unitless.

Proposition 2 *Consider any hypothesis space $\mathcal{F}$, target instance $\mathbf{x}^A$ and cost function A. If there exists some $\bar{\varepsilon} > 0$ such that no efficient query-based algorithm can find an ε-IMAC$^{(*)}$ for any $0 < \varepsilon \leq \bar{\varepsilon}$, then there is no efficient query-based algorithm that can find an η-IMAC$^{(+)}$ for any $0 < \eta \leq \bar{\varepsilon} \cdot MAC(f,A)$. In particular consider a sequence of classifiers f_n admitting unbounded MACs, and a sequence $\varepsilon_n > 0$ such that $1/\varepsilon_n = o(MAC(f_n,A))$. Then if no general algorithm can efficiently find an ε_n-IMAC$^{(*)}$ on each f_n then no general algorithm can efficiently find an η_n-IMAC$^{(+)}$ for $\eta_n \to \infty$.*

Proof Consider any classifier $f \in \mathcal{F}$ such that $MAC(f,A) > 0$. Suppose there exists some $\mathbf{x} \in \eta$-*IMAC*$^{(+)}$ for some $\eta > 0$. Let $\varepsilon = \eta / MAC(f,A)$ then by definition

$$A\left(\mathbf{x} - \mathbf{x}^A\right) \leq \eta + MAC(f,A) = (1+\varepsilon) MAC(f,A) \quad ,$$

implying that $\mathbf{x} \in \varepsilon$-*IMAC*$^{(*)}$. Then by the contrapositive, if no ε-*IMAC*$^{(*)}$ can be efficiently found for any $0 < \varepsilon \leq \bar{\varepsilon}$, then no η-*IMAC*$^{(+)}$ can be efficiently found for any $0 < \eta \leq \bar{\varepsilon} \cdot MAC(f,A)$. The last result is an immediate corollary. ■

The last statement is, in fact, applicable to many common settings. For instance, for any of the weighted ℓ_p costs (with $0 < p \leq \infty$ and $0 < c_d < \infty$ for all d) the family of linear classifiers and the family of hypersphere classifiers are both sufficiently diverse to yield such a sequence of classifiers that admit unbounded *MAC*s as required by the last statement. Thus, the family of convex-inducing classifiers can also yield such a sequence. Moreover, as we show in Section 4, there are indeed ℓ_p costs for which there exists $\bar{\varepsilon} > 0$ such that no efficient query-based algorithm can find an ε-*IMAC*$^{(*)}$ for any $0 < \varepsilon \leq \bar{\varepsilon}$. The consequence of this is that there is no general algorithm capable of achieving additive optimality for any fixed η with respect to the convex-inducing classifiers for these ℓ_p costs.

For the remainder of this paper, we will address ε-multiplicative optimality for an ε-*IMAC* (except where explicitly noted) and define $L_\varepsilon = L_\varepsilon^{(*)}$ and $G_t = G_t^{(*)}$. Nonetheless, our algorithms can be immediately adapted to additive optimality by simply changing the proposal step, stopping condition, and the definitions of $L_\varepsilon^{(*)}$ and G_t; the binary searches for additive and multiplicative optimality differ in their proposal steps and stopping criteria only. Finally, while we express query complexity in the sequel in terms of multiplicative L_ε, note that $L_\varepsilon^{(*)} = \Theta(\log \frac{1}{\varepsilon})$ and so in this way our query complexities can be rewritten to directly depend on ε.

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 an $\mathbf{x} \in \varepsilon$-*IMAC*$(f, A)$ using only polynomially-many membership queries in terms of the dimension D, the encoded size of f, and the encoded size of $\mathbf{x}^+$ and $\mathbf{x}^-$.

In this definition, Lowd and Meek use encoded size to refer to the length of the string of digits used to encode f, $\mathbf{x}^+$, and $\mathbf{x}^-$. In generalizing their result, we slightly alter their definition of query complexity. First, to quantify query complexity we use only the dimension, D, and the number of steps, L_ε, required by a univariate binary search to narrow the gap to within the desired accuracy. By including L_ε in our definition of query complexity, we do not require the encoded size of $\mathbf{x}^+$ and $\mathbf{x}^-$ since L_ε implicitly captures the size of the distance between these points as discussed above. 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 used a third $\mathbf{x}^+ \in \mathcal{X}_f^+$): we restrict our setting to the case of $\mathbf{x}^+ = \mathbf{x}^A$, yielding simpler search procedures.[2] 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 some $\mathbf{x} \in \varepsilon$-*IMAC*$(f, A)$

2. As is apparent in our algorithms, 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.

using polynomially-many membership queries in D and L_ε. We will refer to such an algorithm as *efficient*.

Our definition does not include the encoded size of the classifier, f, because our approach to near-optimal evasion does not reverse engineer the classifier's parameters. Unlike Lowd and Meek's approach for continuous spaces, our algorithms construct queries to provably find an ε-*IMAC* without reverse engineering the classifier's decision boundary; that is, estimating the decision surface of f or estimating the parameters that specify it. Efficient query-based reverse engineering for $f \in \mathcal{F}$ is sufficient for minimizing A over the estimated negative space. However, generally reverse engineering 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 as we demonstrate in this paper.[3] In fact, 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 $\hat{\mathcal{F}} \subset \mathcal{F}$; that is, the set of classifiers consistent with the adversary's membership queries. However reverse-engineering approaches minimize the expected number of disagreements between members of $\hat{\mathcal{F}}$. To find an ε-*IMAC*, by contrast, we need only provide a single instance, $\mathbf{x}^\dagger \in \varepsilon\text{-}IMAC(f,A)$, for all $f \in \hat{\mathcal{F}}$, while leaving the classifier largely unspecified; that is, we need to show that

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

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. Similarly, for linear classifiers in Boolean spaces, Lowd and Meek demonstrated an efficient algorithm for near-optimal evasion that does not reverse engineer the classifier—it too searches directly for an ε-*IMAC* and it shows that this family is 2-*IMAC* searchable for ℓ_1 costs with uniform feature weights, $\mathbf{c}$.

3. Evasion of Convex Classes for ℓ_1 Costs

We generalize ε-*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 convex-inducing classifiers include the linear classifiers studied by Lowd and Meek (2005), anomaly detectors using bounded PCA (Lakhina et al., 2004) and using 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 with a decision function of the form $\mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c \geq 0$ if $\mathbf{A}$ is semidefinite (see Boyd and Vandenberghe, 2004, Chapter 3). The convex-inducing classifiers also include 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. In our approach to this problem, we divide $\mathcal{F}^{\text{convex}}$, the family of convex-inducing classifiers, into

3. Lowd and Meek (2005) also previously showed that the reverse-engineering technique of finding a feature's sign witness is NP-complete for linear classifiers with Boolean features but also that this family was nonetheless 2-*IMAC* searchable.

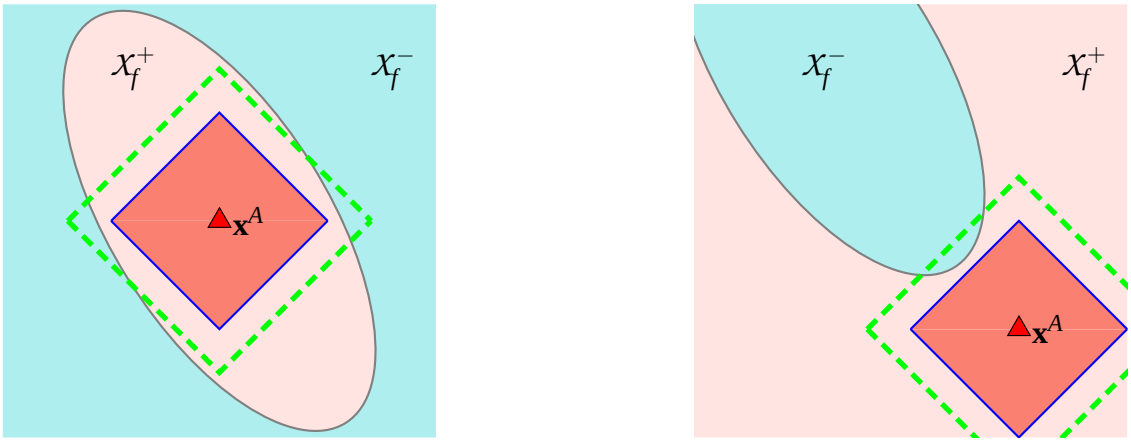


Figure 1: Geometry of convex sets and ℓ_1 balls. (a) If the positive set $\mathcal{X}_f^+$ is convex, finding an ℓ_1 ball contained within $\mathcal{X}_f^+$ establishes a lower bound on the cost, otherwise at least one of the ℓ_1 ball's corners witnesses an upper bound. (b) If the negative set $\mathcal{X}_f^-$ 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.

$\mathcal{F}^{\text{convex,'}-\text{'}}$ and $\mathcal{F}^{\text{convex,'}+\text{'}}$ corresponding to the classifiers that induce a convex set $\mathcal{X}_f^-$ or $\mathcal{X}_f^+$, respectively (of course, linear classifiers belong to both). When the negative class $\mathcal{X}_f^-$ is convex (i.e., $f \in \mathcal{F}^{\text{convex,'}-\text{'}}$), the problem reduces to minimizing a (convex) function A constrained to a convex set—if $\mathcal{X}_f^-$ were known to the adversary, then this would correspond to solving a convex program. When the positive class $\mathcal{X}_f^+$ is convex (i.e., $f \in \mathcal{F}^{\text{convex,'}+\text{'}}$), 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 an 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 (weighted) ℓ_1 cost $A_1^{(\mathbf{c})}$ of Equation (1) for the family of convex-inducing classifiers. There is an asymmetry to this problem depending on whether the positive or negative class is convex as illustrated in Figure 1. When the positive set is convex, determining whether the ℓ_1 ball $\mathcal{B}^C(A_1^{(\mathbf{c})})$ is a subset of $\mathcal{X}_f^+$ only requires querying the vertices of the ball as depicted in Figure 1(a). When the negative set is convex, determining whether $\mathcal{B}^C(A_1^{(\mathbf{c})}) \cap \mathcal{X}_f^- = \emptyset$ is non-trivial since the intersection need not occur at a vertex as depicted in Figure 1(b). We present an efficient algorithm for optimizing (weighted) ℓ_1 costs when $\mathcal{X}_f^+$ is convex and a polynomial randomized algorithm for optimizing any convex cost when $\mathcal{X}_f^-$ is convex. In both cases, we consider only convex sets with non-empty interiors. The algorithms we present achieve multiplicative optimality via the binary search strategies discussed in the previous section. In the sequel, we use Equation (4) to define L_ε and $C_0^- = A_1^{(\mathbf{c})}\left(\mathbf{x}^- - \mathbf{x}^A\right)$ as an initial upper bound on the *MAC*. We also assume there is some $C_0^+ > 0$ that lower bounds the *MAC*.

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

Solving the ε-*IMAC* Search problem when $f \in \mathcal{F}^{\text{convex,'+'}}$ is hard in the general case of a convex cost A. Here we introduce algorithms for the ℓ_1 cost 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 $\mathcal{B}^{C_t}(A_1) \subset \mathcal{X}_f^+$ 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 given in Equation (2) is to obtain multiplicative optimality, our steps will take the form $C_t = \sqrt{C_t^+ \cdot C_t^-}$. We now explain how we exploit the properties of the ℓ_1 ball and convexity of $\mathcal{X}_f^+$ to efficiently determine whether $\mathcal{B}^C(A_1)$ is a subset of $\mathcal{X}_f^+$ 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 d^{th} feature) and can be expressed as

$$\mathbf{x}^A \pm \frac{C}{c_d} \cdot \delta_d \quad , \tag{5}$$

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 d^{th} feature). We now formalize the second fact as follows.

Lemma 3 *For all $C > 0$, if there exists some $\mathbf{x} \in \mathcal{X}_f^-$ that achieves a cost of $C = A_1^{(\mathbf{c})}\left(\mathbf{x} - \mathbf{x}^A\right)$, then there is some feature d such that a vertex of the form of Equation* (5) *is in $\mathcal{X}_f^-$ (and also achieves cost C by Equation 1).*

Proof Suppose not; then there is some $\mathbf{x} \in \mathcal{X}_f^-$ such that $A_1^{(\mathbf{c})}\left(\mathbf{x} - \mathbf{x}^A\right) = C$ and $\mathbf{x}$ has $M \geq 2$ features that differ from $\mathbf{x}^A$ (if $\mathbf{x}$ only differs in one feature it would be of the form of Equation 5). 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 $\mathbf{e}_{d_i} = \mathbf{x}^A + \frac{C}{c_{d_i}} \cdot b_{d_i} \cdot \delta_{d_i}$ be a vertex of the form of Equation (5) which has a cost C (from Equation 1). The M vertices $\mathbf{e}_{d_i}$ form an M-dimensional equi-cost simplex of cost C on which $\mathbf{x}$ lies; that is, $\mathbf{x} = \sum_{i=1}^M \alpha_i \cdot \mathbf{e}_{d_i}$ 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 at least one vertex of the form Equation (5) 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 $1 + \varepsilon$. This yields an ε-*IMAC* of the form of Equation (5).

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

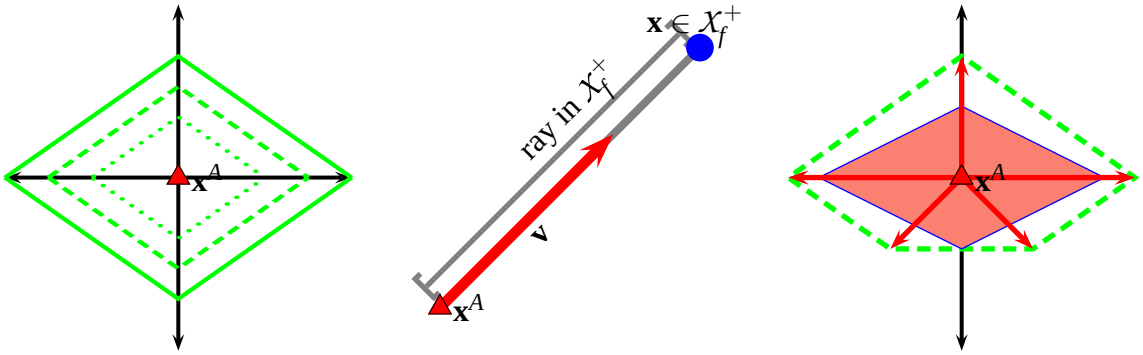


Figure 2: The geometry of search. (a) Weighted ℓ_1 balls are centered around the target $\mathbf{x}^A$ and have $2 \cdot D$ vertices; (b) Search directions in multi-line search radiate from $\mathbf{x}^A$ to probe specific costs; (c) In general, we leverage convexity of the cost function when searching 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.

directions that radiate from their origin at $\mathbf{x}^A$ and that are vectors of unit cost; that is, $A(\mathbf{w}) = 1$ for every $\mathbf{w} \in \mathcal{W}$. (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*—a practice that will lead to better bounds for a worst-case classifier. Further, when an upper bound is established for a cost C (a negative vertex is found), our algorithm prunes all directions that were positive at cost C. This pruning is sound; by convexity, 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 an ε-*IMAC* with no more than $|\mathcal{W}| \cdot L_\varepsilon$ queries but at least $|\mathcal{W}| + L_\varepsilon$ queries. Thus, this algorithm is $O(|\mathcal{W}| \cdot L_\varepsilon)$.

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 $A(\mathbf{0}) = 0$. Without this assumption, the ray-constrained cost function $A(s \cdot \mathbf{w})$ 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 any ray from $\mathbf{x}^A$; that is, $A(s \cdot \mathbf{w}) = |s| \cdot A(\mathbf{w})$. 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 Equation (1) satisfy both assumptions since they are norms centered at $\mathbf{x}^A$.

Algorithm 2 uses MULTILINESEARCH for ℓ_1 costs by taking $\mathcal{W}$ to 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 $\mathcal{B}^C(A_1)$ is a subset of $\mathcal{X}_f^+$ and so Algorithm 2 is $O(L_\varepsilon \cdot D)$. However, MULTILINESEARCH 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

Algorithm 1 MULTI-LINE SEARCH

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

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

ConvexSearch $\left(\mathbf{x}^A, \mathbf{x}^-, \mathbf{c}, \varepsilon, C^+\right)$
$D \leftarrow dim\left(\mathbf{x}^A\right)$
$C^- \leftarrow A^{(\mathbf{c})}\left(\mathbf{x}^- - \mathbf{x}^A\right)$
$\mathcal{W} \leftarrow \emptyset$
for $i = 1$ to D **do**
  $\mathbf{e}^i \leftarrow \frac{1}{c_i} \cdot \delta_i$
  $\mathcal{W} \leftarrow \mathcal{W} \cup \left\{\pm \mathbf{e}^i\right\}$
end for
return: *MLS* $\left(\mathcal{W}, \mathbf{x}^A, \mathbf{x}^-, C^+, C^-, \varepsilon\right)$

Figure 3: Algorithms for multi-line search. Algorithm 1 is a generic procedure for performing simultaneous binary searches along multiple search directions emanating from $\mathbf{x}^A$; each direction, $\mathbf{e} \in \mathcal{W}$, must be a unit-cost direction. Algorithm 2 uses this MULTILINESEARCH procedure to minimize weighted ℓ_1 costs when the positive class of a classifier is convex. For this procedure, every weight, c_i, must be on the range $(0, \infty)$ although extensions are discussed in Section 3.1.3.

and can be adapted for any set of directions that can provide a sufficiently tight bound on the cost using the convexity of $\mathcal{X}_f^+$(see Section 4.1.1 for the bounding requirements the search directions must satisfy). However, as we show in Section 4.1, the number of search directions required to adequately bound an ℓ_p cost ball 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 consists of $2 \cdot |\mathcal{W}|$ simultaneous binary searches (a breadth-first strategy). This strategy prunes directions most effectively when the convex body is asymmetrically elongated relative to $\mathbf{x}^A$ but fails to prune for symmetrically rounded bodies. We could instead search each direction sequentially (a depth-first strategy) and still obtain a worst case of $O(L_\varepsilon \cdot D)$ queries. This strategy uses fewer queries to shrink the cost gap on symmetrically rounded bodies but is unable to do so for asymmetrically elongated bodies. We therefore propose an algorithm that mixes these strategies.

At each phase, the K-STEP MULTILINESEARCH (Algorithm 3) chooses a single direction $\mathbf{e}$ and queries it for K steps to generate candidate bounds B^- and B^+ on the *MAC*. The algorithm makes substantial progress towards reducing G_t without querying other directions (a depth-first strategy). It then iteratively queries all remaining directions at the candidate lower bound B^+ (a breadth-first strategy). 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 B^+. Thus, in every iteration, either the gap is substantially decreased or at least one search direction is pruned. We show that for $K = \lceil\sqrt{L_\varepsilon}\rceil$, the algorithm achieves a delicate balance between the usual breadth-first and depth-first approaches to attain a better worst-case complexity than either.

Theorem 4 *Algorithm 3 will find an* ε*-IMAC with at most* $O\left(L_\varepsilon + \sqrt{L_\varepsilon}|\mathcal{W}|\right)$ *queries when* $K = \lceil\sqrt{L_\varepsilon}\rceil$.

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

3.1.2 LOWER BOUND

Here we find a lower bound 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., Equation 1 for $p \geq 1$). Below we present a theorem that provides a lower bound for multiplicative optimality (for additive optimality, there is an analogous lower bound for any $0 < \eta < C_0^- - C_0^+$). Notably, since an ε-*IMAC* uses multiplicative optimality, we incorporate a bound $C_0^+ > 0$ on the *MAC* into our statement.

Theorem 5 *For any* $D > 0$*, any positive convex function* $A : \Re^D \to \Re^+$*, any initial bounds* $0 < C_0^+ < C_0^-$ *on the MAC, and* $0 < \varepsilon < \frac{C_0^-}{C_0^+} - 1$*, all algorithms must submit at least* $\max\{D, L_\varepsilon^{(*)}\}$ *membership queries in the worst case to be* ε*-multiplicatively optimal on* $\mathcal{F}^{\text{convex},'+'}$.

The proof of this result is in Appendix B. In this theorem, we restrict ε to the interval $\left(0, \frac{C_0^-}{C_0^+} - 1\right)$ since, outside of this interval, the strategy is trivial. For $\varepsilon = 0$ no approximation algorithm terminates and for $\varepsilon \geq \frac{C_0^-}{C_0^+} - 1$, $\mathbf{x}^-$ is an ε-*IMAC*, so no queries are required.

Theorem 5 shows that ε-multiplicative optimality requires $\Omega(L_\varepsilon^{(*)} + D)$ queries. Thus, we see that our K-STEP MULTILINESEARCH algorithm (Algorithm 3) has close to the optimal query complexity for ℓ_1-costs with its $O(L_\varepsilon + \sqrt{L_\varepsilon}D)$ queries. This lower bound also applies to any ℓ_p cost with $p > 1$, but in Section 4 we show lower bounds for $p > 1$ that substantially improve this result.

3.1.3 SPECIAL CASES

Here we present a number of special cases that require minor modifications to Algorithms 1 and 3 primarily as preprocessing steps.

Revisiting Linear Classifiers: Lowd and Meek originally developed a method for reverse engineering linear classifiers for an ℓ_1 cost. First their method isolates a sequence of points from $\mathbf{x}^-$ to $\mathbf{x}^A$

Algorithm 3 *K*-STEP MULTI-LINE SEARCH

$KMLS\left(\mathcal{W}, \mathbf{x}^A, \mathbf{x}^-, C_0^+, C_0^-, \varepsilon, K\right)$
$\mathbf{x}^* \leftarrow \mathbf{x}^-$
$t \leftarrow 0$
while $C_t^- / C_t^+ > 1 + \varepsilon$ **do**
  Choose a direction $\mathbf{e} \in \mathcal{W}$
  $B^+ \leftarrow C_t^+$
  $B^- \leftarrow C_t^-$
  for K steps **do**
    $B \leftarrow \sqrt{B^+ \cdot B^-}$
    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 $\mathbf{i} \in \mathcal{W} \setminus \{\mathbf{e}\}$ **do**
    Query: $f_{\mathbf{i}}^t \leftarrow f\left(\mathbf{x}^A + (B^+) \cdot \mathbf{i}\right)$
    if $f_{\mathbf{i}}^t =$ '−' **then**
      $\mathbf{x}^* \leftarrow \mathbf{x}^A + (B^+) \cdot \mathbf{i}$
      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: $\mathbf{x}^*$

Figure 4: Algorithm for multi-line search. It performs simultaneous binary searches along multiple unit search directions emanating from $\mathbf{x}^A$. Algorithm 3 is asympototically more efficient than Algorithm 1 when $K = \lceil \sqrt{L_\varepsilon} \rceil$ and can be used as a substitute for it in Algorithm 2.

that cross the classifier's boundary and then the method estimates the hyperplane's parameters using D binary 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. Because linear classifiers are a special case of convex-inducing classifiers, Algorithm 2 can be applied, and our K-STEP MULTILINESEARCH algorithm improves on complexity of Lowd and Meek's reverse-engineering technique's $O(L_\varepsilon \cdot D)$ queries and applies to a broader family of classifiers.

While Algorithm 2 has superior complexity, it uses $2 \cdot D$ search directions rather than the D directions used in the approach of Lowd and Meek, which may require our technique to issue more queries in some practical settings. However, for some restrictive classifier families, it is also pos-

sible to eliminate search directions proved to be infeasible based on the current set of queries. For instance, given a set $\mathcal{W}$ of search directions, t queries $\{\mathbf{x}^i\}_{i=1}^t$ and their corresponding responses $\{y^i\}_{i=1}^t$, a search direction $\mathbf{e}$ can be eliminated from $\mathcal{W}$ if for all $C_t^+ \leq \alpha < C_t^-$ there does not exist any classifier $f \in \mathcal{F}$ consistent with all previous queries (i.e., $f(\mathbf{x}^-) = '-'$, $f(\mathbf{x}^A) = '+'$ and for all $i \in \{1,\ldots,t\}$, $f(\mathbf{x}^i) = y^i$) that also satisfies $f(\alpha \cdot \mathbf{e}) = '-'$ and $f(\alpha \cdot \mathbf{i}) = '+'$ for every $\mathbf{i} \in \mathcal{W} \setminus \{\mathbf{e}\}$. That is, $\mathbf{e}$ is feasible if and only if it is the only search direction among the set of remaining search directions, $\mathcal{W}$, that would be classified as a negative for a cost α by some consistent classifier. Further, since subsequent queries only restrict the feasible space of α and the set of consistent classifiers $\hat{\mathcal{F}}$, pruning these infeasible directions is sound for the remainder of the search.

For restrictive families of convex-inducing classifiers, these feasibility conditions can be efficiently verified and may be used to prune search directions without issuing further queries. In fact, for the family of linear classifiers written as $f(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$ for a normal vector $\mathbf{w}$ and displacement b, the above conditions become a set of linear inequalities along with quadratic inequalities corresponding to the constraint involving search directions. This can be cast as the following optimization program with respect to α, $\mathbf{w}$ and b:

$$\begin{aligned}
\min_{\alpha,\mathbf{w},b} \quad & \alpha \cdot \mathbf{w}^\top \mathbf{e} + b \\
& \alpha \in [C_t^+, C_t^-) \\
& \mathbf{w}^\top \mathbf{x}^- + b \leq 0 \\
\text{s.t.} \quad & \mathbf{w}^\top \mathbf{x}^A + b \geq 0 \\
& y^i(\mathbf{w}^\top \mathbf{x}^i + b) \geq 0 \quad \forall\, i \in \{1,\ldots,t\} \\
& \alpha \cdot \mathbf{w}^\top \mathbf{i} + b \geq 0 \quad \forall\, \mathbf{i} \in \mathcal{W} \setminus \{\mathbf{e}\}.
\end{aligned}$$

If the resulting minimum is less than zero, direction $\mathbf{e}$ is feasible, otherwise, $\mathbf{e}$ can be pruned. Such programs can be efficiently solved and may allow the adversary to rapidly eliminate infeasible search directions without issuing additional queries. However, refining these pruning procedures further is beyond the scope of this paper.

Extending MULTILINESEARCH *Algorithms to Weights $c_d = \infty$ or $c_d = 0$:* In Algorithm 2, we reweighted the d^{th} axis-aligned directions by a factor $\frac{1}{c_d}$ to make unit cost vectors by implicitly assuming $c_d \in (0,\infty)$. The case of immutable features where $c_d = \infty$ is dealt with by simply removing those features from the set of search directions $\mathcal{W}$ used in the MULTILINESEARCH. In the case of useless or unconstrained features when $c_d = 0$, MULTILINESEARCH-like algorithms no longer ensure near-optimality because they implicitly assume that cost balls are bounded sets. If $c_d = 0$, then $\mathcal{B}^0(A)$ is no longer bounded and 0 cost can 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 t^{th} iteration) as we repeatedly rerun 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 it terminates with a non-zero cost instance, which is an ε-*IMAC* if no 0-cost negative instances exist. This algorithm does not ensure near-optimality but may be suitable for practical settings using some fixed T runs.

Lack of an Initial Lower Cost Bound: Thus far, to find an 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 the SPIRALSEARCH algorithm (Algorithm 4) that efficiently establishes 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 this candidate bound. Either the lower bound is verified or directions that were positive can be pruned for the remainder of the search.

Algorithm 4 SPIRAL SEARCH

SpiralSearch $\left(\mathcal{W}, \mathbf{x}^A, C_0^-\right)$
$t \leftarrow 0$ 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: $f_\mathbf{e} \leftarrow f\left(\mathbf{x}^A + C_0^- \cdot 2^{-2^t} \cdot \mathbf{e}\right)$
  if $f_\mathbf{e} = \text{'}-\text{'}$ **then**
    $\mathcal{W} \leftarrow \mathcal{W} \cup \{\mathbf{e}\}$ and $\mathcal{V} \leftarrow \emptyset$
    $t \leftarrow t + 1$
  end if
until $\mathcal{W} = \emptyset$
$C_0^+ \leftarrow C_0^- \cdot 2^{-2^t}$
if $t > 0$ **then** $C_0^- \leftarrow C_0^- \cdot 2^{-2^{t-1}}$
return: $(\mathcal{V}, C_0^+, C_0^-)$

Figure 5: Algorithm for establishing an initial lower bound on the cost.

At the t^{th} iteration of SPIRALSEARCH, a direction is selected and queried at the candidate lower bound of $(C_0^-)2^{-2^t}$. If the query is positive, that direction is added to the set $\mathcal{V}$ of directions consistent with the lower bound. Otherwise, all positive directions in $\mathcal{V}$ are pruned, a new upper bound is established, and the candidate lower bound is reduced with an exponentially decreasing exponent. By definition of the *MAC*, this algorithm will terminate after $t = \left\lceil \log_2 \log_2 \frac{C_0^-}{MAC(f,A)} \right\rceil$ iterations. Further, in this algorithm, multiple directions are probed only during iterations with positive queries and it makes at most one positive query for each direction. Thus, given that some lower bound $C_0^+ > 0$ does exist, SPIRALSEARCH will establish a lower bound with $O(L'_\varepsilon + D)$ queries, where L'_ε is given by Equation (4) defined using $C_0^+ = MAC(f,A)$; the largest possible lower bound.

This algorithm can be used as a precursor to any of the previous searches.[4] Upon completion, the upper and lower bounds it establishes have a multiplicative gap of $2^{2^{t-1}}$ for $t > 0$ or 2 for $t = 0$. From the definition of t provided above in terms of the *MAC*, MULTILINESEARCH can hence proceed using $L_\varepsilon = L'_\varepsilon$. Further, the search directions pruned by SPIRALSEARCH are also invalid for the subsequent MULTILINESEARCH so the set $\mathcal{V}$ returned by SPIRALSEARCH will be used as the initial set $\mathcal{W}$ for the subsequent search. Thus, the query complexity of the subsequent search is the same as if it had started with the best possible lower bound.

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

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 t^{th} 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 $(C_0^+)2^{2^{T-1}} < MAC(f,A) \leq (C_0^+)2^{2^T}$; 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 $C_0^- = 2^{2^T}$; that is, $\log_2 G_0 = 2^{T-1}$. This precursor step requires at most $|\mathcal{W}| \cdot T$ queries to initialize the MULTILINESEARCH algorithm with a gap such that $L_\varepsilon = \left\lceil (T-1) + \log_2 \frac{1}{\log_2(1+\varepsilon)} \right\rceil$ according to Equation (4).

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

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

Here, we minimize a convex cost function A with bounded cost balls (we focus on weighted ℓ_1 costs in Equation 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 an ellipsoid or interior point method; 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 $\mathcal{B}^{C^t}(A)$; that is, whether C^t is a new lower or upper bound on the *MAC*. With high probability, this approach can find an ε-*IMAC* in no more than L_ε repetitions using binary search. The following theorem is the main result of this section.

Theorem 6 *Let cost function A be convex and have bounded balls; that is, bounded sublevel sets. Let the feasible set $\mathcal{X}_f^-$ be convex and assume there is some $r > 0$ and $\mathbf{y} \in \mathcal{X}_f^-$ such that $\mathcal{X}_f^-$ contains the cost ball $\mathcal{B}^r(A;\mathbf{y})$. Then given access to an oracle returning separating hyperplanes for the A cost balls, Algorithm 7 will find an ε-IMAC using $O^*\left(D^5\right)$ queries with high probability.*[5]

The proof of this result is outlined in the remainder of this section, and is based on Bertsimas and Vempala (2004, Theorem 14). We first introduce their randomized ellipsoid algorithm, then we elaborate on their procedure for efficient sampling from a convex body, and finally we present our application to optimization. In this section, we focus only on weighted ℓ_1 costs (Equation 1) and return to more general cases in Section 4.2.

3.2.1 INTERSECTION OF CONVEX SETS

Bertsimas and Vempala present a query-based procedure for determining whether two convex sets (e.g., $\mathcal{X}_f^-$ and the A_1-ball of radius C^t) intersect. Their INTERSECTSEARCH procedure, which we

5. $O^*(\cdot)$ denotes the standard complexity notation $O(\cdot)$ without logarithmic terms. The dependence on ε is in these logarithmic terms, see Bertsimas and Vempala (2004) for details.

Algorithm 5 INTERSECT SEARCH

IntersectSearch $\left(\mathcal{P}^0, Q = \left\{\mathbf{x}^j \in \mathcal{P}^0\right\}, \mathbf{x}^A, C\right)$
for $s = 1$ to T **do**
  (1) Generate $2N$ samples $\left\{\mathbf{x}^j\right\}_{j=1}^{2N}$
    Choose $\mathbf{x}$ from Q
    $\mathbf{x}^j \leftarrow$ *HitRun* $\left(\mathcal{P}^{s-1}, Q, \mathbf{x}^j\right)$
  (2) If any $\mathbf{x}^j$, $A\left(\mathbf{x}^j - \mathbf{x}^A\right) \leq C$ 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 $\mathcal{S} \leftarrow \left\{\mathbf{x}^j\right\}_{j=N+1}^{2N}$
  (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 Equation (7)
  (6) $\mathcal{P}^s \leftarrow \mathcal{P}^{s-1} \cap \mathcal{H}_{\mathbf{z}^s}$
  (7) Keep samples in $\mathcal{P}^s$
    $Q \leftarrow \{\mathbf{x} \in \mathcal{S} \wedge \mathbf{x} \in \mathcal{P}^s\}$
end for
Return: the found $[\mathbf{x}_j, \mathcal{P}^s, Q]$; or No Intersect

Algorithm 6 HIT-AND-RUN

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

Figure 6: Algorithms used for the randomized ellipsoid algorithm of Bertsimas and Vempala. INTERSECTSEARCH is used to find the intersection between a pair of convex sets: $\mathcal{P}^0$ is queryable and $\mathcal{B}$ provides has a separating hyperplane from Equation (7). Note that the ROUNDING algorithm discussed in Section 3.2.2 can be used as a preprocessing step so that $\mathcal{P}^0$ is near-isotropic and to obtain the samples for Q. The HIT-AND-RUN algorithm is used to efficiently obtain uniform samples from a bounded near-isotropic convex set, $\mathcal{P}$, based on a set of uniform samples from it, $\left\{\mathbf{y}^j\right\}$, and a starting point $\mathbf{x}^0$.

present as Algorithm 5, 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 (for our problem these sets correspond to $\mathcal{X}_f^-$ and $\mathcal{B}^{C^t}(A_1)$ respectively). 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 $\mathcal{B}$ significantly reduces 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 $\mathcal{P} \cap \mathcal{B}$ or it is highly likely that the intersection is 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 cost balls, so we can instead use the set $\mathcal{P}^0 = \mathcal{X}_f^- \cap \mathcal{B}^{2R}(A_1; \mathbf{x}^-)$ (where $R = A\left(\mathbf{x}^- - \mathbf{x}^A\right) > C^t$), which is a convex bounded subset of $\mathcal{X}_f^-$. Since, by the triangle inequality, the ball $\mathcal{B}^{2R}(A_1; \mathbf{x}^-)$ centered at $\mathbf{x}^-$ envelops all of $\mathcal{B}^{C^t}\left(A_1; \mathbf{x}^A\right)$ centered at $\mathbf{x}^A$, the set $\mathcal{P}^0$ contains the entirety of the desired 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 $\mathcal{X}_f^-$; that is, there exists $\mathbf{y} \in \mathcal{X}_f^-$ such that the r-ball centered at $\mathbf{y}$, $\mathcal{B}^r(A_1; \mathbf{y})$, is a subset of $\mathcal{X}_f^-$. This assumption both ensures that $\mathcal{X}_f^-$ has a non-empty interior (a requirement for the HIT-AND-RUN

Algorithm 7 CONVEX $\mathcal{X}_f^-$ SET SEARCH

SetSearch $\left(\mathcal{P}, Q = \left\{\mathbf{x}^j \in \mathcal{P}\right\}, \mathbf{x}^A, \mathbf{x}^-, C_0^+, C_0^-, \varepsilon\right)$
$\mathbf{x}^* \leftarrow \mathbf{x}^-$ and $t \leftarrow 0$
while $C_t^- / C_t^+ > 1 + \varepsilon$ **do**
  $C_t \leftarrow \sqrt{C_t^- \cdot C_t^+}$
  $[\mathbf{x}^*, \mathcal{P}', Q'] \leftarrow$ *IntersectSearch* $\left(\mathcal{P}, Q, \mathbf{x}^A, C_t\right)$
  if intersection found **then**
    $C_{t+1}^- \leftarrow A\left(\mathbf{x}^* - \mathbf{x}^A\right)$ and $C_{t+1}^+ \leftarrow C_t^+$
    $\mathcal{P} \leftarrow \mathcal{P}'$ and $Q \leftarrow Q'$
  else
    $C_{t+1}^- \leftarrow C_t^-$ and $C_{t+1}^+ \leftarrow C_t$
  end if
  $t \leftarrow t + 1$
end while
Return: $\mathbf{x}^*$

Figure 7: Algorithm that efficiently implements the randomized ellipsoid algorithm of Bertsimas and Vempala. SETSEARCH performs a binary search for an ε-*IMAC* using the randomized INTERSECTSEARCH procedure to determine, with high probability, whether or not $\mathcal{X}_f^-$ contains any points less than a specified cost, C_t. Note that the ROUNDING algorithm discussed in Section 3.2.2 can be used as a preprocessing step so that $\mathcal{P}$ is near-isotropic and to obtain the samples for Q.

algorithm discussed below) and it provides a stopping condition for the overall intersection search algorithm.

The foundation of Bertsimas and Vempala's search algorithm is the capability to sample uniformly from an unknown but bounded convex body by means of the HIT-AND-RUN random walk technique (Algorithm 6) introduced by Smith (1996). 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 $\Omega = \left\{\omega \geq 0 \;\middle|\; \mathbf{x}^j + \omega\mathbf{v} \in \mathcal{P}^{s-1}\right\}$ 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 $\mathbf{x}^j + \omega\mathbf{v}$. As noted above, this random walk will not make progress if the interior of $\mathcal{P}^{s-1}$ is empty (which we preclude by assuming that $\mathcal{X}_f^-$ contains an r-ball), and efficient sampling also requires that $\mathcal{P}^{s-1}$ is sufficiently round. However, under the conditions discussed in Section 3.2.2, the HIT-AND-RUN random walk generates a sample uniformly from the convex body after $O^*\left(D^3\right)$ steps (Lovász and Vempala, 2004). We now detail the overall INTERSECTSEARCH procedure (Algorithm 5) and then discuss the mechanism used to maintain efficient sampling after each successive cut. It is worth noting that Algorithm 5 requires $\mathcal{P}^0$ to be in near-isotropic position and that Q is a set of samples from it; these requirements are met by using the ROUNDING algorithm of Lovász and Vempala discussed at the end of Section 3.2.2.

Randomized Ellipsoid Method: We use HIT-AND-RUN to obtain $2N$ samples $\left\{\mathbf{x}^j\right\}$ from $\mathcal{P}^{s-1} \subset \mathcal{X}_f^-$ for a single phase of the randomized ellipsoid method. If any sample $\mathbf{x}^j$ satisfies $A_1\left(\mathbf{x}^j - \mathbf{x}^A\right) \leq$

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 is given by

$$h_d^{\mathbf{y}} = c_d \operatorname{sign}\left(y_d - x_d^A\right) \quad , \tag{6}$$

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 $\mathbf{z} = N^{-1}\sum_{\mathbf{x}\in\mathcal{R}}\mathbf{x}$ of the half of our samples in $\mathcal{R}$; the other half will be used in Section 3.2.2. We cut $\mathcal{P}^{s-1}$ with the hyperplane $\mathbf{h}^{\mathbf{z}}$ through $\mathbf{z}$; that is, $\mathcal{P}^s = \mathcal{P}^{s-1} \cap \mathcal{H}_{\mathbf{z}}$ where $\mathcal{H}_{\mathbf{z}}$ is the halfspace

$$\mathcal{H}_{\mathbf{z}} = \left\{ \mathbf{x} \;\middle|\; \mathbf{x}^\top \mathbf{h}^{\mathbf{z}} < \mathbf{z}^\top \mathbf{h}^{\mathbf{z}} \right\} \quad . \tag{7}$$

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

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

3.2.2 SAMPLING FROM A QUERYABLE CONVEX BODY

In the randomized ellipsoid method, 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*: for any unit vector $\mathbf{v}$, $\mathbb{E}_{\mathbf{x}\sim\mathcal{P}}\left[\left(\mathbf{v}^\top(\mathbf{x} - \mathbb{E}_{\mathbf{x}\sim\mathcal{P}}[\mathbf{x}])\right)^2\right]$ is bounded between $1/2$ and $3/2$ of $vol(\mathcal{P})$.

If the body is not near-isotropic, we must rescale $\mathcal{X}$ with an appropriate affine transformation $\mathbf{T}$ so the resulting transformed 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 Q of sufficiently many uniform samples from the body $\mathcal{P}^s$, and in the HIT-AND-RUN algorithm (Algorithm 6), we sample the direction $\mathbf{v}$ based on this set. Intuitively, because the samples in Q are distributed uniformly in $\mathcal{P}^s$, the directions we sample based on the points in 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 and covariance of $\mathcal{P}$.

We must ensure Q is a set of sufficiently many samples from $\mathcal{P}^s$ after each cut taking $\mathcal{P}^s \leftarrow \mathcal{P}^{s-1} \cap \mathcal{H}_{\mathbf{z}^s}$. 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 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, for this sampling approach to succeed, we need the initial set $\mathcal{P}^0$ to be transformed into near-isotropic position and we also need an initial set Q of uniform samples from the transformed $\mathcal{P}^0$ as input to Algorithm 5. However in our problem, we only have a single point $\mathbf{x}^- \in \mathcal{X}_f^-$ and our set, $\mathcal{P}^0$, need not be near-isotropic. Fortunately, there is an iterative procedure that uses the HIT-AND-RUN algorithm to simultaneously transform the initial convex set, $\mathcal{P}^0$, into a near-isotropic position and construct our initial set of samples, Q. This algorithm, the ROUNDING algorithm as described by Lovász and Vempala (2003), uses $O^*\left(D^4\right)$ membership queries to find a transformation that places $\mathcal{P}^0$ into a near-isotropic position and produces an initial set of samples from it. We use this as a preprocessing step for Algorithms 5 and 7; that is, given $\mathcal{X}_f^-$ and $\mathbf{x}^- \in \mathcal{X}_f^-$, we construct $\mathcal{P}^0 = \mathcal{X}_f^- \cap \mathcal{B}^{2R}(A_1; \mathbf{x}^-)$ and then can use the ROUNDING algorithm to transform $\mathcal{P}^0$ and produce an initial uniform sample from it; that is, $Q = \left\{\mathbf{x}^j \in \mathcal{P}^0\right\}$. 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. These improvements are reflected in our final procedure SET-SEARCH in Algorithm 7—the total number of queries required is also $O^*\left(D^5\right)$. Again, Algorithm 7 requires $\mathcal{P}$ to be near-isotropic and that Q is a set of samples from it, which is accomplished by the ROUNDING algorithm discussed at the end of Section 3.2.2. First, notice that $\mathbf{x}^A$ and $\mathbf{x}^-$ are the same for every iteration of the optimization procedure. Further, in each iteration of Algorithm 7, the new set, $\mathcal{P}$, remains near-isotropic and the new Q is a set of samples from it since the sets returned by Algorithm 5 retain these properties. Thus, the set, $\mathcal{P}$, and the set of samples, $Q = \left\{\mathbf{x}^j \in \mathcal{P}\right\}$, maintained by Algorithm 7 are sufficient to initialize INTERSECTSEARCH at each stage of its overall binary search over C^t, and we only need to execute the ROUNDING procedure once as a preprocessing step rather than re-invoking it before every invocation of INTERSECTSEARCH. Second, the separating hyperplane $\mathbf{h}^{\mathbf{y}}$ given by Equation (6) does not depend on the target cost C^t but only on $\mathbf{x}^A$, the common center of all the ℓ_1 balls used in this search. In fact, the separating hyperplane at point $\mathbf{y}$ is valid for all ℓ_1-balls of cost $C < A\left(\mathbf{y} - \mathbf{x}^A\right)$. Further, if $C < C^t$, we have $\mathcal{B}^C(A_1) \subset \mathcal{B}^{C^t}(A_1)$. Thus, the final state from a successful call to INTERSECTSEARCH for the C^t-ball can be used as the starting state for any subsequent call to INTERSECTSEARCH for all $C < C^t$. Hence, in Algorithm 7, we update $\mathcal{P}$ and Q only when Algorithm 5 succeeds.

4. Evasion for 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 $0 < p \leq \infty$. As we demonstrate in this section, many ℓ_p costs are not generally ε-*IMAC* searchable for all $\varepsilon > 0$ 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 L_ε). In

fact, only the weighted ℓ_1 costs have known (randomized) polynomial query strategies when either the positive or negative set is convex.

4.1 Convex Positive Set

Here we explore the ability of the 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 8 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 using an ℓ_1 cost.

Lemma 7 *The largest ℓ_p ($p > 1$) ball enclosed within a C-cost ℓ_1 ball has a cost of $C \cdot D^{\frac{1-p}{p}}$ and for $p = \infty$ the cost is $C \cdot D^{-1}$.*

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 a lower bound of C_0^+. 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 $C^\dagger \leq C_0^+$. In order to achieve ε-multiplicative optimality, we need $\frac{C_0^-}{C^\dagger} \leq 1+\varepsilon$, which we can rewrite as

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

This allows us to break the problem into two parts. The first ratio C_0^-/C_0^+ is controlled solely by the accuracy ε achieved by running the multiline search algorithm for L_ε steps whereas the second ratio $C_0^+/C^\dagger$ depends only on how well the ℓ_p ball is approximated by the convex hull of the M search directions. These two ratios separate our task into choosing M and L_ε so that their product is less than $1+\varepsilon$. First we can choose parameters $\alpha \geq 0$ and $\beta \geq 0$ so that $(1+\alpha)(1+\beta) \leq 1+\varepsilon$. Then we choose M so that $\frac{C_0^+}{C^\dagger} = 1+\beta$ and use L_α steps 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 can achieve ε-multiplicative optimality.

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 by taking $\alpha = \varepsilon$, we recover our original multiline search result.

We now address costs where $\beta > 0$. For a MULTILINESEARCH algorithm to be efficient, it is necessary that $\frac{C_0^+}{C^\dagger} = 1+\beta$ can be achieved with polynomially-many search directions (in D and L_ε) for some $\beta \leq \varepsilon$; otherwise, $(1+\alpha)(1+\beta) > 1+\varepsilon$ and the MULTILINESEARCH approach cannot succeed for any $\alpha > 0$. Thus, we quantify how many search directions (or queries) are required to achieve $\frac{C_0^+}{C^\dagger} \leq 1+\varepsilon$. Note that this ratio is independent of the relative size of these costs, so without loss of generality we will only consider bounds for unit-cost balls. Thus, we compute the largest value of $C^\dagger$ that can be achieved for the unit-cost ℓ_p ball (i.e., we make $C_0^+ = 1$) within the convex

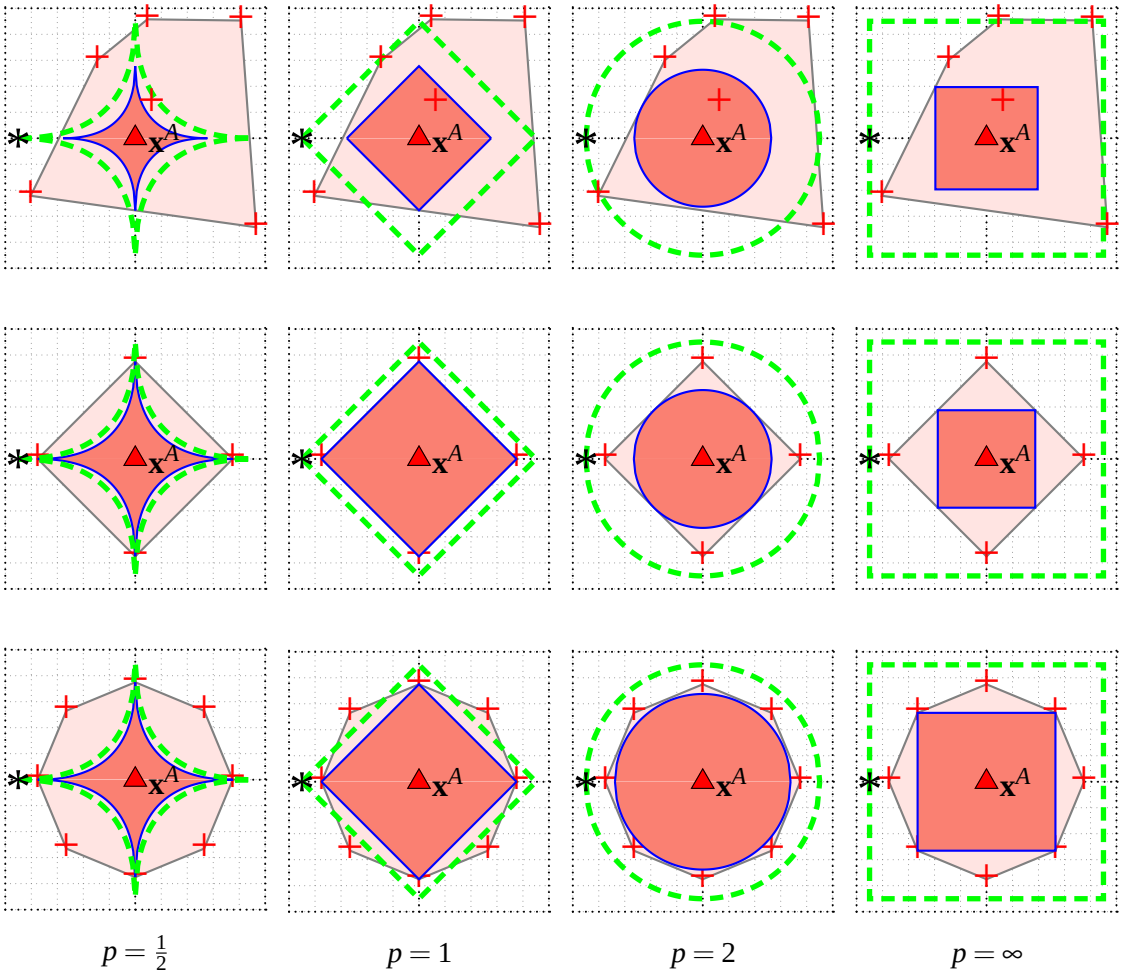


Figure 8: Convex hull for a set of queries and the resulting bounding balls for several ℓ_p costs. Each row represents a unique set of positive (red '+' points) and negative (black '∗' points) queries and each column shows the implied upper bound (the green dashed ball) and lower bound (the solid blue ball) for a different ℓ_p cost. In the first row, the body is defined by a random set of seven queries, in the second, the queries are along the coordinate axes, and in the third, the queries are around a circle.

hull of M queries. In particular, we quantify how many queries are required to achieve

$$C^\dagger \geq \frac{1}{1+\varepsilon} \ .$$

We would like to show that only polynomially-many are required for at least some values of ε as this is sufficient for a MULTILINESEARCH approach to be efficient.

Lemma 8 *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* MULTILINESEARCH *algorithms can use those search*

directions to achieve ε*-multiplicative optimality with a query complexity that is polynomial in* M *and* $L_\varepsilon^{(*)}$ *for any*

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

Moreover, if the M *search directions yield* $C^\dagger = 1$ *for the cost function* A*, then* MULTILINESEARCH *algorithms can achieve* ε*-multiplicative optimality with a query complexity that is polynomial in* M *and* $L_\varepsilon^{(*)}$ *for any* $\varepsilon > 0$*.*

Notice that this lemma also reaffirms that for $p = 1$ using the $M = 2 \cdot D$ axis-aligned directions allows MULTILINESEARCH algorithms to achieve ε-multiplicative optimality for any $\varepsilon > 0$ with a query complexity that is polynomial in M and $L_\varepsilon^{(*)}$.

4.1.2 MULTILINE SEARCH FOR $0 < p < 1$

A simple result holds here. Namely, since the unit ℓ_1 ball bounds any unit ℓ_p balls with $0 < p < 1$, we can achieve $C_0^+ / C^\dagger = 1$ using only the $2 \cdot D$ axis-aligned search directions. Thus, for any $0 < p < 1$, we can efficiently search for any value of $\varepsilon > 0$. Whether or not any ℓ_p ($0 < p < 1$) cost function can be efficiently optimized 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 9 *For* $1 < p < \infty$ *and* $\varepsilon \in \left(D^{\frac{p-1}{p}} - 1, \infty\right)$ *any multi-line search algorithm can achieve* ε*-multiplicative optimality on* A_p *using* $M = 2 \cdot D$ *search directions. Similarly for* $\varepsilon \in (D-1, \infty)$ *any multi-line search algorithm can achieve* ε*-multiplicative optimality on* A_∞ *also using* $M = 2 \cdot D$ *directions.*

Proof From Lemma 7, the largest co-centered ℓ_p ball contained within the unit ℓ_1 ball has radius $D^{\frac{1-p}{p}}$ cost (or D for $p = \infty$). The bounds on ε then follow from Lemma 8. ■

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 D as the following result shows.

Theorem 10 *For* $p > 1$, $D > 0$, *any initial bounds* $0 < C_0^+ < C_0^-$ *on the MAC, and* $\varepsilon \in \left(0, 2^{\frac{p-1}{p}} - 1\right)$ *(or* $\varepsilon \in (0,1)$ *for* $p = \infty$*), all algorithms must submit at least* $\alpha_{p,\varepsilon}^D$ *membership queries (for some constant* $\alpha_{p,\varepsilon} > 1$*) in the worst case to be* ε*-multiplicatively optimal on* $\mathcal{F}^{\text{convex},'+'}$ *for* ℓ_p *costs.*

The proof of this theorem and the definition of $\alpha_{p,\varepsilon}$ are provided in Appendix C. A consequence of this result is that there is no query-based algorithm that can efficiently find an ε-*IMAC* of any ℓ_p cost ($p > 1$) for any *fixed* ε within the range $0 < \varepsilon < 2^{\frac{p-1}{p}} - 1$ (or $0 < \varepsilon < 1$ for $p = \infty$) on the family $\mathcal{F}^{\text{convex},'+'}$. However, from Theorem 9 and Lemma 8, multiline-search type algorithms efficiently find the ε-*IMAC* of any ℓ_p cost ($p > 1$) for any $\varepsilon \in \left(D^{\frac{p-1}{p}} - 1, \infty\right)$ (or $D - 1 < \varepsilon < \infty$ 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 $p = 2$.

4.1.4 MULTILINE SEARCH FOR $p = 2$

Theorem 11 *For any $D > 1$, any initial bounds $0 < C_0^+ < C_0^-$ on the MAC, and $0 < \varepsilon < \frac{C_0^-}{C_0^+} - 1$, all algorithms must submit at least $\alpha_\varepsilon^{\frac{D-2}{2}}$ membership queries (where $\alpha_\varepsilon = \frac{(1+\varepsilon)^2}{(1+\varepsilon)^2-1} > 1$) in the worst case to be ε-multiplicatively optimal on $\mathcal{F}^{\text{convex},'+'}$ for ℓ_2 costs.*

The proof of this result is in Appendix D.

This result says that there is no algorithm that can generally achieve ε-multiplicative optimality for ℓ_2 costs for any *fixed* $\varepsilon > 0$ using only polynomially-many queries in D since the ratio $\frac{C_0^-}{C_0^+}$ could be arbitrarily large. It may appear that Theorem 11 contradicts Corollary 9. However, Corollary 9 only applies for an interval of ε that depends on D; that is, $\varepsilon > \sqrt{D} - 1$. Interestingly, substituting this lower bound on ε into the bound given by Theorem 11, we get that the number of required queries for $\varepsilon > \sqrt{D} - 1$ need only be

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

which is a monotonically increasing function in D that asymptotes at $\sqrt{e} \approx 1.64$. Thus, Theorem 11 and Corollary 9 are in agreement since for $\varepsilon > \sqrt{D} - 1$, the former only requires that we need at least 2 queries.

4.2 Convex Negative Set

Algorithm 7 generalizes immediately to all weighted ℓ_p costs ($p \geq 1$) centered at $\mathbf{x}^A$ since they are convex. For these costs, an equivalent separating hyperplane for $\mathbf{y}$ can be used in place of Equation (6). They are given by the equivalent (sub)-gradients for ℓ_p cost balls:

$$\begin{aligned} h^{\mathbf{y}}_{p,d} &= c_d \cdot \text{sign}\left(y_d - x^A_d\right) \cdot \left(\frac{|y_d - x^A_d|}{A_p^{(\mathbf{c})}\left(\mathbf{y} - \mathbf{x}^A\right)}\right)^{p-1} \quad , \\ h^{\mathbf{y}}_{\infty,d} &= c_d \cdot \text{sign}\left(y_d - x^A_d\right) \cdot \mathbb{I}\left\{|y_d - x^A_d| = A_\infty^{(\mathbf{c})}\left(\mathbf{y} - \mathbf{x}^A\right)\right\} \quad . \end{aligned}$$

By only changing the cost function A and the separating hyperplane $\mathbf{h}^{\mathbf{y}}$ used for the halfspace cut in Algorithms 5 and 7, the randomized ellipsoid method can also be applied for any weighted ℓ_p cost $A_p^{(\mathbf{c})}$ with $p > 1$.

For more general convex costs A, we still have that every C-cost ball is a convex set (i.e., the sublevel set of a convex function is a convex set; see Boyd and Vandenberghe 2004, Chapter 3) and thus has a separating hyperplane. Further, since for any $D > C$, $\mathcal{B}^C(A) \subset \mathcal{B}^D(A)$, the separating hyperplane of the D-cost ball is also a separating hyperplane of the C cost ball and can be re-used in our Algorithm 7. Thus, this procedure is applicable for any convex cost function, A, so long as we can compute the separating hyperplanes of any cost ball of A for any point $\mathbf{y}$ not in the cost ball.

For non-convex costs A such as weighted ℓ_p costs with $0 < p < 1$, minimization over a convex set $\mathcal{X}_f^-$ is generally hard. 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 efficient techniques that outperform the previous reverse-engineering approaches for linear classifiers in a continuous space. When the negative class is convex, we apply the randomized ellipsoid method introduced by Bertsimas and Vempala 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 for a number of special circumstances. Most importantly, we demonstrate that these algorithms can succeed without reverse engineering the classifier. Instead, these approaches 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 the family of ℓ_p costs and show that $\mathcal{F}^{convex}$ is only generally ε-*IMAC* searchable for all $\varepsilon > 0$ when $p = 1$. 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 the randomized ellipsoid method when the negative set is convex. 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 $\varepsilon > 0$. 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 present are invaluable tools not for an adversary to develop better attacks but rather for analysts to better understand the vulnerabilities of their filters: our framework provides the query complexity in the worst-case setting when an adversary can directly query the classifier. However, our analysis and algorithms do not completely answer the evasion problem and also generally can not be easily used by an adversary since there are several real-world obstacles that are not incorporated into our framework. Queries may only be partially observable or noisy, and the feature set may only be partially known. Most importantly, 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)? Can an adversary efficiently perform ε-*IMAC* search when his cost is defined in an alternate feature space to the classifier's? 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? These and other open problems for near-optimal evasion are discussed in Nelson et al. (2011).

Acknowledgments

We would like to thank Shing-hon Lau, Anthony Tran, Peter Bartlett, Marius Kloft, Peter Bodík, and the JMLR editor and reviewers for their helpful feedback on this project.

We gratefully acknowledge the support of our sponsors. This work was supported in part by TRUST (Team for Research in Ubiquitous Secure Technology), which receives support from the National Science Foundation (NSF award #CCF-0424422) and AFOSR (#FA9550-06-1-0244); RAD Lab, which receives support from California state MICRO grants (#06-148 and #07-012); DETER-lab (cyber-DEfense Technology Experimental Research laboratory), which receives support from DHS HSARPA (#022412) and AFOSR (#FA9550-07-1-0501); NSF award #DMS-0707060; the Siebel Scholars Foundation; and the following organizations: Amazon, BT, Cisco, DoCoMo USA Labs, EADS, ESCHER, Facebook, Google, HP, IBM, iCAST, Intel, Microsoft, NetApp, ORNL, Pirelli, Qualcomm, Sun, Symantec, TCS, Telecom Italia, United Technologies, and VMware. The opinions expressed in this paper are solely those of the authors and do not necessarily reflect the opinions of any funding agency, the State of California, or the U.S. government.

Appendix A. Query Complexity for K-STEP MULTILINESEARCH Algorithm

We consider the evasion problem as a game between *classifier* (playing first) and *adversary* (playing second) who wishes to evade detection by the classifier. To analyze the worst-case query complexity of K-STEP MULTILINESEARCH (Algorithm 3), we consider a *worst-case classifier* that seeks to maximize the number of queries submitted by the adversary. The worst-case classifier is completely aware of the state of the adversary; that is, the dimension of the space D, the adversary's goal L_ε, the cost function A, the bounds on the cost function C_t^+ and C_t^-, and so forth.

Proof of Theorem 4 At each iteration of Algorithm 3, the adversary chooses 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 worst-case 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 worst-case classifier responds, the candidate multiplicative gap (see Section 2.2) along $\mathbf{e}$ will shrink by an exponent of 2^{-K}; that is,

$$\begin{aligned} \frac{B^-}{B^+} &= \left(\frac{C^-}{C^+}\right)^{2^{-K}}, \\ \log(G'_{t+1}) &= \log(G_t) \cdot 2^{-K}. \end{aligned}$$

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

Case 1 ($t \in C_1$): Respond with '+' for all remaining directions. Here the bound 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 ($t \in C_2$): Choose at least one direction to respond with '−'. Here since only the value of C^- changes, the worst-case classifier can choose to respond to the first K queries so that the gap decreases by a negligible amount (by always responding with '+' during the first K queries along $\mathbf{e}$, the gap only decreases by an exponent of

$\left(1-2^{-K}\right)$). However, the worst-case classifier must choose 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. By this assumption, if $t \in \mathcal{C}_1$ we have $G_t = G_{t-1}^{2^{-K}}$ whereas if $t \in \mathcal{C}_2$ then $G_t = G_{t-1}$. By analyzing the gap before and after the final iteration T, it can be shown that

$$|\mathcal{C}_1| = \lceil L_\varepsilon / K \rceil \tag{8}$$

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

At every iteration in case one, the adversary makes exactly $K + |\mathcal{W}'_t| - 1$ queries where $\mathcal{W}'_t$ is the set of feasible directions remaining at the t^{th} iteration. While $\mathcal{W}'_t$ is controlled by the worst-case classifier, we can apply the bound $|\mathcal{W}'_t| \leq |\mathcal{W}|$. Using this and the relation from Equation (8), we can bound the number of queries, Q_1, used in case 1 by

$$\begin{aligned} Q_1 &\leq \sum_{t \in \mathcal{C}_1} (K + |\mathcal{W}| - 1) \\ &= \left\lceil \frac{L_\varepsilon}{K} \right\rceil \cdot (K + |\mathcal{W}| - 1) \\ &\leq \left(\frac{L_\varepsilon}{K} + 1 \right) \cdot K + \left\lceil \frac{L_\varepsilon}{K} \right\rceil \cdot (|\mathcal{W}| - 1) \\ &= L_\varepsilon + K + \left\lceil \frac{L_\varepsilon}{K} \right\rceil \cdot (|\mathcal{W}| - 1) \quad . \end{aligned}$$

For each case two iteration, we make exactly $K + E_t$ queries, and each eliminates $E_t \geq 1$ directions; hence, $|\mathcal{W}'_{t+1}| = |\mathcal{W}'_t| - E_t$. A worst-case classifier will always make $E_t = 1$ 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 and at least one direction must remain. Thus,

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

The total number of queries used by Algorithm 3 is then

$$\begin{aligned} Q = Q_1 + Q_2 &\leq L_\varepsilon + K + \left\lceil \frac{L_\varepsilon}{K} \right\rceil \cdot (|\mathcal{W}| - 1) + (|\mathcal{W}| - 1)(K + 1) \\ &= L_\varepsilon + \left\lceil \frac{L_\varepsilon}{K} \right\rceil \cdot |\mathcal{W}| + K \cdot |\mathcal{W}| + |\mathcal{W}| - \left\lceil \frac{L_\varepsilon}{K} \right\rceil - 1 \\ &\leq L_\varepsilon + \left(\left\lceil \frac{L_\varepsilon}{K} \right\rceil + K + 1 \right) |\mathcal{W}| \quad . \end{aligned}$$

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

$$Q \leq L_\varepsilon + \left(2\lceil\sqrt{L_\varepsilon}\rceil + 1\right)|\mathcal{W}| \ ,$$

establishing the result. ■

Appendix B. Proof of Lower Bound

Here we prove the lower bound from Section 3.1.2. Recall that D is the dimension of the space, $A : \Re^D \to \Re^+$ is any positive convex function, and $0 < C_0^+ < C_0^-$ are initial upper and lower bounds on the *MAC*. We also have that $\hat{\mathcal{F}}^{\text{convex,'+'}} \subset \mathcal{F}^{\text{convex,'+'}}$ is the set of classifiers consistent with the constraints on the *MAC*; that is, 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 $\mathcal{B}^{C_0^-}(A) \not\subset \mathcal{X}_f^+$. As above, we consider a worst-case classifier.

Proof of Theorem 5 Suppose a query-based algorithm submits $N < D+1$ membership queries $\mathbf{x}^1, \ldots, \mathbf{x}^N \in \Re^D$ 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:

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

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. $A\left(\mathbf{x} - \mathbf{x}^A\right) < C_0^-$ therefore $MAC(f, A) = C_0^-$, and any ε-optimal points $\mathbf{x}' \in \varepsilon\text{-}IMAC^{(*)}(f, A)$ must satisfy $C_0^- \leq A\left(\mathbf{x}' - \mathbf{x}^A\right) \leq (1+\varepsilon)C_0^-$.

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 remainder are negative. Let $\mathcal{G} = conv\left(\mathbf{x}^1, \ldots, \mathbf{x}^M\right)$; that is, the convex hull of the M positive queries. Now let $\mathcal{X}_g^+$ be the convex hull of $\mathcal{G}$ 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 $C_0^+ < C_0^-$, 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 $\Re^D$ and hence the interior of $\mathcal{G}$ is empty; that is, $int(\mathcal{G}) = \emptyset$. Hence, there is always some point from $\mathcal{B}^{C_0^+}(A)$ that is on the boundary of $\mathcal{X}_g^+$; that is, $\mathcal{B}^{C_0^+}(A) \not\subset int(\mathcal{G})$ 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, $MAC(g, A) = \inf_{\mathbf{x} \in \mathcal{X}_g^-}\left[A\left(\mathbf{x} - \mathbf{x}^A\right)\right] = C_0^+$. Since the accuracy $\varepsilon < \frac{C_0^-}{C_0^+} - 1$, any $\mathbf{x} \in \varepsilon\text{-}IMAC^{(*)}(g, A)$ must have

$$A\left(\mathbf{x} - \mathbf{x}^A\right) \leq (1+\varepsilon)C_0^+ < \frac{C_0^-}{C_0^+}C_0^+ = C_0^- \ ,$$

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

Suppose instead that a query-based algorithm submits $N < L_\varepsilon^{(*)}$ membership queries. 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. Here, the worst-case classifier f responds with

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

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 $G_t \geq \sqrt{G_{t-1}}$. These responses ensure the invariant $G_t \geq \sqrt{G_{t-1}}$ and since the algorithm can not terminate until $G_N \leq 1+\varepsilon$, we have $N \geq L_\varepsilon^{(*)}$ from Equation (4). Otherwise, there are still two convex-inducing classifiers with consistent query responses but with no common ε-*IMAC*. The first classifier's positive set is the smallest cost-ball enclosing all positive queries, while the second classifier's positive set is the largest cost-ball enclosing all positive queries but no negatives. The *MAC* values for these classifiers differ by more than a factor of $(1+\varepsilon)$ if $N < L_\varepsilon^{(*)}$, so they have no common ε-*IMAC*. ■

Appendix C. Proof of Theorem 10

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 12 *For any $0 < \delta \leq 1/2$ and $D \geq 1$, 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 $H(\delta) = -\delta \log_2 \delta - (1-\delta)\log_2(1-\delta)$ 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 following bound (see Flum and Grohe, 2006, Page 427)

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

to the denominator,[6] which is valid for any $0 < \delta \leq 1/2$. ■

6. Gottlieb et al. (2011) present a better entropy bound on this sum of binomial coefficients, but it is unnecessary for our result.

Lemma 13 *The minimum of the ℓ_p cost function A_p from the target $\mathbf{x}^A$ to the halfspace $\mathcal{H}_{\mathbf{w},\mathbf{b}} = \left\{\mathbf{x} \;\middle|\; \mathbf{x}^\top\mathbf{w} \geq \mathbf{b}^\top\mathbf{w}\right\}$ can be expressed in terms of the equivalent 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*

$$\min_{\mathbf{x}\in\mathcal{H}_{\mathbf{w},d}} A_p\left(\mathbf{x}-\mathbf{x}^A\right) = \begin{cases} d\cdot\|\mathbf{w}\|_{\frac{p}{p-1}}^{-1}, & \text{if } d>0 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

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

$$\min_{\mathbf{x}\in\mathcal{H}_{\mathbf{w},d}} A_\infty\left(\mathbf{x}-\mathbf{x}^A\right) = \begin{cases} d\cdot\|\mathbf{w}\|_1^{-1}, & \text{if } d>0 \\ 0, & \text{otherwise} \end{cases} . \tag{10}$$

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 Equation (9). 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

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

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 $A_p\left(\mathbf{x}^*-\mathbf{x}^A\right) = d\cdot\|\mathbf{w}\|_{\frac{p}{p-1}}^{-1}$, which is the first case of Equation (9).

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 Equation (10). 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 Equation (10). 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 the 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 coordinates 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, w_i. These constraints, along with the hyperplane constraint, lead to the following formula for an optimal solution: $x_i = d \cdot \text{sign}(w_i)\|\mathbf{w}\|_1^{-1}$ for all i. The ℓ_∞ cost of this solution is simply $d \cdot \|\mathbf{w}\|_1^{-1}$. ■

Finally, for the proof of Theorem 10, 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 its *canonical representation* which is a vector of D positive or negative ones; that is, 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:

$$x_i \in \begin{cases} [0, +\infty], & \text{if } a_i = +1 \\ [-\infty, 0], & \text{if } a_i = -1 \end{cases} .$$

Proof of Theorem 10 Suppose a query-based algorithm submits N membership queries $\mathbf{x}^1, \ldots, \mathbf{x}^N \in \Re^D$ 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

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

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. $A_p\left(\mathbf{x} - \mathbf{x}^A\right) < C_0^-$ therefore $MAC(f, A_p) = C_0^-$, and any ε-optimal points $\mathbf{x}' \in \varepsilon\text{-}IMAC^{(*)}(f, A_p)$ must satisfy $C_0^- \leq A_p\left(\mathbf{x}' - \mathbf{x}^A\right) \leq (1+\varepsilon)C_0^-$.

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 within relative to the center, $\mathbf{x}^A$ (a data point may lie within more than one orthant but, to cover it, we need only have one orthant that contains 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 $\mathcal{X}_f^+$ consistent with all the query responses of f; that is, each of the M positive queries are in $\mathcal{X}_g^+$ and all the negative queries are in $\mathcal{X}_g^-$. Moreover, $\mathcal{G}$ contains 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 one. 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 a $K-1$ 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 positive 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

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

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

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

Since all the terms in the second summation are non-positive, the second sum is at most 0. Thus, by maximizing the first summation, we upper bound $s(\mathbf{x})$. The summation $\sum_{\{i \mid a_i = +1\}} x_i$ (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 $p = \infty$; that is, it is upper bounded by d. Thus, we have that $s(\mathbf{x}) \leq 0$, and this hyperplane separates 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. This hyperplane also separates the scaled vertex from $\mathcal{G}$ by the properties of the convex hull. Since the displacement d defined above is greater than 0, by applying Lemma 13, 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

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

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

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

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

For the case $1 < p < \infty$, by Lemma 12, there must be

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

vertices of the hypercube in the covering to achieve any accuracy $0 < \varepsilon < 2^{\frac{p-1}{p}} - 1$, for which $\delta = \frac{(1+\varepsilon)^{\frac{p}{p-1}}-1}{(1+\varepsilon)^{\frac{p}{p-1}}} < \frac{1}{2}$ as required by the lemma. Moreover, since $H(\delta) < 1$ for $\delta < \frac{1}{2}$,

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

and we have $M \geq \alpha_{p,\varepsilon}^D$.

Similarly for $p = \infty$, Lemma 12 can be applied yielding $M \geq 2^{D\left(1-H\left(\frac{\varepsilon}{1+\varepsilon}\right)\right)}$ to achieve any desired accuracy $0 < \varepsilon < 1$ (for which $\varepsilon/(1+\varepsilon) < 1/2$ as required by the Lemma). Again, by the properties of entropy, the constant $\alpha_{\infty,\varepsilon} = 2^{\left(1-H\left(\frac{\varepsilon}{1+\varepsilon}\right)\right)} > 1$ for $0 < \varepsilon < 1$ and we have $M \geq \alpha_{\infty,\varepsilon}^D$. ■

Appendix D. Proof of Theorem 11

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 outward region formed by the intersection of a hypersphere and a halfspace as depicted in Figure 9. We parameterize the caps by the hypersphere's radius R and the half-angle ϕ about a central radius (through the caps's peak) as in the right-most diagram of Figure 9.

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

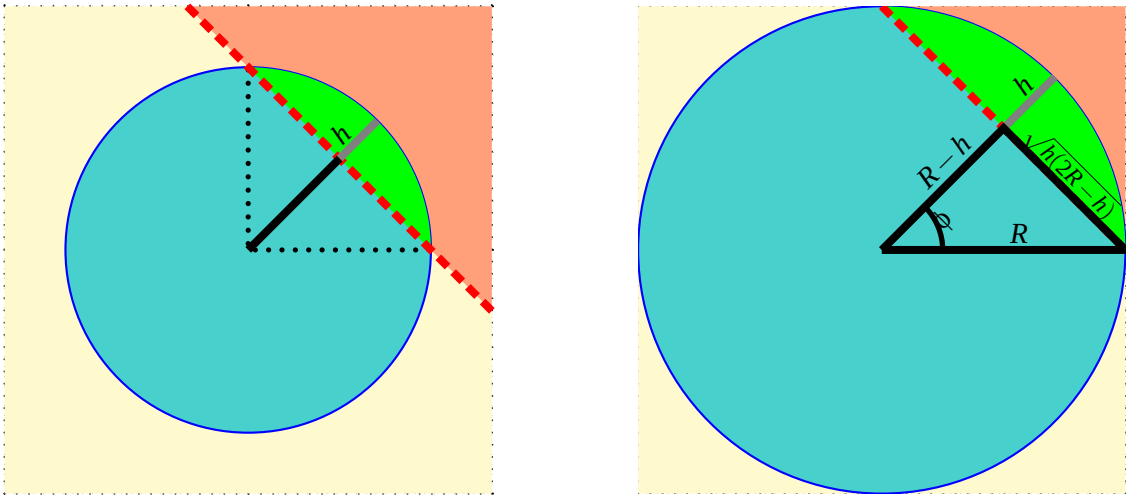
Lemma 14 ***(Result based on Wyner 1965)*** *Covering the surface of D-dimensional hypersphere of radius R requires at least*

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

spherical caps of half-angle $\phi \in (0, \frac{\pi}{2})$.

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 of radius R is given by

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



(a) (b)

Figure 9: This figure depicts the geometry of spherical caps. (a) A spherical cap of height h, which is created by a plane passing through the sphere. The green region represents the area of the cap. (b) The geometry of the spherical cap; the intersecting halfspace forms a right triangle with the centroid of the hypersphere. The length of the side of this triangle adjacent to the centroid is $R-h$, its hypotenuse has length R, and the side opposite the centroid has length $\sqrt{h(2R-h)}$. The half angle ϕ, given by $\sin(\phi) = \frac{\sqrt{h(2R-h)}}{R}$, of the right circular cone is used to parameterize the cap.

where $\Gamma(x)$ is the usual gamma function. This result follows directly from computing the surface area of the hypersphere and that of each spherical cap.

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; compare with 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 becomes

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

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 11 Suppose a query-based algorithm submits N membership queries $\mathbf{x}^1,\ldots,\mathbf{x}^N \in \Re^D$ 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

$$f(\mathbf{x}) = \begin{cases} +1, & \text{if } A_2\left(\mathbf{x}-\mathbf{x}^A\right) < C_0^- \\ -1, & \text{otherwise} \end{cases} .$$

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, $\nexists\, \mathbf{x} \in \mathcal{X}_f^-$ such that $A_2\left(\mathbf{x}-\mathbf{x}^A\right) < C_0^-$. Therefore, $MAC(f,A_2) = C_0^-$, and any ε-optimal points $\mathbf{x}' \in \varepsilon\text{-}IMAC^{(*)}(f,A_2)$ must satisfy $C_0^- \le A_2\left(\mathbf{x}'-\mathbf{x}^A\right) \le (1+\varepsilon)C_0^-$.

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 $M \le N$ queries are positive and the remaining are negative. Let $\mathcal{G} = conv\left(\mathbf{x}^1,\ldots,\mathbf{x}^M\right)$ be the convex hull of these M positive queries. We will assume $\mathbf{x}^A \in \mathcal{G}$, since otherwise, we construct the set $\mathcal{X}_g^+$ as in the proof for Theorem 5 above and achieve $MAC(f,A_2) = C_0^+$ thereby achieving our desired result. Now consider the projection of each of the positive queries onto the surface of the ℓ_2 ball $\mathcal{B}^{C_0^-}(A_2)$, given by the points $\mathbf{z}^i = C_0^- \frac{\mathbf{x}^i}{A_2(\mathbf{x}^i-\mathbf{x}^A)}$. 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 $\mathcal{G} \subset conv\left(\mathbf{z}^1,\mathbf{z}^2,\ldots,\mathbf{z}^M\right)$—we will call this enlarged hull $\hat{\mathcal{G}}$. These M projected points $\left\{\mathbf{z}^i\right\}$ must form a covering of the C_0^--hypersphere as the locii of caps of half-angle $\phi_\varepsilon^* = \arccos\left((1+\varepsilon)^{-1}\right)$. If not, then there exists some point on the surface of this hypersphere that is at least an angle ϕ_ε^* from all $\mathbf{z}^i$ points and the resulting ϕ_ε^*-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_ε^*-cap, it achieves a minimal ℓ_2 cost of $C_0^- \cos\phi_\varepsilon^*$. Thus, if we fail to achieve a ϕ_ε^*-covering of the C_0^--hypersphere, the alternative classifier g has $MAC(g,A_2) < C_0^- \cos\phi_\varepsilon^* = C_0^-/(1+\varepsilon)$ and any $\mathbf{x} \in \varepsilon\text{-}IMAC^{(*)}(g,A_2)$ must have

$$A_2\left(\mathbf{x}-\mathbf{x}^A\right) \le (1+\varepsilon)MAC < (1+\varepsilon)\frac{C_0^-}{1+\varepsilon} = C_0^- ,$$

whereas any $\mathbf{y} \in \varepsilon\text{-}IMAC^{(*)}(f,A)$ must have cost $A\left(\mathbf{y}-\mathbf{x}^A\right) \ge C_0^-$. Thus, we would have $\varepsilon\text{-}IMAC^{(*)}(f,A) \cap \varepsilon\text{-}IMAC^{(*)}(g,A) = \emptyset$ and would fail to achieve ε-multiplicative optimality. Hence, we have shown that an ϕ_ε^*-covering is necessary for ε-multiplicative optimality. Moreover, from our definition of ϕ_ε^*, for any $\varepsilon \in (0,\infty)$, $\phi_\varepsilon^* \in (0,\frac{\pi}{2})$ and thus, Lemma 14 is applicable for all ε. From Lemma 14, to have an ϕ_ε^*-covering we must have

$$M \ge \left(\frac{1}{\sin\phi_\varepsilon^*}\right)^{D-2}$$

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

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

■

References

Dana Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

Keith Ball. An elementary introduction to modern convex geometry. In *Flavors of Geometry*, volume 31 of *MSRI Publications*, pages 1–58. Cambridge 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.

Richard P. Brent. *Algorithms for Minimization without Derivatives*. Prentice-Hall, 1973.

Michael Brückner and Tobias Scheffer. Nash equilibria of static prediction games. In *Advances in Neural Information Processing Systems (NIPS)*, volume 22, pages 171–179. 2009.

Richard L. Burden and J. Douglas Faires. *Numerical Analysis*. Brooks Cole, 7th edition, 2000.

Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *Proceedings of the* 10^{th} *International Conference on Knowledge Discovery and Data Mining*, pages 99–108, 2004.

Sanjoy Dasgupta, Adam Tauman Kalai, and Claire Monteleoni. Analysis of perceptron-based active learning. *Journal of Machine Learning Research*, 10:281–299, 2009.

Vitaly Feldman. On the power of membership queries in agnostic learning. *Journal of Machine Learning Research*, 10:163–182, 2009.

Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Secaucus, NJ, USA, 2006.

Lee-Ad Gottlieb, Aryeh Kontorovich, and Elchanan Mossel. VC bounds on the cardinality of nearly orthogonal function classes. Technical Report arXiv:1007.4915v2 [math.CO], arXiv, 2011.

Donald R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001.

Donald R. Jones, Cary D. Perttunen, and Bruce E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal Optimization Theory and Application*, 79(1):157–181, 1993.

Murat Kantarcioglu, Bowei Xi, and Chris Clifton. Classifier evaluation and attribute selection against active adversaries. Technical Report 09-01, Purdue University, 2009.

Tamara G. Kolda, Robert Michael Lewis, and Virginia Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45(3):385–482, 2003.

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

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

László Lovász and Santosh Vempala. Hit-and-run from a corner. In *Proceedings of the* 36^{th} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 310–314, 2004.

Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the* 11^{th} *International Conference on Knowledge Discovery in Data Mining*, pages 641–647, 2005.

John A. Nelder and Roger Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

Blaine Nelson, Benjamin I. P. Rubinstein, Ling Huang, Anthony D. Joseph, Shing hon Lau, Steven Lee, Satish Rao, Anthony Tran, and J. D. Tygar. Near-optimal evasion of convex-inducing classifiers. In *Proceedings of the* 13^{th} *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 549–556, 2010a.

Blaine Nelson, Benjamin I. P. Rubinstein, Ling Huang, Anthony D. Joseph, Steven Lee, Satish Rao, and J. D. Tygar. Query strategies for evading convex-inducing classifiers. Technical Report arXiv:1007.0484v1 [cs.LG], arXiv, 2010b.

Blaine Nelson, Benjamin I. P. Rubinstein, Ling Huang, Anthony D. Joseph, and J. D. Tygar. Classifier evasion: Models and open problems. In *Privacy and Security Issues in Data Mining and Machine Learning*, volume 6549 of *Lecture Notes in Computer Science*, pages 92–98. 2011.

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

Greg Schohn and David Cohn. Less is more: Active learning with support vector machines. In *Proceedings of the* 17^{th} *International Conference on Machine Learning (ICML)*, pages 839–846, 2000.

Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

Robert L. Smith. The hit-and-run sampler: A globally reaching Markov chain sampler for generating arbitrary multivariate distributions. In *Proceedings of the* 28^{th} *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* 5^{th} *International Conference on Recent Advances in Intrusion Detection (RAID)*, volume 2516 of *Lecture Notes in Computer Science*, pages 54–73, 2002.

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

Ke Wang and Salvatore J. Stolfo. Anomalous payload-based network intrusion detection. In *Proceedings of the* 7^{th} *International Conference on Recent Advances in Intrusion Detection (RAID)*, volume 3224 of *Lecture Notes in Computer Science*, pages 203–222, 2004.

Ke Wang, Janak J. Parekh, and Salvatore J. Stolfo. Anagram: A content anomaly detector resistant to mimicry attack. In *Proceedings of the* 9^{th} *International Conference on Recent Advances in Intrusion Detection (RAID)*, volume 4219 of *Lecture Notes in Computer Science*, pages 226–248, 2006.

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