On Robustness to Adversarial Examples and Polynomial Optimization

Pranjal Awasthi
Department of Computer Science
Rutgers University
pranjal.awasthi@rutgers.edu

Abhratanu Dutta
Department of Computer Science
Northwestern University
abhratanudutta2020@u.northwestern.edu

Aravindan Vijayaraghavan
Department of Computer Science
Northwestern University
aravindv@northwestern.edu

Abstract

We study the design of computationally efficient algorithms with provable guarantees, that are robust to adversarial (test time) perturbations. While there has been an explosion of recent work on this topic due to its connections to test time robustness of deep networks, there is limited theoretical understanding of several basic questions like *(i) when and how can one design provably robust learning algorithms? (ii) what is the price of achieving robustness to adversarial examples in a computationally efficient manner?*
The main contribution of this work is to exhibit a strong connection between achieving robustness to adversarial examples, and a rich class of polynomial optimization problems, thereby making progress on the above questions. In particular, we leverage this connection to (a) design computationally efficient robust algorithms with provable guarantees for a large class of hypothesis, namely linear classifiers and degree-2 polynomial threshold functions (PTFs), (b) give a precise characterization of the price of achieving robustness in a computationally efficient manner for these classes, (c) design efficient algorithms to certify robustness and generate adversarial attacks in a principled manner for 2-layer neural networks. We empirically demonstrate the effectiveness of these attacks on real data.

1 Introduction

The empirical success of deep learning has led to numerous unexplained phenomena about which our current theoretical understanding is limited. Examples include the ability of complex models to generalize well and effectiveness of first order methods on optimizing training loss. The focus of this paper is on the phenomenon of *adversarial robustness*, that was first pointed out by Szegedy et al. [33]. On many benchmark data sets, deep networks optimized on the training set can often be fooled into misclassifying a test example by making a small adversarial perturbation that is imperceptible to a human labeler. This has led to a proliferation of work on designing robust algorithms that defend against such adversarial perturbations, as well as attacks that aim to break these defenses.

In this work we choose to focus on *perturbation defense*, the most widely studied formulation of adversarial robustness [24]. In the perturbation defense model, given a classifier f, an adversary can take a test example x generated from the data distribution and perturb it to $\tilde{x}$

such that $\|x - \tilde{x}\| \leq \delta$. Here δ characterizes the amount of power the adversary has and the distance is typically measured in the ℓ_∞ norm (other norms that have been studied include the ℓ_2 norm). Given a loss function $\ell(\cdot)$, the goal is to optimize the *robust loss* defined as

$$\mathbb{E}_{(x,y)\sim D}\Big[\max_{\tilde{x}:\|x-\tilde{x}\|_\infty \leq \delta} \ell(f(\tilde{x}), y)\Big].$$

One would expect that when δ is small the label y of an example does not change, thereby motivating the robust loss objective. Despite a recent surge in efforts to theoretically understand adversarial robustness [36, 37, 38, 21, 30, 13, 4, 10, 16, 34, 25, 26, 11], several central questions remain open. *How can one design provable polynomial time algorithms that are robust to adversarial perturbations? Given a classifier and a test input, how can one provably construct an adversarial example in polynomial time or certify that none exists? What computational barriers exist when designing adversarially robust learning algorithms?*

In this work we identify and study a natural class of *polynomial optimization* problems that are intimately connected to adversarial robustness, and help us shed new light on all three of the above questions simultaneously! As a result we obtain the first polynomial time learning algorithms for a large class of functions that are optimally robust to adversarial perturbations. Furthermore, we also provide nearly matching computational intractability results that, together with our upper bounds give a sharp characterization of the price of achieving adversarial robustness in a computationally efficient manner. We now summarize our main results.

Our Contributions Polynomial optimization and Adversarial Robustness. We identify a natural class of polynomial optimization problems that provide a common and principled framework for studying various aspects of adversarial robustness. These problems are also closely related to a rich class of well-studied problems that include the Grothendiëck problem and its generalizations [2, 9, 1, 22]. Given a classifier of the form $sgn(g(x))$ with $g : \mathbb{R}^n \to \mathbb{R}$, input x, and budget $\delta > 0$, the optimization problem is

$$\max_{z \in \mathbb{R}^n : \|z\|_\infty \leq \delta} g(x + z).$$

Usually, such problems are NP-hard and one relaxes them to find a $\hat{z}$ such that $g(x + \hat{z})$ comes as close to $g(x + z^*)$ in the objective value, where z^* is the optimal solution. We instead require the algorithm to output a $\hat{z}$ such that $g(x + \hat{z}) \geq g(x + z^*)$ at the cost of violating the ℓ_∞ constraint by a factor $\gamma \geq 1$. An efficient algorithm for producing such a $\hat{z}$ leads to an adversarial attack (in the relaxed ℓ_∞ neighborhood of radius $\gamma\delta$) when an adversarial example exists. On the other hand, if the algorithm produces no $\widehat{z}$, then this guarantees that there is no adversarial example within the ℓ_∞ neighborhood of radius δ. We then design such algorithms based on convex programming relaxations to get the *first* provable polynomial time adversarial attacks when the given classifier is a degree-1 or a degree-2 polynomial threshold function (PTF).

Algorithms for Learning Adversarially Robust Classifiers. Next we use the algorithm for finding adversarial examples to design polynomial time algorithms for learning robust classifiers for the class of degree-1 and degree-2 polynomial threshold functions (PTFs). To incorporate robustness we introduce a parameter γ, that helps clarify the tradeoff when computational efficiency is desired. We focus on the 0/1 error and say that a class $\mathcal{F}$ of PTFs of VC dimension Δ is γ-approximately robustly learnable if there exists a (randomized) polynomial time algorithm that, for any given $\varepsilon, \delta > 0$, takes as input poly$(\Delta, \frac{1}{\varepsilon})$ examples generated from a distribution and labeled by a function in $\mathcal{F}$ that has zero δ-robust error (realizable case), outputs a classifier from $\mathcal{F}$ that has (δ/γ)-robust error upper bounded by ε. See Section 2 for the formal definition. We design polynomial time algorithms for degree-1 and degree-2 PTFs with $\gamma = 1$ and $\gamma = O(\sqrt{\log n})$ respectively. Our next result that we discuss below a nearly matching lower bound. Together this gives nearly optimal approximately robust polynomial time algorithms for learning PTFs of degree at most 2.

Computational Hardness. While our algorithm for degree-1 PTFs is optimal, i.e., has $\gamma = 1$, for degree-2 and higher PTFs, we show that one indeed has to pay a price for computational robustness. We establish this by proving that robust learning of degree-2 PTFs is computationally hard for $\gamma = o(\log^c n)$, for some constant $c > 0$ (see Section 5 for formal statements). This is in sharp contrast to the non-robust setting ($\delta = 0$), where there exist polynomial time algorithms for constant degree PTFs (in the literature this is referred to as proper PAC learning in the realizable setting). More importantly, our lower bound again leverages the connection to polynomial optimization and in fact shows that robust learning of degree-2 PTFs for $\gamma = o(\sqrt{\eta_{approx}})$ is NP-hard where η_{approx} is *precisely* the hardness of approximation factor of a well-studied combinatorial optimization problem called *Quadratic Programming.* Hence, any significant improvement in the approximation factor in our upper bound is unlikely. While our hardness result applies to algorithms that output a classifier of low error, we also prove a more robust hardness result showing that for learning degree-2 and higher PTFs without any loss in the robustness parameter, i.e, $\gamma = 1$, it is computationally hard to even find a classifier of any constant error in the range $(0, \frac{1}{4})$.

Application to Neural Networks. Finally, we show that the connection to polynomial optimization also leads to new algorithms for generating adversarial attacks on neural networks. We focus on 2-layer neural networks with ReLU activations. We show that given a network and a test input, the problem of finding an adversarial example can also be phrased as an optimization problem of the kind studied for PTFs. We design a semi-definite programming (SDP) based polynomial time algorithm to generate an adversarial attack for such networks and compare our attack to the state-of-the-art attack of Madry et al. [24] on the MNIST data set.

Comparison to Related Work. Among the several recent and concurrent works on this topic, the most relevant to our result is the work of Bubeck et al. [7, 8] that studies computational complexity of robust learning. We defer other related work to Section C. In the rest of the paper, we define our model formally and give an overview of our techniques in Section 2. We then describe the connection to polynomial optimization in Section 3 and use it to design robust learning algorithms in Section 4, and derive computational intractability results in Section 5. In Section 6, we design adversarial attacks for 2 layer neural networks, followed by conclusions in Section 7.

2 Model and Preliminaries

We focus on binary classification, and adversarial perturbations are measured in ℓ_∞ norm. For a vector $x \in \mathbb{R}^n$, we have $\|x\|_\infty = \max_i |x_i|$. We study robust learning of *polynomial threshold functions* (PTFs). These are functions of the form $sgn(p(x))$, where $p(x)$ is a polynomial in n variables over the reals. Here $sgn(t)$ equals $+1$, if $t \geq 0$ and -1 otherwise. Given $y, y' \in \{-1, 1\}$, we study the 0/1 loss defined as $\ell(y, y') = 1$ if $y \neq y'$ and 0 otherwise. Given a binary classifier $\text{sgn}(g(x))$, an input x^*, and a budget $\delta > 0$, we say that $x^* + z$ is an *adversarial example* (for input x^*) if $\text{sgn}(g(x^* + z)) \neq \text{sgn}(g(x^*))$ and that $\|z\|_\infty \leq \delta$. One could similarly define the notion of adversarial examples for other norms. For a classifier with multiple outputs, we say that $x^* + z$ is an adversarial example iff the largest co-ordinate of $g(x^* + z)$ differs from the largest co-ordinate of $g(x^*)$. We now define the notion of robust error of a classifier.

Definition 2.1 (δ-robust error)**.** Let $f(x)$ be a Boolean function mapping $\mathbb{R}^n$ to $\{-1, 1\}$. Let D be a distribution over $\mathbb{R}^n \times \{-1, 1\}$. Given $\delta > 0$, we define the δ-robust error of f with respect to D as $err_{\delta,D}(f) = \mathbb{E}_{(x,y)\sim D}\big[\sup_{z \in B^n_\infty(0,\delta)} \ell(f(x + z), y)\big]$. Here $B^n_\infty(0, \delta)$ denotes the ℓ_∞ ball of radius δ, i.e., $B^n_\infty(0, \delta) = \{x \in \mathbb{R}^n : \|x\|_\infty \leq \delta\}$.

Analogous to empirical error in PAC learning, we denote $\hat{err}_{\delta,S}(f)$ to be the δ-robust empirical error of f, i.e., the robust error computed on the given sample S. To bound generalization gap, we will use the notion of adversarial VC dimension as introduced in [10] (See Appendix A). Next we define robust learning for PTFs.

Definition 2.2 (γ-approximately robust learning)**.** Let $\mathcal{F}$ be the class of degree-d PTFs from $\mathbb{R}^n \mapsto \{-1, 1\}$ of VC dimension $\Delta = O(n^d)$. For $\gamma \geq 1$, an algorithm $\mathcal{A}$ γ-approximately robustly learns $\mathcal{F}$ if the following holds for any $\varepsilon, \delta, \eta > 0$: Given $m = \text{poly}(\Delta, \frac{1}{\varepsilon}, \frac{1}{\eta})$ samples from a distribution D over $\mathbb{R}^n \times \{-1, 1\}$, if $\mathcal{F}$ contains a function f^* such that $err_{\delta,D}(f^*) = 0$, then with probability at least $1 - \eta$, $\mathcal{A}$ runs in time polynomial in m and outputs $f \in \mathcal{F}$ such that $err_{\delta/\gamma,D}(f) \leq \varepsilon$. If $\mathcal{F}$ admits such an algorithm then we say that $\mathcal{F}$ is γ-approximately robustly learnable. Here γ quantifies the price of achieving computationally efficient robust learning, with $\gamma = 1$ implying optimal learnability.

A Note about the Model and the Realizability Assumption Our definition of an adversarial example requires that $\text{sgn}(g(x^* + z)) \neq \text{sgn}(g(x^*))$, whereas for robust learning we require a classifier that satisfies $\text{sgn}(g(x^* + z)) \neq y$, where y is the given label of x^*. This might create two sources of confusion to the reader: a) In general the two requirements might be incompatible, and b) It might happen that initially $\text{sgn}(g(x^*))$ predicts the true label incorrectly but there is a perturbation z such that $\text{sgn}(g(x^* + z))$ predicts the true label correctly. In this case one should not count z as an adversarial example. To address (a) we would like to stress that all our guarantees hold under the realizability assumption, i.e., we assume that there is true function c^* such that for all examples x in the support of the distribution and all perturbations of magnitude upto δ, $\text{sgn}(c^*(x^* + z)) = \text{sgn}(c^*(x^*))$. Hence, there will indeed be a target concept for which no adversarial example exists and as a result will have zero robust error. To address (b) we would like to point out that in Section 4 where we use the subroutine for finding adversarial examples to learn a good classifier $\text{sgn}(g)$, we always enforce the constraint that on the training set $\text{sgn}(g(x^*)) = \text{sgn}(c^*(x^*))$ and g is as robust as possible. Hence when we find an adversarial example for a point x^* in our training set, it will indeed satisfy that $\text{sgn}(g(x^* + z)) \neq \text{sgn}(c^*(x))$ and correctly penalize g for the mistake. More generally, we could also define an adversarial example as one where given pair (x^*, y) the goal is to find a z such that $\text{sgn}(g(x^*+z)) \neq y$. All of our guarantees from Section 3 apply to this definition as well. Finally, in the non-realizable case, the distinction between defining adversarial robustness as either $\text{sgn}(g(x^* + z)) \neq \text{sgn}(g(x^*))$, or $\text{sgn}(g(x^* + z)) \neq y$, or even $\text{sgn}(g(x^* + z)) \neq \text{sgn}(c^*(x))$ matters and has different computational and statistical implications [11, 18]. Understanding when one can achieve computationally efficient robust learning in the non-realizable case is an important direction for future work.

The definition of γ-approximately robustly learnability has the realizability assumption built into it. So, when we prove that a class $\mathcal{F}$ is γ-approximately robustly learnable, we find an approximate robust learner from $\mathcal{F}$ under the realizability assumption on $\mathcal{F}$ i.e. for a set of points from the distribution, the algorithm guarantees to return an approximate robust learner only if there exists a perfect robust learner in the class $\mathcal{F}$ of learners.

3 Finding Adversarial Examples using Polynomial Optimization

In this section we introduce the broad class of polynomial optimization problems which are useful in designing algorithms with provable guarantees for generating adversarial examples for large classes like PTFs, and will later be useful for two layer neural networks in Section 6. These polynomial optimization problems are generalizations of well-studied combinatorial optimization problems like the *Grothëndieck problem* and computing operator norms of matrices [19, 2, 9]. We then design algorithms with provable guarantees for some of these classes. The following simple proposition illustrates the connection.

Proposition 3.1. *Let $\gamma \geq 1$. There is an efficient algorithm that given a classifier $sgn(f(x))$ and a point x^*, guarantees to either (a) find an adversarial example in $B_\infty^n(x^*, \gamma\delta)$, or (b) certify the absence of any adversarial example in $B_\infty^n(x^*, \delta)$, given an efficient algorithm that given x and a polynomial $g(z) \in \{ f(x^* + z), -f(x^* + z) \}$ finds a $\widehat{z}$ such that $g(\widehat{z}) \geq \max_{\|z\|_\infty \leq \delta} g(z)$ with $\|\widehat{z}\|_\infty \leq \gamma\delta$.*

When the classifier is a degree-d PTF of the form $sgn(f)$, we get the following problem: given as input a degree d polynomial g (potentially different from f), and any $\eta, \delta > 0$, find in time $\text{poly}(n, \log(\frac{1}{\eta}))$ and with probability at least $1 - \eta$, outputs a point $\hat{x}$ s.t.

$$g(\hat{x}) \geq \max_{x \in B_\infty^n(0,\delta)} g(x) \text{ and } \widehat{x} \in B_\infty^n(0, \gamma\delta). \quad (1)$$

This is closely related to the standard approximation variant of polynomial maximization problem where the goal is to obtain, in polynomial time, an objective value as close to the optimal one, without violating the B_∞^n ball constraint. Instead, our problem asks for the same objective value at the cost of an increase in the radius of the optimization ball (this is

```
1. Given $(A, b, c)$ that defines the polynomial $g(z) := z^T Az + b^T z + c$.
2. Solve the SDP given by following vector program:
   max $\sum_{i,j} A_{ij}\langle u_i, u_j\rangle + \sum_i b_i\langle u_i, u_0\rangle + c$ subject to $\|u_i\|_2^2 \le \delta^2\ \forall i \in [n]$, $\|u_0\|_2^2 = 1$.
3. Let $u_i^\perp$ represent the component of $u_i$ orthogonal to $u_0$. Draw $\zeta \sim N(0, I)$ a
   standard Gaussian vector, and set $\widehat{z}_i := \langle u_i, u_0\rangle + \langle u_i^\perp, \zeta\rangle$ for each $i \in \{0, 1, \dots, n\}$.
4. Repeat rounding $O(\log(1/\eta))$ random choices of $\zeta$ and pick the best choice.
```

Figure 1: The SDP-based algorithm for the degree-2 optimization problem.

sometimes called a $(1, \gamma)$-bicriteria approximation). This changes the flavor of the problem, and introduces new challenges particularly when the polynomial g is non-homogenous. We begin with the following simple claim.

Claim 3.2. *There is a deterministic linear-time algorithm that given any linear threshold function $sgn(b^T x + c)$, a point x^* and $\delta > 0$, provably finds an adversarial example ℓ_∞ ball of δ around x^* when it exists.*

In Section 4, this will be used to give robust learning algorithms for linear classifiers. Our main result of this section is a provable algorithm for degree-2 PTFs.

Theorem 3.3. *For any $\delta, \eta > 0$, there is a polynomial time algorithm that given a degree-2 PTF $sgn(f(x))$ and a example $(x^*, sgn(f(x^*)))$, guarantees at least one of the following holds with probability at least $(1 - \eta)$: (a) finds an adversarial example $(x^* + \widehat{z})$ i.e., $sgn(f(x^*)) \ne sgn(f(x^* + \widehat{z}))$, with $\|\widehat{z}\|_\infty \le C\delta\sqrt{\log n}$, or (b) certifies that $\forall z : \|z\|_\infty \le \delta$, $sgn(f(x^*)) = sgn(f(x^* + z))$ for some constant $C > 0$.*

To establish the above theorem using Proposition 3.1, we need to design a polynomial time algorithm that given any degree-2 polynomial $g(x) = x^T Ax + b^T x + c$ with $A \in \mathbb{R}^{n\times n}, b \in \mathbb{R}^n, c \in \mathbb{R}$, finds a solution $\widehat{x}$ with $\|\widehat{x}\|_\infty \le O(\sqrt{\log n}) \cdot \delta$ such that $g(\widehat{x}) \ge \max_{\|x\|_\infty \le \delta} g(x)$. We design such an algorithm via an semi-definite programming (SDP) based approach that is directly inspired by the algorithm for quadratic programming (QP) by [27, 9]. However, further complications arise due to non-homogeneity, and as our goal is to preserve the objective function while potentially relaxing the constraint. We defer to the appendix for a detailed discussion. In Fig. 1 we describe the SDP that we use and the corresponding rounding algorithm to solve the optimization problem. The vector program given in step 2 of Algorithm 1 is an SDP where the variables are $X_{ij} = \langle u_i, u_j\rangle$, and can be solved in polynomial time up to any additive error (using the Ellipsoid algorithm). We defer the the details Appendix D.

4 From Adversarial Examples to Robust Learning Algorithms

In this section we show how to leverage algorithms for finding adversarial examples to design polynomial time robust learning algorithms for degree-1 and degree-2 PTFs. We obtain our upper bounds by establishing a general algorithmic framework that relates robust learnability of PTFs to the polynomial maximization problem studied in Section 3.

Definition 4.1 (γ-factor admissibility)**.** For $\gamma \ge 1$, we say that a class $\mathcal{F}$ of PTFs is γ-factor admissible if $\mathcal{F}$ has the following properties:
(1) For any $a, b, c \in \mathbb{R}$ s.t. $\text{sgn}(f(x)), \text{sgn}(g(x)) \in \mathcal{F}$, we have $\text{sgn}(af(x) + bg(x) + c) \in \mathcal{F}$. Further for any $r \in \mathbb{R}^n$, we have $sgn(g(x + r)) \in \mathcal{F}$.
(2) There is a polynomial time algorithm that solves the optimization problem of maximizing $g(x + z)$ around any point x, i.e., given a $g \in \mathcal{F}$, an x and $\delta > 0$, the algorithm outputs a $\hat{z}$ such that $g(x + \hat{z}) \ge \max_{z\in B_\infty(0,\delta)} g(x + z)$and $\|z\|_\infty \le \gamma\delta$.

The first two conditions above are natural and are satisfied by many classes of PTFs. The third condition in the above definition concerns the optimization problem studied in Section 3. The main result of this section, stated below, is the claim that any admissible class of PTFs is also robustly learnable in polynomial time.

Theorem 4.2. *Let $\mathcal{F}$ be a class of PTFs that is γ-factor admissible for $\gamma \ge 1$. Then $\mathcal{F}$ is γ-approximate robustly learnable.*

1. Let $S = (x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$ be the given training set.
2. Find a degree-d polynomial g with $sgn(g(x)) \in \mathcal{F}$ that satisfies:
   $\forall i \in [m], \quad \sup_{z \in B^n_\infty(0,\delta)} (-y_i) g(x_i + z) < 0.$

Figure 2: Convex program to find a PTF $sgn(g(x)) \in \mathcal{F}$ with zero robust empirical error.

To learn a $g \in \mathcal{F}$ we formulate robust empirical risk minimization as a convex program, shown in Figure 2. Here we use the fact that the value of any polynomial g of degree d at a given point x can be expressed as the inner product between the co-efficient vector of g and an appropriate vector $\psi(x) \in \mathbb{R}^D$ where $D = \binom{n+d-1}{d}$. It is easy to see that the constraints in the program above are linear in the coefficients of g. Furthermore, checking the validity of each constraint is really asking to check the robustness of g at a given point (x_i, y_i), which is an NP-hard problem [9]. Instead, we will use the fact that $\mathcal{F}$ is γ-factor admissible to design an approximate separation oracle for the type of constraints enforced in the program. Below we give a proof sketch of Theorem 4.2 and defer the full proof to Appendix E.

Proof Sketch of Theorem 4.2. Let $\mathcal{B}$ be an algorithm that achieves the γ-factor admissibility for the class $\mathcal{F}$. Given S, we will run the Ellipsoid algorithm on the convex program in Figure 2. In each iteration, for each $i \in [m]$, we run $\mathcal{B}$ on the polynomial $y_i g(x_i + z)$, where z is the variable and x_i is fixed to be the ith data point. From the guarantee of $\mathcal{B}$ we get that if there exists an i and z with $\|z\|_\infty \le \delta/\gamma$, such that $(-y_i)g(x_i + z) > 0$, then with high probability, $\mathcal{B}$ will output a violated constraint of the convex program, i.e., an index $i \in [m]$ and $\hat{z} \in B^n_\infty(0, \delta)$ such that $(-y_i)g(x_i + \hat{z}) > 0$. This gives us a separating hyperplane of the form $sgn(-y_i g(x_i + \hat{z}))$, and the algorithm continues. This means that when the algorithm terminates, we would have the empirical robust error $e\hat{r}r_{\delta/\gamma,S}(sgn(g)) = 0$. Using the uniform convergence bound from Lemma A.1, this would imply that $err_{\delta/\gamma,D}(sgn(g)) \le \varepsilon$. □

As a result we get the following corollaries about linear classifier and degree-2 PTFs. The proof for linear classifiers just follows from Claim 3.2, and Theorem 3.3 immediately implies the result for degree-2 PTFs.

Corollary 4.3. *The class of linear classifiers is optimally robustly learnable. The class of degree-*2 *PTFs is* $O(\sqrt{\log n})$*-approximately robustly learnable.*

5 Computational Intractability of Learning Robust Classifiers

In this section, we leverage the connection to polynomial optimization to complement our upper bound with the following nearly matching lower bound.We give a reduction from *Quadratic Programming (QP)* where given a polynomial $p(x) = \sum_{i<j} a_{ij} x_i x_j$, and a value s, the goal is to distinguish whether $max_{x \in \{-1,1\}^n} p(x) < s$ or whether exists an x such that $p(x) > s\eta_{approx}$. It is known that the distinguishing problem is hard for $\eta_{approx} = O(\log^c n)$ for some constant $c > 0$ [3]; moreover the state-of-the-art algorithms give a $\eta_{approx} = O(\log n)$ factor approximation [9] and improving upon this factor is a major open problem. By appropriately scaling the instance, this immediately implies the hardness of checking whether a given degree-2 PTF is robust around a given point.

However, this does not suffice for hardness of learning, since given a distribution supported at a single point, there is a trivial constant classifier that robustly classifies the instance correctly. More generally, there could exist a different degree-2 PTF that could be easy to certify for the given point. Instead, given a degree-2 PTF $\text{sgn}(p(x))$, we carefully construct a set of $O(n^2)$ points such that any classifier that is robust on an instance supported on the set will have to be close to the given polynomial p. Having established this, we can distinguish between the two cases of the *QP* problem by whether the learning algorithm is able to output a robust classifier or not. This is formalized below.

Theorem 5.1. *There exists* $\delta, \varepsilon > 0$*, such that assuming* $NP \ne RP$ *there is no algorithm that given a set of* $N = poly(n, \frac{1}{\varepsilon})$ *samples from a distribution* D *over* $\mathbb{R}^n \times \{-1, +1\}$*, runs in time* $\text{poly}(N)$ *and distinguishes between the following two cases for any* $\delta' = o(\sqrt{\eta_{approx}}\delta)$*:*

- Yes: *There exists a degree-*2 *PTF that has* δ*-robust error of* 0 *w.r.t.* D*.*

- No: *There exists no degree-2 PTF that has δ'-robust error at most ε w.r.t. D.*

Here η_{approx} is the hardness of approximation factor of the QP *problem.*

Remark 5.2. The above theorem proves that any polynomial time algorithm that always outputs a robust classifier (or declares failure if it does not find one) will have to incur an extra factor of $\Omega(\sqrt{\eta_{approx}})$ in the robustness parameter δ. Our upper bound in Section 4 on the other hand matches this bound. While our lower bound applies to algorithms that output a classifier of low error, in Appendix (see Theorem G.6) we also prove a more robust lower bound that rules out the possibility of an efficient robust learner that incurs an error less than 1/4.

6 Finding Adversarial Examples for Two Layer Neural Networks

Next we use the framework in Section 3 to design new algorithms for finding adversarial examples in two layer neural networks with ReLU activations. We describe the setting for binary classification below. A two layer neural network with ReLU gates is given by parameters (v_1, v_2, W) and outputs $f_1(x) = v_1^T \sigma(Wx), f_2(x) = v_2^T \sigma(Wx)$ where $x \in \mathbb{R}^n, v_1, v_2 \in \mathbb{R}^k$ and $W \in \mathbb{R}^{k\times n}$. Here $\sigma : \mathbb{R}^m \to \mathbb{R}^m$ is a co-ordinate wise non-linear operator $\sigma(y_i) = \max\{0, y_i\}$ for each $i \in [m]$. The binary classifier corresponding to the network is $sgn(f_1(x) - f_2(x)) = sgn(v^T\sigma(Wx))$ where $v = v_1 - v_2$. The optimization problem that arises is the following: given an instance with $A \in \mathbb{R}^{m_1\times n}, \beta \in \mathbb{R}^{m_2}, B \in \mathbb{R}^{m_2\times n}, c_1 \in \mathbb{R}^n, c_2 \in \mathbb{R}^{m_1}, c_0 \in \mathbb{R}$, the goal is to find $opt(A, B, \beta, c)$, defined as :

$$\begin{aligned} opt(A,B,\beta,c) &:= \max_{z:\|z\|_\infty\le\delta} \|c_2 + Az\|_1 + c_1^T z - \|\beta + Bz\|_1 + c_0 \\ &= \max_{z:\|z\|_\infty\le\delta}\ \max_{y:\|y\|_\infty\le 1} y^T Az + c_1^T z + c_2^T y - \sum_{j=1}^{m_2} |\beta_j + B_j^T z|. \qquad (2) \end{aligned}$$

Here B_j is the jth row of B. Let c denote (c_0, c_1, c_2), and let $opt(A, B, \beta, c)$ be the optimal value of the above problem. The following proposition holds in a slightly more general setting where there can be an extra linear term as described below.

Proposition 6.1. *Let $\gamma \ge 1$. Suppose there is an efficient algorithm that given an instance of problem (2) finds a solution $\widehat{z}, \widehat{y}$ with $\|\widehat{z}\|_\infty \le \gamma\delta, \|\widehat{y}\|_\infty \le 1$ such that $f(\widehat{y}, \widehat{z}) > 0$ when $opt(A, b, \beta, c) > 0$. Then there is an efficient algorithm that given a two layer neural net $sgn(f(x))$ where $f(x) := v^T\sigma(Wx) + (v')^T x$ and an example x^*, guarantees to either (a) find an adversarial example in the $B^n_\infty(x^*, \gamma\delta)$ around x^*, or (b) certify the absence of any adversarial example in $B^n_\infty(x^*, \delta)$.*

Our algorithm for solving (2) given in Figure 3 is inspired by Algorithm 1 for polynomial optimization. However, the rounding algorithm differs because the variables y_j and variables z_i serve different purposes in (2), and we need to simultaneously satisfy different constraints on them to produce a valid perturbation. Moreover when the SDP is negative, then this gives a certificate of robustness around x.

Please see Section F for a simple proof and more details. We remark that one can obtain provable guarantees similar to Theorem 4.2 for Algorithm 3 under certain regularity conditions about the SDP solution. However, this is unsatisfactory as this depends on the SDP solution to the given instance, as opposed to an explicit structural property of the instance. Obtaining provable guarantees of the latter kind is an interesting open question.

Experiments

Next, we evaluate the performance of the proposed attack in Figure 3 and compare it with the state of the art projected gradient descent(PGD) based attack of Madry et al. [24]. Our approach indeed finds more adversarial examples, although at a higher computational cost since we need to solve an SDP per example and per pair of classes. We use the MNIST data set and our 2-layer neural network has $d = 784$ input units, $k = 1024$ hidden units and 10 output units. The SDP has $d + k + 1$ vector variables, and takes about $200s$ per instance on a standard desktop. Hence we perform our experiments on randomly chosen subsets of the MNIST data set. Another optimization we perform for computational reasons is that given

1. Given instance $I = (A, B, \beta, c)$ of (2), solve SDP with parameter $\eta \in (0,1)$:

$$sdp = \max \sum_{j\in[m_1], i\in[n]} A_{j,i}\langle v_j, u_i\rangle + \sum_{i=1}^{n} c_1(i)\langle u_i, u_0\rangle + \sum_{j=1}^{m_1} c_2(j)\langle u_0, v_j\rangle - \sum_{j\in[m_2]} r_j + c_0$$

$$\text{s.t.} \forall j \in [m_1] \quad \|v_i\|^2 \le 1, \quad \forall i \in \{1,\dots,n\} \quad \|u_i\|^2 \le \delta^2, \quad \text{and } \|u_0\|^2 = 1$$

$$\forall j \in [k_2] \quad r_j \ge (\beta_j + \sum_j B_{j,i}\langle u_i, u_0\rangle), \quad \text{and } r_j \ge -(\beta_j + \sum_j B_{j,i}\langle u_i, u_0\rangle).$$

2. Let $u_i^\perp, v_j^\perp$ represent the components of u_i, v_j orthogonal to u_0. Let $\varepsilon \in (0,1)$ with $\varepsilon = \Omega(1)/\sqrt{\log m_1}$. Let $\zeta \sim N(0,I)$ be a Gaussian vector; set $\forall i \in \{0,1,\dots,n\}$, $\widehat{z}_i := \langle u_i, u_0\rangle + \frac{1}{\varepsilon}\langle u_i^\perp, \zeta\rangle$, $\widehat{y}_j := \langle v_j, u_0\rangle + \varepsilon\langle v_j^\perp, \zeta\rangle$.
3. Repeat rounding with $poly(n)$ random choices of ζ and pick the best choice.

Figure 3: The SDP-based algorithm for Problem (2).

$\delta = 0.3$	PGDpass (6×50 random samples)	PGDfail (8×100 random samples)
SDP succeeds	297 out of 300 total Mean : 49.5 out of 50, Std : 0.76	244 out of 800 total Mean 30.6 out of 100, Std : 2.87
$\delta = 0.01$	PGDpass (138 samples)	PGDfail (100 ranked)
SDP succeeds	138	45

Table 1: For $\delta = 0.3$, we report mean and standard deviation across batches of the number of adversarial examples found by running our SDPattack algorithm on 6 batches of 50 random examples from PGDpass and 8 batches of 100 random samples from PGDfail. For $\delta = 0.01$, we run SDPattack on all 138 examples in PGDpass and first 100 sorted examples from PGDfail.

an example x with predicted class i, we use a greedy heuristic to pick a class $j \neq i$ for the potential adversarial example $x + z$. So the numbers we report below are an underestimate of the effectiveness of the full SDP based algorithm. See Appendix B for a detailed discussion.

We consider two settings of the parameter δ, the maximum amount by which each pixel can be perturbed to produce a valid attack example. As in [24] we first choose $\delta = 0.3$ and train a robust 2-layer network using the algorithm in [24]. This network has an accuracy of 82.32% and adversarial accuracy (allowing for adversarial perturbations) of 31.7% on the test set. We then run the PGD attack and divide the test set into examples where the PGD attack succeeds (PGDPass) and examples where the PGD attack fails (PGDfail). We then run our attack on batches of random subsets chosen from each set. The first row of Table 1 shows the precision and recall of our method, along with the average and the standard deviation across the chosen batches. As one can see, our method has very high recall, i.e., whenever the PGD attack succeeds, our SDP based algorithm also finds adversarial examples. Furthermore, on examples where the PGD attack fails, our method is still able to discover new adversarial examples 30% of the time. See a sample of the perturbed images produced by our method in Section B. In particular, Figure 4 shows images of some of the examples where the SDP based attack succeeds, but the PGDattack fails and Figure 5 shows some images where both the PGDattack and SDP based attack succeed. A visual inspection of both the figures reveals that our attack often produces sparse targeted attacks as opposed to PGDattack.

We also run the PGD attack on the network with $\delta = 0.01$. Here we notice that attack succeeds on only 138 test examples and hence we can afford to run our attack on all of them. As can be seen from the second row of Table 1 our attack succeeds on all of these examples. Further, when we run our algorithm on the first 100 examples from PGDfail picked according to a greedy heuristic (see Section B for details), our method finds 45 new adversarial examples. This implies at least a $(138 + 45)/138 = 1.33$-fold advantage here.

The experiments above suggest that our theoretical claims and algorithms can lead to improved attacks. We would like to note that the recent work of [29] also studied SDP based methods for providing adversarial certificates for 2-layer neural networks. However, our SDP as outlined in Figure 3 is strictly stronger. The SDP of [29] is in fact independent of the given example x, so we expect our method to produce better certificates. We leave as future work the task of making our theoretical analysis practical for large scale applications.

7 Future Directions

Design of polynomial time algorithms that provably achieve adversarial robustness is an important direction of research. Several open questions remain to be explored further. In Section 4 we provide a general algorithmic framework for designing polynomial time robust algorithms. It would be interesting to use our framework to design robust algorithms for general degree-d PTFs. While there are algorithms to approximately maximize degree-d polynomials, they focus on the homogeneous case which does not suffice for our purposes. Another important direction for future work is to convert our adversarial attack algorithm for 2-layer neural networks into a provably robust learning algorithm via the framework of Section 4. A straightforward invocation of the framework does not lead to a convex constraint set. It would also be interesting to design provable adversarial attacks for higher depth networks. Finally, our experimental results suggest that making our SDP based attack work on a large scale could lead to improved adversarial attacks.

Acknowledgements

The second and third authors were supported by the National Science Foundation (NSF) under Grant No. CCF-1652491 and CCF-1637585. Additionally, the second author was funded by the Morrison Fellowship from Northwestern University.

References

[1] Noga Alon, Konstantin Makarychev, Yury Makarychev, and Assaf Naor. Quadratic forms on graphs. *Inventiones mathematicae*, 163(3):499–522, 2006.

[2] Noga Alon and Assaf Naor. Approximating the cut-norm via grothendieck's inequality. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 72–80. ACM, 2004.

[3] Sanjeev Arora, Eli Berger, Hazan Elad, Guy Kindler, and Muli Safra. On non-approximability for quadratic programs. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 206–215. IEEE, 2005.

[4] Idan Attias, Aryeh Kontorovich, and Yishay Mansour. Improved generalization bounds for robust learning. *arXiv preprint arXiv:1810.02180*, 2018.

[5] Chiranjib Bhattacharyya. Robust classification of noisy data using second order cone programming approach. In *Intelligent Sensing and Information Processing, 2004. Proceedings of International Conference on*, pages 433–438. IEEE, 2004.

[6] Alberto Bietti, Grégoire Mialon, and Julien Mairal. On regularization and robustness of deep neural networks. *arXiv preprint arXiv:1810.00363*, 2018.

[7] Sébastien Bubeck, Yin Tat Lee, Eric Price, and Ilya Razenshteyn. Adversarial examples from cryptographic pseudo-random generators. *arXiv preprint arXiv:1811.06418*, 2018.

[8] Sébastien Bubeck, Eric Price, and Ilya Razenshteyn. Adversarial examples from computational constraints. *arXiv preprint arXiv:1805.10204*, 2018.

[9] M Charikar and A Wirth. Maximizing quadratic programs: extending grothendieck's inequality. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 54–60. IEEE, 2004.

[10] Daniel Cullina, Arjun Nitin Bhagoji, and Prateek Mittal. Pac-learning in the presence of evasion adversaries. *arXiv preprint arXiv:1806.01471*, 2018.

[11] Dimitrios Diochnos, Saeed Mahloujifar, and Mohammad Mahmoody. Adversarial risk and robustness: General definitions and implications for the uniform distribution. In *Advances in Neural Information Processing Systems*, pages 10380–10389, 2018.

[12] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. In *Advances in Neural Information Processing Systems*, pages 1632–1640, 2016.

[13] Uriel Feige, Yishay Mansour, and Robert Schapire. Learning and inference in the presence of corrupted inputs. In *Conference on Learning Theory*, pages 637–657, 2015.

[14] Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.

[15] Justin Gilmer, Ryan P Adams, Ian Goodfellow, David Andersen, and George E Dahl. Motivating the rules of the game for adversarial example research. *arXiv preprint arXiv:1807.06732*, 2018.

[16] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018.

[17] Amir Globerson and Sam Roweis. Nightmare at test time: robust learning by feature deletion. In *Proceedings of the 23rd international conference on Machine learning*, pages 353–360. ACM, 2006.

[18] Pascale Gourdeau, Varun Kanade, Marta Kwiatkowska, and James Worrell. On the hardness of robust classification. *arXiv preprint arXiv:1909.05822*, 2019.

[19] A. Grothendieck and V. Losert. *"Résumé de la théorie métrique des produits tensoriels topologiques"*. Univ., 1976.

[20] Michael J Kearns, Umesh Virkumar Vazirani, and Umesh Vazirani. *An introduction to computational learning theory.* MIT press, 1994.

[21] Justin Khim and Po-Ling Loh. Adversarial risk bounds for binary classification via function transformation. *arXiv preprint arXiv:1810.09519*, 2018.

[22] Subhash Khot and Assaf Naor. Linear equations modulo 2 and the l1 diameter of convex bodies. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pages 318–328. IEEE, 2007.

[23] Subhash Khot and Ryan O'Donnell. Sdp gaps and ugc-hardness for maxcutgain. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 217–226. IEEE, 2006.

[24] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[25] Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure. *arXiv preprint arXiv:1809.03063*, 2018.

[26] Saeed Mahloujifar and Mohammad Mahmoody. Can adversarially robust learning leverage computational hardness? *arXiv preprint arXiv:1810.01407*, 2018.

[27] Yu Nesterov. Semidefinite relaxation and nonconvex quadratic optimization. *Optimization methods and software*, 9(1-3):141–160, 1998.

[28] Ryan O'Donnell. *Analysis of Boolean Functions.* Cambridge University Press, New York, NY, USA, 2014.

[29] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

[30] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. *arXiv preprint arXiv:1804.11285*, 2018.

[31] Pannagadatta K Shivaswamy, Chiranjib Bhattacharyya, and Alexander J Smola. Second order cone programming approaches for handling missing and uncertain data. *Journal of Machine Learning Research*, 7(Jul):1283–1314, 2006.

[32] Aman Sinha, Hongseok Namkoong, and John Duchi. Certifying some distributional robustness with principled adversarial training. 2018.

[33] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[34] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. 2018.

[35] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5283–5292, 2018.

[36] Huan Xu, Constantine Caramanis, and Shie Mannor. Robustness and regularization of support vector machines. *Journal of Machine Learning Research*, 10(Jul):1485–1510, 2009.

[37] Huan Xu and Shie Mannor. Robustness and generalization. *Machine learning*, 86(3):391–423, 2012.

[38] Dong Yin, Kannan Ramchandran, and Peter Bartlett. Rademacher complexity for adversarially robust generalization. *arXiv preprint arXiv:1810.11914*, 2018.