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 proliferation 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. [35]. 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 [26]. 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 [38, 39, 40, 23, 32, 16, 4, 12, 19, 36, 27, 28, 13], 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, 11, 1, 24]. 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 $\mathrm{poly}(\Delta, \frac{1}{\varepsilon})$ 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 3 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 6 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. [26] on the MNIST data set.

Paper Outline. In the rest of the paper, we give an overview of related work in Section 2. We define our model formally and give an overview of our techniques in Section 3. We then describe the connection to polynomial optimization in Section 4 and use it to design robust learning algorithms in Section 5, and derive computational intractability results in Section 6. In Section 7, we design adversarial attacks for 2 layer neural networks, followed by conclusions in Section 9.

2 Related Work

As mentioned in the introduction, there has been a recent explosion of works on understanding adversarial robustness from both empirical and theoretical aspects. Here we choose to discuss the theoretical works that are the most relevant to our paper. We refer the interested reader to a recent paper by [18] for a broader discussion. Prior to their relevance for deep networks, robust optimization problems have been studied in machine learning and other domains. The works of [7, 20, 33] studies optimization heuristics for optimizing a robust loss that can handle noisy or missing data. The works of [38, 39] proved an equivalence between robust optimization and various regularized variants of SVMs. They used this relation to re-derive standard generalization bounds for SVMs and their kernel versions. Akin to classifier stability, these bounds depend on the robustness of the classifier on the training set. A recent work of [8] views deep networks as functions in an RKHS and designs new norm based regularization algorithms to achieve robustness.

Motivated by connections to deep networks a recent line of work studies generalization bounds for robust learning. The work of [32] provides specific constructions of a linear binary classification task where a single example is enough to learn the problem in the usual sense, i.e., to achieve low test error, whereas learning the problem robustly requires a significantly large training set. The authors also show that in certain cases, non-linearity can help reduce the sample complexity of robust learning. The work of [12] proposes a PAC model for robust learning and defines adversarial VC dimension as a combinatorial quantity that captures robust learning via robust empirical risk minimization (ERM). The authors show that for linear classifiers the adversarial VC dimension is the same as the VC dimension, although there are functions classes and distributions where the gap between the two quantities could be much higher. The recent works of [40] and [23] analyze Rademacher complexity of robust loss functions classes. In particular, it is observed that even for linear models with bounded weight norm, there is an unavoidable dependence on the data dimension in the Rademacher complexity of robust loss function classes. These results point to the fact that for many distributions robust learning could require many more training samples than their non-robust counterpart. The work of [16, 4] studies algorithms and generalization bounds for a model where the adversary can choose perturbations from a known finite set of small size k.

Another recent line of work studies the trade-off between traditional test error and robust error. The work of [36] designs a classification task that is efficiently learnable with a linear classifier to low standard error, but has the property that any classifier that achieves low test error will have high robust error on the task. The work of [19] designs a task that is learnable by a degree-2 polynomial and relates the test error of any model to its robust error. Similar conclusions have been observed in [27, 28, 13] and have been used to design various data poisoning attacks. These results essentially follows from the use of isoperimetric inequalities for distributions such as the Gaussian and the uniform distribution over the Boolean hypercube. However, as noted in [19], it is not clear if the same relation holds between test error and robust error for real world data distributions. The work of [15] relates robustness to the curvature of the decision boundary

and uses it to quantify robustness to random perturbations.

Yet another line of work concerns the design of certificates of perturbation robustness or distributional robustness of a given classifier (e.g., deep neural networks) at a given point [37, 31, 34]. This is achieved by the use of convex relaxations of the optimal robustness at a given point. These works also conclude that by augmenting the training objective with a penalty that depends on the certificates, one can empirically achieve increased robustness. However these algorithms do not give any guarantees for relating the bound achieved by the certificate of robustness to the optimal robustness around a given point.

The work of Bubeck et al. [9, 10] provides a cryptographic lower bound by designing a computational task in $\mathbb{R}^n$ that is robustly learnable using $\text{poly}(n)$ samples to any given robustness parameter M, but is hard to learn robustly to any non-trivial robustness parameter $\varepsilon > 0$, in polynomial time. When translated to our model, this provides an instance of a cryptographic learning task that is computationally hard to γ-approximately robustly learn for any constant $\gamma \geq 1$. However, this does not rule out the possibility that natural function classes can be robustly learned without any loss in robustness parameter. Our result rules this out for the class of degree-2 and higher PTFs, even in the realizable setting, i.e., when there exists a robust classifier of zero error! Finally, to the best of our knowledge, our upper bounds are the first to establish the robustness tradeoff for computationally efficient learning for a large natural class of functions.

3 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 3.1 (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 [12]. Next we define robust learning for PTFs.

Definition 3.2 (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 may 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 5 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 4 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 [13, 21]. 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.

The work of [12] defines the notion of adversarial VC dimension to bound the generalization gap for robust empirical risk minimization. Additionally, the authors show that for linear classifiers the adversarial VC dimension remains the same as that of the original class. The bound below then follows by viewing PTFs as linear classifiers in a higher dimensional space.

Lemma 3.3. *Let $\mathcal{F}$ be a class of degree-d polynomial threshold functions from $\mathbb{R}^n \mapsto \{-1, 1\}$ of VC dimesion $\Delta = O(n^d)$. Given $\delta, \eta > 0$, and a set S of m examples $(x_1, y_1), \ldots, (x_m, y_m)$ generated from a distribution D over $\mathbb{R}^n \times \{-1, 1\}$, with probability at least $1 - \eta$, we have that $\sup_{f \in \mathcal{F}} |err_{\delta,D}(f) - \hat{err}_{\delta,S}(f)| \leq 2\sqrt{2\Delta \log m/m} + \sqrt{\log(1/\eta)/(2m)}$.*

4 Finding Adversarial Examples using Polynomial Optimization

In this section we introduce the broad class of polynomial optimization problems which are useful in designing adversarial (test-time) examples with provable guarantees for polynomial threshold functions (PTFs), and depth-2 neural networks with RELU gates. These polynomial optimization problems are generalizations of well-studied combinatorial optimization problems like the *Grothëndieck problem* and computing operator norms of matrices. We then design algorithms with provable guarantees for some of these classes. Proposition 4.1 restated below illustrates the connection and motivates the family of optimization problems that arise when designing algorithms with provable guarantees for finding adversarial examples for $sgn(g(x))$. While our theory below is stated for binary classifiers, it is easily extended to multiclass classification.

Proposition 4.1. *Let $\gamma \geq 1$. There is an efficient algorithm that given a classifier $sgn(f(x))$ and a point x^*, and budget $\delta > 0$, guarantees to either (a) find an adversarial example in $B^n_\infty(x^*, \gamma\delta)$, or (b) certify the absence of any adversarial example in $B^n_\infty(x^*, \delta)$, given access to an efficient optimization algorithm that takes x^* and a polynomial $g(z) \in \{ f(x^* + z), -f(x^* + z) \}$ as input and finds a $\widehat{z}$ such that $g(\widehat{z}) \geq \max_{\|z\|_\infty \leq \delta} g(z)$ with $\|\widehat{z}\|_\infty \leq \gamma\delta$.*

Proof of Proposition 4.1. Let ALG_γ be the optimization algorithm. Suppose there exists an adversarial example $x^* + z^*$ with $\|z^*\|_\infty \leq \delta$, and let $y^* := sgn(f(x^*))$ be the label for the point x^*. Then we have that $\max_{z:\|z\|_\infty \leq \delta} (-y^*) f(x^* + z) > (-y^*) f(x^* + z^*) > 0$. Now for $g(z) = -y^* f(x^* + z)$ (a polynomial in z), we get that ALG_γ finds a point $\widehat{z}$ with $\|\widehat{z}\|_\infty \leq \gamma\delta$ that also satisfies $(-y^*) f(x^* + \widehat{z}) > 0$ i.e., $sgn(f(x^*)) \neq sgn(f(x^* + \widehat{z}))$, as required. Furthermore, if ALG_γ fails, i.e., outputs a $\hat{z}$ such that $(-y^*) f(x^* + \widehat{z}) < 0$, then from the guarantee of the algorithm we know that $\max_{z:\|z\|_\infty \leq \delta} (-y^*) f(x^* + z) < 0$ and hence no adversarial example exists within a δ ball around x^*. □

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 $i \in \{0, 1, \ldots, n\}$.
4. Repeat rounding $O(\log(1/\eta))$ random choices of ζ and pick the best choice.

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

The proposition above also holds for randomized algorithms. While the proof of the proposition only requires that the algorithm returns $\widehat{z}$ with $g(\widehat{z}) > 0$, it effectively requires that $\widehat{z}$ attains at least as large an objective value because the constant term can be arbitrary. When the classifier is a degree-d PTF of the form $sgn(f)$, it leads to the following approximate optimization problem: given as input a degree d polynomial $g : \mathbb{R}^n \to \mathbb{R}$ (potentially different from f) and any $\eta, \delta > 0$, find in time $\mathrm{poly}(n, \log(\frac{1}{\eta}))$ and w.p. at least $1 - \eta$ a point $\hat{x}$ s.t.

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

The above problem 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 $\|\|_\infty$ ball constraint. Instead, our problem asks for the same objective value at the cost of an increase in the radius of the optimization ball. [1] 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 about degree-1 PTFs.

Claim 4.2. *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 in the ℓ_∞ ball of δ around x^* when it exists.*

Proof. We use Proposition 4.1 applied with linear functions. For linear function $g(x)$ represented by $g(x) := b^T x + c$ where $b \in \mathbb{R}^n, c \in \mathbb{R}$, we can easily find a solution $\widehat{x} \in B_\infty^n(0, \delta)$ such that $g(\widehat{x}) = \max_{x \in B_\infty^n(0,\delta)} g(x)$. This is because the linear form $b^T x + c$ is maximized within $B_\infty^n(0, \delta)$ by setting each variable x_i to be δ if the corresponding $b_i \ge 0$, and $-\delta$, otherwise. □

As we will see in Section 5, this will further be used to give robust learning algorithms for linear threshold functions. Our main theoretical result of this section gives an algorithm for provably finding adversarial examples for degree-2 PTFs.

Theorem 4.3. *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 4.1, 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 $g(\widehat{x}) \ge \max_{\|x\|_\infty \le \delta} g(x)$.

To prove the theorem we use a semi-definite programming (SDP) based algorithm shown in Figure 1, that is directly inspired by the SDP-based algorithm for quadratic programming (QP) by [29, 11]. However, the goal in quadratic programming is to find an assignment $x \in \{-1, 1\}^n$ that maximizes $\sum_{i \ne j} a_{ij} x_i x_j$. There are three main differences from the QP problem. Firstly, unlike QP which finds a solution with $\|x\|_\infty = 1$ with sub-optimal objective value, our goal is to output a solution which attains at least as large a value as $\max_{\|x\|_\infty \le \delta} g(x)$ while violating the ℓ_∞ length of the vector. Secondly, unlike QP where the diagonal terms are all 0, in our problem the diagonal terms can be non-zero and hence it is no longer true that the solution

[1] In approximation algorithms literature this will correspond to obtaining a $(1, \gamma)$-bicriteria approximation.

with $\|x\|_\infty \leq 1$ will have each co-ordinate being $\{\pm 1\}$. Finally and most crucially, QP corresponds to optimizing a homogeneous degree 2 polynomial, with no linear term. These challenges necessitates non-trivial modifications to the algorithm and in the analysis. We also remark that it seems unlikely that the upper bound of $O(\sqrt{\log n})$ on the approximation factor can be improved even for the special case of homogenous degree-2 polynomials, based on the current state of the approximability of Quadratic Programming (see Remark 4.5 for details).

The SDP we consider is given by the following equivalent vector program (the SDP variables correspond to $X_{ij} = \langle u_i, u_j \rangle$), which can be solved in polynomial time up to arbitrary additive error (using the Ellipsoid algorithm).

$$\max_{\{u_0, u_1, \ldots, u_n\}} \quad \sum_{i,j=1}^{n} A_{ij} \langle u_i, u_j \rangle + \sum_{i=1}^{n} b_i \langle u_i, u_0 \rangle + c \tag{2}$$

$$\text{s.t.} \quad \|u_i\|_2^2 \leq \delta^2 \quad \forall i \in \{1, 2, \ldots, n\}, \text{ and } \|u_0\|_2^2 = 1. \tag{3}$$

Let SDP_{val} denote the optimal value of the above SDP relaxation. Clearly the above SDP is a valid relaxation of the problem; for any valid solution $x \in [-\delta, \delta]^n$, consider the solution given by $\big(u_i = x_i u_0 : i \in [n]\big)$ for any unit vector u_0. Hence $\text{SDP}_{val} \geq \max_{\|x\|_\infty \leq \delta} g(x)$. Moreover, when the SDP value SDP_{val} is negative, this certifies that the classifier is robust around the give sample x^*. We prove Theorem 4.3 by designing a polynomial time rounding algorithm that takes the SDP solution and obtained a valid $\hat{z}$ satisfying the requirements of the theorem.

Rounding Algorithm. Given the SDP solution, let $u_i^\perp$ represent the component of u_i orthogonal to u_0. Consider the following randomized rounding algorithm that returns a solution $\{\widehat{x}_i : i \in [n]\}$:

$$\forall i \in \{0, 1, \ldots, n\}, \quad \widehat{x}_i := \langle u_i, u_0 \rangle + \langle u_i, \zeta \rangle = \langle u_i, u_0 \rangle + \langle u_i^\perp, \zeta \rangle, \text{ with } \zeta \sim N\Big(0, \Pi^\perp\Big), \tag{4}$$

where $\Pi^\perp$ is the projection matrix onto the subspace of $span(\{u_1, \ldots, u_n\})$ that is orthogonal to u_0. For convenience, we can assume without loss of generality that $u_0 = e_0$, where e_0 is a standard basis vector, and $u_i \in \mathbb{R}^{n+1}$. Let $e_0, e_1, \ldots, e_n$ represent an orthogonal basis for $\mathbb{R}^{n+1}$. Then

$$\forall i \in \{0, 1, \ldots, n\}, \ \widehat{x}_i = \langle u_i, u_0 \rangle + \langle u_i^\perp, \zeta \rangle \text{ where } \langle \zeta, e_0 \rangle = 0, \ \langle \zeta, v \rangle \sim N(0, \|v\|_2^2) \text{ for every } v \perp e_0,$$

and $\widehat{x}_0 = 1$. The rounding algorithm just tries $O(\log(1/\eta))$ independent random draws for ζ, and picks the best of these solutions.

We now give the analysis of the algorithm. We prove Theorem 4.3 by showing the following guarantee for the rounding algorithm.

Lemma 4.4. *There is a polynomial time randomized rounding algorithm that takes as input the solution of the SDP as defined in Equations 2, and 3, and outputs a solution $\widehat{x}$ such that*

$$\mathop{\mathbb{P}}_{\widehat{x}} \Big[g(\widehat{x}) \geq \max_{\|x\|_\infty \leq \delta} g(x) \text{ and } \|\widehat{x}\|_\infty \leq O(\sqrt{\log n}) \cdot \delta \Big] \geq \Omega(1). \tag{5}$$

Assuming (5), we can repeat the algorithm at least $O(\log(1/\eta))$ times to get the guarantee of Theorem 4.3.

Proof of Lemma 4.4. We start with a simple observation that follows from the standard properties of spherical Gaussians. For any $i, j \in [n]$, we have $\mathbb{E}_\zeta[\langle u_i^\perp, \zeta \rangle \langle u_j^\perp, \zeta \rangle] = (u_i^\perp)^T \Pi^\perp u_j^\perp = \langle u_i^\perp, u_j^\perp \rangle$. Hence we get the key observation that for $\forall i, j \in \{0, \ldots, n\}$,

$$\begin{aligned} \mathbb{E}\left[\widehat{x}_i \widehat{x}_j\right] &= \mathop{\mathbb{E}}_{\zeta} \Big[\Big(\langle u_i, u_0 \rangle + \langle u_i^\perp, \zeta \rangle \Big) \Big(\langle u_j, u_0 \rangle + \langle u_j^\perp, \zeta \rangle \Big) \Big] = \langle u_i, u_0 \rangle \langle u_j, u_0 \rangle + \mathop{\mathbb{E}}_{\zeta} \Big[\langle u_i^\perp, \zeta \rangle \langle u_j^\perp, \zeta \rangle \Big] \\ &= \langle u_i, u_0 \rangle \langle u_j, u_0 \rangle + \langle u_i^\perp, u_j^\perp \rangle = \langle u_i, u_j \rangle. \end{aligned} \tag{6}$$

Note that this also holds when $i = j$. We now consider the expected value of $g(\widehat{x})$. Using (6), $\widehat{x}_0 = 1$ and since $\mathbb{E}_\zeta[\langle u_i^\perp, \zeta\rangle] = 0$, we have

$$\begin{aligned}\mathbb{E}[g(\widehat{x})] &= \sum_{i,j=1}^{n} A_{ij} \mathop{\mathbb{E}}_{\zeta}\left[\widehat{x}_i \widehat{x}_j\right] + \sum_{i=1}^{n} b_i \mathop{\mathbb{E}}_{\zeta}[\widehat{x}_i \widehat{x}_0] + c \mathop{\mathbb{E}}_{\zeta}[\widehat{x}_0^2] \\ &= \sum_{i,j=1}^{n} A_{ij}\langle u_i, u_j\rangle + \sum_{i=1}^{n} b_i \langle u_i, u_0\rangle + c\|u_0\|_2^2 = \mathrm{SDP}_{val}. \end{aligned} \tag{7}$$

We now show that $\widehat{x}_i \le O(\sqrt{\log n}) \cdot \delta$ w.h.p. For each fixed $i \in \{1, \ldots, n\}$, $\langle u_i^\perp, \zeta\rangle$ is distributed as a Gaussian with mean 0 and variance $\|u^\perp\|_2^2 \le \delta^2$,

$$|\widehat{x}_i| \le |\langle u_i, u_0\rangle| + |\langle u_i^\perp, \zeta\rangle| \le \delta + |\langle u_i^\perp, \zeta\rangle| \le \sqrt{C \log n} \cdot \delta \text{ with probability at least } 1 - 1/n^{C/2},$$

using standard tail properties of Gaussians. Hence, using a union bound over all $i \in [n]$, we have that

$$\mathbb{E}[g(\widehat{x})] \ge \max_{\|x\|_\infty \le \delta} g(x), \quad \text{and } \mathbb{P}\left[\|\widehat{x}\|_\infty \le O(\sqrt{\log n}) \cdot \delta\right] \ge 1 - \frac{1}{n^2}. \tag{8}$$

for $C \ge 4$. Further note that $g(\widehat{x})$ can be expressed a degree-d polynomial of the Gaussian vector ζ. Hence using hypercontractivity of low-degree polynomials (Theorem 10.23 of [30]), we have

$$\mathop{\mathbb{P}}_{\zeta}\left[g(\widehat{x}) \ge \mathop{\mathbb{E}}_{\zeta} g(\widehat{x})\right] \ge \Omega(1).$$

Hence (5) follows. □

Remark 4.5. Obtaining an approximation factor of $O(\gamma)$ in the ℓ_∞ norm of $\hat{z}$, even for the special case of homogeneous degree-2 polynomials $\sum_{i<j=1}^{n} a_{ij} x_i x_j$ with no diagonal entries ($a_{ii} = 0\ \forall i \in [n]$) over $\|x\|_\infty \le \delta$ is equivalent to obtaining a $O(\gamma^2)$-factor approximation algorithm for the problem called Quadratic Programming (QP) which maximizes $\sum_{i<j=1}^{n} a_{ij} x_i x_j$ over $x \in \{-1, 1\}^n$ (this is also called the Grothendieck problem on complete graphs). The best known approximation algorithm for Quadratic Programming (QP) gives an $O(\log n)$-factor approximation in polynomial time [29, 11]. Further [3] showed that it is hard to approximate QP within a $O(\log^c n)$ for some universal constant $c > 0$ assuming NP does not have quasi-polynomial time algorithms. Moreover integrality gaps for SDP relaxations [1, 25] suggest that $O(\log n)$ factor maybe be tight for polynomial time algorithms. Hence even for the special case of homogeneous degree-2 polynomials, improving upon the bound of $\sqrt{\log n}$ in the approximation factor seems unlikely.

5 From Adversarial Examples to Robust Learning Algorithms

In this section we will show how to leverage the algorithms for finding adversarial examples to design polynomial time robust learning algorithms for various sub-classes of Polynomial Threshold Functions (PTF). In particular, these include general degree-1 and degree-2 polynomial threshold functions. 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 4.This is formalized in the definition below:

Definition 5.1 (γ-factor admissibility)**.** For $\gamma \ge 1$, we say that a sub-class $\mathcal{F}$ of PTFs is γ-factor admissible if $\mathcal{F}$ has the following properties:

1. For any $a, b, c \in \mathbb{R}, \mathrm{sgn}(f(x)), \mathrm{sgn}(g(x)) \in \mathcal{F}$, $\mathrm{sgn}(af(x) + bg(x) + c) \in \mathcal{F}$.
2. For any $b \in \mathbb{R}^n$ and $sgn(g(x)) \in \mathcal{F}$, we have that $sgn(g(x + b)) \in \mathcal{F}$.
3. There is a γ-admissible approximation for $\{ g : sgn(g) \in \mathcal{F} \}$.

The first two conditions above are natural and are satisfied by many sub-classes of PTFs. The third condition in the above definition concerns the optimization problem studied in Section 4. The main result of this section, stated below, is the claim that any admissible sub-class of PTFs is also robustly learnable in polynomial time.

Theorem 5.2. *Let $\mathcal{F}$ be a sub-class of PTFs that is γ-factor admissible for $\gamma \geq 1$. Then $\mathcal{F}$ is γ-approximate robustly learnable.*

Remark 5.3. While we state our upper bounds for perturbations measured in the ℓ_∞ norm, we would like to point out that one can define analogously γ-factor admissibility for any ℓ_p norm and the above theorem will still hold true with the new definition.

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 (denoted by $\text{coeff}(g) \in \mathbb{R}^D$) and an appropriate vector $\psi(x) \in \mathbb{R}^D$ where $D = \binom{n+d-1}{d}$. Our goal is to find a polynomial $g \in \mathcal{F}$ that correctly classifies all the training examples (x_i, y_i). This corresponds to the constraint $y_i g(x_i) > 0$ expressed as $y_i \langle \text{coeff}(g), \psi(x) \rangle > 0$, a linear constraint in the unknown coefficients $\text{coeff}(g)$ of the polynomial g. For example, if $g(x)$ is a degree-2 polynomial of the form $x^T Ax + b^T x + c$, then the constraint $y_i g(x_i) > 0$ is linear in the unknown coefficients, $a_{i,j}, b_i$ and c, of the polynomial. Here $a_{i,j}$ corresponds to the (i,j) entry of the matrix A and b_i is the ith coordinate of vector b. We also want to ensure that g is robust around each point in the training set. These two constraints together can be enforced by the convex program in Figure 2, where the r_i's are additional variables apart from the coefficients of g. Note that the set of all g is convex because of condition 1 of Definition 5.1. While constraints in (10) are linear in the variables and easy to implement, (11) is really asking to check the robustness of g at a given point (x_i, y_i), which is an NP-hard problem [11]. 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 (11). We would like to mention that the classical literature on robust optimization of linear and convex programs studies a similar setting where typically the goal is to bound the probability of each constraint being violated while achieving the maximum objective value [5, 14, 6]. In contrast, we are interested in precisely quantifying how much a constraint can be violated by and relate the bound to the robustness of the final classifier obtained. We are now ready to prove the main theorem of this section.

Proof of Theorem 5.2. Let $\eta > 0$ be the success probability desired for the robust learning algorithm and $\varepsilon > 0$ be the final robust error that is desired. 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. Let $T(m,n)$ be a (polynomial) upper bound on the number of iterations of the algorithm. In each iteration, given $g, r_1, r_2, \ldots, r_m$, we will first check whether $y_i g(x_i) > r_i$. If not, then we have found a violated constraint with the corresponding separating hyperplane being $sgn(r_i - y_i g(x_i))$, and the algorithm proceeds to the next iteration. If all the constraints in (10) are satisfied, then for each $i \in [m]$, we run $\mathcal{B}$ on the polynomial $y_i(g(x_i) - g(x_i + z))$, where z is the variable and x_i is fixed to be the ith data point. Furthermore, we will set η', the failure probability of $\mathcal{B}$, to be equal to $\eta/(mT(m,n))$ and set δ' that is input to $\mathcal{B}$ to be δ/γ. From the guarantee of $\mathcal{B}$ we get that if there exists an i such that

$$r_i < \sup_{z \in B_\infty^n(0, \frac{\delta}{\gamma})} y_i \Big(g(x_i) - g(x_i + z) \Big), \tag{9}$$

with probability at least $1 - \eta/T(m,n)$, the $\mathcal{B}$ will output a violated constraint of the convex program, i.e., an index $i \in [m]$ and $\hat{z} \in B_\infty^n(0, \delta)$ such that

$$r_i < \sup_{z \in B_\infty^n(0, \delta)} y_i \Big(g(x_i) - g(x_i + \hat{z}) \Big).$$

This gives us a separating hyperplane of the form $sgn(y_i(g(x_i) - g(x_i + \hat{z})) - r_i)$, and the algorithm continues. Hence, we get that when the Ellipsoid algorithm terminates, with probability at least $1 - \eta$, it will

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 polynomial $g \in \mathcal{F}$ that satisfies

$$y_i g(x_i) > r_i, \ \ \forall i \in [m] \tag{10}$$

$$r_i \geq \sup_{z \in B_\infty^n(0,\delta)} y_i \Big(g(x_i) - g(x_i + z) \Big), \ \ \forall i \in [m] \tag{11}$$

Figure 2: The convex program for finding a polynomial $g \in \mathcal{F}$ with zero robust empirical error.

output a polynomial $g \in \mathcal{F}$ such that the constraints in (10) and (9) are satisfied. This means that we would have the empirical robust error $e\hat{r}r_{\delta/\gamma,S}(sgn(g)) = 0$. Hence, by Lemma 3.3, we get that

$$err_{\delta/\gamma,D}(sgn(g)) \leq 2\sqrt{\frac{2\Delta \log m}{m}} + \sqrt{\frac{\log \frac{1}{\eta}}{2m}},$$

where Δ is the VC dimension of $\mathcal{F}$. Choosing $m = c\frac{\Delta + \log(1/\eta)}{\varepsilon^2}$, makes $err_{\delta/\gamma,D}(sgn(g)) \leq \varepsilon$. □

It is easy to check that for any fixed $d \in \mathbb{N}$, general degree-d PTFs satisfy conditions 1 and 2 of Definition 5.1 (however homogenous degree d polynomials do not satisfy condition 2). We conclude the section by stating the following corollaries about robust learnability of general degree-1 and degree-2 PTFs.We begin with the following claim about admissibility and hence robust learnability of degree-1 PTFs.

Corollary 5.4. *The class of degree-*1 *PTFs is optimally robustly learnable.*

The proof just follows from Claim 4.2 and since any linear combination or shift of a linear function is also linear. Similarly, the following corollary about degree-2 PTFs is immediate from Theorem 4.3.

Corollary 5.5. *The class of degree-*2 *PTFs is $O(\sqrt{\log n})$-approximately robustly learnable.*

6 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 [11] 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 6.1. *There exists $\delta, \varepsilon > 0$, such that assuming $NP \neq RP$ 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 6.2. 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 5 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 6.7) 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.

We will represent an instance of *Quadratic Programming (QP)* by a polynomial $p(x) = x^T Ax$ where A is a symmetric matrix with zeros on the diagonal, and $A_{ij} = A_{ji} = a_{ij}/2$. Formally, the NP-hard problem $\mathcal{QP}$ [3, 17] is the following: given $\beta > 0$ and a polynomial $p(x) = x^T Ax$ distinguish whether

No Case : there exists an assignment $x^* \in \{-1, 1\}^n$ such that $p(x^*) > \beta\eta_{approx}$,

Yes Case : for every assignment $x \in \{-1, 1\}^n$, $p(x) < \beta$.

We prove that there exists a $\delta > 0$ and a set of $N = \text{poly}(n)$ points such that it is hard to distinguish whether there exists a degree-2 PTF that is δ robust at all the points or that no degree-2 PTF is $\eta\delta$ robust for $\eta = \Omega(1/\sqrt{\eta_{approx}})$.

Theorem 6.3. *[Hardness] There exists $\delta > 0$, such that assuming $NP \neq RP$ there is no polynomial time algorithm that given a set of $N = O(n^2)$ labeled points $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$ with $(x^{(j)}, y^{(j)}) \in \mathbb{R}^{n+1} \times \{-1, 1\}$ for all $j \in [N]$ can distinguish between the following two cases*

***YES Case:** There exists a degree-2 PTF that has δ-robust empirical error of 0 on these N points.*

***NO Case:** No degree-2 PTF is $\eta\delta$-robust on these points for $\eta = \Omega(1/\sqrt{\eta_{approx}})$.*

Theorem 6.1 follows from Theorem 6.3 and the standard fact used in establishing learning theoretic hardness [22], namely if there were a robust learning algorithm for every distribution and $\varepsilon > 0$, the one could use it on the uniform distribution over the instance from Theorem 6.3 with $\varepsilon = \frac{1}{2N}$ to determine whether there exists a degree-2 PTF that has δ-robust empirical error of 0 on the points in the instance. Hence our main goal is to prove Theorem 6.3. In order to get hardness of approximation, we need to pick the set of points carefully. Our set of points will have the property that in the YES case of the QP instance, the polynomial $x^T Ax - z$ will be δ robust at all the points (see Claim 6.6). Furthermore, the points will enforce the property that any other degree-2 PTF that classifies the points correctly will have to be very close to $x^T Ax - z$ in terms of the parameters. This will help use rule out the existence of an $\eta\delta$ robust classifier in the NO case, since if one exists, it must be close to $x^T Ax - z$, thereby implying an upper bound on the value of $x^T Ax$ around the neighborhood of zero. This is established in the following key lemma.

Lemma 6.4. *Let $p(x, z) = x^T Ax - z$ be a given polynomial where A is a symmetric matrix with zeros on the diagonal. For any $\varepsilon, \delta < 1/10$, consider the labeled set $S = S_1 \cup S_2 \cup S_3 \cup S_4 \cup S_5$ where,*

$$S_1 = \{((\mathbf{0}, 1), -1), ((\mathbf{0}, -1), +1), ((\mathbf{0}, \tau'), -1), ((\mathbf{0}, -\tau'), +1), ((\mathbf{0}, 2\delta), -1), ((\mathbf{0}, -2\delta), +1)\},$$

$$S_2 = \{((\mathbf{e}_i, \gamma), -1), ((\mathbf{e}_i, -\gamma), +1), ((-\mathbf{e}_i, \gamma), -1), ((-\mathbf{e}_i, -\gamma), +1)\},\ \forall i \in [n],$$

$$S_3 = \{((\mathbf{e}_{i,j}, 2), -1), ((\mathbf{e}_{-i,j}, 2), -1), ((\mathbf{e}_{i,-j}, 2), -1), ((\mathbf{e}_{-i,-j}, 2), -1)\},\ \forall i \neq j \in [n],$$

$$S_4 = \{((2\mathbf{e}_{i,j}, 1), \text{sgn}(a_{i,j})), ((2\mathbf{e}_{-i,j}, 1), -\text{sgn}(a_{i,j})), ((2\mathbf{e}_{i,-j}, 1), -\text{sgn}(a_{i,j})),$$
$$((2\mathbf{e}_{-i,-j}, 1), \text{sgn}(a_{i,j}))\},\ \forall i \neq j \in [n],$$

and

$$S_5 = \{((\mathbf{e}_{i,j}, -2), +1), ((\mathbf{e}_{-i,j}, -2), +1), ((\mathbf{e}_{i,-j}, -2), +1), ((\mathbf{e}_{-i,-j}, -2), +1)\},\ \forall i \neq j \in [n],$$

Here $\mathbf{e}_i$ *is the vector* $(0,0,\ldots,\tau,0,\ldots,0)$ *and* $\mathbf{e}_{i,j}$ *is the vector* $(0,0,\ldots,\frac{1}{\sqrt{2(\varepsilon+|a_{i,j}|)}},0,\ldots,\frac{1}{\sqrt{2(\varepsilon+|a_{i,j}|)}},0,\ldots,0)$. *For every general degree 2 polynomial* $q'(x,z)$ *with the coefficient of* $z=c_z$, *such that* $sgn(q')$ *has zero error on* S, *we must have* $c_z \neq 0$. *Moreover, let* $q(x,z)=\frac{1}{-c_z}q'(x,z)=x^TA'x+c_1^Tx+c_2z^2-z+c_4+\sum_i\beta_izx_i$, *where* A' *be a symmetric matrix. Then we must have that*

$$\max(|c_2|,\|\beta\|_\infty,|a'_{i,i}|)\leq\varepsilon,$$

$$|c_4|\leq 4\delta,$$

$$|c_{1,i}|\leq\min_{j\neq i}8\delta\sqrt{\varepsilon+|a_{i,j}|},$$

and

$$\frac{1}{4}-\delta-\frac{\varepsilon}{4}\leq\max(\frac{|a'_{i,j}|}{\varepsilon+|a_{i,j}|})\leq 2+4\delta+\varepsilon$$

provided $\tau'=\Omega(\frac{n^2}{\varepsilon})\max(1,1/(\varepsilon+\min_{i\neq j}|a_{i,j}|))$, $\tau=\Omega(\frac{n}{\varepsilon})\max(1,1/(\varepsilon+\min_{i\neq j}|a_{i,j}|))$, $\gamma=4n\tau$.

We first prove Theorem 6.3 assuming the lemma above and finally end the section with the proof of the lemma.

Proof of Theorem 6.3. Given an $n\times n$ symmetric matrix A with zeros on diagonals and given $s>100$, we assume that the following cases are hard to distinguish for some $\eta_{approx}>1$,
YES Case: $\max_{x\in\{-1,1\}^n}x^TAx<s$.
NO Case: $\max_{x\in\{-1,1\}^n}x^TAx>s\eta_{approx}$. The reduction from the instance of the QP problem is sketched below. Next we establish completeness and soundness of the reduction.

1. Scale the entries of A such that each non zero entry is greater than 10. Scale s by the same factor. Set $\delta=1/s$ and $\varepsilon=200/n^2$.
2. Generate the labeled point set S in $\mathbb{R}^{n+1}$ as specified in Lemma 6.4 with $\tau'=\Omega(\frac{n^2}{\varepsilon})\max(1,1/(\varepsilon+\min_{i\neq j}|a_{i,j}|))$, $\tau=\Omega(\frac{n}{\varepsilon})\max(1,1/(\varepsilon+\min_{i\neq j}|a_{i,j}|))$, $\gamma=4n\tau$.

Figure 3: Reduction from the QP problem.

NO Case: The following claim captures the soundness analysis of the reduction.

Claim 6.5. *There does not exist an* $\eta\delta$*-robust degree-2 polynomial on* S *for* $\eta=\Omega(1/\sqrt{\eta_{approx}})$.

Proof. We will prove by contradiction. Let $q(x,z)=x^TA'x+c_1^Tx+c_2z^2-z+c_4+\sum_i\beta_izx_i$ be an $\eta\delta$-robust polynomial on S.[2] The fact that q is correct on $(\mathbf{0},2\delta)$ gives us

$$4c_2\delta^2-2\delta+c_4<0 \tag{12}$$

Furthermore, the fact the fact that q is $\eta\delta$-robust on $(\mathbf{0},2\delta)$ gives us that

$$\max_{x\in B_\infty^n(0,\eta\delta),z\in(2\delta-\eta\delta,2\delta+\eta\delta)}q(x,z)<|4c_2\delta^2-2\delta+c_4| \tag{13}$$

From Lemma 6.4 this implies that

$$\max_{x\in B_\infty^n(0,\eta\delta)}x^TA'x<|4c_2\delta^2-2\delta+c_4|+(2\delta+\eta\delta)+12\delta+\varepsilon(2\delta+\eta\delta)^2+n\varepsilon\eta\delta(2\delta+\eta\delta) \tag{14}$$

[2]We can always scale q to get it into this form.

Substituting the value of ε we get that

$$\max_{x \in B_\infty^n(0,\eta\delta)} x^T A' x < 20\delta. \tag{15}$$

Again using Lemma 6.4 we get that

$$\max_{x \in B_\infty^n(0,\delta)} x^T A x < \frac{50\delta}{\eta^2}. \tag{16}$$

But since we are in the NO case we also know that

$$\max_{x \in B_\infty^n(0,\delta)} x^T A x > \delta^2 s \eta_{approx} = \delta \eta_{approx}. \tag{17}$$

This contradicts the fact that $\eta = \Omega(1/\sqrt{\eta_{approx}})$. □

YES Case: The analysis of the YES case relies on the following claim which establishes δ-robustness of the PTF given by $p(x, z)$ on the point in S.

Claim 6.6. *The polynomial $p(x, z) = x^T Ax - z$ is δ-robust on S.*

We defer the proofs of Claim 6.6 and Lemma 6.4 to Section A.1.

A Lower Bound for Weak Robust Learning. We also prove a robust lower bound that rules out the possibility of weak robust learning with $\gamma = 1$. This hardness result allows the algorithm to output a robust classifier that makes errors on constant fraction of the points! Hence, even when there is a degree-2 PTF that has δ robust error of 0, it is computationally hard to output a degree-2 PTF that has δ-robust error of $\varepsilon \leq \frac{1}{4}$.

Theorem 6.7. *[Stronger Distributional Hardness] For every $\delta > 0$ and $\varepsilon \in (0, \frac{1}{4})$, assuming $NP \neq RP$ there is no polynomial time algorithm that given a set of $N = poly(n, \frac{1}{\varepsilon})$ samples from a distribution D over $\mathbb{R}^n \times \{-1, +1\}$ can distinguish between the following two cases:*

- 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.*

The proof of the above theorem uses non-distributional hardness in Theorem A.10. Please see Section A.2.

7 Finding Adversarial Examples for Two Layer Neural Networks

Next we use the framework in Section 4 to design new algorithms for finding adversarial examples in two layer neural networks with ReLU activations. The description that follows applies to binary classification and can be easily extended to multiclass classification. 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 :

$$\begin{aligned} opt(A, B, \beta, c) &:= \max_{z:\|z\|_\infty \leq \delta} \|c_2 + Az\|_1 + c_1^T z - \|\beta + Bz\|_1 + c_0 \\ &= \max_{z:\|z\|_\infty \leq \delta} \max_{y:\|y\|_\infty \leq 1} y^T Az + c_1^T z + c_2^T y - \sum_{j=1}^{m_2} |\beta_j + B_j^T z|. \end{aligned} \tag{18}$$

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.

To see the connection to polynomial optimization, notice that if $B = 0$, then the above problem is exactly the one we considered in Section 4 in the context of degree-2 PTFs. Furthermore, if $A = 0$, then 18 is a linear

1. Given instance $I = (A, B, \beta, c)$ of (18), solve SDP with parameter $\eta \in (0, 1)$:

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

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

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

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 $\forall i \in \{0, 1, \ldots, n\}$, $\widehat{z}_i := \langle u_i, u_0\rangle + \frac{1}{\varepsilon}\langle u_i^\perp, \zeta\rangle$, $\widehat{y}_j := \langle v_j, u_0\rangle + \varepsilon\langle v_j^\perp, \zeta\rangle$.
3. Repeat rounding with $poly(n)$ random choices of ζ and pick the best choice.

Figure 4: The SDP-based algorithm for Problem (18).

program. However, the presence of both the terms involving A and B make 18 a challenging optimization problem. Next we discuss how the problem is related to finding adversarial examples for 2-layer neural networks. 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 classifier corresponding to the network is $sgn(f_1(x) - f_2(x)) = sgn((v_1 - v_2)^T\sigma(Wx)) = sgn(v^T\sigma(Wx))$.

Our algorithm for solving (18) given in Figure 4 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 (18), 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.

We remark that one can obtain provable guarantees similar to Theorem 5.2 for Algorithm 4 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. The following proposition holds in a more general setting where there can be an extra linear term as described below.

Proposition 7.1. *Let $\gamma \ge 1$. Suppose there is an algorithm that given an instance of problem* (18) *finds a solution $\widehat{z}, \widehat{y}$ with $\|\widehat{z}\|_\infty \le \gamma\delta, \|\widehat{y}\|_\infty \le 1$ such that $\widehat{y}^T A\widehat{z} + c_1^T\widehat{z} + c_2^T\widehat{y} - \|\beta + B\widehat{z}\|_1 + c_0 > 0$ when $opt(A, B, \beta, , c) > 0$, then there is a polynomial time algorithm that given a classifier $sgn(f(x))$ corresponding to a two layer neural net where $f(x) := v^T\sigma(Wx) + (v')^T x$ and an example x^*, guarantees to either (a) find an adversarial example in the ℓ_∞ ball of $\gamma\delta$ around x^*, or (b) certify the absence of any adversarial example in the ℓ_∞ ball of δ.*

Proof. Let $\ell(x^*) = sgn(f(x^*))$. We first observe that $\sigma(y_j) = \frac{1}{2}(|y_j| + y_j)$, and $\sigma(Wx)_j = \frac{1}{2}(|\langle W_j, x\rangle| + \langle W_j, x\rangle)$, where W_j is the jth row of W. We want to find a $\widehat{z}$ with $\|\widehat{z}\|_\infty \le \gamma\delta$, such that $(-\ell(x^*))f(x^* + \widehat{z}) > 0$, or certify that there is no such $\widehat{z}$ with $\|\widehat{z}\|_\infty \le \delta$.

Let $S_+ = \{j \in [k] : -\ell(x^*)v_j \ge 0\}$ and $S_- = [k] \setminus S_+$ and let $k_1 = |S_+|$. We now split the rows of W into two (A and B) as follows: for every $j \in S_+$, define the row $A_j := \frac{1}{2}|v_j|W_j$; otherwise let $B_j := \frac{1}{2}|v_j|W_j$.

$$\begin{aligned} -\ell(x^*)f(x^* + z) &= \tfrac{1}{2}\sum_{j\in S_+} |v_j||\langle W_j, x^* + z\rangle| + \tfrac{1}{2}\langle v^T W, x^* + z\rangle - \tfrac{1}{2}\sum_{j\in S_-} |v_j||\langle W_j, x^* + z\rangle| \\ &= \max_{y\in\{-1,1\}^{k_1}} \sum_{j\in S_+} y_j\langle A_j, x^* + z\rangle - \sum_{j\in S_-} |\langle B_j, x^* + z\rangle| + c_1^T z + c_0, \end{aligned}$$

where $c_1^T = \frac{1}{2}v^T W + (v')^T$ and $c_0 = \frac{1}{2}v^T W x^*$ are constants. Since the dependence on y is linear we also get

by substituting $c_2 := Ax^*$, $\beta := Bx^*$,

$$\max_{\|z\|_\infty \le \delta} (-\ell(x^*)) f(x^* + z) = \max_{\|z\|_\infty \le \delta} \max_{y : \|y\|_\infty \le 1} \sum_{j \in S_+} y_j \langle A_j, z \rangle + c_2^T y + c_1^T z - \sum_{j \in S_-} |\beta_j + \langle B_j, z \rangle| + c_0,$$

as required. Now the proposition follows from the same argument as in Proposition 4.1. □

8 Experiments

In this section, we evaluate the performance of the SDP based rounding algorithm outlined in Figure 4 to generate adversarial examples for depth-2 neural networks with ReLU gates, and compare it with the projected gradient descent(PGD) based attack of Madry et al. [26]. We will show that our approach indeed finds more adversarial examples. This however, comes at a computational cost since we need to solve one SDP per example and per pair of classes. We use the MNIST data set and our two layer neural network has $d = 784$ input units, $k = 1024$ hidden units and 10 output units. This leads to an SDP with $d + k + 1$ vector variables. On an standard desktop with Intel i5 4590 processor, and 4 cores 3.30GHz, solving one SDP instance takes 200 seconds on average. As a consequence we perform our experiments on randomly chosen subsets of the MNIST data set. Another optimization we perform for computational reasons is that given an example x with predicted class i, rather than checking for every class j, if one can find an attack example z that misclassifies $x + z$ to be in class j, we simply pick j to be the class label of the second highest prediction at x. Hence, the numbers we report below are an underestimate of the effectiveness of the full SDP based algorithm

We compare the effectiveness of our attack in finding adversarial examples when compared to the the PGD based attack of Madry et al. [26]. 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 [26] we first choose $\delta = 0.3$ and train a robust 2-layer network using the algorithm of Madry et al. [26]. 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. In the algorithm we set $\delta' = \alpha\delta$ for a hyperparameter $\alpha \le 1$. This is because we want to ensure that the rounded solutions have ℓ_∞ norm of at most δ. In our experiment we set $\alpha = 0.07$. The first row of Table 1 shows the precision and recall of our method. We report 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. Please see Figure 5 for the images corresponding to some of the examples where the SDP based attack succeeds, but the PGDattack fails and Figure 6 for the images of some examples 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 repeat the same methodology with $\delta = 0.01, \alpha = 0.2$. Here we notice that PGD 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. Furthermore, we rank the examples in PGDfail according to the difference of the highest and the second highest of the ten network outputs. The smaller the difference, the easier it should be to find an adversarial example. Indeed as can be seen from the table, our method finds 45 new adversarial examples out of the first 100 such ranked examples.

The experiments above suggest that our algorithms can lead to improved adversarial attacks. We would like to note that the recent work of [31] also studied semi-definite programming based methods for providing adversarial certificates for 2-layer neural networks. However, our SDP as outlined is Figure 4 is strictly stronger. In particular, the SDP of [31] is independent of the given example x and as a result 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.

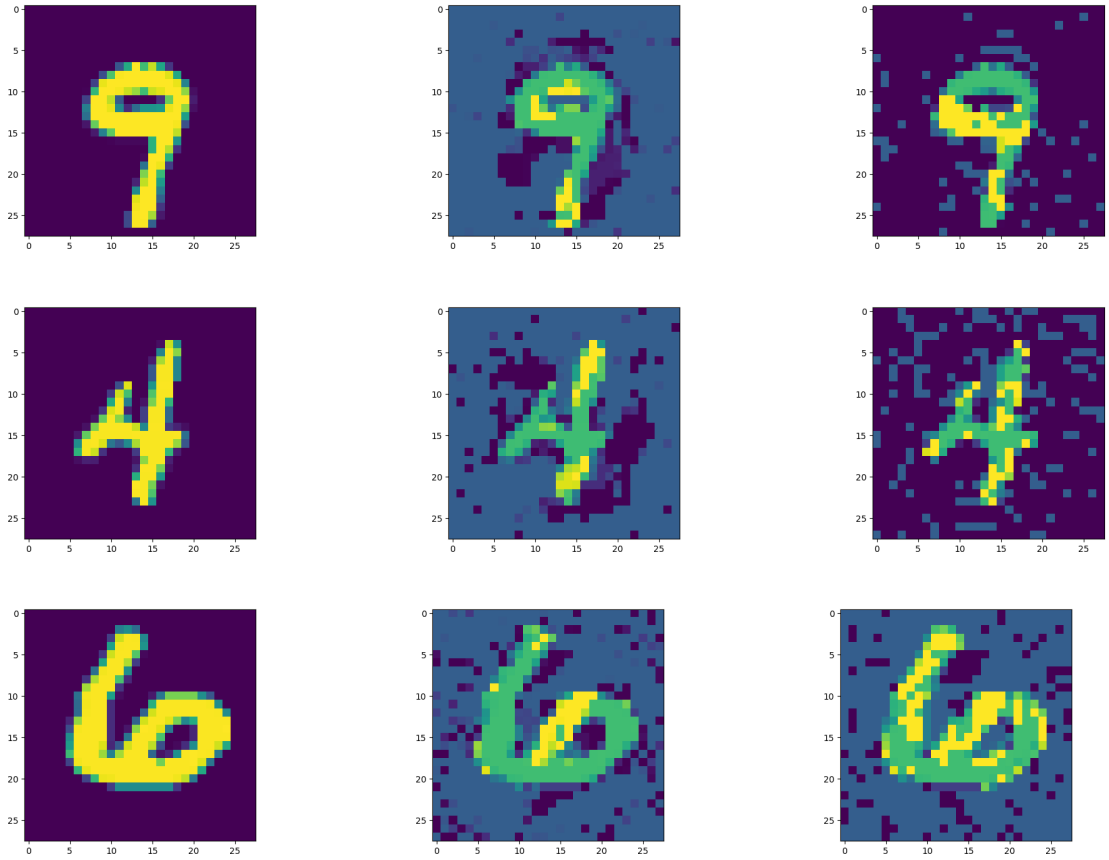

Figure 5: The figure shows three MNIST random samples from PDGfail (i.e., examples where PGDattack failed to find an adversarial perturbation), where SDPattack successfully finds adversarial perturbations for $\delta = 0.3$. The images in the first column represent the original images corresponding to three, the second column represents the perturbed images produced by the failed PGDattack, and perturbed images produced by the successful SDPattack. Visual inspection of these examples suggest that our method often produces sparse targeted perturbations.

$\delta = 0.3$	PGDpass (6×50 random samples)	PGDfail (8×100 random samples)
SDP succeds	297 out of 300 total Mean : 49.5 of 50, Std : 0.76	244 out of 800 total Mean 30.6 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 of 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.

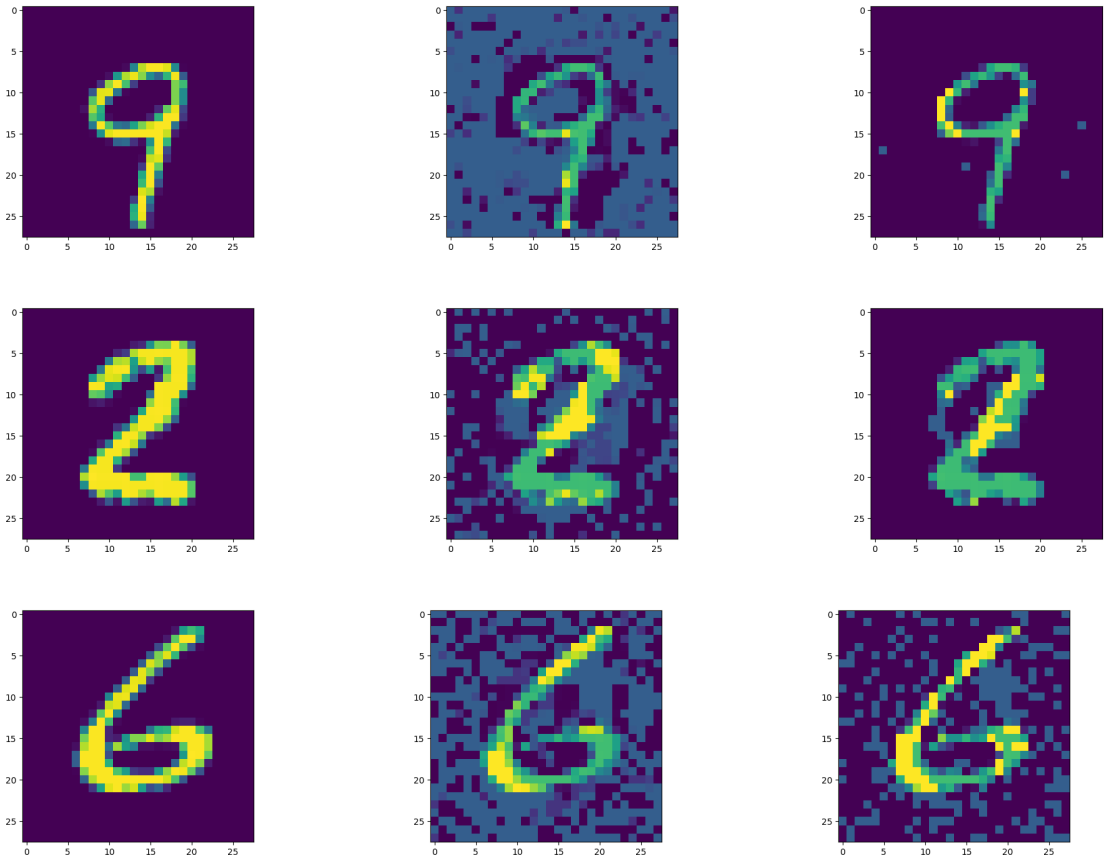

Figure 6: The figure shows three MNIST random samples from PDGpass (i.e., examples where PGDattack succeeded to find an adversarial perturbation), where SDPattack successfully finds adversarial perturbations for $\delta = 0.3$. The images in the first column represent the original images corresponding to three, the second column represents the perturbed images produced by the successful PGDattack, and perturbed images produced by the successful SDPattack. Visual inspection of these examples suggest that our method often produces sparse targeted perturbations.

9 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 5 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 5. 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] Aharon Ben-Tal and Arkadi Nemirovski. Robust solutions of uncertain linear programs. *Operations research letters*, 25(1):1–13, 1999.

[6] Dimitris Bertsimas and Melvyn Sim. The price of robustness. *Operations research*, 52(1):35–53, 2004.

[7] 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.

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

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

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

[11] 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.

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

[13] 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.

[14] Laurent El Ghaoui and Hervé Lebret. Robust solutions to least-squares problems with uncertain data. *SIAM Journal on matrix analysis and applications*, 18(4):1035–1064, 1997.

[15] 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.

[16] 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.

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

[18] 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.

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

[20] 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.

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

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

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

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

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

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

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

[33] 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.

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

[35] 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.

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

[37] 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.

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

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

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

A Proofs for the Computational Hardness Results

A.1 Proofs of Claim 6.6 and Lemma 6.4

We first prove Claim 6.6 that helps establish the YES case.

Proof of Claim 6.6.

Proof. It is easy to check that $sgn(x^TAx - z)$ classifies all of S correctly.
Robustness at $((\mathbf{0}, 2\delta), -1)$. Follows from the fact that we are in the YES case and hence $\max_{x \in B_\infty^n(0,\delta)} x^TAx < \delta^2 s = \delta$.
Robustness at $((\mathbf{0}, 1), -1), ((\mathbf{0}, \tau'), -1), ((\mathbf{0}, -1), +1), ((\mathbf{0}, -\tau'), +1)$. Follows from the fact that we are in the YES case and hence $\max_{x \in B_\infty^n(0,\delta)} x^tAx < \delta^2 s = 1/s < 1/100$ and that $\tau' > n/(20\delta) > 5n$.
Robustness at $((\mathbf{0}, 2\delta), -1), ((\mathbf{0}, -2\delta), +1)$. Follows from the fact that we are in the YES case and hence $\max_{x \in B_\infty^n(0,\delta)} x^TAx < \delta^2 s = \delta$ and that $\varepsilon n/10 = 20\delta$.
Robustness at $((\mathbf{e}_i, \gamma), -1), ((\mathbf{e}_i, -\gamma), +1), ((-\mathbf{e}_i, \gamma), -1), ((-\mathbf{e}_i, -\gamma), +1)$. Let's argue robustness at $((\mathbf{e}_i, \gamma), -1)$ and the other calculations are similar. The maximum value of x^TAx in a δ-ball around $\mathbf{e}_i$ is at most

$$(\tau + \delta)\delta \sum_j |a_{i,j}| + \delta^2 s.$$

Hence to establish robustness we need that

$$(\tau+\delta)\delta\sum_j |a_{i,j}| + \delta^2 s \leq \gamma - \delta. \tag{19}$$

Substituting the value of δ and noticing that γ, τ are much larger than $\delta = 1/s < 1/100$ we get that it is enough for the following to hold

$$2\tau\delta\sum_j |a_{i,j}| \leq \frac{\gamma}{2}. \tag{20}$$

In other words we need that

$$\frac{\gamma}{\tau} \geq 4\delta\sum_j |a_{i,j}| \tag{21}$$

Substituting the values of γ, τ we get that

$$n \geq \delta\sum_j |a_{i,j}| \tag{22}$$

This is true since $\delta = 1/s$ and the fact that $s \geq \frac{1}{n}\sum_{i,j}|a_{i,j}| > \frac{1}{n}\sum_j |a_{i,j}|$ where the first inequality is from [11].

Robustness at $((\mathbf{e}_{i,j}, 2), -1), ((\mathbf{e}_{-i,j}, 2), -1), ((\mathbf{e}_{i,-j}, 2), -1), ((\mathbf{e}_{-i,-j}, 2), -1)$. Let's argue robustness at $((\mathbf{e}_{i,j}, 2), -1)$ and the other calculations are similar. The maximum value of $x^T Ax$ in a δ-ball around $\mathbf{e}_{i,j}$ is at most

$$\frac{2\delta \max_i \sum_j |a_{i,j}|}{\sqrt{2(\varepsilon + |a_{i,j}|)}} + \delta^2 s + 1$$

Hence to establish robustness we need that

$$\frac{2\delta \max_i \sum_j |a_{i,j}|}{\sqrt{2(\varepsilon + |a_{i,j}|)}} + \delta^2 s + 1 \leq 2 - \delta. \tag{23}$$

Noticing that $\delta = 1/s$ and much smaller than $1/100$, we get that it is enough for the following to hold

$$\frac{\delta \max_i \sum_j |a_{i,j}|}{\sqrt{2(\varepsilon + |a_{i,j}|)}} \leq \frac{1}{4}. \tag{24}$$

This is again true since $\delta = 1/s$ and by our assumption $|a_{i,j}| \geq 4$ for non-zero entries of A. □

Robustness at $((2\mathbf{e}_{i,j}, 1), \operatorname{sgn}(a_{i,j})), ((2\mathbf{e}_{-i,j}, 1), -\operatorname{sgn}(a_{i,j})), ((2\mathbf{e}_{i,-j}, 1), -\operatorname{sgn}(a_{i,j})), ((2\mathbf{e}_{-i,-j}, 1), \operatorname{sgn}(a_{i,j}))$. We'll argue robustness at $((2\mathbf{e}_{i,j}, 1), +1)$ and the other calculations are similar. Also for simplicity, assume $\operatorname{sgn}(a_{i,j} > 0$. The other case is similar. The minimum value of $x^T Ax$ in a δ-ball arond $\mathbf{e}_{i,j}$ is at least

$$2 - \frac{2\delta \max_i \sum_j |a_{i,j}|}{\sqrt{2(\varepsilon + |a_{i,j}|)}} - \delta^2 s$$

So for robustness, we need

$$2 - \frac{2\delta \max_i \sum_j |a_{i,j}|}{\sqrt{2(\varepsilon + |a_{i,j}|)}} - \delta^2 s > 1 + \delta$$

This is true since we have

$$\frac{\delta \max_i \sum_j |a_{i,j}|}{\sqrt{2(\varepsilon + |a_{i,j}|)}} \leq \frac{1}{4}.$$

□

Proof of Lemma 6.4. We now prove the key lemma that is used in the analysis of our reduction.

Proof. First we prove that if $q'(x,z)$ has zero error on S then c_z must be non zero. Then it is clear that if $q'(x,z)$ has zero error on S, then so does $q(x,z)$. Consider the case when $c_z = 0$. Now $q'(x,z)$ classifies S_1 correctly. More specifically, it classifies the two points $((\mathbf{0},1),-1)$ and $((\mathbf{0},-1),1)$ correctly. This gives us the following equations

$$c_2 + c_4 < 0$$
$$c_2 + c_4 > 0$$

and hence we get a contradiction. Moving on to the main part of the proof about the coefficients of $q(x,z)$, the constraints at $(\mathbf{0},1),(\mathbf{0},-1),(\mathbf{0},\tau'),(\mathbf{0},-\tau')$ give us

$$c_2 - 1 + c_4 < 0 \tag{25}$$

$$c_2 + 1 + c_4 > 0 \tag{26}$$

$$\tau'^2 c_2 - \tau' + c_4 < 0 \tag{27}$$

$$\tau'^2 c_2 + \tau' + c_4 > 0 \tag{28}$$

From (25) and (26) we get that

$$-1 < c_2 + c_4 < 1 \tag{29}$$

Similarly, from (27) and (28) we get that

$$-\tau' < \tau'^2 c_2 + c_4 < \tau' \tag{30}$$

This implies that $|c_2| < 1/(\tau' - 1) < \varepsilon/10$ for $\tau' = \Omega(1/\varepsilon)$.

The constraints at $((\mathbf{0},2\delta),-1),((\mathbf{0},-2\delta)$ gives us that

$$4c_2\delta^2 - 2\delta + c_4 < 0$$
$$4c_2\delta^2 + 2\delta + c_4 > 0$$

From the above equations we get that

$$|c_4| \leq c_2(2\delta)^2 + 2\delta < 4\delta. \tag{31}$$

The constraints at $(\mathbf{e}_i,\gamma),(-\mathbf{e}_i,\gamma),(\mathbf{e}_i,-\gamma),(-\mathbf{e}_i,-\gamma)$ give us

$$\tau^2 a'_{i,i} + \tau c_{1,i} + c_2\gamma^2 - \gamma + c_4 + \tau\gamma\beta_i < 0 \tag{32}$$

$$\tau^2 a'_{i,i} - \tau c_{1,i} + c_2\gamma^2 - \gamma + c_4 - \tau\gamma\beta_i < 0 \tag{33}$$

$$\tau^2 a'_{i,i} + \tau c_{1,i} + c_2\gamma^2 + \gamma + c_4 - \tau\gamma\beta_i > 0 \tag{34}$$

$$\tau^2 a'_{i,i} - \tau c_{1,i} + c_2\gamma^2 + \gamma + c_4 + \tau\gamma\beta_i > 0 \tag{35}$$

From (32) and (35) we get that

$$\tau c_{1,i} < \gamma \tag{36}$$

Similarly, from (33) and (34) we get that

$$\tau c_{1,i} > -\gamma \tag{37}$$

Plugging back into the equations above we get that

$$-(4\delta + 2\gamma + \frac{\gamma^2}{\tau' - 1}) < \tau^2 a'_{i,i} + \tau\gamma\beta_i < (4\delta + 2\gamma + \frac{\gamma^2}{\tau' - 1}) \tag{38}$$

and

$$-(4\delta + 2\gamma + \frac{\gamma^2}{\tau' - 1}) < \tau^2 a'_{i,i} - \tau\gamma\beta_i < (4\delta + 2\gamma + \frac{\gamma^2}{\tau' - 1}) \tag{39}$$

This implies that

$$|a'_{i,i}| \leq \frac{1}{\tau^2}(4\delta + 2\gamma + \frac{\gamma^2}{\tau' - 1}) \leq \varepsilon/10$$

for $\tau' = \Omega(\frac{n^2}{\varepsilon}) \max(1, 1/\min_{i,j} |a_{i,j}|), \tau = \Omega(\frac{n}{\varepsilon}) \max(1, 1/\min_{i,j} |a_{i,j}|)$, $\gamma = 4n\tau$. We also get that

$$|\beta_i| \leq \frac{1}{\tau\gamma}(4\delta + 2\gamma + \frac{\gamma^2}{\tau' - 1}) \leq \varepsilon/10$$

for $\tau' = \Omega(\frac{n^2}{\varepsilon}) \max(1, 1/\min_{i,j} |a_{i,j}|), \tau = \Omega(\frac{n}{\varepsilon}) \max(1, 1/\min_{i,j} |a_{i,j}|)$, $\gamma = 4n\tau$.

The constraints at $(\mathbf{e}_{i,j}, 2), (\mathbf{e}_{-i,j}, 2), (\mathbf{e}_{i,-j}, 2), (\mathbf{e}_{-i,-j}, 2)$ give us

$$\frac{a'_{i,i}}{2\tilde{a}_{i,j}} + \frac{a'_{j,j}}{2\tilde{a}_{i,j}} + \frac{a'_{i,j}}{\tilde{a}_{i,j}} + \frac{c_{1,i}}{\sqrt{2\tilde{a}_{i,j}}} + \frac{c_{1,j}}{\sqrt{2\tilde{a}_{i,j}}} + 4c_2 - 2 + c_4 + \frac{2\beta_i}{\sqrt{2\tilde{a}_{i,j}}} + \frac{2\beta_j}{\sqrt{2\tilde{a}_{i,j}}} < 0 \tag{40}$$

$$\frac{a'_{i,i}}{2\tilde{a}_{i,j}} + \frac{a'_{j,j}}{2\tilde{a}_{i,j}} - \frac{a'_{i,j}}{\tilde{a}_{i,j}} - \frac{c_{1,i}}{\sqrt{2\tilde{a}_{i,j}}} + \frac{c_{1,j}}{\sqrt{2\tilde{a}_{i,j}}} + 4c_2 - 2 + c_4 - \frac{2\beta_i}{\sqrt{2\tilde{a}_{i,j}}} + \frac{2\beta_j}{\sqrt{2\tilde{a}_{i,j}}} < 0 \tag{41}$$

$$\frac{a'_{i,i}}{2\tilde{a}_{i,j}} + \frac{a'_{j,j}}{2\tilde{a}_{i,j}} - \frac{a'_{i,j}}{\tilde{a}_{i,j}} + \frac{c_{1,i}}{\sqrt{2\tilde{a}_{i,j}}} - \frac{c_{1,j}}{\sqrt{2\tilde{a}_{i,j}}} + 4c_2 - 2 + c_4 + \frac{2\beta_i}{\sqrt{2\tilde{a}_{i,j}}} - \frac{2\beta_j}{\sqrt{2\tilde{a}_{i,j}}} < 0 \tag{42}$$

$$\frac{a'_{i,i}}{2\tilde{a}_{i,j}} + \frac{a'_{j,j}}{2\tilde{a}_{i,j}} + \frac{a'_{i,j}}{\tilde{a}_{i,j}} - \frac{c_{1,i}}{\sqrt{2\tilde{a}_{i,j}}} - \frac{c_{1,j}}{\sqrt{2\tilde{a}_{i,j}}} + 4c_2 - 2 + c_4 - \frac{2\beta_i}{\sqrt{2\tilde{a}_{i,j}}} - \frac{2\beta_j}{\sqrt{2\tilde{a}_{i,j}}} < 0 \tag{43}$$

where $\tilde{a}_{i,j} = \varepsilon + |a_{i,j}|$. Combining (40) and (43) we get

$$\frac{a'_{i,i}}{2\tilde{a}_{i,j}} + \frac{a'_{j,j}}{2\tilde{a}_{i,j}} + \frac{a'_{i,j}}{\tilde{a}_{i,j}} + 4c_2 - 2 + c_4 < 0 \tag{44}$$

From this we get that

$$\frac{a'_{i,j}}{\tilde{a}_{i,j}} < 2 + 4\delta + 4\frac{\varepsilon}{10} + \frac{4\delta + 2\gamma + \frac{\gamma^2}{\tau' - 1}}{\tau^2 \min_{i,j} |a_{i,j}|} < 2 + 4\delta + \varepsilon \tag{45}$$

for large enough τ. Similarly, combining (41) and (42) we get

$$\frac{a'_{i,i}}{2\tilde{a}_{i,j}} + \frac{a'_{j,j}}{2\tilde{a}_{i,j}} - \frac{a'_{i,j}}{\tilde{a}_{i,j}} + 4c_2 - 2 + c_4 < 0 \tag{46}$$

From this we get that

$$\frac{a'_{i,j}}{\tilde{a}_{i,j}} > -2 - 4\delta - \varepsilon. \tag{47}$$

The constraints at $(\mathbf{e}_{i,j}, -2), (\mathbf{e}_{-i,j}, -2), (\mathbf{e}_{i,-j}, -2), (\mathbf{e}_{-i,-j}, -2)$ give us

$$\frac{a'_{i,i}}{2\tilde{a}_{i,j}} + \frac{a'_{j,j}}{2\tilde{a}_{i,j}} + \frac{a'_{i,j}}{\tilde{a}_{i,j}} + \frac{c_{1,i}}{\sqrt{2\tilde{a}_{i,j}}} + \frac{c_{1,j}}{\sqrt{2\tilde{a}_{i,j}}} + 4c_2 + 2 + c_4 + \frac{2\beta_i}{\sqrt{2\tilde{a}_{i,j}}} + \frac{2\beta_j}{\sqrt{2\tilde{a}_{i,j}}} > 0 \tag{48}$$

$$\frac{a'_{i,i}}{2\tilde{a}_{i,j}} + \frac{a'_{j,j}}{2\tilde{a}_{i,j}} - \frac{a'_{i,j}}{\tilde{a}_{i,j}} - \frac{c_{1,i}}{\sqrt{2\tilde{a}_{i,j}}} + \frac{c_{1,j}}{\sqrt{2\tilde{a}_{i,j}}} + 4c_2 + 2 + c_4 - \frac{2\beta_i}{\sqrt{2\tilde{a}_{i,j}}} + \frac{2\beta_j}{\sqrt{2\tilde{a}_{i,j}}} > 0 \tag{49}$$

$$\frac{a'_{i,i}}{2\tilde{a}_{i,j}} + \frac{a'_{j,j}}{2\tilde{a}_{i,j}} - \frac{a'_{i,j}}{\tilde{a}_{i,j}} + \frac{c_{1,i}}{\sqrt{2\tilde{a}_{i,j}}} - \frac{c_{1,j}}{\sqrt{2\tilde{a}_{i,j}}} + 4c_2 + 2 + c_4 + \frac{2\beta_i}{\sqrt{2\tilde{a}_{i,j}}} - \frac{2\beta_j}{\sqrt{2\tilde{a}_{i,j}}} > 0 \tag{50}$$

$$\frac{a'_{i,i}}{2\tilde{a}_{i,j}} + \frac{a'_{j,j}}{2\tilde{a}_{i,j}} + \frac{a'_{i,j}}{\tilde{a}_{i,j}} - \frac{c_{1,i}}{\sqrt{2\tilde{a}_{i,j}}} - \frac{c_{1,j}}{\sqrt{2\tilde{a}_{i,j}}} + 4c_2 + 2 + c_4 - \frac{2\beta_i}{\sqrt{2\tilde{a}_{i,j}}} - \frac{2\beta_j}{\sqrt{2\tilde{a}_{i,j}}} > 0 \tag{51}$$

Combining (40) and (49) we get

$$\frac{a'_{i,j}}{\tilde{a}_{i,j}} + \frac{c_{1,i}}{\sqrt{2\tilde{a}_{i,j}}} - 2 + \frac{2\beta_i}{\sqrt{2\tilde{a}_{i,j}}} < 0 \tag{52}$$

From this we get that

$$c_{1,i} < (4\delta + \varepsilon)\sqrt{2\tilde{a}_{i,j}} \tag{53}$$

for large enough τ. Similarly, from (50) and (42) we get

$$c_{1,i} > -(4\delta + \varepsilon)\sqrt{2\tilde{a}_{i,j}}. \tag{54}$$

Finally, the constraints at $(2\mathbf{e}_{i,j}, 1), (2\mathbf{e}_{-i,j}, 1), (2\mathbf{e}_{i,-j}, 1), (2\mathbf{e}_{-i,-j}, 1)$ give us

$$2\frac{a'_{i,i}}{\tilde{a}_{i,j}} + 2\frac{a'_{j,j}}{\tilde{a}_{i,j}} + 4\frac{a'_{i,j}}{\tilde{a}_{i,j}} + \frac{2c_{1,i}}{\sqrt{2\tilde{a}_{i,j}}} + \frac{2c_{1,j}}{\sqrt{2\tilde{a}_{i,j}}} + c_2 - 1 + c_4 + \frac{4\beta_i}{\sqrt{2\tilde{a}_{i,j}}} + \frac{4\beta_j}{\sqrt{2\tilde{a}_{i,j}}} > 0 \tag{55}$$

$$2\frac{a'_{i,i}}{\tilde{a}_{i,j}} + 2\frac{a'_{j,j}}{\tilde{a}_{i,j}} - 4\frac{a'_{i,j}}{\tilde{a}_{i,j}} - \frac{2c_{1,i}}{\sqrt{2\tilde{a}_{i,j}}} + \frac{2c_{1,j}}{\sqrt{2\tilde{a}_{i,j}}} + c_2 - 1 + c_4 - \frac{4\beta_i}{\sqrt{2\tilde{a}_{i,j}}} + \frac{4\beta_j}{\sqrt{2\tilde{a}_{i,j}}} < 0 \tag{56}$$

$$2\frac{a'_{i,i}}{\tilde{a}_{i,j}} + 2\frac{a'_{j,j}}{\tilde{a}_{i,j}} - 4\frac{a'_{i,j}}{\tilde{a}_{i,j}} + \frac{2c_{1,i}}{\sqrt{2\tilde{a}_{i,j}}} - \frac{2c_{1,j}}{\sqrt{2\tilde{a}_{i,j}}} + c_2 - 1 + c_4 + \frac{4\beta_i}{\sqrt{2\tilde{a}_{i,j}}} - \frac{4\beta_j}{\sqrt{2\tilde{a}_{i,j}}} < 0 \tag{57}$$

$$2\frac{a'_{i,i}}{\tilde{a}_{i,j}} + 2\frac{a'_{j,j}}{\tilde{a}_{i,j}} + 4\frac{a'_{i,j}}{\tilde{a}_{i,j}} - \frac{2c_{1,i}}{\sqrt{2\tilde{a}_{i,j}}} - \frac{2c_{1,j}}{\sqrt{2\tilde{a}_{i,j}}} + c_2 - 1 + c_4 - \frac{4\beta_i}{\sqrt{2\tilde{a}_{i,j}}} - \frac{4\beta_j}{\sqrt{2\tilde{a}_{i,j}}} > 0 \tag{58}$$

Combining (55) and (58) we get

$$2\frac{a'_{i,i}}{\tilde{a}_{i,j}} + 2\frac{a'_{j,j}}{\tilde{a}_{i,j}} + 4\frac{a'_{i,j}}{\tilde{a}_{i,j}} + c_2 - 1 + c_4 > 0 \tag{59}$$

From this we get that

$$\frac{a'_{i,j}}{\tilde{a}_{i,j}} > \frac{1}{4} - \delta - \frac{\varepsilon}{4} \tag{60}$$

for large enough τ. Similarly, combining (56) and (57) we get

$$2\frac{a'_{i,i}}{\tilde{a}_{i,j}} + 2\frac{a'_{j,j}}{\tilde{a}_{i,j}} - 4\frac{a'_{i,j}}{\tilde{a}_{i,j}} + c_2 - 1 + c_4 < 0 \tag{61}$$

From this we get that

$$\frac{a'_{i,j}}{\tilde{a}_{i,j}} > -\frac{1}{4} - \delta - \frac{\varepsilon}{4} \tag{62}$$

for large enough τ.

□

A.2 Proof of Weak Robust Learning

In this section we prove Theorem 6.3, which in turns uses the non-distributional hardness in Theorem A.10. But to begin with we first prove an alternate NP hardness result. Although weaker than the hardness result of the previous section, this will help us prove the more robust bound. More formally, we will prove that

Theorem A.1. *[Hardness] For every $\delta > 0$, assuming $NP \neq RP$ there is no polynomial time algorithm that given a set of $N = O(n^2)$ labeled points $\{ (x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)}) \}$ with $(x^{(j)}, y^{(j)}) \in \mathbb{R}^{n+1} \times \{ -1, 1 \}$ for all $j \in [N]$ can determine whether there exists a degree-2 PTF that has δ-robust empirical error of 0 on these N points.*

The above theorem immediately implies the following result about hardness of optimal robust learning of degree-2 PTFs.

Corollary A.2. *[Distributional Hardness] For every $\delta > 0$, there exists an $\varepsilon > 0$ such that assuming $NP \neq RP$ 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 $\mathrm{poly}(N)$ and distinguishes between the following two cases:*

- 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.*

We again reduce from the QP problem (Problem $\mathcal{QP}$) which is known to be NP hard. The reduction is sketehced below.

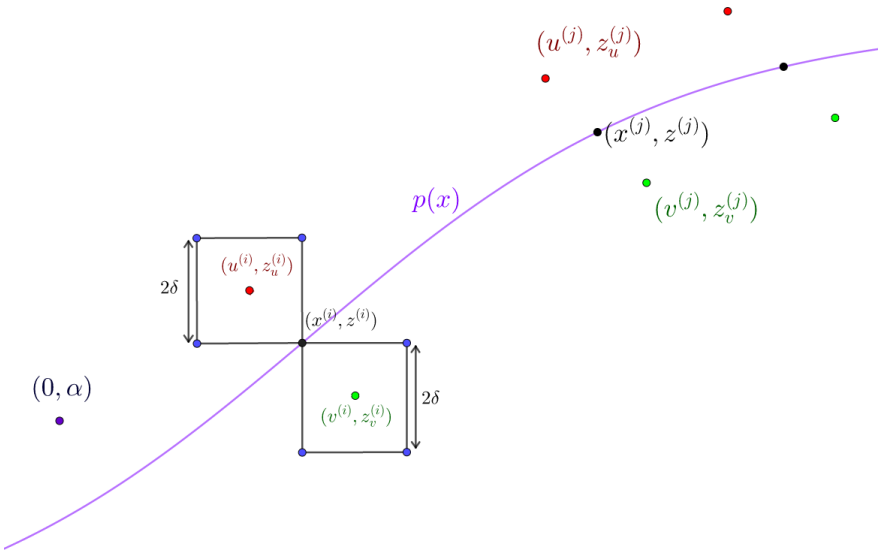

Figure 8: *The figure shows the construction of a hard instance for the robust learning problem. First, points $(x^{(j)}, z^{(j)})$ are sampled randomly and staisfying $z^{(j)} = p(x^{(j)})$. Each such point is then perturbed along the direction of the sign vector of the gradient at $(x^j, z^{(j)})$ to get two data points of the training set, one labeled as $+1$, and the other labeled as -1.*

1. Let $p(x) := x^T Ax$ be the polynomial given by Problem $\mathcal{QP}$, and let β, δ be the given parameters. Set $\alpha := \delta^2\beta + \delta, \rho := c_3\delta n^{3/2}m$, for some sufficiently large constant $c_3 \geq 1$.
2. Using A we generate m points $(x^{(j)}, z^{(j)}) \in \mathbb{R}^{n+1}$ as follows. Sample point $x^{(j)}$ from $\mathbb{N}(0, \rho^2)^n$, then set $z^{(j)} = p(x^{(j)}) = (x^{(j)})^T Ax^{(j)}$ for each $j \in [m]$.
3. Define $s^{(j)} = \text{sgn}(\nabla p(x^{(j)}))$ where the $\text{sgn}(x) \in \{-1, 1\}^n$ refers to a vector with entry-wise signs, and ∇p stands for the gradient of p at $x^{(j)}$. From each $(x^{(j)}, z^{(j)})$ generate $(u^{(j)}, z_u^{(j)}) = (x^{(j)} - \delta s^{(j)}, z^{(j)} + \delta)$ with label $y_u^{(j)} = \text{sgn}(z_u^{(j)} - p(u^{(j)}))$ and $(v^{(j)}, z_v^{(j)}) = (x^{(j)} + \delta s^{(j)}, z^{(j)} - \delta)$ with label $y_v^{(j)} = \text{sgn}(z_v^{(j)} - p(v^{(j)}))$.
4. Generate α (depends on δ and β from problem $\mathcal{QP}$) and input the $2m + 1$ points in $\mathbb{R}^{n+1} \times \{\pm 1\}$ given by $((u^{(j)}, z_u^{(j)}), y_u^{(j)})$, $((v^{(j)}, z_v^{(j)}), y_u^{(j)})$ for each $j \in [m]$ and $(0, \alpha, +1)$ to the algorithm.

Figure 7: Reduction from the QP problem.

To argue the soundness and the completeness of our reduction, we will first analyze the robustness of degree-2 PTFs on the $2m$ added labeled examples $((u^{(\ell)}, z_u^{(\ell)}), y_u^{(\ell)})$ and $((v^{(\ell)}, z_v^{(\ell)}), y_v^{(\ell)})$. We will show that the "intended" PTF $\text{sgn}(z - p(x))$ is the *unique* degree-2 PTF (up to scaling) that is robust at all these $2m$ points. Note that a degree-2 PTF $\text{sgn}(q(x, z))$ on the $n+1$ variables (x, z) may *not* necessarily be of the form $\text{sgn}(z - g(x))$ for some degree-2 polynomial $g(x)$. We need to rule out the existence of any other degree-2 PTF of the form $\text{sgn}(q(x, z))$ that is δ-robust at these points. Once we have established this, we will then show that the "intended" PTF $\text{sgn}(z - p(x))$ is δ-robust at $((0, \alpha), +1)$ in the YES case, and not δ-robust at

$((0,\alpha),+1)$ in the NO case.

We proceed by first proving that the intended PTF $\mathrm{sgn}(z-p(x))$ is robust at the $2m$ added examples. Recall that the points $x^{(j)} \in \mathbb{R}^n$ are chosen according to a Gaussian distribution with variance ρ^2 in every direction. The following lemma shows a property that holds w.h.p. for the points $\{\, x^{(\ell)} : \ell \in [m] \,\}$ that will be key in proving the robustness of $\mathrm{sgn}(z-p(x))$ at the $2m$ added points in Lemma A.5.

Lemma A.3. *There exists some universal constant $C>0$ such that for any $\eta>0$, assuming $\rho \geq C\delta n^{3/2}m/\eta$ we have with probability at least $1-\eta$ that*

$$\forall \ell \in [m],\ \forall i \in [n],\ \frac{|\langle A_i, x^{(\ell)}\rangle|}{\|A_i\|_1} > \delta, \tag{63}$$

where A_i denotes the ith row of A.

Proof. The proof follows from the following standard anti-concentration fact about Gaussians.

Fact A.4. *Let x^* be sampled from $\mathbb{N}(0,\rho^2)^n$. Let $a \in \mathbb{R}^n$. There exists a universal constant $C>0$ such that for any $\eta'>0$,*

$$\mathbb{P}\Big[|\langle a, x^*\rangle| \leq C\|a\|_2\rho\eta\Big] \leq \eta'.$$

Set $\eta' := \eta/(mn)$. Fix $\ell \in [m], i \in [n]$. Using Fact A.4 we have with probability at least $1-\eta'$

$$|\langle A_i, x^{(j)}\rangle| \geq \|A_i\|_2\rho\eta' \geq \frac{\|A_i\|_1}{\sqrt{n}} \cdot \rho \cdot \frac{\eta}{mn} \geq \delta,$$

from our assumption on ρ. The lemma follows from a union bound over all $\ell \in [m], i \in [n]$. □

We now prove the δ-robustness of the "intended" degree-2 PTF $\mathrm{sgn}(z-p(x))$ at the $2m$ added points w.h.p.

Lemma A.5. *There exists constant $C>0$ such that for any $\eta>0$, assuming $\rho \geq C\delta n^{3/2}m/\eta$, then with probability at least $1-\eta$, the degree-2 PTF $\mathrm{sgn}(z-p(x)) = \mathrm{sgn}(z-x^TAx)$ is δ-robust at all the $2m$ points $\{\, ((u^{(\ell)}, z_u^{(\ell)}), y_u^{(\ell)}), ((v^{(\ell)}, z_v^{(\ell)}), y_v^{(\ell)}) : \ell \in [m] \,\}$.*

Proof. Consider a fixed $\ell \in [m]$. For convenience let x^*, z^*, u, v, z_u, z_v denote $x^{(\ell)}, z^{(\ell)}, u^{(\ell)}, v^{(\ell)}, z_u^{(\ell)}, z_v^{(\ell)}$ respectively, and let $s = \mathrm{sgn}(\nabla p(x^{(\ell)})) \in \{\,-1,1\,\}^n$. Hence $z^* = x^{*T}Ax^*$, $(u, z_u) = (x^* - \delta s, z^* + \delta)$ and $(v, z_v) = (x^* + \delta s, z^* - \delta)$. We want to prove that the points (u, z_u) and (v, z_v) are δ robust i.e., these points are δ away in ℓ_∞ distance from the decision boundary of $\mathrm{sgn}(z-p(x))$. We now prove the following claim:
Claim. *Any point $(x,z) \in B_\infty^{n+1}(u, z_u)$ is on the 'positive' side i.e., $z - x^TAx > 0$.*

Note that (u, z_u) itself lies inside the ball, and hence the claim will show that $\mathrm{sgn}(z-x^TAx)$ is δ-robust at (u, z_u). An analogous proof also holds that δ-robustness at (v, z_v).
Proof of Claim. Let's now define $\tilde{x} = x - x^*$, $\tilde{z} = z - z^*$. A simple observation is that (x,z) lies on the opposite orthant with respect to (x^*, z^*) as s , and we have (as shown in Figure 9)

$$\forall j \in [d], -2\delta \leq s(j)\tilde{x}(j) \leq 0, \quad \text{and } \tilde{z} \geq 0.$$

Using $z^* = p(x^*)$ and $\tilde{z} \geq 0$, for all $(x,z) \in B^{n+1}((u, z_u), \delta)$ we have

$$\begin{aligned}
z - p(x) &= z^* + \tilde{z} - p(\tilde{x} + x^*) = \tilde{z} + p(x^*) - p(\tilde{x} + x^*) = \tilde{z} - \langle \nabla p, \tilde{x}\rangle - \frac{1}{2}\tilde{x}^T\nabla^2 p\tilde{x} \\
&\geq -\sum_{i=1}^n \tilde{x}(i)\Big(\sum_{j=1}^n a_{ij}x^*(j)\Big) - \frac{1}{2}\sum_{i=1}^n \tilde{x}(i)\Big(\sum_{j=1}^n a_{ij}\tilde{x}(j)\Big) \\
&= \sum_{i=1}^n (-\tilde{x}(i)s(i))\Big|\sum_{j=1}^n a_{ij}x^*(j)\Big| - \frac{1}{2}\sum_{i=1}^n \tilde{x}(i)\sum_{j=1}^n a_{ij}\tilde{x}(j) \\
&\geq \sum_{i=1}^n |\tilde{x}(i)|\Big(\Big|\sum_{j=1}^n a_{ij}x^*(j)\Big| - \delta\sum_{j=1}^n |a_{ij}|\Big),
\end{aligned}$$

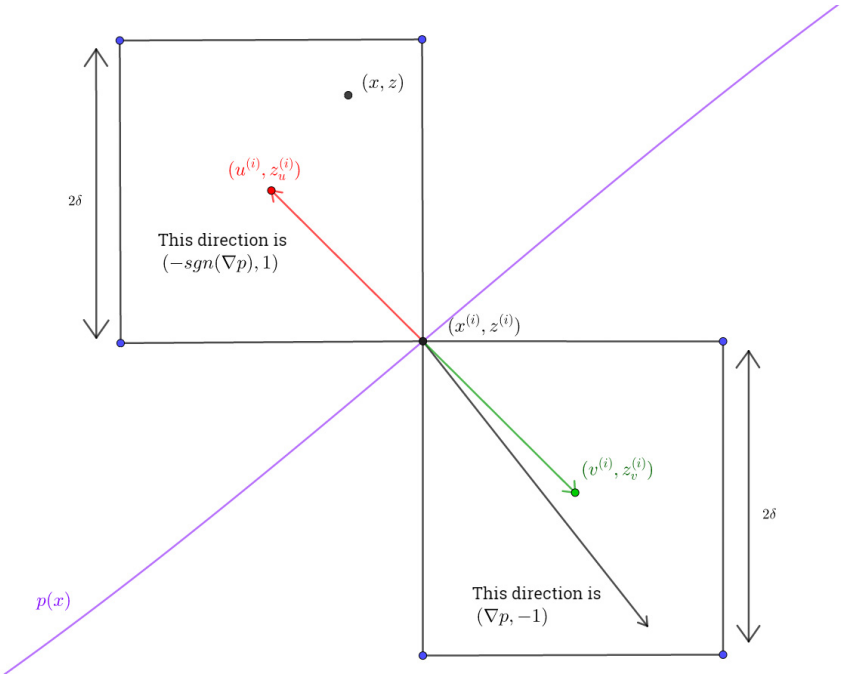

Figure 9: *The figure shows the radius of robustness around the point* $(x^{(i)}, z^{(i)})$. Any degree-2 PTF that is δ-robust at all the data points, must take a value of $+1$ in the upper ball around each $(x^{(i)}, z^{(i)})$ of ℓ_∞ radius of 2δ and must take a value of -1 in the lower ball around each $(x^{(i)}, z^{(i)})$ of ℓ_∞ radius of 2δ. We use this fact to establish that such a PTF must pass through the points $(x^{(i)}, z^{(i)})$.

using the fact that $\tilde{x}(i)s(i) \in [-2\delta, 0]$ for each $i \in [n]$. Applying Lemma A.3 we see that with probability at least $(1-\eta)$, (63) holds, and hence $|\langle x^*, A_i \rangle| > \delta \|A_i\|_1$ for each $i \in [n]$ as required. This establishes the claim, and proves the lemma.

□

We now prove that the "intended" PTF $\text{sgn}(z - p(x))$ is the only degree-2 PTF that is robust at the added $2m$ examples.

Lemma A.6. *Consider any degree-2 PTF* $\text{sgn}(q(x,z))$ *that is* δ*-robust at the* $2m$ *labeled points* $\{ ((u^{(\ell)}, z_u^{(\ell)}), +1) : \ell \in [m] \}$ *and* $\{ ((v^{(\ell)}, z_v^{(\ell)}), -1) : \ell \in [m] \}$ *and is consistent with their labels. Then* $q(x,z) = C(z - p(x))$ *for some* $C \neq 0$.

The proof of Lemma A.6 follows immediately from the following two lemmas (Lemma A.7 and Lemma A.8).

Lemma A.7. *Consider any degree-2 PTF on* $n+1$ *variables* $\text{sgn}(q(x,z))$ *that satisfies the conditions of Lemma A.6. Then* $q(x^{(\ell)}, z^{(\ell)}) = 0$ *for each* $\ell \in [m]$.

Proof. Since $\text{sgn}(q(u^{(\ell)}, z_u^{(\ell)})) \neq sgn(q(v^{(\ell)}, z_v^{(\ell)}))$, by the Intermediate Value Theorem,

$$\exists \gamma \in [0,1] \text{ s.t. } (\widehat{x}, \widehat{z}) = \gamma(u^{(\ell)}, z_u^{(\ell)}) + (1-\gamma)(v^{(\ell)}, z_v^{(\ell)}) \text{ and } q(\widehat{x}, \widehat{z}) = 0.$$

Also, since q is δ-robust at $(u^{(\ell)}, z_u^{(\ell)})$ and $(v^{(\ell)}, z_v^{(\ell)})$, we must have that $(\widehat{x}, \widehat{y})$ is atleast δ far away in ℓ_∞ distance from both $(u^{(\ell)}, z_u^{(\ell)})$ and $(v^{(\ell)}, z_v^{(\ell)})$. Further by design two points are separated by exactly 2δ in each co-ordinate (see Figure 9 for an illustration)! Hence it is easy to see that $\gamma = 1/2$ i.e., $(\widehat{x}, \widehat{z}) = (x^{(\ell)}, z^{(\ell)})$ as required.

□

We now show that $q(x,z) = z - p(x)$ is the only polynomial over $(n+1)$ variables that evaluates to 0 on all points $\{ (x^{(\ell)}, z^{(\ell)}) : \ell \in [m] \}$. Together with Lemma A.7 this establishes the proof of Lemma A.6.

Lemma A.8. *Let $m > (n+1)^2$ and let $q : \mathbb{R}^{n+1} \to \mathbb{R}$ be any degree-2 polynomial with $q(x^{(\ell)}, z^{(\ell)}) = 0$ for all $\ell \in [m]$, where $z^{(\ell)} = (x^{(\ell)})^T A^* x^{(\ell)}$ and $x^{(\ell)} \sim N(0, \rho^2)^n$ with $\rho > 0$. Then with probability 1, $q(x,z) = C(z - x^T A^* x)$ for $C \neq 0$.*

Proof. We can represent a general degree-2 polynomial $q : \mathbb{R}^{n+1} \to \mathbb{R}$ given by

$$q(x,z) = x^T A x + b_1^T x + c_1 + z b_2^T x + c_2 z^2 + c_3 z, \text{ where } x \in \mathbb{R}^n, z \in \mathbb{R}.$$

This polynomial is parameterized by a vector $w = (A, b_1, c_1, b_2, c_2, c_3) \in \mathbb{R}^r$ where $r = \binom{n+1}{2} + 2n + 3$ (since A is symmetric). Now given a point $(x^{(\ell)}, z^{(\ell)})$, the equation $q(x^{(\ell)}, z^{(\ell)}) = 0$ is a linear equation over the coefficients w of q. Hence, the set of conditions $q(x^{(\ell)}, z^{(\ell)}) = 0$ can be expressed as a systems of linear equations $Mw = 0$ over the (unknown) co-efficients w. Hence any valid polynomial q corresponds to a solution of the linear system $Mw = 0$ and vice-versa. We now describe the matrix $M \in \mathbb{R}^{m \times r}$. Define

$$\begin{aligned} f(x,z) &:= (1) \oplus (x_1, \ldots, x_n) \oplus (x_i x_j : i \leq j \in [n]) \oplus (x_1 z, \ldots x_n z) \oplus (z^2), \oplus(z) \in \mathbb{R}^r, \\ \text{and } M_\ell &:= f(x^{(\ell)}, z^{(\ell)}) \; \forall \ell \in [m], \end{aligned}$$

where $u \oplus v$ refers to the concatenation of vectors u and v, and M_ℓ represents the row ℓ of M. In other words $f(x,z) = (1, x_1, \ldots, x_n, x_1^2, \ldots, x_j x_k, \ldots, x_n^2, x_1 z, \ldots, x_j z \ldots, x_n z, z^2, z)$, where x_j is the jth component of x and $z = x^T A^* x$. Observe that the "intended" polynomial $q^*(x,z) = z - x^T A^* x$ is a valid solution to this system of equations. Hence, it will suffice to prove that M has rank exactly $r-1$ i.e., M has full column rank minus one. First observe that as polynomials over the formal variables x, z, *all but one* of the columns of f are linearly independent – in fact the only linear dependency in $f(x,z)$ corresponds to the column z that can be expressed as a linear combination of degree-2 monomials $\{ x_i x_j : i \leq j \}$ since $z := x^T A^* x$ is a homogenous degree-2 polynomial. Further the columns $\{ x_j z : j \in [n] \}$ have degree 3 and z^2 has degree 4. Hence excluding the column corresponding to z, it is easy to see that the rest of the columns are linearly independent (either they correspond to different monomials, or the degrees are different). Now define $g(x,z), M'$ analogously to $f(x,z)$ and M respectively, without the last column that corresponds to z i.e.,

$$\begin{aligned} g(x,z) &:= (1) \oplus (x_1, \ldots, x_n) \oplus (x_i x_j : i \leq j \in [n]) \oplus (x_1 z, \ldots x_n z) \oplus (z^2) \in \mathbb{R}^{r-1}, \\ \text{and } M'_\ell &:= g(x^{(\ell)}, z^{(\ell)}) \; \forall \ell \in [m]. \end{aligned}$$

From our earlier discussion, the columns of $g(x,z)$ when seen as polynomials over the formal variables x, z are linearly independent. Hence, it suffices to prove the following claim:
Claim: *M' has full column rank i.e., rank of M' is r.*

To see why the claim holds consider the first ℓ rows of the matrix M' and look at their span $S(R_\ell)$. If $\ell \leq r-1$ then the space orthogonal to $S(R_\ell)$ i.e., $S(R_\ell)^\perp$ is non-empty. Consider any direction v in $S(R_\ell)^\perp$.

$$\langle v, M'_{\ell+1} \rangle = \widehat{q}(x^{(\ell+1)}, z^{(\ell+1)}), \text{ where } \widehat{q}(x,z) := \langle v, g(x,z) \rangle$$

is a non-zero polynomial of degree 2 in x, z (it is not identically zero because the columns of $g(x,z)$ are linearly independent as polynomials over x, z). Hence using a standard result about multivariate polynomials evaluated at randomly chose points (See Fact A.9), we get that $\widehat{q}(x^{(\ell+1)}, z^{(\ell+1)}) \neq 0$ and so $\langle v, M'_{\ell+1} \rangle \neq 0$ with probability 1. Taking a union bound over all the $\ell \in \{ 1, \ldots, r \}$ completes the proof.

□

Fact A.9. *A non-zero multivariate polynomial $p : \mathbb{R}^n \to \mathbb{R}$ evaluated at a point $x \sim N(0, \rho^2)^n$ with $\rho > 0$ evaluates to zero with zero probability.*

We remark that the statement of Lemma A.8 can also be made robust to inverse polynomial error by using polynomial anti-concentration bounds (e.g., Carbery-Wright inequality) instead of Fact A.9; however this is not required for proving NP-hardness. We now complete the proof of Theorem A.1.

Proof of Theorem A.1. We start with the NP-hardness of $\mathcal{QP}$, and for the reduction in Figure 7, we will show that in the YES case, we will show that there is a δ-robust degree-2 PTF (completeness), and in the NO case we will show that there is no δ robust degree-2 PTF (soundness). As a reminder, the NP-hard problem $\mathcal{QP}$ is the following: given a symmetric matrix $A \in \mathbb{R}^{n\times n}$ with zeros on diagonals, and $\beta > 0$ distinguish whether

No Case : there exists an assignment y^* with $\|y^*\|_\infty \le 1$ such that $q(y^*) = (y^*)^T A y^* > \beta$,

Yes Case : $\max_{\|y\|_\infty \le 1} y^T A y < \beta$.

Completeness (**Yes Case**): Consider the degree-2 PTF given by $\mathrm{sgn}(z - p(x)) = \mathrm{sgn}(z - x^T A x)$. From Lemma A.5, we have that it is δ robust at the $2m$ points $\{ ((u^{(\ell)}, z_u^{(\ell)}), y_u^{(\ell)}) : \ell \in [m] \}$ and $\{ ((v^{(\ell)}, z_v^{(\ell)}), y_v^{(\ell)}) : \ell \in [m] \}$ with probability at least $1 - \eta$ (for η being any sufficiently small constant). Further, from multilinearity of p we have that,

$$\max_{\|y\|_\infty \le \delta} y^T A y = \delta^2 \max_{\|y\|_\infty \le 1} y^T A y < \delta^2 \beta = \alpha - \delta.$$
$$\text{Hence } (\alpha - \delta) - \max_{\|y\|_\infty \le \delta} y^T A y > 0,$$

which establishes robustness at $((0, \alpha), +1)$ for $\mathrm{sgn}(z - x^T A x)$. Hence $\mathrm{sgn}(z - p(x))$ is δ-robust at the $2m + 1$ points with probability at least $1 - \eta$ (for η being any sufficiently small constant).

Soundness (**No Case**): From Lemma A.6, we see that the degree-2 PTF given by $\mathrm{sgn}(z - p(x)) = \mathrm{sgn}(z - x^T A x)$ is the only degree-2 PTF that can potentially be robust at all the $2m + 1$ points with probability 1. Again analyzing robustness at the example $((0, \alpha), +1)$, we see that from multilinearity of p,

$$\max_{\|y\|_\infty \le \delta} y^T A y = \delta^2 \max_{\|y\|_\infty \le 1} y^T A y > \delta^2 \beta = \alpha - \delta.$$
$$\text{Hence } (\alpha - \delta) - \max_{\|y\|_\infty \le \delta} y^T A y < 0,$$

which shows that the degree-2 PTF $\mathrm{sgn}(z - p(x))$ is *not* robust at $(0, \alpha)$. Hence there is no δ-robust degree-2 PTF at the $2m + 1$ given points, with probability 1. This completes the analysis of the reduction, and establishes the theorem.

□

Stronger Hardness. We now prove the robust lower bound stated below.

Theorem A.10. *[Stronger Hardness] For every $\delta > 0$ and $\varepsilon \in (0, \frac{2}{7})$, assuming NP $\neq$ RP there is no polynomial time algorithm that given a set of $N = \mathrm{poly}(n, 1/\varepsilon)$ labeled points $\{ (x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)}) \}$ in $\mathbb{R}^{n+1} \times \{ -1, 1 \}$ such that there is a degree-2 PTF with δ-robust empirical error of 0, can output a degree-2 PTF that has δ-robust empirical error of at most ε on these N points.*

Proof. The proof of this theorem closely follows the proof of Theorem A.1 (the $\varepsilon = 0$ case), so we only point out the differences here. The reduction uses the same gadget (Figure 7) used in Theorem A.1. The main challenge is the soundness analysis (NO case), where we need to rule out the existence of degree-2 PTFs which are δ-robust and consistent at all but an ε fraction of the points. To handle this, we introduce "redundancy" by including more points (of both kinds) to ensure that even when an arbitrary ε fraction of these points are ignored (the PTF makes errors on them), we can still use the arguments in the soundness analysis of Theorem A.1.

Recall that our reduction (see Figure 7) generated two sets of points. We have one point of the form $(0, \alpha)$ (let us denote this type as *Type A*) and m pairs of points $\{ (u^{(\ell)}, z_u^{(\ell)}), (v^{(\ell)}, z_v^{(\ell)}) : \ell \in [m] \}$ which are obtained by modifying $(x^{(\ell)}, z^{(\ell)} = p(x^{(\ell)}))$ with $x^{(\ell)}$ generated randomly (let us denote these $2m$ points as of *Type B*).

Set $N_1 := n^3, N_2 := 2n^3$. In our modified instance, we will have N_1 points of Type A i.e., N_1 identical points $(0, \alpha)$ (note that we can also perturb these points slightly so that they are all distinct, if required). Further, we will have N_2 points of Type B i.e., we will generate $N_2/2$ pairs of points $\{ (u^{(\ell)}, z_u^{(\ell)}) : \ell \in [N_2/2] \}$

which are generated as described in Figure 7 after drawing $x^{(\ell)} \sim N(0, \rho^2)^n$ for $\ell \in [N_2/2]$ (here a larger $\rho = O(\delta n^{3/2} N_2)$ will suffice). Hence, we have in total $N = N_1 + N_2 = 3n^3$ points.

The *completeness* analysis (YES case) is identical to that of Theorem A.1, as $\mathrm{sgn}(z - p(x))$ will be δ-robust at all of the N points (from Lemma A.5 and our choice of α).

We now focus on the *soundness* analysis (NO case). From $\varepsilon < \frac{1}{3}$ and our choice of N_1 and N_2,

$$N_1 > \varepsilon(N_1 + N_2) \tag{64}$$

$$(1 - \varepsilon)(N_1 + N_2) > N_1 + \frac{N_2}{2} + (n+1)^2 \tag{65}$$

From (65) and a pigeonhole argument, any subset of size $(1 - \varepsilon)(N_1 + N_2)$ is guaranteed to have $(n+1)^2$ pairs of points of the form $(u^{(\ell)}, z_u^{(\ell)})$ and $(v^{(\ell)}, z_v^{(\ell)})$. This is because the LHS of (65) represents a lower bound on the number of points the candidate degree-2 PTF is robust on. The RHS of (65) represents the number of points needed to ensure that atleast $(n+1)^2$ pairs of points from Type B are picked. Hence using Lemma A.6 along with a union bound over all the $\binom{N_2}{(n+1)^2}$ choices of the pairs (note that the failure probability in Lemma A.8 is 0), the "intended" PTF $\mathrm{sgn}(z - p(x))$ is the only surviving degree-2 PTF.

Again from (64) and the pigeonhole principle, any $(1 - \varepsilon)$ fraction of the points will contain *atleast one* point of the Type A i.e., $(0, \alpha)$. Hence in the NO case, the "intended" PTF $\mathrm{sgn}(z - p(x))$ is not δ-robust. This completes the soundness analysis and establishes the theorem.

□