Adversarial Examples in Simpler Settings

by

Tony T. Wang

S.B., Mathematics and Computer Science and Engineering,
Massachusetts Institute of Technology (2020)

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2021

Author .....
Department of Electrical Engineering and Computer Science
May 20, 2021

Certified by .....
Gregory W. Wornell
Sumitomo Professor of Engineering
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Adversarial Examples in Simpler Settings

by

Tony T. Wang

Submitted to the Department of Electrical Engineering and Computer Science on May 20, 2021, in partial fulfillment of the requirements for the degree of Master of Engineering in Electrical Engineering and Computer Science.

Abstract

In this thesis we explore adversarial examples for simple model families and simple data distributions, focusing in particular on linear and kernel classifiers. On the theoretical front we find evidence that natural accuracy and robust accuracy are more likely than not to be misaligned. We conclude from this that in order to learn a robust classifier, one should explicitly aim for it either via a good choice of model family or via optimizing explicitly for robust accuracy. On the empirical front we discover that kernel classifiers and neural networks are non-robust in similar ways. This suggests that a better understanding of kernel classifier robustness may help unravel some of the mysteries of adversarial examples.

Thesis Supervisor: Gregory W. Wornell
Title: Sumitomo Professor of Engineering

Acknowledgments

First and foremost thanks goes out to my advisor Greg Wornell for supporting me on this project and for teaching me the value in both big picture thinking and starting small.

Second, thanks goes out to my collaborator Yang Yan, who made this whole endeavor much less lonely.

Third, thanks goes out to Yuheng Bu, who provided many thoughtful comments and literature suggestions and was pleasure to work with in and out of this project.

Next, I wish to thank all those that I spoke with over the past year who gave me important insights into adversarial examples, including but not limited to: Simon Alford, Dylan Hadfield-Menell, Horace He, Qian Huang, and David Rolnick. Thanks also goes out to Gary Lee and Tejas Jayashankar for helping me access our lab's computing resources.

Finally, I am grateful for all my friends and family without whom I would be truly lost.

Contents

List of Figures

List of Tables

Chapter 1

Adversarial examples: an overview

In its 2012 debut at the ImageNet competition, AlexNet [10] set a record of 84.7% for top-5 accuracy, beating the second place model by over 10% [8]. In the years since, deep vision networks have achieved top-5 accuracies of 98.7% [26], surpassing humans who top out at around 95% [19]. Given these amazing results, one may be tempted to conclude that we have solved the problem of image classification. Adversarial examples are evidence that this is not true.

It turns out deep neural networks trained using standard methods are actually very fragile. Given an arbitrary input x, one can usually perturb x a tiny amount to $x + \epsilon$ such that the network outputs nonsense when fed $x + \epsilon$ as input [11]. A visualization of this effect for an image classifier is given in Figure 1.0.1.

For a more concrete sense of the extent of this fragility, consider a ResNet-50 neural network trained on ImageNet. Such a model achieves 76% top-1 accuracy, which corresponds to around 93% top-5 accuracy [15]. However, if we adversarially perturb the test inputs by at most 2/255 intensity-levels per pixel, the same ResNet-50 achieves a top-1 accuracy of **0.036%** [22]. 2/255 intensity-levels per pixel is imperceptible to a human, but this is enough to completely break a standard ResNet-50.

Moreover, adversarial vulnerability is also extremely hard to get rid of. While we do have methods that increase the robustness of a network, even state of the art robust models don't come close to reaching human levels of robustness.

Figure 1.0.1: An adversarial chair. The left image was taken by the author using a smartphone and is of a chair in his residence. When fed to CLIP [16], a cutting edge multimodal image classification model, CLIP says the image is a chair with 99.99% confidence. However, by perturbing the left image in a slight[1] but adversarial manner to the image on the right, CLIP changes its prediction and says the right image is a hat with 95.56% confidence. The almost imperceptibly perturbed image on the right is called an "adversarial example".

[1] Up to 2.55/255 intensity levels per pixel.

For example, on CIFAR10, human accuracy is around 94% [9]. We feel this number is roughly unchanged even when an adversary is able to perturb up to 8/255 intensity-levels per pixel, since from Figure 1.0.2 it appears that such perturbations are near imperceptible. So let's be generous and say human robust accuracy is 90% under the 8/255 threat model. State of the art neural networks on the other hand are only able to achieve 66.56% robust accuracy [4].

We can summarize the empirical phenomena of adversarial examples thusly:

1. Adversarial examples are ubiquitous and severe in naturally trained networks.

2. Adversarial examples are less severe but ever-present in our best attempts at robust models.

This is a sobering state of affairs, especially when it comes to deploying neural networks in the real world where there are potentially life threatening consequences to adversarial examples.[1]

[1]For example, researchers have performed proof of concept adversarial attacks in which they placed small stickers on a road that caused a Tesla to swerve into the wrong lane [1].

However, adversarial examples are not all bad news. Fundamentally, the existence of adversarial examples suggests that artificial neural networks may be doing some critically different things than biological neural networks, which are much more robust. Thus understanding the phenomenon of adversarial examples may 1) help us understand what current ANNs are really doing under the hood and 2) reveal key things biological NNs have that current ANNs don't have. Advancing this understanding is the goal of this thesis.

1.1 Open problems in adversarial examples

We feel there are three major open problems in the field of adversarial examples:

(A) What are naturally trained neural networks learning that enables them to have high natural accuracy but simultaneously abysmal robust accuracy?

(B) How can we train neural networks to have high robust accuracy? Is it even possible?

(C) How do we build a machine in practice that has high robust accuracy (like a human)?

Below, we give a (incomplete) summary of what existing literature has to say about these questions.

1.1.1 Open problem A: What are ANNs learning?

In the vision domain, [7] show that neural networks heavily utilize non-robust features that are easily flipped via small imperceptible perturbations, but nonetheless encode a large amount of signal. The utilization of these non-robust features is so great that by manipulating just these non-robust features in the training data, networks can be made to learn arbitrary classification functions.

In [6], it is shown that when trained to classify two nested high dimensional spheres (using standard training methods), simple fully connected neural networks

are highly non-robust. This is shocking given the fact that it is easy to construct even by hand a fully connected NN that robustly classifies the two nested spheres.

In [14], a simple demonstration is given in which label noise causes adversarial examples in a neural network. Thus ANNs are very sensitive to label noise, which makes sense given that they are often trained to perfectly fit their training data. It is unclear how much this influences adversarial examples in practice though, since adversarial examples appear even on close to perfectly clean data.

1.1.2 Open problem B: How can we train robust ANNs, if at all?

In [12], the authors introduce *adversarial training*, a block coordinate ascent/descent algorithm for training robust models. Standard adversarial training yields close to state of the art results for many tasks.

1.1.3 Open problem C: Building robust machines

Much of the existing theoretical work on adversarial examples is most relevant towards open problem C. The results of this theoretical work is best understood in terms of guidelines for what to do when it comes to building a robust machine. We review a few of the most important guidelines below.

- **Optimize explicitly for robustness.** In [23], it is shown that there exist scenarios in which natural accuracy and robust accuracy are at odds with each other. That is to say, making one too big will necessarily make the other small. Thus if robustness is the goal, it is generally wise to explicitly optimize for it.

- **Pick the right model family.** In [13] it is shown that there are scenarios in which using the wrong model family causes non-robustness to be inevitable, and using the right model family makes learning a robust classifier possible and easy. In addition, [20] gives a similar construction where using the wrong model family causes the sample complexity of learning a robust classifier to

blow up. Overall, the message here is that the right model family can make all the difference when it comes building a robust machine.

- **Have enough training data.** In [20], it is shown that learning a robust model can sometimes take significantly more training data than learning a well-performing but non-robust model.

- **Try to have clean training data.** In [2], it is shown that when data labels are noisy, interpolative kernel methods (i.e., methods that get 100% training accuracy) learn classifiers that blow in norm and are non-robust. Likewise, [14] also gives a simple demonstration of this fact for neural networks, which is also an interpolative method in practice. Though a robust machine need not be an interpolator, it still seems wise to try and have clean training data when possible.

- **Have enough compute.** In [3], the authors construct a scenario in which learning a robust model is computationally intractable even when learning an accurate but non-robust model is computationally easy. Though the construction is a bit pathological and is not very reflective of real data distributions, it nonetheless shows it is possible for robust learning to take more compute. Thus having ample compute at one's disposal would probably not hurt when it comes to building a robust machine.

1.2 Outline of thesis

This thesis is centered on the exploration of simple datasets and models that demonstrate some degree of adversarial example phenomena. We use our observation in these simple settings to comment on and conjecture solutions to open problems A, B, and C. The remaining chapters are laid out in the following manner:

- In Chapter 2 we study the joint behavior of natural and robust accuracy, and conclude that there are at least two different ways of tackling problem C. One

could either design optimization processes that better promote robustness, or design better model architectures that are "automatically" robust.

- In Chapter 3, we analyze linear classifiers on a simple Gaussian dataset. We discover that on this dataset, misalignment between natural accuracy and robust accuracy is the norm rather than the exception. This reinforces the idea that solving open problem C must involve explicitly optimizing/designing for robustness.

- In Chapter 4, we study adversarial example for kernel classifiers. We find that classical kernel classifiers are vulnerable to adversarial examples in similar ways to neural networks. From this, we conjecture that classical kernel classifiers may hold some secrets to what neural networks are doing under the hood (problem A).

  We also provide some weak evidence that classical kernel classifiers are fundamentally unable to be robust. Due to the similarity between kernel classifiers and neural networks, this is also weak evidence for the impossibility of open problem B.

- Finally in Chapter 5, we study some empirical properties of neural networks. We find further evidence suggesting that neural networks and classical kernel classifiers may be non-robust for similar reasons (problem A).

Figure 1.0.2: Visualization of different levels of pixel perturbations on CIFAR10 images. The images are the first ten images of the CIFAR10 test set. For each perturbation size ϵ, the pixel intensities in the left-half of each image are decreased by ϵ and clipped to zero if they go negative. A maximum perturbation of size 255/255 results in an completely black left-half. Note that a perturbation of $\epsilon = 8/255$ is near-imperceptible.

Chapter 2

Alignment

In this chapter we develop a formal framework for studying the interplay between natural accuracy and robust accuracy. Our main idea is to look at the global joint behavior of the two losses, and we develop notions for when this joint behavior is aligned and when it is not.

We then introduce a simple dataset on which natural accuracy is misaligned with robust accuracy, and demonstrate how modifying the model family can eliminate misalignment. We thus conclude that there are multiple roads to robustness.

Finally, we show under certain conditions the tradeoff curve between natural and robust error is convex.

2.1 Formalizing adversarial robustness

When people say a classifier "is vulnerable to adversarial examples" or "lacks adversarial robustness" what they generally mean is that the robust accuracy of the classifier is much lower than the natural accuracy of the classifier. Let us formalize this notion.

Let $F : \mathcal{X} \to \mathcal{Y}$ be a classifier, let $X, Y \sim P_{X,Y}$ be random variables representing a random datapoint and its label, and let $\mathcal{A} : \mathcal{X} \to 2^{\mathcal{X}}$ denote a perturbative threat

model (i.e., a datapoint x can be adversarially perturbed to any point in $\mathcal{A}(x)$).[1]

We can write the natural error of F on $P_{X,Y}$ as

$$L^{\text{nat-err}}(F) \triangleq \mathbb{P}_{P_{X,Y}}(F(X) \neq Y) \tag{2.1.1}$$

and the robust error of F on $P_{X,Y}$ with respect to threat model $\mathcal{A}$ as

$$L^{\text{rob-err}}_{\mathcal{A}}(F) \triangleq \mathbb{P}_{P_{X,Y}}(F(\mathcal{A}(X)) \neq \{Y\}) \tag{2.1.2}$$

Thus a more formal way to say F "lacks adversarial robustness" is to say

$$L^{\text{nat-err}}(F) \ll L^{\text{rob-err}}_{\mathcal{A}}(F). \tag{2.1.3}$$

To understand when Equation (2.1.3) occurs, it will be helpful to understand the joint behavior of $L^{\text{nat-err}}$ and $L^{\text{rob-err}}$ over a family of classifiers $\mathcal{F}$. This will be the focus of later sections. Before we move on to this joint behavior however, let us first introduce a useful alternate form of robust error.

2.1.1 Adversarial robustness as distributional robustness

An alternative way of writing robust error is

$$L^{\text{rob-err}}_{\mathcal{A}}(F) = \max_{Q_{X,Y} \in B_{\mathcal{A}}(P_{X,Y})} \mathbb{P}_{X,Y \sim Q_{X,Y}}(F(X) \neq Y), \tag{2.1.4}$$

where $B_{\mathcal{A}}(P_{X,Y})$ is the set of all probability distributions over X, Y that can be obtained from $P_{X,Y}$ be perturbing X to some element in $\mathcal{A}(X)$. More precisely, $Q_{X,Y} \in B_{\mathcal{A}}(P_{X,Y})$ if and only if there exist three random variables X, X', Y where the variables X, Y have joint distribution $P_{X,Y}$, the variables X', Y have joint distribution $Q_{X,Y}$, and $\mathbb{P}(X' \in \mathcal{A}(X)) = 1$.

The expression in Equation (2.1.4) is written in distributional robustness form,

[1] The most commonly studied perturbative threat models in the literature [21] are those of ϵ-ℓ_p perturbations for $p \in \{0, 2, \infty\}$, where $\mathcal{A}(x) = \{x + \Delta \,|\, \|\Delta\|_p \leq \epsilon\}$. In this work, our analysis will center primarily on the $p = 2$ case.

and reveals that adversarial robustness is a special case of distributional robustness.[2] This form will prove useful for us both for mathematical convenience, and because it allows us to treat minimizing robust error as a minimax game, enabling us to make statements about equilibria.

2.2 Joint loss behavior and loss alignment

Let us now cover some general properties of of joint loss behavior. We define a loss function $L : \mathcal{Y}^{\mathcal{X}} \to [0,1]$ to be a function that maps a classifier $F : \mathcal{X} \to \mathcal{Y}$ to a real number $L(F) \in [0,1]$. As the name implies, smaller values of loss functions are more desirable. The joint behavior of two loss functions L_1, L_2 is captured via the notion of a joint achievable region, defined as follows.

Definition 2.2.1. *The **joint achievable region** of two losses $L_1, L_2 : \mathcal{Y}^{\mathcal{X}} \to [0,1]$ across a family of classifiers $\mathcal{F} \subseteq \mathcal{Y}^{\mathcal{X}}$ is the subset of $[0,1]^2$ defined by*

$$\mathcal{R}(L_1, L_2, \mathcal{F}) \triangleq \{(L_1(F), L_2(F)) \,|\, F \in \mathcal{F}\}. \tag{2.2.1}$$

To avoid cumbersome technical conditions, we assume throughout this work that joint achievable regions are closed sets.[3]

Since smaller losses are more desirable, we can also define a notion of classifier efficiency and a notion of an efficient frontier.

Definition 2.2.2. *A classifier $F \in \mathcal{F}$ is **efficient** with respect to two losses L_1, L_2 if there does not exist a classifier in $F' \in \mathcal{F}$ with strictly better performance, i.e. $L_1(F') \leq L_1(F)$ and $L_2(F') \leq L_2(F)$ with at least one of the inequalities strict.*

Definition 2.2.3. *The **efficient frontier** with respect to losses L_1, L_2 across a family of classifiers $\mathcal{F}$ is the subset of $\mathcal{R}(L_1, L_2, \mathcal{F})$ generated by efficient classifiers. In other words:*

$$\mathcal{E}(L_1, L_2, \mathcal{F}) \triangleq \{(L_1(F), L_2(F)) \,|\, F \in \mathcal{F}, F \text{ efficient}\}. \tag{2.2.2}$$

[2] A related but separate problem of study from that of adversarial examples. Some have proposed that adversarial example research may benefit from taking more of a distributional robustness viewpoint [6].

[3] A sufficient condition for this would be for $\mathcal{F}$ to be compact and L_1 and L_2 to be continuous over $\mathcal{F}$.

Now, let us define some notions of loss alignment, which describes the extent to which minimizing loss L_1 is equivalent to minimizing loss L_2.

Definition 2.2.4. *Two losses L_1 and L_2 are **strongly aligned** (or alternatively have **strong alignment**) over $\mathcal{F}$ if*

$$L_1(F) < L_1(F') \quad \iff \quad L_2(F) < L_2(F')$$

for all $F, F' \in \mathcal{F}$.

Note that strong alignment is both symmetric and reflexive (i.e. any loss is always strongly aligned with itself). When two losses are strongly aligned, optimizing for one is just a good as optimizing for the other (modulo computational concerns).

The following weaker notion of alignment will also prove to be useful:

Definition 2.2.5. *Two losses L_1 and L_2 are **aligned in the limit** (or alternatively have **limit alignment**) over $\mathcal{F}$ if*

$$F_* \in \operatorname*{argmin}_{F \in \mathcal{F}} L_1(F) \qquad \iff \qquad f_* \in \operatorname*{argmin}_{F \in \mathcal{F}} L_2(F).$$

Strong alignment always implies limit alignment, but not necessarily the other way around. Limit alignment of L_1 and L_2 means that minimizing L_1 *perfectly* is equivalent to minimizing L_2 *perfectly*. Emphasis here on *perfectly* here, since weak positive alignment makes no guarantees on alignment outside of perfect minimization.

The complement to limit alignment is limit misalignment:

Definition 2.2.6. *Two losses L_1 and L_2 are **misaligned in the limit** (or alternatively have **limit misalignment**) over $\mathcal{F}$ if the efficient frontier $\mathcal{E}(L_1, L_2, \mathcal{F})$ has more than one element.*

Limit misalignment implies there is a genuine tradeoff involved in optimizing L_1 and L_2, i.e., minimizing one objective necessarily leaves the other sub-optimal. When limit misalignment is present, one should explicitly optimize for the objective they care about, since optimizing for the other may be actively harmful.

On occasion we will also speak of the "alignment" of two functions where larger values of the functions more desirable. In this case we really refer to the alignment of properly scaled negations of the quantities, which are loss functions. As an example, the two statements “natural accuracy is strongly aligned with robust accuracy” and “natural error is strongly aligned with robust error” are equivalent when accuracy = 1 − error.

2.3 Alignment between natural and robust error

We are now ready to study the alignment of $L^{\text{nat-err}}$ and $L^{\text{rob-err}}_{\mathcal{A}}$. We assume in this section that $\mathcal{A}$ is non-degenerate, meaning $x \in \mathcal{A}(x)$ always holds. Due to this non-degeneracy, it holds for any classifier F that

$$L^{\text{nat-err}}(F) \leq L^{\text{rob-err}}_{\mathcal{A}}(F). \tag{2.3.1}$$

Next we explore a simple example where natural error and robust error are misaligned in the limit.

Consider the following data model with data X and labels Y,

$$\begin{aligned} Y &\sim \text{Unif}(\{\text{False}, \text{True}\}) \\ X &= (X_1, X_2) \\ X_1 &= Y \\ X_2 &= \begin{cases} Y, & \text{w. prob } (1-p) \\ \neg Y, & \text{w. prob } p \end{cases} \end{aligned}$$

with an adversary that can perturb X_1 at will:

$$\mathcal{A}(x_1, x_2) = \{(\text{False}, x_2), (\text{True}, x_2)\}$$

The joint achievable region for this dataset over the set $\mathcal{F}$ of all possible *random-*

ized[4] classifiers is visualized in Figure 2.3.1.

This dataset is a simple demonstration of the fact that natural error and robust error can be negatively aligned. By varying p, the degree of negative alignment can be changed. In particular, for $|p| < 1$, any classifier achieving optimal natural error necessarily achieves 100% robust error.

Thus in order to get a classifier with low robust error on this simple dataset, we must explicitly optimize for robustness. This is not the only way to get low robust error though. We could instead change the model family to get robustness. Indeed when we restrict the model family to randomized classifiers that only use the X_2 feature, natural error and robust error become perfectly aligned.

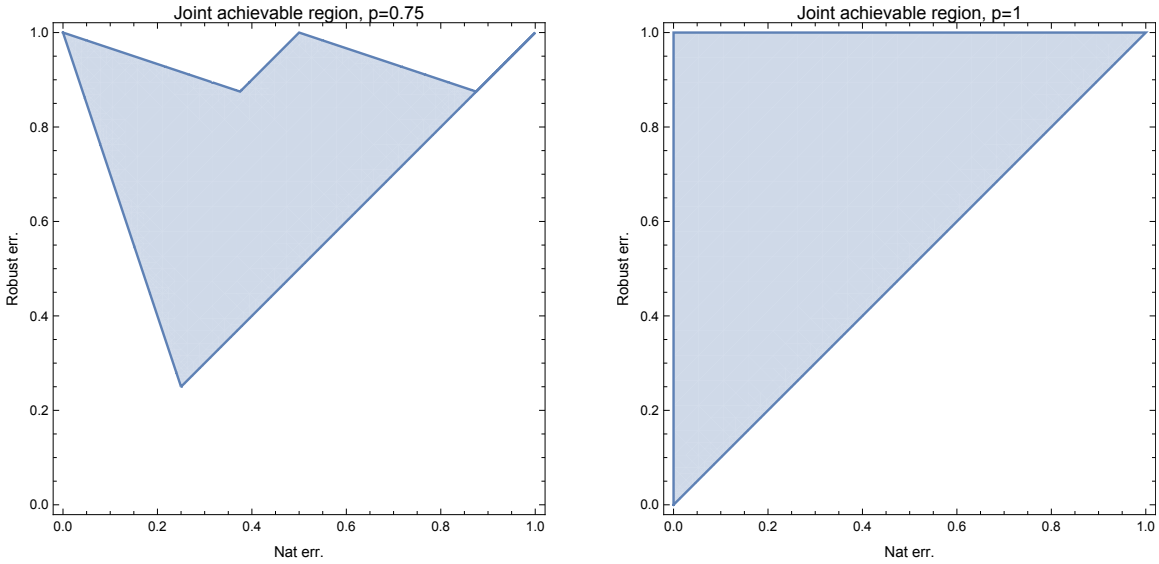

Figure 2.3.1: Joint achievable regions of natural and robust error for a simple dataset with a non-robust feature for different settings of p.

2.4 Shape of tradeoff between natural and robust error

In the simple dataset of the previous section, the efficient frontier of natural and robust error is linear (as can be seen in Figure 2.3.1). In general this efficient frontier can be of any shape (and in the case of strong alignment is a single point).

[4]We assume that the adversary does not have access to the classifiers randomness.

However when the model family is sufficiently rich, the efficient frontier is convex. This is captured by the following theorem:

Theorem 2.4.1. *Let $\mathcal{F}$ be a family of randomized classifiers closed under taking mixtures. That is to say, if F_0 and F_1 are in $\mathcal{F}$ then for any $\lambda \in [0,1]$, the randomized classifier*

$$F_\lambda(x) = \begin{cases} F_0(x) & \text{with probability } 1-\lambda \\ F_1(x) & \text{with probability } \lambda \end{cases}$$

is also in $\mathcal{F}$. It then holds that the efficient frontier of natural and robust error over $\mathcal{F}$ is a convex.

This theorem is a consequence of the fact that F_λ will perform at least as well as the corresponding weighted average of the performances of F_0 and F_1:

Lemma 2.4.2. *Let F_0 and F_1 be two classifiers, and let F_λ denote a randomized mixture classifier that behaves as*

$$F_\lambda(x) = \begin{cases} F_0(x) & \text{with probability } 1-\lambda, \\ F_1(x) & \text{with probability } \lambda, \end{cases} \qquad \lambda \in [0,1].$$

Then for any $\lambda \in [0,1]$, it holds that

$$L_{\mathcal{A}}^{\text{rob-err}}(F_\lambda) \leq (1-\lambda) \cdot L_{\mathcal{A}}^{\text{rob-err}}(F_0) + \lambda \cdot L_{\mathcal{A}}^{\text{rob-err}}(F_1). \tag{2.4.1}$$

Proof. Using the distributional robustness form of robust error as given defined in

Equation (2.1.4), we have

$$
\begin{aligned}
&L_{\mathcal{A}}^{\text{rob-err}}(f_\lambda) \\
&= \max_{Q_{X,Y} \in B_{\mathcal{A}}(P_{X,Y})} \mathbb{P}_{X,Y\sim Q_{X,Y}}(F_\lambda(X) \neq Y) \\
&= \max_{Q_{X,Y} \in B_{\mathcal{A}}(P_{X,Y})} \left[(1-\lambda)\cdot \mathbb{P}_{X,Y\sim Q_{X,Y}}(F_0(X) \neq Y) + \lambda \cdot \mathbb{P}_{X,Y\sim Q_{X,Y}}(F_1(X) \neq Y)\right] \\
&\leq (1-\lambda)\cdot \left[\max_{Q_{X,Y} \in B_{\mathcal{A}}(P_{X,Y})} \mathbb{P}_{X,Y\sim Q_{X,Y}}(F_0(X) \neq Y)\right] \\
&\qquad + \lambda \cdot \left[\max_{Q_{X,Y} \in B_{\mathcal{A}}(P_{X,Y})} \mathbb{P}_{X,Y\sim Q_{X,Y}}(F_1(X) \neq Y)\right] \\
&= (1-\lambda)\cdot L_{\mathcal{A}}^{\text{rob-err}}(F_0) + \lambda \cdot L_{\mathcal{A}}^{\text{rob-err}}(F_1).
\end{aligned}
$$

□

Having established Lemma 2.4.2, the proof of Theorem 2.4.1 follows easily.

Proof of Theorem 2.4.1. Let F_0 and F_1 be two arbitrary points on the efficient frontier of $\mathcal{F}$. To show the efficient frontier is convex, we must show that for any $\lambda \in [0,1]$,

$$
\left\{
\begin{array}{ll}
\inf\limits_{F\in\mathcal{F}} & L_{\mathcal{A}}^{\text{rob-err}}(F) \\
\text{s.t.} & L^{\text{nat-err}}(F) = (1-\lambda)\cdot L^{\text{nat-err}}(F_0) + \lambda \cdot L^{\text{nat-err}}(F_1) \\
& \qquad \leq (1-\lambda)\cdot L_{\mathcal{A}}^{\text{rob-err}}(F_0) + \lambda \cdot L_{\mathcal{A}}^{\text{rob-err}}(F_1).
\end{array}
\right\}
$$

Lemma 2.4.2 easily implies this inequality, since F_λ has natural error $(1-\lambda)\cdot L^{\text{nat-err}}(F_0) + \lambda \cdot L^{\text{nat-err}}(F_1)$ and robust error at most $(1-\lambda)\cdot L_{\mathcal{A}}^{\text{rob-err}}(F_0) + \lambda \cdot L_{\mathcal{A}}^{\text{rob-err}}(F_1)$.

□

Chapter 3

Linear classification of symmetric Gaussians

In this chapter we study the problem of robustly classifying two symmetric Gaussians using a linear model. The data distribution under study is heavily inspired by the work in [20] and [7].

The major result of this chapter is that on this dataset, it is the rule rather than the exception that natural and robust accuracy are limit misaligned. This suggests that such misalignment may be commonplace in practice, in which case it is imperative that one optimizes/designs explicitly for robustness if one desires robustness.

This result follows from the main theorem of this chapter, Theorem 3.3.1, which presents a fairly precise characterization of linear classifiers that are efficient with respect to natural and robust accuracy on the symmetric Gaussian dataset.

3.1 Problem setup

The linear models we study are parameterized functions $F_w : \mathbb{R}^d \to \{\pm 1\}$ parameterized by $w \in (\mathbb{R}^d - \{0\}) \times \mathbb{R}$ of the form[1]

$$F_w(x) \triangleq \operatorname{sgn}(f_w(x)) \triangleq \operatorname{sgn}(\langle w, x \rangle), \tag{3.1.1}$$

where the smaller $f_w : \mathbb{R}^d \to \mathbb{R}$ denotes the pre-sgn activation of F_w.

The particular data distribution of interest is the following symmetric two-class Gaussian,

$$\begin{aligned} Y &\sim \text{Unif}(\{\pm 1\}), \\ X \mid Y &\sim \mathcal{N}(Y \cdot \mu, \Sigma), \end{aligned}$$

where Y is the class label and $X \in \mathbb{R}^d$ for $d \geq 2$. We assume that $\mu \neq 0$ and that $\Sigma \in \mathbb{R}^{d \times d}$ is positive definite.

Our study of robustness will be with respect to an ϵ-ℓ_2 perturbative threat model, where

$$\mathcal{A}_\epsilon(x) = \{x + \Delta \mid \|\Delta\|_2 \leq \epsilon\}.$$

This threat model plays nicely with the geometry of the classifier and data distribution. As shorthand, we will write $L_\epsilon^{\text{rob-err}}$ in place of the more cumbersome $L_{\mathcal{A}_\epsilon}^{\text{rob-err}}$.

Under the symmetric two-class Gaussian data distribution, a linear classifier F_w has natural and robust error

$$L^{\text{nat-err}}(F_w) = Q\left(\frac{w^\top \mu}{\|w\|_\Sigma}\right), \tag{3.1.2}$$

$$L_\epsilon^{\text{rob-err}}(F_w) = Q\left(\frac{w^\top \mu - \epsilon \|w\|_2}{\|w\|_\Sigma}\right), \tag{3.1.3}$$

[1]Note that the absolute scale of w do not effect the behavior of the classifier. Many of the expressions in this chapter will be scale invariant in w for this reason.

where $\|w\|_\Sigma \triangleq \sqrt{w^\top \Sigma w}$ and $Q(\cdot)$ is the tail distribution function of a standard Gaussian. We comment on the behavior of biased classifiers in Appendix A.1.

3.2 Isotropic covariance

We first consider the isotropic covariance setting, where $\Sigma = \sigma^2 \cdot I_d$ for some constant $\sigma > 0$. In this setting the natural and robust errors for a classifier F_w can be written as

$$L^{\text{nat-err}}(F_w) = Q\left(\frac{w^\top \mu}{\sigma \|w\|_2}\right), \tag{3.2.1}$$

$$L_\epsilon^{\text{rob-err}}(F_w) = Q\left(\frac{w^\top \mu}{\sigma \|w\|_2} - \frac{\epsilon}{\sigma}\right). \tag{3.2.2}$$

Since Q is a strictly decreasing function, we see that natural error and robust error are strongly aligned. This will actually turn out to be an exceptional case, and thus we will note this result down as a theorem.

Theorem 3.2.1. *For linear classifiers on a symmetric two-class Gaussian dataset, natural error is strongly aligned with robust error.*

Let us now return to the expressions in Equation (3.2.1) and Equation (3.2.2). We note that

$$\left\{ \frac{w^\top \mu}{\|w\|_2} \;\middle|\; w \in \mathbb{R} - \{0\} \right\} = \left[-\|\mu\|, \|\mu\|\right],$$

which means

$$\left\{ L^{\text{nat-err}}(F_w) \;\middle|\; w \in \mathbb{R} - \{0\} \right\} = \left[Q\left(-\frac{\|\mu\|}{\sigma}\right), Q\left(\frac{\|\mu\|}{\sigma}\right) \right]$$

$$\left\{ L_\epsilon^{\text{rob-err}}(F_w) \;\middle|\; w \in \mathbb{R} - \{0\} \right\} = \left[Q\left(-\frac{\|\mu\| - \epsilon}{\sigma}\right), Q\left(\frac{\|\mu\| - \epsilon}{\sigma}\right) \right]$$

Thus, the joint achievable region of natural and robust loss is governed entirely by the two ratios $\|\mu\|/\sigma$ and ϵ/σ. We will refer to these ratios as the signal-to-noise ratio (SNR) and signal-to-adversary ratio (SAR) respectively.

We visualize the effect that SNR and SAR have on the joint behavior of natural and robust error, in Figure 3.2.1 and Figure 3.2.2. These figures show the joint achievable region as a one-dimensional line that is strictly increasing in natural error, which is to be expected given Theorem 3.2.1.

We also make two trivial but important observations:

1. Any point in top right quadrant of the plot is strictly worse than a constant classifier, which has natural and robust error 1/2.

2. Any point in the top left quadrant of the plot has strictly greater robust error than a constant classifier, which has robust error 1/2.

Thus, looking at Figure 3.2.1, observation (2) tells us that when SNR = 1 and SAR = 2, natural accuracy and robust accuracy are misaligned in the limit.

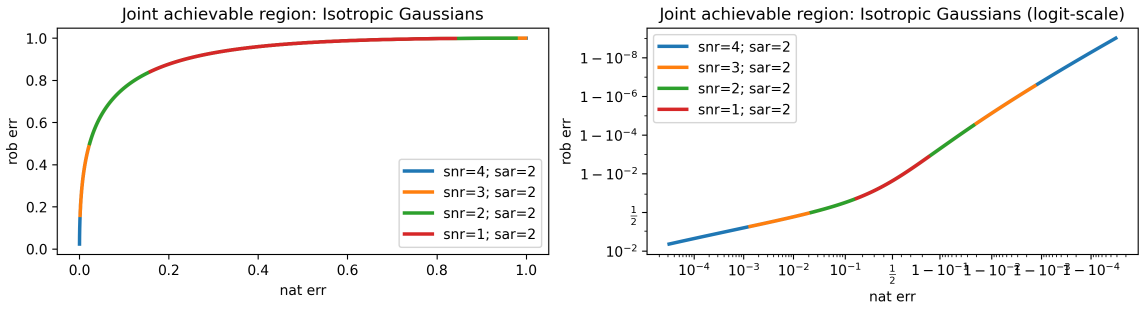

Figure 3.2.1: Joint achievable region for an isotropic Gaussian: SAR fixed. When the SNR and SAR is fixed, changing the SAR increases the width of the joint achievable region along a fixed tradeoff curve. The right plot is a logit-scaled version of the left plot.

3.3 General covariance

When the covariance is not isotropic, the story is considerably more nuanced. In particular assuming general position (explained below), natural error and robust error will be misaligned in the limit. This fact (Corollary 3.3.2) follows from the following theorem, which characterizes which linear classifiers are efficient with respect to natural and robust error.

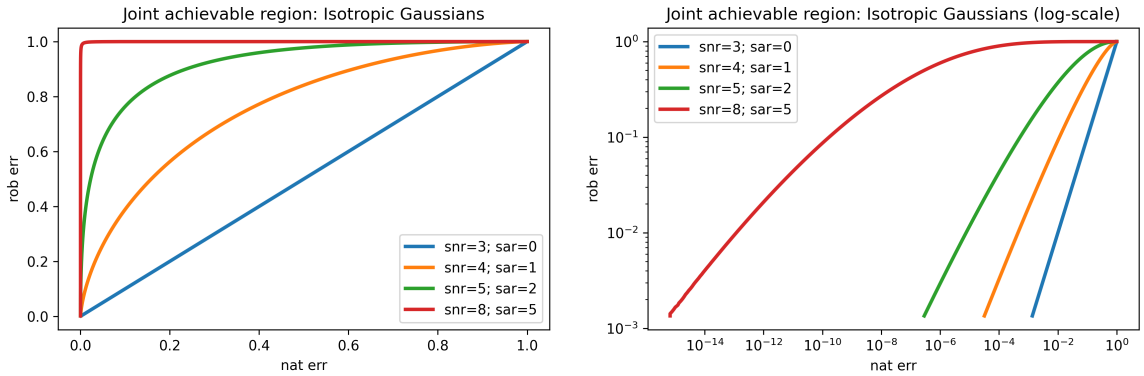

Figure 3.2.2: Joint achievable region for an isotropic Gaussian: SNR – SAR fixed. When the difference between SNR and SAR is fixed, the optimal robust accuracy is fixed. When we keep this difference fixed and increase SNR, we sharpen the tradeoff between robust and natural accuracy. In this manner we can construct a dataset on which we can get arbitrarily low natural error while keeping robust error near 100%. The right plot is a log-scaled version of the left plot.

Theorem 3.3.1. *For a symmetric two-class Gaussian data distribution in general position*[†]*, the efficient frontier of natural and ϵ-ℓ_2-robust error over linear models $\{F_w \mid w \in \mathbb{R}^d - \{0\}\}$ is generated by models $F_{w_*(\lambda)}$ where*

$$w_*(\lambda) \triangleq \frac{(I + \lambda\Sigma)^{-1}\mu}{\|(I + \lambda\Sigma)^{-1}\mu\|_2}, \qquad \lambda \in [\lambda_*^{\text{rob}}, \infty].$$

Here we adopt the convention that $w_*(\infty) = \Sigma^{-1}\mu / \|\Sigma^{-1}\mu\|_2$.

Over the range $[\lambda_*^{\text{rob}}, \infty]$, natural error $L^{\text{nat-err}}(F_{w_*(\lambda)})$ is strictly decreasing in λ and robust error $L_\epsilon^{\text{rob-err}}(F_{w_*(\lambda)})$ is strictly increasing in λ. So $F_{w_*(\infty)}$ is a zero-bias linear model that minimizes natural error, and $F_{w_*(\lambda_*^{\text{rob}})}$ is a zero-bias linear model that minimizes robust error.

Finally, λ_*^{rob} is the unique constant in $(-\frac{1}{\sigma_{\max}(\Sigma)}, \infty)$ satisfying

$$\left\|(I + \lambda_*^{\text{rob}}\,\Sigma)^{-1}\mu\right\|_2 = \epsilon. \tag{3.3.1}$$

[†] *General position here means μ is not an eigenvector of Σ and μ is not orthogonal to the eigenspace of Σ with the largest eigenvalue.*

Proof. See Appendix A.2. □

Limit misalignment of natural and robust error is an immediate corollary of this theorem.

Corollary 3.3.2. *For a symmetric two-class Gaussian data distribution in general position, over linear classifiers natural error and robust error are misaligned in the limit.*

Another slightly more involved corollary of Theorem 3.3.1 is that in this Gaussian case, a Nash equilibrium exists for minimizing robust error:

Corollary 3.3.3. *If ϵ is small enough such that $\lambda_*^{\text{rob}} > 0$, then $F_{w_*(\lambda_*^{\text{rob}})}$ is both the optimal ϵ-ℓ_2-robust classifier for the base distribution where $X|Y \sim \mathcal{N}(Y\mu, \Sigma)$, and the optimal natural classifier when*

$$X \mid Y \sim \mathcal{N}(Y(\mu - (I + \lambda_*^{\text{rob}}\,\Sigma)^{-1}\mu), \Sigma). \tag{3.3.2}$$

In fact, this perturbed data distribution and $F_{w_*(\lambda_*^{\text{rob}})}$ form a strong Nash equilibrium in the game of minimizing robust loss:

$$\min_{F:\mathbb{R}^d \to \{\pm 1\}} \max_{Q_{X,Y} \in B_{\mathcal{A}_\epsilon}(P_{X,Y})} \mathbb{P}_{X,Y \sim Q_{X,Y}}(F(X) \neq Y), \tag{3.3.3}$$

where $P_{X,Y}$ is the base distribution. Note that this Nash equilibrium is over all classifiers, not just linear ones.

Proof. See Appendix A.2. □

When Σ and μ are known, Equation (3.3.1) also provides an efficient way of computing the optimal robust classifier:

Corollary 3.3.4. *Since the LHS of Equation (3.3.1) is strictly decreasing in λ, the value of λ_*^{rob} can be computed efficiently using binary search.*

This algorithm is used to generate the plots in Figure 3.3.1, which provide some visualizations of the joint behavior of natural and robust error for Gaussians with general covariance structure. We note that

- A large Σ causes natural error and robust error to be more aligned, and a small Σ causes natural error and robust error to be more misaligned.

- Natural and robust error are most severely misaligned when Σ is small and μ is close to the maximum eigenvector of Σ.

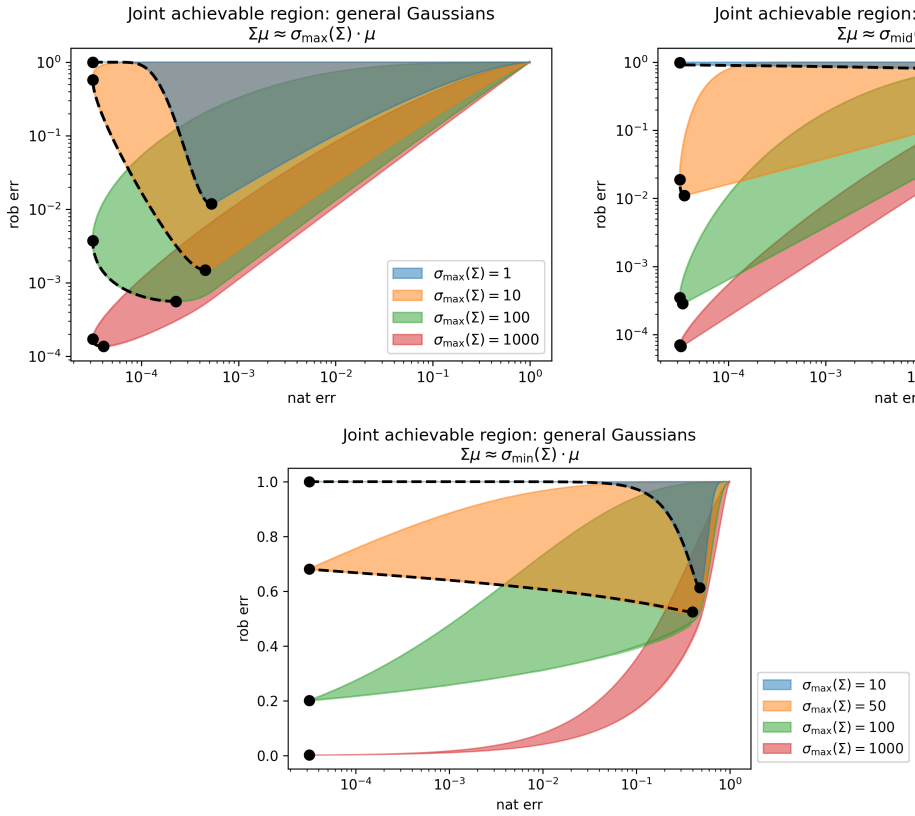

Figure 3.3.1: Joint achievable regions for general Gaussian. In the figures above, $\Sigma \in \mathbb{R}^{32\times32}$ with condition number $\sigma_{\max}(\Sigma)/\sigma_{\min}(\Sigma) = 1000$. The eigenvalues of Σ are evenly log-spaced. In each plot, μ is approximately a maximal eigenvector (i.e. with largest eigenvalue), a minimal eigenvector, or a middle eigenvector (with eigenvalue $\sigma_{\text{mid}}(\Sigma) = \sqrt{\sigma_{\min}(\Sigma)\sigma_{\max}(\Sigma)}$). We show in each plot the effect of scaling Σ. The black points correspond to the optimal classifiers for natural and robust accuracy, and the black dashed lines trace out the efficient frontier.

Chapter 4

Kernel classifiers

In this chapter, we study the robustness of binary kernel classifiers. We again focus on ϵ-ℓ_2 perturbative threat models.

Our major empirical finding is that naturally trained classical kernel classifiers are vulnerable to adversarial examples in similar ways to naturally trained neural networks. Due to this similarity, we conjecture that the non-robustness of naturally trained neural networks may have the same root cause as the non-robustness of naturally trained kernel methods.

We then develop some theory to explain the non-robustness of kernel methods and evaluate our theory empirically.

Finally, we provide experimental evidence that suggests classical kernel classifiers are fundamentally unable to achieve anything more than trivial robust accuracy. If this is indeed the case, it suggests that perhaps neural networks in their current form are also fundamentally unable to achieve robust accuracies on the level of humans.

4.1 Model definitions

A binary kernel classifier $\Phi : \mathbb{R}^d \to \{\pm 1\}$ can be thought of as the composition of an embedding $\varphi : \mathbb{R}^d \to \mathcal{H}$ (where $\mathcal{H}$ is a Hilbert space) and a linear classifier

$F_{w,b} : \mathcal{H} \to \{\pm 1\}$ where $w, b \in \mathcal{H} \times \mathbb{R}$. Written out explicitly, Φ behaves as

$$\Phi(x) = F_{w,b}(\varphi(x)) = \operatorname{sgn}(f_{w,b}(\varphi(x))) = \operatorname{sgn}\left(\langle w, \varphi(x)\rangle_{\mathcal{H}} + b\right), \tag{4.1.1}$$

where $f_{w,b} : \mathcal{H} \to \mathbb{R}$ is the pre-sgn activation of $F_{w,b}$. Linear classifiers are a special case of kernel classifiers where φ is the identity function on $\mathbb{R}^d$, giving $\Phi = F_{w,b} \circ \varphi = F_{w,b} \circ \mathrm{Id} = F_{w,b}$.

The expression in Equation (4.1.1) the primal form of a kernel classifier. Though conceptually useful, in practice we often work with kernel classifiers in their dual form:

$$\Phi(x) = G_{\mathcal{X},\alpha,b}(x) \triangleq \operatorname{sgn}\left(g_{\mathcal{X},\alpha,b}(x)\right) \triangleq \operatorname{sgn}\left(b + \sum_{i=1}^{n} \alpha_i \, k(x_i, x)\right), \tag{4.1.2}$$

where like above $g_{\mathcal{X},\alpha,b} : \mathbb{R}^d \to \mathbb{R}$ is the pre-sgn activation of $G_{\mathcal{X},\alpha,b}$. Here $\mathcal{X} = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^n$ is a set of datapoints that we call the *data-basis* of Φ, $\alpha \in \mathbb{R}^n$ is a vector of coefficients, and $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a kernel function that implicitly defines (and is defined by) the embedding φ via the relation

$$k(x, x') \triangleq \langle \varphi(x), \varphi(x') \rangle_{\mathcal{H}}. \tag{4.1.3}$$

The primal and dual forms of a kernel classifier are connected by the following equivalence:

$$w = \sum_{i=1}^{n} \alpha_i \, \varphi(x_i) \qquad \Longleftrightarrow \qquad f_{w,b} \circ \varphi \equiv g_{\mathcal{X},\alpha,b}. \tag{4.1.4}$$

Put another way, the dual form of a kernel classifier is the decomposition of the primal form into the contributions from a set of datapoints $\mathcal{X}$.

4.1.1 Choice of kernel

The choice of the kernel / embedding for a kernel classifier has an immense impact on its behavior. Indeed with an arbitrary kernel, a kernel classifier can represent arbitrary classification functions. When learning a kernel classifier from data, a choice of kernel can be interpreted as a choice of prior for the learning algorithm.

In this chapter, we will study kernel classifiers with classical kernels. This is a loose term, and we mean by it simply those kernels with simple closed forms that are commonly used in pedagogy and practice. A list of the classical kernels used in this chapter is given in Table 4.1.

Table 4.1: A list of the classical kernels we study in Chapter 4. In this chapter our data always lies in $\mathbb{R}^d$, so the kernels we study are functions from $\mathbb{R}^d \times \mathbb{R}^d$ to $\mathbb{R}$.

Kernel	**Parameters**	Kernel expression
Linear		$k(x, x') = x^\top x'$
Polynomial	degree: k	$k(x, x') = (x^\top x' + 1)^k$
RBF	bandwidth: σ	$k(x, x') = \exp\left(-\frac{\|x-x'\|_2^2}{\sigma^2}\right)$
Laplace	bandwidth: σ	$k(x, x') = \exp\left(-\frac{\|x-x'\|_2}{\sigma}\right)$

4.2 Learning kernel classifiers

There are many ways of learning a kernel classifiers from data, but for this chapter we will focus on support vector machine (SVM) inspired learning algorithms.

The base SVM algorithm works as follows. Given a labeled dataset $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \{\pm 1\}$, we learn a kernel classifier $\Phi = \operatorname{sgn} \circ f_{w,b} \circ \varphi$ optimized for natural accuracy by minimizing the empirical SVM loss

$$L^{\text{svm}}(\Phi) \triangleq \frac{1}{n} \sum_{i=1}^{n} \left(1 - y_i \cdot f_{w,b}(\varphi(x_i))\right)^+, \tag{4.2.1}$$

where $(x)^+ = \max(0, x)$ denotes the ReLU function. One interesting fact about the SVM loss is that the minimizing w^* lies in the span of $\varphi(x_1), \ldots, \varphi(x_n)$, which means we can equivalently optimize the dual form of the classifier $G_{\{x_1,\ldots,x_n\},\alpha,b}$ and optimize over α instead of w. This is known as the kernel trick.

Whether in primal or dual form, the SVM loss is convex, which means stochastic gradient descent or more advanced convex optimization routines can be used to minimize it. We will use both such methods in this chapter.

4.3 Kernel classifiers have adversarial examples

It turns out that on image classification tasks like MNIST and CIFAR10, classical kernel classifiers are vulnerable to adversarial examples in ways that are similar to neural networks.

For example, a RBF kernel classifier trained for natural accuracy on a binarized MNIST task (where the task is to distinguish 0-4 from 5-9), achieves 97.72% natural accuracy and 0% robust accuracy on the test set.[1] This shares striking similarities with neural networks, which achieves 98.67% natural and 6.69% robust test accuracy on the same task under natural training.

From conducting experiments (results summarized in Table 4.2) involving varied choices of kernel, training procedure, and dataset, we conclude that severe non-robustness seems to be an inherent property of naturally trained kernel classifiers with standard kernels. This mirrors the endemic non-robustness of naturally trained deep vision networks.

Since the robust accuracies in Table 4.2 are well below 50%, it must hold that natural accuracy and robust accuracy are misaligned in the limit on binarized image classification for MNIST and CIFAR10. While the degree of this misalignment may be surprising or at least non-obvious, its mere existence should not come as a surprise given results like Corollary 3.3.2 from the previous chapter.

[1] We also binarize CIFAR10 in a similar way, forming two groups of labels 0-4 and 5-9 and making the classification task to distinguish between the two groups.

Table 4.2: Natural and robust accuracies of naturally trained kernel classifiers and neural networks on binarized MNIST and CIFAR10. Accuracies are for the test set. All models are trained to maximize natural accuracy except for the last row, which has numbers for a robustly trained neural network. Robust accuracies are measured under an ϵ-ℓ_2 perturbative adversary with $\epsilon = 2$ on MNIST and $\epsilon = 0.5$ on CIFAR10. Pixel intensities are normalized to lie in $[0, 1]$ and perturbations are clipped to this range. We use [24] to train kernel models and picked the SVM regularization constant C for each model by hand to maximize test accuracy. The Random Fourier feature (RFF) [17] architectures were trained using TensorFlow, and neural networks were taken from [4], [18], and [5].

* Evaluated on a subset of the train set due to computational limitations.

Architecture	**MNIST nat.**	**MNIST rob.**	**CIFAR nat.**	**CIFAR rob.**
Linear	88.07	0.06	60.37	0.00
Poly kernel (deg 2)	97.54	0.00	65.46	10.71
RBF kernel	97.72	0.00	66.27	16.87*
RBF RFF	95.17	0.00	64.66	3.97
Laplace RFF	93.77	0.21	67.81	0.29
Natural NN	98.67	6.69	96.29	4.39
Robust NN	99.28	88.09	96.06	85.16

Finally, though neural networks and kernel classifiers are both non-robust, their respective adversarial examples have characteristic visual differences that can be used to tell them apart. We visualize some of these differences for the MNIST dataset in Figure 4.3.1.

4.4 Margin and robustness

We now develop some theoretical tools for thinking about kernel classifier robustness. In particular, we will focus on the idea of **margin**. At a high level, the margin of a datapoint x with respect to a classifier F is a measure of the distance from x to the decision boundary of F. The larger this margin is, the more robustly F will classify x.

There are actually a few different notions of margin that we will care about for kernel classifiers. The first such notion of margin measures distances in the pre-sgn output space of kernel classifier Φ:

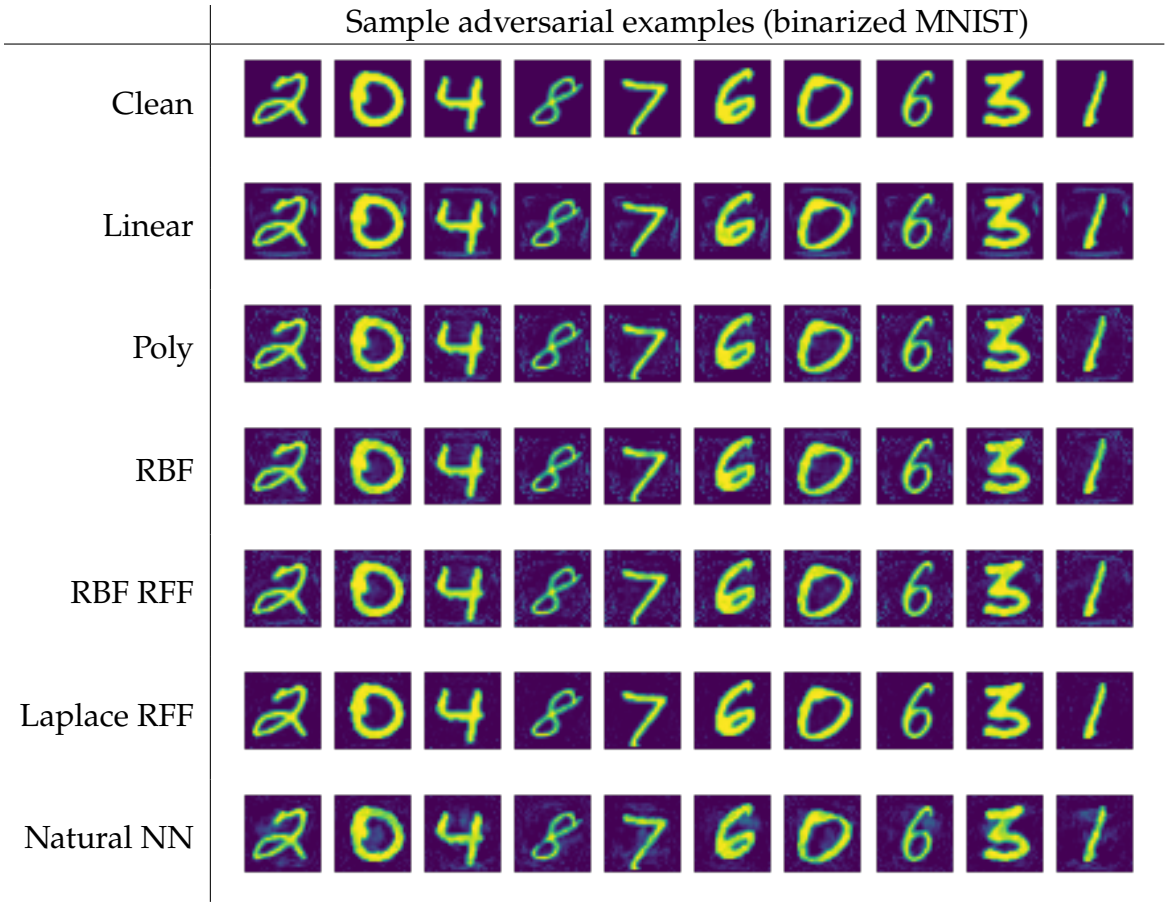

Figure 4.3.1: Visualization of adversarial examples for kernel classifiers and a neural network on binarized MNIST. The top row is a set of clean test set images, and the subsequent rows are adversarial images for different models under a ϵ-ℓ_2 perturbative adversary with $\epsilon = 2$. These models are the same as the ones evaluated in Table 4.2. The adversarial images for each model type have their own distinctive characteristics. Interestingly, the Laplace RFF's adversarial examples are almost indistinguishable from the clean images, setting it apart from the other models.

Definition 4.4.1. *The **functional-margin** of a labeled datapoint $(x, y) \in \mathbb{R}^d \times \{\pm 1\}$ with respect to a kernel classifier $\Phi = F_{w,b} \circ \varphi$ is defined as*

$$\text{Fmarg}_\Phi(x, y) \triangleq y \cdot \big(\langle w, \varphi(x)\rangle + b\big) \tag{4.4.1}$$

*The **absolute functional-margin** of an unlabeled datapoint x is just the absolute value of the above quantity, and is defined as*

$$\text{Fmarg}_\Phi(x) \triangleq \text{Fmarg}_\Phi(x, \Phi(x)). \tag{4.4.2}$$

This definition of functional-margin is relevant to SVM methods. For example, a kernel classifier Φ with zero SVM loss will have a functional-margin of at least one on every labeled datapoint. The functional margin however, is not scale invariant in w and b. This motivates the following definition.

Definition 4.4.2. *The **geometric-margin** of a datapoint $(x, y) \in \mathbb{R}^d \times \{\pm 1\}$ with respect to a kernel classifier $\Phi = F_{w,b} \circ \varphi$ is defined as*

$$\text{Gmarg}_\Phi(x, y) \triangleq \frac{\text{Fmarg}_\Phi(x, y)}{\|w\|_\mathcal{H}}. \tag{4.4.3}$$

*The **absolute geometric-margin** of an unlabeled datapoint x is just the absolute value of the above quantity, and is defined as*

$$\text{Gmarg}_\Phi(x) \triangleq \text{Gmarg}_\Phi(x, \Phi(x)). \tag{4.4.4}$$

The geometric-margin measures the signed distance between $\varphi(x)$ and the decision boundary of Φ in the embedding space using the norm $\|\cdot\|_\mathcal{H}$. When φ is an isometric embedding[2], Φ robustly classifies x as y under an ϵ-ℓ_2 adversary if and only if $\text{Gmarg}_\Phi(x, y) > \epsilon$. In general though, φ is not isometric and warps the geometry of $\mathbb{R}^d$, complicating the relationship between geometric-margin and robustness.

To avoid this warping caused by φ, we can define a final notion of margin that measures distances directly in the data space.

[2] i.e., when $\langle x, x' \rangle = \langle \varphi(x), \varphi(x') \rangle_\mathcal{H}$.

Definition 4.4.3. *The **data-margin** of a datapoint $(x, y) \in \mathbb{R}^d \times \{\pm 1\}$ with respect to a generic classifier $F : \mathbb{R}^d \to \{\pm 1\}$ is defined as*

$$\text{Dmarg}_F(x, y) \triangleq y \cdot F(x) \cdot \left\{ \begin{array}{ll} \min\limits_{x'} & \|x' - x\|_2 \\ s.t. & F(x') \neq y \end{array} \right\}. \tag{4.4.5}$$

*The **absolute data-margin** of an unlabeled datapoint x is just the absolute value of the above quantity, and is defined as*

$$\text{Dmarg}_\Phi(x) \triangleq \text{Dmarg}_\Phi(x, \Phi(x)). \tag{4.4.6}$$

Data-margin directly measures the robustness of a classifier. Indeed a classifier F robustly classifies x as y under an ϵ-ℓ_2 adversary if and only if $\text{Dmarg}_F(x, y) > \epsilon$. Moreover data-margin is more general than functional-margin or geometric-margin, as it applies to a generic binary classifier. For example, we could speak of the data-margin for a neural network or even a human.

4.4.1 A margin bound for shift-invariant kernels

It turns out that for kernels of special shift-invariant form, we can lower bound the absolute data-margin by the absolute geometric-margin (see Theorem 4.4.4). This is a consequence of the following theorem from [27]:

Theorem 4.4.4. *Let $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a shift-invariant kernel of the form $k(x, x') = \psi(\|x - x'\|_2)$ for some strictly decreasing function $\psi : [0, \infty) \to \mathbb{R}$. Let Φ be a kernel classifier with kernel k. Then for any $x \in \mathbb{R}^d$, it holds that*

$$\text{Dmarg}_G(x) \geq \delta^{-1}\left(\text{Gmarg}_G(x)\right). \tag{4.4.7}$$

where $\delta(\cdot)$ is the strictly increasing function

$$\delta(d) = \sqrt{2\psi(0) - 2\psi(d)}, \qquad d \geq 0, \tag{4.4.8}$$

and $\delta^{-1} : \left[0, \sqrt{2\psi(0) - 2\psi(\infty)}\right) \to [0, \infty)$ *is its inverse.*

Proof. See Appendix B.1. □

Theorem 4.4.4 holds in particular for RBF and Laplace kernels. Using the forms of these kernels as given in Table 4.1, we have

$$\psi_{\text{RBF}}(d) = \exp\left(-\frac{d^2}{\sigma^2}\right) \qquad \text{and} \qquad \psi_{\text{Laplace}}(d) = \exp\left(-\frac{d}{\sigma}\right),$$

with corresponding $\delta^{-1}(\cdot)$ functions

$$\delta^{-1}_{\text{RBF}} = \sigma \cdot \sqrt{-\log\left(1 - \frac{d^2}{2}\right)} \qquad \text{and} \qquad \delta^{-1}_{\text{Laplace}} = -\sigma \cdot \log\left(1 - \frac{d^2}{2}\right).$$

A plot of these two functions is given in Figure 4.4.1.

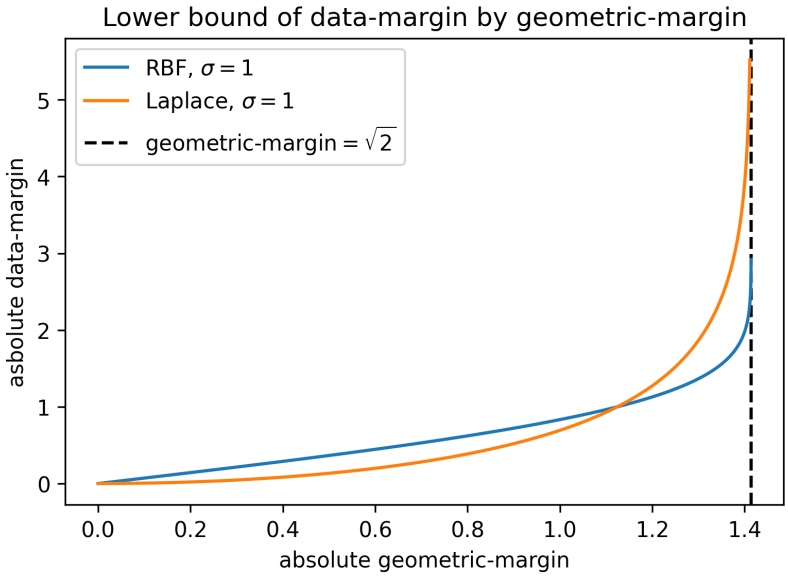

Figure 4.4.1: Visualization of the lower bound in Theorem 4.4.4 for a RBF and Laplace kernel (with bandwidth $\sigma = 1$). Note that both lower bounds functions have a vertical asymptote at $\sqrt{2}$. Thus when using a RBF or Laplace kernel, as the absolute geometric-margin of a point x approaches $\sqrt{2}$, the absolute data-margin approaches ∞.

4.5 Empirical behavior of margin

Let us now examine the empirical behavior of classifier margin.

In Figure 4.5.1 we plot the joint distribution of geometric-margin and (approximate) data-margin for some RBF classifiers. We make two observations about these plots:

1. The bound given by Theorem 4.4.4 is not tight. This means the RBF classifiers in Figure 4.5.1 are more robust than they appear from looking at geometric-margin alone.

2. **Data-margin is close to a perfect linear function of geometric-margin!** Thus we can tell if a point is robustly classified under an ϵ-ℓ_2 perturbative adversary simply by seeing if its geometric-margin is greater than some threshold $c_0\epsilon$, where c_0 is the proportionality constant linking geometric-margin and data-margin.

In general, the data-margin is difficult to analyze theoretically due to the non-linear effects of the kernel embedding, whereas the geometric-margin is easier to analyze because it falls under the umbrella of linear classification theory. Observation 2 is particularly exciting because it describes scenario in which there is a simple and precise relationship between data-margin and geometric-margin, allowing us to study data-margin using the better understood geometric-margin. The following set of open questions are thus of theoretical interest:

(a) When does the linear relationship in Observation 2 hold? It is specific to the RBF kernel classifiers, or can we see similar behavior in other types of classifiers?

(b) When the linear relationship does hold, what determines the proportionality constant c_0?

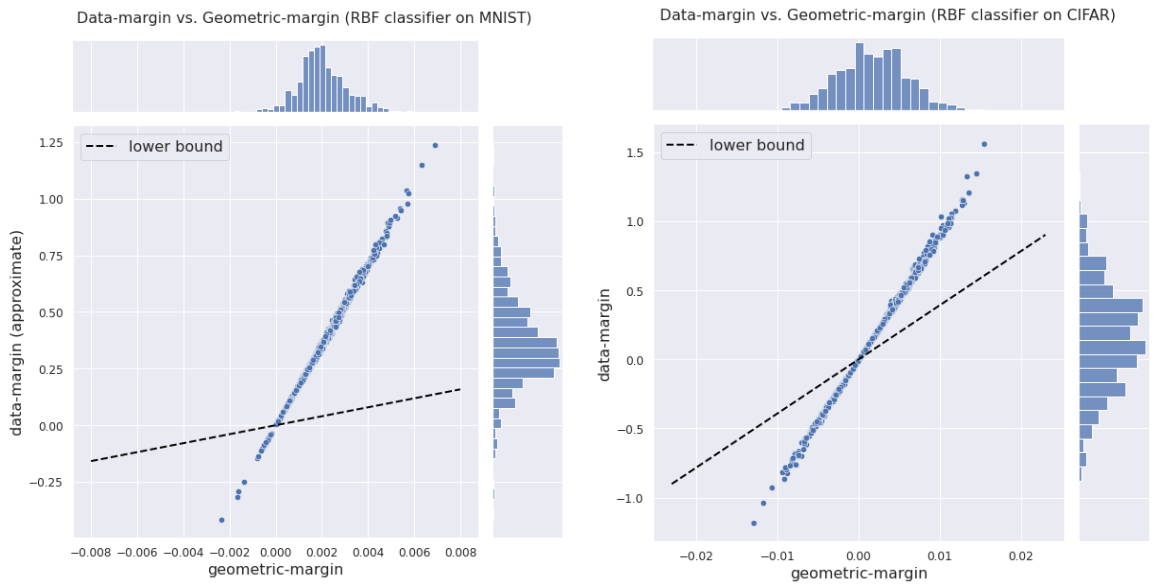

Figure 4.5.1: Joint distribution of geometric-margin and (approximate) data-margin of RBF classifiers on binarized MNIST and CIFAR. Each plot is for 1024 test samples, and the classifiers are the same ones as in Table 4.2. A sample is classified correctly if its corresponding point on the plot lies in the upper right quadrant, and it is classified incorrectly if the corresponding point is in the lower left quadrant. A point is robustly classified correctly (for an ϵ-ℓ_2 adversary) if its y-coordinate is greater than ϵ. Finally, the black dashed line represents the bound given by Theorem 4.4.4, which looks linear due to us being near the origin (see Figure 4.4.1 for a more complete picture). We compute the approximate data-margin by first performing PGD with early stopping, and then doing binary search to find the smallest perturbation from the base point that still flips the label.

4.6 Learning robust kernel classifiers

The empirical results from Section 4.3 tell us that naturally accuracy and robust accuracy are heavily misaligned for kernel classifiers on image classification. Thus if we want to obtain a robust kernel classifier we must explicitly optimize for robustness. In this section, we propose some algorithms for doing precisely this. These algorithms aim to learn robust kernel classifiers from data, and are all extensions to the basic SVM algorithm described in Section 4.2.

4.6.1 Robust SVM loss

The basic idea behind the SVM algorithm is that a classifier $\Phi = \mathrm{sgn} \circ f_{w,b} \circ \varphi$ with low SVM loss

$$L^{\text{svm}}(\Phi) = \frac{1}{n} \sum_{i=1}^{n} \left(1 - y_i \cdot f_{w,b}(\varphi(x_i))\right)^+$$

tends also to have low natural error $L^{\text{nat-err}}(\Phi)$. If we take this implication for granted, then to guarantee low robust error $L_\epsilon^{\text{rob-err}}(\Phi)$ it suffices to ensure $L^{\text{svm}}(\Phi)$ is small under all possible ϵ-ℓ_2 adversarial data perturbations.

This inspires our first algorithm for learning a robust kernel classifier – we simply find a classifier that has low robust SVM loss:

$$L^{\text{svm-rob}}(\Phi) \triangleq \frac{1}{n} \sum_{i=1}^{n} \max_{\|\Delta_i\| \le \epsilon} \left(1 - y_i \cdot f_{w,b}(\varphi(x_i + \Delta_i))\right)^+ . \tag{4.6.1}$$

One of the standard methods for minimizing a robust loss like $L^{\text{svm-rob}}$ is to perform *adversarial training* [12], which can be thought of as block coordinate ascent/descent where we alternate between minimizing the loss with respect to v, b while fixing the Δ_i, and maximizing the loss with respect to Δ_i while fixing v, b. So when φ is an explicit finite dimensional embedding, we can apply adversarial training to optimize $L^{\text{svm-rob}}$.

Things are problematic when φ is infinite-dimensional or implicitly defined by a kernel however. For the vanilla SVM loss, the solution here is to use the kernel trick

and optimize the dual coefficients of the kernel classifier. However for $L^{\text{svm-rob}}$ this no longer works because the optimal w^* may not lie in the span of $\varphi(x_1), \ldots, \varphi(x_n)$ (since we are allowed to perturb the x_i). Thus the kernel trick can no longer be safely applied. There are a few potential solutions to this dilemma:

1. Apply the kernel trick anyways with a well chosen data-basis $\{z_1, \ldots, z_m\}$ and hope that the optimal w^* is "close" to the span of $\varphi(z_1), \ldots, \varphi(z_m)$.

2. Approximate the infinite-dimensional or implicitly defined φ with an explicit finite dimensional $\tilde{\varphi} : \mathbb{R}^d \to \mathbb{R}^m$ where

$$\langle \varphi(x), \varphi(x') \rangle_{\mathcal{H}} \approx \tilde{\varphi}(x)^\top \tilde{\varphi}(x').$$

   Then apply primal adversarial training using $\tilde{\varphi}$ in place of φ.

4.6.2 Robust SVM loss with data-margin bound

This approach is directly inspired by [25].

When φ is the embedding of a shift-invariant kernel that satisfies the conditions specified in Theorem 4.4.4, it holds that

$$\begin{aligned} L^{\text{svm-rob}}(\Phi) &= \frac{1}{n} \sum_{i=1}^{n} \max_{\|\Delta_i\| \le \epsilon} \left(1 - y_i \cdot f_{w,b}(\varphi(x_i + \Delta_i))\right)^+ \\ &\le \frac{1}{n} \sum_{i=1}^{n} \left(1 - y_i \cdot f_{w,b}(\varphi(x_i + \Delta_i)) + \delta(\epsilon) \cdot \|w\|_{\mathcal{H}}\right)^+ . \end{aligned}$$

Thus instead of minimizing $L^{\text{svm-rob}}$ we could minimize

$$L_\epsilon^{\text{svm-rob-ub}}(F_{v,b} \circ \varphi) \triangleq \frac{1}{n} \sum_{i=1}^{n} \left(1 - y_i \cdot f_{w,b}(\varphi(x_i + \Delta_i)) + \delta(\epsilon) \cdot \|w\|_{\mathcal{H}}\right)^+ . \tag{4.6.2}$$

Since Theorem 4.4.4 is not tight, a low $L_\epsilon^{\text{svm-rob-ub}}(\Phi)$ may not imply a low $L^{\text{svm-rob}}(\Phi)$. However $L_\epsilon^{\text{svm-rob-ub}}$ has two advantages over $L^{\text{svm-rob}}$:

1. $L_\epsilon^{\text{svm-rob-ub}}$ does not have an inner maximization problem. Thus we can apply standard convex optimization techniques (since the loss is convex).

2. The kernel trick is valid for $L_\epsilon^{\text{svm-rob-ub}}$.

Finally, to account for that fact that Theorem 4.4.4 may be a loose bound, we could adjust the ϵ in $L_\epsilon^{\text{svm-rob-ub}}$ via a procedure like cross-validation.

4.6.3 L2-Regularized SVM loss

In the same way that we upper bounded $L^{\text{svm-rob}}$ by the simpler loss $L_\epsilon^{\text{svm-rob-ub}}$, we can upper bound $L_\epsilon^{\text{svm-rob-ub}}$ by an even simpler loss in the following way

$$\begin{aligned} L_\epsilon^{\text{svm-rob-ub}}(\Phi) &= \frac{1}{n}\sum_{i=1}^{n}\left(1 - y_i \cdot f_{w,b}(\varphi(x_i + \Delta_i)) + \delta(\epsilon)\cdot\|w\|_{\mathcal{H}}\right)^+ \\ &\leq \left[\frac{1}{n}\sum_{i=1}^{n}\left(1 - y_i \cdot f_{w,b}(\varphi(x_i + \Delta_i))\right)^+\right] + \delta(\epsilon)\cdot\|w\|_{\mathcal{H}} \\ &= L^{\text{svm-rob}}(\Phi) + \delta(\epsilon)\cdot\|w\|_{\mathcal{H}} \end{aligned} \tag{4.6.3}$$

Thus instead of minimizing $L_\epsilon^{\text{svm-rob-ub}}$ we could minimize this upper bound.

Note now that $\|\cdot\|_{\mathcal{H}}$ and $L^{\text{svm-rob}}$ are both convex in w and b. Thus by convex optimization theory, minimizing Equation (4.6.3) is equivalent to the constrained optimization problem

$$\begin{aligned} \min_{w,b} \quad & L^{\text{svm-rob}}(\Phi) \\ \text{s.t.} \quad & \|w\|_{\mathcal{H}} \leq C \end{aligned} \tag{4.6.4}$$

for some constant C that depends on $\delta(\epsilon)$. But again by convex optimization theory, Equation (4.6.4) is equivalent to the unconstrained minimization of the standard L2-regularized SVM loss

$$L_\lambda^{\text{svm-reg}}(\Phi) \triangleq L^{\text{svm-rob}}(\Phi) + \lambda\cdot\|w\|_{\mathcal{H}}^2 \tag{4.6.5}$$

for some λ that depends on C (and thus ϵ implicitly).

Thus with an appropriate choice of regularization constant, the standard L2-regularized SVM loss is an upper bound on robust SVM loss and to some extent its minimization promotes low robust error.

4.7 Classical kernel classifiers seem unable to be robust

Having developed some robust SVM losses in the previous section, we move onto trying to optimize them in practice. Surprisingly however, regardless of the loss we use, we were unable in our experiments to do better than trivial adversarial accuracy (50% for our binary classification setting).

Unfortunately this is only weak evidence as our optimization methods are only approximate and the consistency properties of our robust losses are unknown. Nonetheless, based on our results we conjecture that perhaps classical kernel methods are fundamentally unable to achieve high robust accuracy on image classification tasks like MNIST and CIFAR10. If this conjecture is true, we believe understanding it's theoretical underpinnings might give us a better understanding of the limitations of neural networks.

Chapter 5

Neural networks

In this chapter we conduct some empirical experiments related to neural network robustness.

We show that kernel classifiers with naturally trained NN embeddings seem fundamentally unable to be robust. Since in the previous chapter we saw evidence that this is also the case for classical kernel embeddings, this supports our conjecture that naturally trained neural networks and classical kernel classifiers may be non-robust for similar reasons.

5.1 Natural neural network embeddings are non-robust

The final set of experiments in the previous chapter weakly suggest that linear models trained on top of standard kernel embeddings can only achieve trivial robustness. In this section we provide evidence suggesting the same is true of naturally trained neural network embeddings.

What do we mean by a naturally trained neural network embedding? Well, most neural network classifiers $F : \mathcal{X} \to \{1, \ldots, C\}$ can be broken up into three components:

1. A parameterized embedding function $\varphi_\theta : \mathcal{X} \to \mathbb{R}^d$ (consisting of various layers and modules) that takes in an input $x \in \mathcal{X}$ and spits out and an embedding $\varphi_\theta(x) \in \mathbb{R}^d$.

2. A linear layer $f_{W,b} : \mathbb{R}^d \to \mathbb{R}^C$ that affinely transforms an embedded input $\varphi_\theta(x)$ to a vector of activations. The action of $f_{W,b}$ looks like

$$f_{W,b}\left(\varphi_\theta(x)\right) = W\varphi_\theta(x) + b, \qquad \text{for some } W \in \mathbb{R}^{C\times d},\ b \in \mathbb{R}^C.$$

3. A final argmax operation that returns the predicted class:

$$F(x) = \operatorname*{argmax}_{c \in \{1,\ldots,C\}} f_{W,b}\left(\varphi_\theta(x)\right)_c .$$

By naturally trained neural network embedding, we simply mean a φ_θ that has its parameters θ optimized for minimizing natural error (or some proxy of it).

To determine whether a robust linear model could be trained on top of a naturally trained neural network embedding, we took a naturally pre-trained model on CIFAR10 and attempted to re-train just its final layer of weights in order to make it robust. Over the course of re-training, this model never reached a robust validation accuracy of over 10%, suggesting that the natural network's embedding could not support robust classification.

To make sure that something wasn't wrong with our fine-tuning procedure, we also ran the same experiment but starting from a robustly pre-trained model instead. Re-training the final layer of this robust model yielded a robust validation accuracy of around 54.4%, showing that with the right embedding nontrivial robust accuracy is possible.

A more comprehensive set of statistics for these two experiments and additional ablation experiments is given in Table 5.1.

Table 5.1: Performance of various neural network embeddings. **Base** indicates the base pretrained model from which re-training commenced, **Finetune** indicates whether only the last layer was re-trained, and **Rand** indicates whether the re-trained layers were re-initialized with random weights at the beginning of training. The last four columns report the natural and robust accuracy at initialization and and their peak during training. Accuracies are on the test set except for the last column which is on the training set. Pretrained models were taken from [5].

∗ Results not collected.

Base	**Finetune**	**Rand**	**Nat init**	**Rob init**	**Rob peak**	**Rob peak (train)**
Nat.	Y	N	95.25	0	10	6.1
Nat.	Y	Y	5.86	0	9.8	3.5
Rob.	Y	N	87.03	53.5	∗	∗
Rob.	Y	Y	10.6	5.07	54.4	87.3
Rand.	N	Y	9.55	1.26	52	86

Appendix A

Linear classifiers: derivations and proofs

A.1 Effect of bias

Let $F_{w,b}$ denote the biased classifier

$$F_w(x) \triangleq \operatorname{sgn}(\langle w, x \rangle + b). \tag{A.1.1}$$

The natural and robust error of this classifier on a symmetric Gaussian dataset is

$$L^{\text{nat-err}}(F_{w,b}) = \frac{1}{2} Q\left(\frac{w^\top \mu + b}{\|w\|_\Sigma}\right) + \frac{1}{2} Q\left(\frac{w^\top \mu - b}{\|w\|_\Sigma}\right), \tag{A.1.2}$$

$$L_\epsilon^{\text{rob-err}}(F_{w,b}) = \frac{1}{2} Q\left(\frac{w^\top \mu - \epsilon\|w\|_2 + b}{\|w\|_\Sigma}\right) + \frac{1}{2} Q\left(\frac{w^\top \mu - \epsilon\|w\|_2 - b}{\|w\|_\Sigma}\right). \tag{A.1.3}$$

In effect, the bias term b allows one to interpolate between any natural-robust-error pair and the (0.5, 0.5) performance point by sending $b \to \infty$.

A.2 Proof of Theorem 3.3.1

Before proving the main result, we establish a key lemma.

Lemma A.2.1. *Define $w_*(\lambda)$ as in Theorem 3.3.1:*

$$w_*(\lambda) \triangleq \frac{(I + \lambda\Sigma)^{-1}\mu}{\|(I + \lambda\Sigma)^{-1}\mu\|_2}.$$

Then for $\lambda \in \left(-\frac{1}{\sigma_{\max}(\Sigma)}, \infty\right)$, the quantities

$$\frac{w_*(\lambda)^\top \mu}{\|w_*(\lambda)\|_\Sigma} \quad \text{and} \quad \frac{\|w_*(\lambda)\|_2}{\|w_*(\lambda)\|_\Sigma}$$

are non-decreasing functions of λ, meaning the parameterized curve

$$t(\lambda) = \left(\frac{w_*(\lambda)^\top \mu}{\|w_*(\lambda)\|_\Sigma}, \frac{\|w_*(\lambda)\|_2}{\|w_*(\lambda)\|_\Sigma}\right)$$

is a function of the first coordinate. Whenever $t(\lambda)$ has non-zero velocity, the slope of this function is $\|(I + \lambda\Sigma)^{-1}\mu\|_2^{-1}$. Thus $t(\lambda)$ as a function of its first coordinate is strictly increasing for $\lambda \in \left(-\frac{1}{\sigma_{\max}(\Sigma)}, \infty\right)$.

Proof. As shorthand we let $T_\lambda \triangleq (I + \lambda\Sigma)^{-1}$. Note that when $\lambda > -\frac{1}{\sigma_{\max}(\Sigma)}$, T_λ is positive definite.

To show $w_*(\lambda)^\top \mu / \|w_*(\lambda)^\top\|_\Sigma$ is non-decreasing in λ, we will show its derivative with respect to λ is non-negative. To begin, we note that

$$\frac{d}{d\lambda}\frac{w_*(\lambda)^\top \mu}{\|w_*(\lambda)\|_\Sigma} = \frac{d}{d\lambda}\frac{\mu^\top T_\lambda \mu}{(\mu^\top \Sigma T_\lambda^2 \mu)^{1/2}} \tag{A.2.1}$$

$$= \frac{(\mu^\top T_\lambda \mu)(\mu^\top \Sigma^2 T_\lambda^3 \mu) - (\mu^\top \Sigma T_\lambda^2 \mu)^2}{(\mu^\top \Sigma T_\lambda^2 \mu)^{3/2}}. \tag{A.2.2}$$

Now without loss of generality rotate coordinates such that $\Sigma = \mathrm{diag}(s)$ with $s \in \mathbb{R}^d$.

The positivity of Equation (A.2.2) then follows from Cauchy-Schwarz:

$$(\mu^\top \Sigma T_\lambda^2 \mu)^2 = \left\langle \frac{|\mu|}{(s+\lambda)^{1/2}}, \frac{s \odot |\mu|}{(s+\lambda)^{3/2}} \right\rangle^2 \tag{A.2.3}$$

$$\leq \left\langle \frac{|\mu|}{(s+\lambda)^{1/2}}, \frac{|\mu|}{(s+\lambda)^{1/2}} \right\rangle \cdot \left\langle \frac{s \odot |\mu|}{(s+\lambda)^{3/2}}, \frac{s \odot |\mu|}{(s+\lambda)^{3/2}} \right\rangle \tag{A.2.4}$$

$$= (\mu^\top T_\lambda \mu)(\mu^\top \Sigma^2 T_\lambda^3 \mu), \tag{A.2.5}$$

where the $\odot$, division, exponents on s, and absolute value on μ all act coordinate-wise.

To show $\|w_*(\lambda)^\top\|_2 / \|w_*(\lambda)^\top\|_\Sigma$ is non-decreasing in λ, note that when $\lambda \neq 0$,

$$\frac{d}{d\lambda} \frac{\|w_*(\lambda)\|_2}{\|w_*(\lambda)\|_\Sigma} = \frac{d}{d\lambda} \left(\frac{\mu^\top T_\lambda^2 \mu}{\mu^\top \Sigma T_\lambda^2 \mu} \right)^{1/2} \tag{A.2.6}$$

$$= \frac{(\mu^\top T_\lambda^2 \mu) \cdot (\mu^\top \Sigma^2 T_\lambda^3 \mu) - (\mu^\top \Sigma T_\lambda^2 \mu) \cdot (\mu^\top \Sigma T_\lambda^3 \mu)}{(\mu^\top T_\lambda^2 \mu)^{1/2} \cdot (\mu^\top \Sigma T_\lambda^2 \mu)^{3/2}} \tag{A.2.7}$$

$$= \frac{(\mu^\top T_\lambda^2 \mu)(\mu^\top (T_\lambda^{-1} - I)^2 T_\lambda^3 \mu) - (\mu^\top (T_\lambda^{-1} - I) T_\lambda^2 \mu)(\mu^\top (T_\lambda^{-1} - I) T_\lambda^3 \mu)}{\lambda^2 \cdot (\mu^\top T_\lambda^2 \mu)^{1/2} \cdot (\mu^\top \Sigma T_\lambda^2 \mu)^{3/2}} \tag{A.2.8}$$

$$= \frac{(\mu^\top T_\lambda \mu)(\mu^\top (T_\lambda^{-1} - I)^2 T_\lambda^3 \mu) - (\mu^\top (T_\lambda^{-1} - I) T_\lambda^2 \mu)^2}{\lambda^2 \cdot (\mu^\top T_\lambda^2 \mu)^{1/2} \cdot (\mu^\top \Sigma T_\lambda^2 \mu)^{3/2}} \tag{A.2.9}$$

$$= \frac{(\mu^\top T_\lambda \mu)(\mu^\top \Sigma^2 T_\lambda^3 \mu) - (\mu^\top \Sigma T_\lambda^2 \mu)^2}{(\mu^\top T_\lambda^2 \mu)^{1/2} \cdot (\mu^\top \Sigma T_\lambda^2 \mu)^{3/2}} \tag{A.2.10}$$

$$= \frac{\frac{d}{d\lambda} \frac{w_*(\lambda)^\top \mu}{\|w_*(\lambda)\|_\Sigma}}{(\mu^\top T_\lambda^2 \mu)^{1/2}}. \tag{A.2.11}$$

By continuity, this relation also holds when $\lambda = 0$.

By Equation (A.2.11), whenever $t(\lambda)$ has nonzero velocity, its slope is $(\mu^\top T_\lambda^2 \mu)^{-1/2}$. Now note that $\mu^\top T_\lambda^2 \mu$ is positive and strictly decreasing in λ, since without loss of

generality letting $\Sigma = \mathrm{Diag}(\sigma_1, \ldots, \sigma_d)$, we have

$$\mu^\top T_\lambda^2 \mu = \sum_{i=1}^{d} \frac{\mu_i^2}{(1+\lambda\sigma_i)^2}.$$

Thus the curve $t(\lambda)$ as a function of the its first coordinate is strictly increasing and convex for $\lambda \in \left(-\frac{1}{\sigma_{\max}(\Sigma)}, \infty\right)$. □

We are now ready to prove the main result.

Proof of Theorem 3.3.1. An efficient classifier F_w must have w that is a solution to the optimization problem

$$\begin{aligned} \min_{w \in \mathbb{R}^d - \{0\}} \quad & Q\left(\frac{w^\top \mu - \epsilon \|w\|_2}{\|w\|_\Sigma}\right) \\ \text{s.t.} \quad & Q\left(\frac{w^\top \mu}{\|w\|_\Sigma}\right) = c. \end{aligned} \tag{A.2.12}$$

for some $c \in (0.5, 1]$. We only need to consider $c \in (0.5, 1]$ because an efficient w must have $w^\top \mu > 0$. We will implicitly use this fact throughout our proof.

Since $Q(\cdot)$ is strictly decreasing, w must also be a solution the optimization problem

$$\left\{ \begin{aligned} \max_{w \in \mathbb{R}^d - \{0\}} \quad & \frac{w^\top \mu - \epsilon \|w\|_2}{\|w\|_\Sigma} \\ \text{s.t.} \quad & \frac{w^\top \mu}{\|w\|_\Sigma} = c. \end{aligned} \right\} \iff \left\{ \begin{aligned} \min_{w \in \mathbb{R}^d - \{0\}} \quad & \frac{\|w\|_2}{\|w\|_\Sigma} \\ \text{s.t.} \quad & \frac{w^\top \mu}{\|w\|_\Sigma} = c. \end{aligned} \right\} \tag{A.2.13}$$

for some other $c \in \mathbb{R}$. Now note that both the objective and constraint of Equation (A.2.13) are scale invariant with respect to w. Thus without loss of generality we can force $\|w\|_\Sigma = 1$ and rewrite the RHS of Equation (A.2.13) as

$$\left\{ \begin{aligned} \min_{w \in \mathbb{R}^d - \{0\}} \quad & \|w\|_2 \\ \text{s.t.} \quad & w^\top \mu = c \\ & \|w\|_\Sigma = 1 \end{aligned} \right\} \iff \left\{ \begin{aligned} \min_{w \in \mathbb{R}^d - \{0\}} \quad & w^\top w \\ \text{s.t.} \quad & w^\top \mu = c \\ & w^\top \Sigma w = 1 \end{aligned} \right\}. \tag{A.2.14}$$

Since both the objective and constraints of the RHS of Equation (A.2.14) are

differentiable in w, by the method of Langrange multipliers a efficient w must satisfy the equation

$$(I + \lambda_1 \Sigma)\, w = \lambda_2 \mu$$

for some $\lambda_1, \lambda_2 \in \mathbb{R}$. Or equivalently, an efficient w must satisfy

$$(I + \lambda \Sigma)\, w \propto \mu \tag{A.2.15}$$

for some $\lambda \in \mathbb{R}$. From here, some careful analysis tells us that it suffices for $\lambda > -1/\sigma_{\max}(\Sigma)$. We can then finish by using the results of Lemma A.2.1. □

With the main theorem proved, let us now prove the claimed Nash equilibrium corollary.

Proof of Corollary 3.3.3. When using $F_{w_*(\lambda_*^{\text{rob}})}$ as the classifier, the optimal adversary should perturb X in the direction $-Yw_*(\lambda_*^{\text{rob}})$ by a distance ϵ. But $w_*(\lambda_*^{\text{rob}}) \propto (I + \lambda_*^{\text{rob}}\, \Sigma)^{-1}\mu$ and by Theorem 3.3.1, $\|(I+\lambda_*^{\text{rob}}\, \Sigma)^{-1}\mu\|_2 = \epsilon$. Thus $\mathcal{N}(Y(\mu-(I+\lambda_*^{\text{rob}}\, \Sigma)^{-1}\mu)$ is indeed the worst case distribution for $F_{w_*(\lambda_*^{\text{rob}})}$.

Now for the other direction. Assume the adversary chooses the distribution $\mathcal{N}(Y(\mu - (I + \lambda_*^{\text{rob}}\, \Sigma)^{-1}\mu)$. In this case the optimal natural accuracy classifier F_{w^*} has

$$\begin{aligned} w^* &\propto \Sigma^{-1}\left(\mu - (I + \lambda_*^{\text{rob}}\, \Sigma)^{-1}\mu\right) \\ &\propto (I + \lambda_*^{\text{rob}}\, \Sigma)^{-1}\mu, \end{aligned}$$

as desired (the proportionality is most easily shown by working in a basis where Σ is diagonal). We need $\lambda_*^{\text{rob}} > 0$ for the signs not to flip in the chain of proportionalities.

□

Finally, we give a sketch of the modifications to Theorem 3.3.1 needed when the data distribution is not in general position.

Comment A.2.2. *When general position does not hold in, the following modifications to Theorem 3.3.1 are needed:*

1. *If μ is an eigenvector of Σ, then $\lambda_*^{\text{rob}} = \infty$.*

2. *If μ is orthogonal to the eigenspace of Σ with largest eigenvalue, lower robust error may be achieved by classifiers $f_{w'_*(\lambda')}$ where $\|w'_*(\lambda')\| = 1$,*

$$w'_*(\lambda') \propto \left(I - \frac{\Sigma}{\sigma_{\max}(\Sigma)}\right)^+ \mu + \lambda' z, \qquad \lambda' \in [-\infty, 0], \tag{A.2.16}$$

and z is a non-zero eigenvector of Σ with maximal eigenvalue, i.e. $\Sigma z = \sigma_{\max}(\Sigma) \cdot z$. Here, we use a similar convention that $w'_*(-\infty) = -z/\|z\|_2$.

Appendix B

Kernel classifiers: Details

B.1 Proof of Theorem 4.4.4

As mentioned in the main text, Theorem 4.4.4 is a result of [27], in particular theorem 14. However, theorem 14 as written in [27] is actually flawed, since it implies that the inequality of Theorem 4.4.4 is an equality, which is false.

In the remainder of this section, we give a proof of the corrected theorem.

Proof of Theorem 4.4.4. This proof is essentially identical to the first half of the proof of theorem 14 in [27].

First, let us decompose Φ as $\Phi = \operatorname{sgn} \circ f_{w,b} \circ \varphi$ where $\varphi : \mathbb{R}^d \to \mathcal{H}$ is an embedding for the kernel k that satisfies

$$k(x, x') = \left\langle \varphi(x), \varphi(x') \right\rangle_{\mathcal{H}}.$$

Since we assume k is shift-invariant with the form $k(x, x') = \psi(\|x - x'\|_2)$, we have

$$\left\langle \varphi(x), \varphi(x') \right\rangle_{\mathcal{H}} = \psi(\|x - x'\|_2).$$

Now for any datapoint $x \in \mathbb{R}^d$ and perturbation $\Delta \in \mathbb{R}^d$, it holds that

$$\begin{aligned}
&\left|f_{w,b}(\varphi(x)) - f_{w,b}(\varphi(x+\Delta))\right| \\
&= \left|\left\langle w, \varphi(x) - \varphi(x+\Delta)\right\rangle_{\mathcal{H}}\right| \\
&\leq \|w\|_{\mathcal{H}} \cdot \|\varphi(x) - \varphi(x+\Delta)\|_{\mathcal{H}} \\
&= \|w\|_{\mathcal{H}} \cdot \sqrt{\left\langle \varphi(x) - \varphi(x+\Delta), \varphi(x) - \varphi(x+\Delta)\right\rangle} \\
&= \|w\|_{\mathcal{H}} \cdot \sqrt{\left\langle \varphi(x), \varphi(x)\right\rangle + \left\langle \varphi(x+\Delta), \varphi(x+\Delta)\right\rangle - 2 \cdot \left\langle \varphi(x), \varphi(x+\Delta)\right\rangle} \\
&= \|w\|_{\mathcal{H}} \cdot \sqrt{2f(0) - 2f(\|\Delta\|_2)} \\
&= \|w\|_{\mathcal{H}} \cdot \delta\left(\|\Delta\|_2\right).
\end{aligned}$$

Thus, any perturbation with Δ that satisfies

$$\delta\left(\|\Delta\|_2\right) < \frac{|f_{w,b}(\varphi(x))|}{\|w\|_{\mathcal{H}}} = \mathrm{Gmarg}_G(x)$$

cannot change the prediction of G (i.e. $G(x) = G(x+\Delta)$). Since $\delta(\cdot)$ is strictly increasing, inverting both sides of the above inequality yields the equivalent implication

$$\|\Delta\|_2 < \delta^{-1}(\mathrm{Gmarg}_G(x)) \qquad \Longrightarrow \qquad G(x) = G(x+\Delta).$$

Since a perturbation of size $\mathrm{Dmarg}_G(x)$ *is* able to change the prediction of G, it must be that

$$\mathrm{Dmarg}_G(x) \geq \delta^{-1}\left(\mathrm{Gmarg}_G(x)\right). \qquad \square$$

Bibliography

[1] Evan Ackerman. Three small stickers in intersection can cause tesla autopilot to swerve into wrong lane - ieee spectrum. https://spectrum.ieee.org/cars-that-think/transportation/self-driving/three-small-stickers-on-road-can-steer-tesla-autopilot-into-oncoming-lane.

[2] Mikhail Belkin, Siyuan Ma, and Soumik Mandal. To understand deep learning we need to understand kernel learning. *arXiv preprint arXiv:1802.01396*, 2018.

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

[4] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

[5] Logan Engstrom, Andrew Ilyas, Hadi Salman, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019.

[6] Justin Gilmer and Dan Hendrycks. A discussion of 'adversarial examples are not bugs, they are features': Adversarial example researchers need to expand what is meant by 'robustness'. *Distill*, 2019. https://distill.pub/2019/advex-bugs-discussion/response-1.

[7] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, pages 125–136, 2019.

[8] http://www.image-net.org/challenges/LSVRC/2012/results.html.

[9] Andrej Karpathy. Lessons learned from manually classifying cifar-10, Apr 2011.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[11] Yandong Li, Lijun Li, Liqiang Wang, Tong Zhang, and Boqing Gong. Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. *arXiv preprint arXiv:1905.00441*, 2019.

[12] 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.

[13] Preetum Nakkiran. Adversarial robustness may be at odds with simplicity. *arXiv preprint arXiv:1901.00532*, 2019.

[14] Preetum Nakkiran. A discussion of 'adversarial examples are not bugs, they are features': Adversarial examples are just bugs, too. *Distill*, 2019. https://distill.pub/2019/advex-bugs-discussion/response-5.

[15] https://paperswithcode.com/sota/image-classification-on-imagenet.

[16] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

[17] Ali Rahimi, Benjamin Recht, et al. Random features for large-scale kernel machines. In *NIPS*, volume 3, page 5. Citeseer, 2007.

[18] Jérôme Rony, Luiz Gustavo, Robert Sabourin, and Eric Granger. Decoupling direction and norm for efficient gradient-based l2 adversarial attacks. 2018.

[19] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

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

[21] Alex Serban, Erik Poll, and Joost Visser. Adversarial examples on object recognition: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 53(3):1–38, 2020.

[22] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *Advances in Neural Information Processing Systems*, pages 3358–3369, 2019.

[23] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

[24] Zeyi Wen, Jiashuai Shi, Qinbin Li, Bingsheng He, and Jian Chen. ThunderSVM: A fast SVM library on GPUs and CPUs. *Journal of Machine Learning Research*, 19:797–801, 2018.

[25] Huimin Wu, Bin Gu, Zhengmian Hu, and Heng Huang. Fast and scalable adversarial training of kernel svm via doubly stochastic gradients. 2021.

[26] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020.

[27] Huan Xu, Constantine Caramanis, and Shie Mannor. Robustness and regularization of support vector machines. *Journal of machine learning research*, 10(7), 2009.