# Universal Noise Models in Machine Learning

A thesis presented
by
Juan Carlos Perdomo Silva

to the
Departments of Computer Science and Mathematics
in partial fulfillment of the honors requirements
for the degree of
Bachelor of Arts

Harvard College
Cambridge, Massachusetts
November 2017

To my grandfathers, Diego Perdomo and Juan Silva

# Acknowledgments

I feel incredibly grateful to have had the privilege of being advised by Yaron for the past year and a half. His thoughtful mentoring and encouragement have inspired me to grow as a computer scientist and to develop a deep passion for research that I hope to continue in the future. Sitting in his office and talking about our work together has been the highlight of my academic experience at Harvard and I feel very lucky and thankful to have been his student. Furthermore, I'd like to thank Boaz Barak for the fruitful discussions we've shared that have led to some of the results in this thesis.

I'm not sure how I can properly thank my parents for all their love and support they have given me during my time here in Cambridge. I wish I was a bit more poetically talented so I could write something truly memorable, but I'm a little short on verses. Thank you for always being there for me and for supporting me in all of my endeavors. I feel blessed to have you as my role models. And to my siblings, Iván and Carla, thank you for making me such a a proud older brother.

Lastly, I'd like to thank all my roommates: Nicholas Larus-Stone, Renan Carneiro, Matthew DiSorbo, CJ Christian, and Demren Sinik. Their friendship has made Harvard feel like home.

# Contents

# List of Figures

# Chapter 1

# Introduction

Imagine an algorithm that, given a picture of a street sign, tells you the name of the sign and the relevant action to take. Furthermore, suppose that the algorithm performs very well. On a test set of several thousand images, it has an accuracy of over 99%. Now, imagine that there was a way by which you could perturb any image in such a way that the changes were almost imperceptible, but the algorithm now reliably misclassifies the identity of the road mark. Would you trust the system to help drive your car?

Far from being a contrived scenario, this example captures some of the current challenges within the field of machine learning. Recent advances in the field have brought forth a new class of models that are capable of ever more impressive tasks. Improvements in computing power and the advent of deep learning have led to programs that equal or surpass human ability in domains such as image classification and speech recognition. Together, these advances have opened the door for groundbreaking new systems, such as facial recognition and self-driving cars, that have the potential to transform the way society operates.

However, a substantial body of research demonstrates that even the most sophisticated classification algorithms are incredibly susceptible to manipulation. Within the field of computer vision, recent papers show how a knowledgeable adversary can inject an imperceptibly small amount of noise to an image in order to induce a deep neural network (DNN) to misclassify the image, or even to predict a target label of his choice [27, 30, 40, 32]. The pervasiveness and ease with which one can find these adversarial examples raises serious doubts about the reliability of these new models in security sensitive settings like self-driving vehicles [24].

More broadly, the behavior of these learning algorithms in the presence of adversarial noise has sparked a shift in the way we think about machine learning. Traditionally, the

performance of a model has been analyzed on the basis of how well it represents a function captured in data or how computationally feasibly it is to train. In domains such as image classification, we say a model is better suited to a particular task if it achieves a higher accuracy on a test set than another model. Yet, what can faithfully be said about the effectiveness of a particular classifier if its behavior can be easily manipulated by a potential adversary? Analyzing machine learning algorithms solely on the basis of traditional attributes such as the number of samples required for training, runtime complexity, or just plain accuracy leads to an incomplete picture of their relative strengths and weaknesses and the effectiveness with which they can be deployed in real world scenarios.

Therefore, the motivation behind this work is to develop the theory behind *robust machine learning* and to better understand the limits of machine learning algorithms in the presence of noise. Robust machine learning combines ideas in statistical learning theory, algorithmic game theory, and robust optimization to develop algorithms for generating and defending against adversarial noise that have strong guarantees of performance. More specifically, in this thesis we focus on the question of constructing universal noise models. When designing algorithms that are robust to adversarial noise, we need to consider noise generated by the worst possible adversary and demonstrate that the classifier can defend against it. Similarly, when analyzing ways of fooling classifiers at a particular learning task, the adversary must generate noise that affects the performance of the best possible model. Universal noise models constitute precisely this worst case analysis for adversarial noise. Rather than attempting to trick a particular classifier, universal noise models generate noise that maximally impairs the performance of the best possible learning algorithm that might be used to approximate a given distribution.

In addition to being a theoretically rich subject, the study of universal noise algorithms has immediate practical applications. Given that they constitute a worst case analysis for adversarial noise, universal noise models serve as a clear basis with which to gauge claims regarding the robustness of different classification algorithms. Furthermore, training under adversarial noise has been demonstrated to serve as an effective way of defending against adversarial attacks [41]. Creating universal noise models would therefore serve to improve the quality of existing classification algorithms. Furthermore, these algorithms serve as a practical tool to protect private information contained in data. By bounding the maximum accuracy of a set of classifiers, universal noise algorithms allow individuals to publicly release data sets with a strong sense of security that no classifier can successfully identify more than a predetermined fraction of the data. We demonstrate this behavior on a series of image classification tasks.

So far, most of the focus on the problem of universal noise models has centered around empirical questions regarding the effectiveness of different approaches, especially within the context of deep learning. However, no one has yet advanced a well-principled theoretical framework with which to analyze the issue of universal noise. The goal of this thesis is to take a step towards filling this current void in our understanding of the problem and to develop solutions grounded in a cohesive theoretical framework that perform well in practice.

## 1.1 Contributions

Our main contribution is a formal characterization of the problem of universal noise models in machine learning as a two-player, zero-sum game. In addition to formalizing the problem, we provide an algorithm for finding solutions to the game that is guaranteed to yield an optimal distribution over adversarial noise perturbations.[1] In particular, we demonstrate how one can leverage the Multiplicative Weights algorithm and a Distributional-Oracle in order to generate a distribution over adversarial perturbations that minimizes the maximum accuracy of a finite set of classifiers over a data set. We show how such an oracle can be implemented in the case of affine classifiers and propose a generalization of the framework to the case of deep learning. Lastly, we illustrate the effectiveness of our approach in a number of applications. In short, our contributions can be summarized as follows:

- **Universal Noise as a Zero Sum Game.** We establish how the problem of generating universal adversarial noise for a set of $n$ classifiers can be framed as solving for the equilibrium strategies of a two-player, zero-sum game.
- **Distributional Oracles.** We examine the geometry of adversarial noise in machine learning in order to construct a Distributional-Oracle that can find noise perturbations which maximize the expected loss of a set of linear classifiers. Furthermore, we show how one can apply this algorithm within the Multiplicative Weights framework to boost the quality of the noise and approximate the equilibrium solution of the zero-sum game.
- **Universal Noise at Scale.** We demonstrate how our approaches to universal noise can be extended to efficiently find solutions even in cases where the number of possible classifiers is large.

[1] Optimal solutions in the context of adversarial noise are perturbations or distributions over perturbations that constitute a worst case analysis for learning algorithms. They maximally impair the performance of a classifier on a data set.

- **Universal Noise in Deep Learning.** We provide an extension of our DISTRIBUTIONAL-ORACLE to the domain of deep learning and demonstrate its effectiveness on a series of image classification tasks.

In Chapter 2, we begin by describing how the problem of universal noise models can be cast as a two-player, zero-sum game and present the relevant mathematical formalism. Afterwards, in Chapter 3 we introduce the reader to the geometry of adversarial noise and illustrate how creating a DISTRIBUTIONAL-ORACLE relates to the convexity of the classifier's decision boundary. In Chapter 4, we present our algorithms for universal noise. We prove how the DISTRIBUTIONAL-ORACLE can be combined with the Multiplicative Weights algorithm to find universal noise vectors by solving for the equilibrium strategies of the zero-sum game. Furthermore, we illustrate how our approaches can be extended to deep learning and how they can be made to run efficiently even in cases where the number of classifiers is large. Lastly, in Chapter 5 we demonstrate the effectiveness of our methods through a series of experiments on the MNIST dataset.

## 1.2 Related Work

One of the distinguishing characteristics of robust machine learning has been the wealth of perspectives from which members of the field have sought to understand the limits of machine learning under noise. Initially, one of the earliest topics that garnered interest was the question of whether or not functions remained PAC learnable if the labels of the available training set were subjected to random classification noise [43, 3]. During the mid to late 20th century, members of the field sought to devise algorithms that could approximate a given data distribution, even if the samples drawn from the distribution had their labels randomly changed [13]. Efforts towards this regard were marked by a series of positive results. Early on, researchers demonstrated how linear threshold functions and boolean conjunctions remained PAC learnable even if the training set was subject to random classification noise [13, 25, 7, 8]. These results were then generalized by Kearns through his *learning from statistical queries* model which demonstrates how most functions that are PAC learnable remain PAC learnable, even under the presence of random classification noise [22, 23].

Another, more recent direction of research builds upon this idea of training on corrupted data to analyze the effects of training on samples that have not been randomly corrupted, but rather purposefully manipulated by an adversary in order to influence the decision boundary of the resulting classifier. The key motivation underlying these so called poisoning attacks is

the notion that an adversary attempting to subvert a classification algorithm could potentially insert malicious examples to the training set. Upon training, the decision boundary of the classifier could therefore be manipulated in such a way that the algorithm incorrectly labels certain inputs. Recent papers demonstrate how these attacks can be implemented in the case of certain models like SVMs [35, 6].

Furthermore, adversarial noise has also been examined in the context of combinatorial optimization. Typically, when we think about optimizing a particular function, we assume that we have access to some information about the function, such as its gradient or its value on a set of points. While this assumption might hold in certain cases, often times the function must first be learned from data and hence is subject to approximation errors. In others, the function can only be accessed through noisy channels and therefore the information we receive is inexact. Given this behavior, significant effort has been recently devoted to understanding the limits of optimization under noise and its connections to machine learning [5, 19].

Conversely to the learnability of functions under random classification noise, work in this field of optimization under noise has been marked by a series of strong impossibility results. Work by Singer and Vondrak demonstrates that, while optimizing a convex function over a convex set can in general be done in polynomial time via zeroth order oracles, if the oracle is no longer exact, but rather subject to small amount $\delta$ of adversarial noise, no algorithm that uses subexponentially many samples can find a solution that is better than a factor of $\frac{1}{2} - \delta$ from optimal [38].

However, much of the recent attention devoted to the topic of robust machine learning and the questions examined in this thesis stem in large part from work done in deep learning and computer vision. These efforts were in large part sparked by discoveries published by Szegedy et al. in 2014. In their paper, *Intriguing Properties of Neural Networks*, the authors demonstrate how one can induce a DNN to misclassify an image by adding a small perturbation that maximizes the network's prediction error on that example (see Figure 1.1) [40]. Since then, there has been an overwhelming amount of publications that study this phenomenon of adversarial examples in deep learning from a variety of perspectives [30, 18, 20, 14, 28]. These have ranged from examining practical applications of adversarial noise in the real world [9, 31, 24], to theoretical analysis of the problem [29] and the design of new algorithms and architectures that defend against these attacks [33].

With regards to the topic of constructing noise algorithms, efforts have fallen into two main classes: gradient methods and optimization-based techniques. Gradient based methods leverage the differentiable nature of the loss function of a neural network to compute its gradient and thereby find a subspace of the input space that maximizes the network's prediction

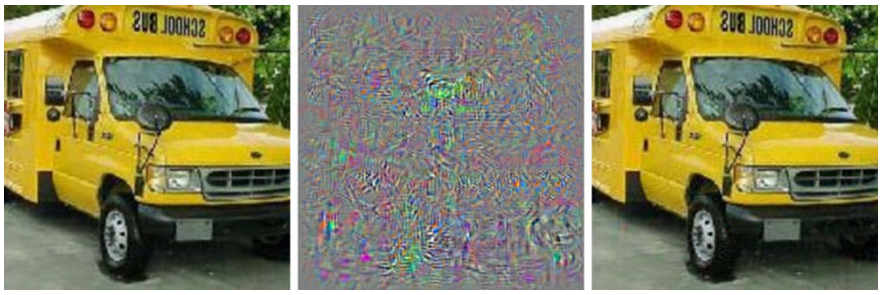


**Figure 1.1:** Szegedy et al. show that while the image on the left is correctly classified as a bus, adding the noise displayed in the middle results in the perturbed image on the right being labeled as an ostrich [40].

error. The fast gradient method developed by Goodfellow et al. was one of the first algorithms that demonstrated the effectiveness of this technique [18]. Interestingly enough, it remains one of the most commonly used algorithms to find adversarial examples since it can be computed very efficiently and has been shown to have good empirical results. However, recent work by Carlini and Wagner has shifted the attention to more optimization-based techniques. Rather than computing the gradient, their proposed algorithm seeks to minimize, via the Adam optimizer, an objective function that captures the prediction error of the neural network. These methods have been shown to yield a higher rate of misclassification than pre-existing gradient-based methods [10].

Most of the adversarial noise algorithms that have been developed for neural networks have been designed with the purpose of targeting a specific model. While the noise generated for particular classifier has been shown to be effective on other classifiers, even those trained on different subsets of the data, recent work in the field has more explicitly addressed this question of generating universal noise and finding adversarial examples that transfer across models [40, 42]. Specifically, researchers have demonstrated how generating noise that fools an ensemble of models tends to yield solutions that also affect a wider range of models [27]. These efforts are perhaps the most closely related to the questions we examine in this work.

Within this thesis, most of the theory behind universal noise generators is developed in the context of affine classifiers. However, we are not the first to examine noise algorithms for this class of models. In their 2016 paper, Moosavi-Dezfooli et al. take a geometric approach to quantify the robustness of a given linear classifier to a particular data point [30]. They build upon these insights into the geometry of adversarial examples to come up with a provably optimal algorithm to fool a single linear classifier. We will revisit some of these ideas later on and demonstrate how they can be extended to generate noise that is optimal not just for a single classifier, but rather for a wide range of possible models.

Lastly, in order to construct our universal noise algorithm we draw upon some of the work done within the field of online learning and boosting. More specifically, we employ the Multiplicative Weights framework as a way of boosting the noise generated by a distributional oracle in such a way that it approximates the equilibrium strategy of a zero-sum game. The Multiplicative Weights framework has been one of the most widely used algorithms of the past century, not just in computer science, but also in related fields such as economics. So much so, that it has been rediscovered multiple times by researchers working on different problems. During the 1950s, it was studied in the context of game theory and proven to find strategies that converge to the equilibrium solution of zero-sum games [34, 17]. In machine learning, it first appeared as the basis for Littlestone's Winnow algorithm to learn disjunctive Boolean formulas [26]. Shortly thereafter, it appeared in the work of Freund and Schapire as a way of boosting a set of weak PAC learners into a strong hypothesis for a learning problem [16, 15, 36]. Furthermore, it has been used in a variety of other contexts such as semidefinite programing as well as in computational geometry as part of Clarkson's algorithm for linear programming [4, 12].

# Chapter 2

# Problem Formulation

The problem of universal noise models is perhaps best motivated by the following example. Consider a privacy-minded individual PI, who wishes to upload a series of pictures from his recent vacation onto his social media page. PI wishes to make these images available to his friends and colleagues, but wants to make sure that no one else can identify the people in the pictures. In particular, PI wants to prevent social media site SM from successfully performing facial recognition on PI's images to identify the people in his album. We assume that SM has a set of pre-trained classifiers that can classify with high accuracy the relevant set of people in the images and that it can apply any of these models to PI's album. What is the best way to prevent SM from learning the identities of the individuals in the images, while still allowing PI's friends to appreciate their content? In essence, we seek a small change to the images that is almost imperceptible to a human, yet is guaranteed to trick SM's classifiers. Notice that, due to the binary nature of classification, SM can either find a classifier that identifies PI or it can't. Therefore, PI needs to ensure that the maximum accuracy of any of SM's classifiers is as small as possible. Minimizing the average accuracy of the models available to SM provides no guarantees of privacy for PI. Furthermore, changes in the underlying utilities for each party sum to zero. If PI improves his objective, it's only because SM has done worse and vice versa.

This example provides the intuition as to why generating universal noise can be formulated as a two-player, zero-sum game. In this chapter, we develop the rest of the mathematical formalism necessary to reason about the problem.

## 2.1 The Learner

The situation presented above is a learning problem. SM can be thought of as a learner who wishes to approximate the function represented by a particular data set. In this example, SM attempts to output a classifier that can correctly relate pictures to the identity of the people in those pictures. The adversary PI, on the other hand, tries to limit the ability of SM to extract the true information captured in the data by perturbing it slightly.

More concretely, the learner wants to output a hypothesis $h : \mathbb{R}^d \to [k]$ where $\mathbb{R}^d$ is the input space of the problem and $[k]$ is a finite set of labels $\{1, \ldots, k\}$. In our running example, images are represented as real valued vectors in $\mathbb{R}^d$ while the identities of the individuals are represented by integers in $[k]$. The goal of the learner is to choose the hypothesis $h$ from a class $\mathcal{H}$ that best approximates the relationships captured within a data set $S \subset \mathbb{R}^d \times [k]$, where $S = \{(x_1, y_1), (x_2, y_2), \ldots (x_m, y_m)\}$.

**Definition 2.1:** A **hypothesis** $h : \mathbb{R}^d \to [k]$ is a function from the input space $\mathbb{R}^d$ to the label set $[k]$.

Formally, $\mathcal{H} = \{h_1, \ldots, h_n\}$ describes a set of hypothesis functions that are available to the learner, such as linear classifiers or DNNs. For the purposes of this thesis, we assume that $\mathcal{H}$ is finite and of cardinality $n$. $S$ is a set of $m$ labeled examples where each example $(x, y)$ is drawn i.i.d from a particular distribution $\mathcal{D}$. $\mathcal{D}$ is a joint distribution over the domain set $\mathbb{R}^d$ and the label set $[k]$ that defines the precise relationship between inputs and outputs for the particular task. In the context of image recognition for example, $\mathcal{D}$ represents the relationship between images and the categories to which those images belong. The quality of a hypothesis $h$ is defined by how well it mimics the distribution $\mathcal{D}$. In particular, when we say that a learner wants to select the best hypothesis, we mean that he wishes to minimize the true risk (or error) $L_{\mathcal{D}}$ of the hypothesis $h$ over $\mathcal{D}$.

**Definition 2.2:** The **true risk** $L_D : \mathcal{H} \to [0, 1]$ of a hypothesis $h$ is the probability that $h$ makes an error when samples are drawn randomly from the distribution $\mathcal{D}$.

$$L_D(h) \stackrel{\text{def}}{=} \mathbb{P}_{(x,y)\sim D}[h(x) \neq y] \stackrel{\text{def}}{=} \mathcal{D}(\{(x, y)|h(x) \neq y\})$$

In practice, however, one does not have access to the distribution $\mathcal{D}$ and hence the learner cannot precisely measure the true risk of a hypothesis. Therefore, the learner resorts to minimizing the empirical risk of a hypothesis, namely how well it approximates the data set $S$.

**Definition 2.3:** The **empirical risk** $L_S : \mathcal{H} \to [0,1]$ of a hypothesis $h$ is the accuracy of the hypothesis over the data set $S$. Assuming $S$ has $m$ examples, we define:

$$L_S(h) \stackrel{\text{def}}{=} \frac{|\{i \in [m] \mid h(x_i) \neq y_i\}|}{m}$$

For the purposes of classification, these definitions are often the most natural since the performance of the learner is inherently binary: the prediction on a particular data point is either right or wrong. Implicit in this definition of the empirical risk of the hypothesis $h$, is the use of the 0-1 loss.

**Definition 2.4:** The **0-1 loss** $\ell_{0\text{-}1} : \mathcal{H} \times \mathbb{R}^d \times [k] \to \{0,1\}$ is defined as:

$$\ell_{0\text{-}1}(h, x_i, y_i) \stackrel{\text{def}}{=} \begin{cases} 0 & h(x_i) = y_i \\ 1 & h(x_i) \neq y_i \end{cases}$$

The empirical risk $L_S(h)$ of a hypothesis $h$ is then just the average 0-1 loss of the function over the data set $S$.

$$L_S(h) = \frac{1}{m} \sum_{i=1}^{m} \ell_{0\text{-}1}(h, x_i, y_i) \tag{2.1}$$

Given a data set $S$ of $m$ examples and a hypothesis class $\mathcal{H} = \{h_1, \ldots, h_n\}$ the goal of the learner is to select the hypothesis that minimizes the empirical risk:

$$\arg\min_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \ell_{0\text{-}1}(h, x_i, y_i) \tag{2.2}$$

So far, we have presented the goal of the learner in the absence of adversarial noise. In the next few sections, we go on to present how the idea of noise can be incorporated into the model and describe how the role of the learner changes once we take into account the fact that an adversary can perturb points in a data set. However, despite these changes, the high level goal of the learner, remains the same. His objective is still to approximate a distribution by selecting a hypothesis that minimizes the empirical risk on a data set.

## 2.2 The Adversary

As discussed previously, the adversary, PI, strives to limit SM's ability to distinguish between images by injecting a small amount of noise to the data set $S$ that maximally impairs the accuracy of SM's classifiers. Therefore, rather than performing classification on inputs drawn directly from the distribution, SM is forced to classify inputs that have been slightly perturbed by PI. One important observation to make is that, in the formulation of the problem presented at the beginning of the chapter, PI is indifferent as to the labels that SM predicts as long as they are not the correct ones. However, this need not be the case in general. Within the domain of adversarial noise models, we distinguish amongst two types of solutions.

The first is untargeted noise in which the adversary attempts to find the minimal distortion $v \in \mathbb{R}^d$ that causes the learner's hypothesis to predict any label other than the true label. Finding untargeted noise for a point $(x, y) \in S$ can be viewed as solution to the following optimization problem:

$$\begin{aligned} \underset{v \in \mathbb{R}^d}{\text{minimize}} \quad & ||v|| \\ \text{subject to} \quad & h(x+v) \neq y \end{aligned} \tag{2.3}$$

The second type of adversarial noise is targeted noise in which the adversary attempts to add noise to a point $(x, y)$ in order to get the learner's predictor to output a specific label (other than the true label). Given a target $j \in [k]$, where $j \neq y$ the relevant optimization problem for targeted noise is:

$$\begin{aligned} \underset{v \in \mathbb{R}^d}{\text{minimize}} \quad & ||v|| \\ \text{subject to} \quad & h(x+v) = j \end{aligned} \tag{2.4}$$

One of the key features of the framework, is that the adversary is restricted to only adding a small amount of noise to the learner's inputs. In our example, PI wants to leave his images relatively intact so that his friends can still appreciate them. This constraint is crucial as the adversary can trivially induce any point to be misclassified provided he is given enough of a noise budget.[1] For the purposes of the model, we suppose that the adversary PI can only perturb each point $x_i$ by adding a noise vector $v$ that has $\ell_2$ norm less than a predetermined value $\alpha$, where $\alpha \in \mathbb{R}^+$ is sufficiently small.[2] Now, we have the set up to properly define an adversarial example and to describe in detail the objective of the adversary.

[1]For example, the adversary could just provide an image whose true label belongs to a different class.
[2]The choice of the $\ell_2$ norm is not arbitrary. We discuss the reasons behind this decision in subsection 3.1.1

**Definition 2.5:** A point $x'$ in $\mathbb{R}^d$ is an **untargeted adversarial example** with regards to a noise budget $\alpha$, a hypothesis $h$, and a sample $(x, y)$ if:

1. $x' = x + v$
2. $||v|| \leq \alpha$
3. $h(x + v) \neq y$

**Definition 2.6:** A point $x'$ in $\mathbb{R}^d$ is a **targeted adversarial example** with target label $j$ if given a noise budget $\alpha$, a hypothesis $h$, and a sample $(x, y)$:

1. $x' = x + v$
2. $||v|| \leq \alpha$
3. $h(x + v) = j$

Given a noise budget $\alpha$, the goal of the adversary in the zero-sum game is to select the set of $m$ noise vectors $v = \{v_1, \ldots, v_m | \, \forall i \; v_i \in \mathbb{R}^d, ||v_i|| \leq \alpha\}$ that maximize the empirical risk of the learner. Assuming that the learner selects a hypothesis $h$, maximizing the empirical risk corresponds exactly to minimizing the accuracy of that hypothesis:

$$\arg\max_{v} \sum_{i=1}^{m} \ell_{0\text{-}1}(h, x_i + v_i, y_i) \tag{2.5}$$

In this presentation, the game is played with regards to the number of examples that are misclassified. Therefore, the noise that the adversary generates is untargeted. This will be the case for the majority of the thesis. Normally within machine learning we measure the performance of a learning algorithm on the basis of how well it approximates the relationship between inputs and outputs that is captured within a particular data set. If our learning algorithm fails at the particular task, we usually do not distinguish between incorrectly predicted labels. Therefore, untargeted noise is perhaps the most natural formulation of the problem. However, given that all the results presented in this thesis are easily generalizable to the case of targeted noise, this distinction is largely inconsequential.

### 2.2.1 A Brief Digression

The key assumption underlying the framework of adversarial noise is that small perturbations to a point in the input space should not alter the true label of that point. In the case of image recognition for example, it shouldn't be the case that adding an imperceptibly small amount of noise to a picture of a bus fundamentally changes the nature of the bus to be an ostrich. The points that are labeled as buses and ostriches should be relatively far apart in the input space under the true distribution $\mathcal{D}$. The fact that these adversarial examples exist in practice is because the learner's hypothesis overfits the data and fails to learn the true decision boundary of $\mathcal{D}$. In this section, we formalize these intuitions mathematically by introducing the concepts of a class separable distribution and an $\alpha$-locally consistent data set.

**Definition 2.7:** A distribution $\mathcal{D}$ over $\mathbb{R}^d \times [k]$ is **class separable** if the input space $\mathbb{R}^d$ can be decomposed into subsets $T_1 \dots T_k$ such that:

1. $T_i \bigcap T_j = \emptyset$ for all $i \neq j$
2. $\mathbb{R}^d \setminus \bigcup_i T_i$ is a set of measure zero
3. For all $T_i$ if $x_1, x_2 \in T_i$ then $y_1 = y_2$

**Definition 2.8:** A data set $S \subset \mathbb{R}^d \times [k]$ consisting of $m$ i.i.d examples drawn from a distribution $\mathcal{D}$ is said to be $\boldsymbol{\alpha}$**-locally consistent** if:

1. $\mathcal{D}$ is class separable and therefore $\mathbb{R}^d$ can be decomposed into sets $T_1, \dots, T_k$
2. For all $(x, y) \in S$, if $x \in T_i$ then $\forall\, x' \in \mathbb{R}^d$ such that $||x - x'|| \leq \alpha$, $x' \in T_i$.

Together, these definitions capture our intuitive notion that small changes to the points in a data set should not truly lead to changes in the underlying labels of those points. For the vast majority of problems that we are interested in solving using machine learning classifiers, the true decision boundary of a class should lie more than a small distance $\alpha$ away from points which we consider to be undisputed members of that class. Recent attention to this problem of adversarial noise stems from a deep seated belief that most data sets we encounter should be $\alpha$-locally consistent and that current classification algorithms are flawed. By developing the theory behind robust machine learning and adversarial noise, we might be able to better understand the shortcomings of current methods and develop new algorithms that move us past these issues.

## 2.3 Universal Noise as a Max-Min Objective

Having presented the roles of the learner and the adversary in our example, in this section we move on to discuss how these can be unified within a single framework and demonstrate how the problem of a universal noise models can be understood as the solution of a two-player, zero-sum game. In particular, we describe in detail how empirical risk minimization and adversarial noise generation can be reframed as the objectives of two players who compete to limit each other's progress at a particular learning task. We formalize the action spaces of both players and illustrate how universal noise can be understood as the solution to a max-min objective that stems from a game-theoretic analysis of the player's strategies.

Beginning with the learner, we propose that SM can be cast as the min player in the zero-sum game. Given a data set $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ of $m$ data samples, the learner strives to minimize the empirical risk of his predictor over the set $S$, bearing in mind that the adversary is capable of adding noise to each of the examples. Formally, the action space of SM consists of individual hypothesis in the set $\mathcal{H} = \{h_1, \ldots h_n\}$. Each individual hypothesis $h$ corresponds to a pure strategy played by the min player. Furthermore, we allow for SM to randomize across classifiers and to play mixed strategies over his action set $\mathcal{H}$. In order to make sense of the perturbations of the adversary, we assume that each of the learner's hypothesis achieves zero loss over the unperturbed inputs. In other words, $L_S(h) = 0$ for all $h \in \mathcal{H}$.

The adversary, PI, is modeled as the max player in the zero-sum game. The action space of the adversary is infinite. As discussed previously, the adversary selects a set of $m$ vectors $v = \{v_1, \ldots, v_m\}$ that are added to the set $S$ in order to yield the perturbed data set $S' = \{(x_1 + v_1, y_1), \ldots, (x_m + v_m, y_m)\}$. The only restriction on the action space of the adversary is that each individual vector $v_i$ must have $\ell_2$ norm less than $\alpha$. Each unique set of $m$ noise vectors corresponds to a pure strategy on the part of the adversary. Mixed strategies consist of distributions over individual sets of $m$ vectors.

Payouts to each player are decided based off the accuracy with which SM can approximate the functional relationships between inputs and outputs over the data set $S'$. More specifically, we specify payoffs as the average 0-1 loss of the learner over the perturbed data set. Given that the learner chooses hypothesis $h_i \in \mathcal{H}$ and the adversary selects a set of vectors $v$ we define the payout to each player to be $M_{S'}(i, v)$.

$$M_{S'}(i, v) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{j=1}^{m} \ell_{0\text{-}1}(h_i, x_j + v_j, y_j) \tag{2.6}$$

Since it is a zero-sum game, the learner technically receives the negative of the payout, while the adversary receives the payout itself. In the case of mixed strategies, we extend the utility function $M$ in the usual way. Let $\boldsymbol{w}$ be a distribution over hypothesis in the set $\mathcal{H}$ and $\boldsymbol{q}$ be a distribution over valid sets of perturbations $v$.[3] We then define:

$$M_{S'}(\boldsymbol{w}, v) = \frac{1}{m} \sum_{j=1}^{m} \mathop{\mathbb{E}}_{i \sim \boldsymbol{w}} \ell_{0\text{-}1}(h_i, x_j + v_j, y_j) \tag{2.7}$$

$$M_{S'}(i, \boldsymbol{q}) = \frac{1}{m} \sum_{j=1}^{m} \mathop{\mathbb{E}}_{v \sim \boldsymbol{q}} \ell_{0\text{-}1}(h_i, x_j + v_j, y_j) \tag{2.8}$$

In order to improve their objectives, each player solves for a strategy that that optimizes their worst-case payout. The learner solves for a distribution $\boldsymbol{w}$ over hypothesis in $\mathcal{H}$ that minimizes that maximum loss he can expect. Similarly, the adversary attempts to find a distribution $\boldsymbol{q}$ over perturbations that maximizes the minimum loss of the learner. Concretely:

Learner's Optimization:

$$\min_{\boldsymbol{w}} \max_{v} M_{S'}(\boldsymbol{w}, v) \tag{2.9}$$

Adversary's Optimization:

$$\max_{\boldsymbol{q}} \min_{i} M_{S'}(i, \boldsymbol{q}) \tag{2.10}$$

Von Neumann's minimax theorem states that the value of these two expressions is exactly the same for zero sum games [44]. Regardless of what order the optimization is carried out, both players will arrive at the same solution. We denote the value associated with this set of strategies by $\lambda$:

$$\min_{\boldsymbol{w}} \max_{v} M_{S'}(\boldsymbol{w}, v) = \max_{\boldsymbol{q}} \min_{i} M_{S'}(i, \boldsymbol{q}) = \lambda \tag{2.11}$$

Notice how the solution to the game is precisely the concept of a universal noise model. By finding the optimal distribution $\boldsymbol{q}$, the adversary is guaranteed to maximally inhibit the performance of any of the classifiers that SM might try to use. This is a much stronger conception of adversarial noise than that which had been analyzed in the literature up until

[3] We adopt the convention of denoting probability distributions in bold (e.g $\boldsymbol{w}, \boldsymbol{q}$)

this point. Researchers within the field of deep learning have been mostly focused on finding the worst case perturbation for a single classifier. In other words, given a hypothesis $h_i$, they have been trying to solve the following problem:

$$\max_{v} M_{S'}(i, v) \tag{2.12}$$

However, we posit that these kinds of analysis are too limited in scope. Ultimately, the topic of adversarial noise grows out of a concern to understand the limits of machine learning under noise. Yet, in order to do so one must consider the possibility that there could be multiple classification algorithms used for a particular learning task. Optimizing noise for a single classifier provides too narrow a perspective on the topic of noise, since parties interested in approximating a particular function are not restricted to using a single model.

Universal noise models are the solution to this set of concerns. They provide a distribution over perturbations that lower bounds the loss of the learner regardless of the particular classification algorithm used. Moreover, they capture the intuition that inherent in this model of adversarial noise is a strong concept of equilibrium. When generating noise for a data set, privacy minded individuals like PI wish to find a solution that is guaranteed to maximally limit the ability of any learning algorithm to successfully pattern match on the data. Rather than expending computational resources having to constantly change the classifier or noise model being used, we expect parties engaged in this kind of scenario to arrive at a stable set of solutions that optimize their individual objectives.

Furthermore, framing the problem as a zero-sum game has the added benefit of demonstrating how there are strict theoretical limits on the amount of misclassification that an adversary can achieve. No noise model can push the minimum loss of a classifier above the minimax value $\lambda$. This realization allows for a precise characterization the relationship between the noise budget $\alpha$ and the maximum accuracy of any classification algorithm, thereby revealing the underlying tradeoff between increasing data corruption and minimizing learning accuracy. By applying our universal noise algorithm, one can determine the value $\lambda$ of the game for a particular set of parameters and empirically verify how the value changes for different noise budgets.[4]

[4]So far, we can only analyze these tradeoffs by calculating $\lambda$ for games with a particular set of classifiers and noise budget. By calculating $\lambda$ for different sets of parameters, we can parametrize the relationship between the amount of noise and the minimum loss of the learner based off of these empirical measures.

# Chapter 3

# The Geometry of Adversarial Noise

Constructing adversarial noise algorithms relies on a deep understanding of the geometric properties of different classification algorithms. At a high level, adversarial examples exist because an adversary who can determine the decision boundary of a particular classifier can inject noise to a point that is correctly labeled in order to push it past the boundary of the relevant class. However, how do we reason about the decision boundary of a classifier in cases where the learner selects not a single hypothesis function but rather a distribution over hypothesis functions? The Multiplicative Weights framework serves as a way of dealing with this uncertainty and approximating the optimal mixed strategy for the adversary. However, in order to be applied, the framework relies on the existence of a Distributional-Oracle that can find the unique noise vector that maximizes the expected loss of the learner for a given distribution. The main result of this thesis is the construction of such an oracle:

**Theorem 3.1:** For affine binary classifiers, implementing a Distributional-Oracle to solve for the set of $m$ noise vectors that maximize the expected loss of the learner with respect to a distribution $\boldsymbol{w}$ over classifiers in $\mathcal{H}$, a noise budget $\alpha$, and a data set $S$ reduces to the problem of minimizing a quadratic function over a set of convex polytopes.

In order to prove this theorem, we begin in Section 3.1 by examining the geometry of adversarial noise in the context of binary classifiers. Then in Section 3.2 we extend our analysis to the case of multilabel classification. Finally, in Section 3.3 we prove Theorem 3.1 and demonstrate the existence of a Distributional-Oracle for linear classifiers.

## 3.1 Linear Binary Classifiers

Examining adversarial noise within the context of linear binary classifiers serves as the ideal starting point to understand the intuitions behind the different approaches one might adopt to generate noise. The hypothesis class of binary, linear classifiers consists of the set of hyperplanes that bisect the input space of a particular learning problem. Without loss of generality, we can view a binary linear classifier as vector $w \in \mathbb{R}^d$ and a scalar $b \in \mathbb{R}$. [1] We predict the label of a point in the input space by computing the following function:[2]

$$h_{w,b}(x) = \text{sign}(\langle x, w \rangle + b) \tag{3.1}$$

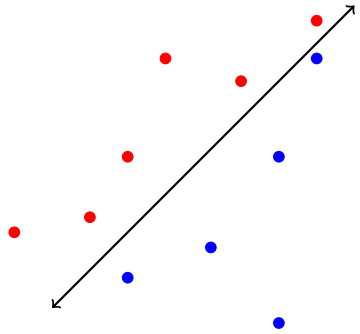


**Figure 3.1:** A typical binary classifer

The resulting sign of the linear function and consequently the predicted class of an individual point are determined by which side of the hyperplane the point lies on. As per Figure 3.1, blue points below the line pertain to one class while red points above it belong to a different class. Given this example, we can ask, what is the optimal way to add noise to a point so that it is misclassified by a linear classifier?

In this setting the answer is clear: the optimal way to fool a linear classifier is to push the point in the direction perpendicular to the decision boundary. This result was published by Moosavi-Dezfooli et al. as a motivation for their DeepFool algorithm for tricking deep neural

[1] In general, we distinguish between homogeneous and nonhomogenous linear functions. Homogenous functions are functions parametrized by a vector $w \in \mathbb{R}^d$. Given a point $x \in \mathbb{R}^d$, they compute the mapping $\langle w, x \rangle$ where $\langle \cdot, \cdot \rangle$ indicates the vector inner product in $\mathbb{R}^d$. Nonhomogenous functions on the other hand incorporate a bias term $b \in \mathbb{R}$ and compute the function $\langle w, x \rangle + b$. While homogenous functions represent hyperplanes that pass through the origin, nonhomogenous functions allow for hyperplanes that can be shifted above or below the origin. In general, these distinctions do not fundamentally change the nature of linear classification. Finding a nonhomogenous linear predictor reduces to the homogenous case if we simply add an extra 1 as a coordinate to every point the in the space. Therefore, for the sake of simplicity, we decide to focus on the case of nonhomogenous classifiers knowing that our results are easily generalized.

[2] In general, we think of labels as the set $[k] = \{1, \ldots k\}$. However, in the case of binary classifiers, we adopt the more natural convention of using the set {+1, -1}.

networks [30]. This perturbation can be quickly computed as the orthogonal projection of a point onto a hyperplane:

**Proposition 3.1:** Given a point $(x, y) \in \mathbb{R}^d \times \{+1, -1\}$, the perturbation of minimal norm that tricks a binary, linear classifier $h_{w,b}$ is

$$v = -\frac{\langle w, x\rangle + b}{||w||^2} w$$

*Proof.* The decision boundary of the linear predictor $h_{w,b}$ is defined by the set

$$D = \{x | \langle w, x\rangle + b = 0\} \tag{3.2}$$

Given a point $x_0$ we want to find the vector $v$ that minimizes the distance from $x_0$ to the set $D$. If we let $x'$ be a point in $D$, the optimal vector is simply the projection of $(x' - x_0)$ onto the vector $w$ since $w$ is perpendicular to the hyperplane. The projection of $(x' - x_0)$ is given by the following expression:

$$\frac{\langle x' - x_0, w\rangle}{||w||^2} w = \frac{\langle x', w\rangle - \langle x_0, w\rangle}{||w||^2} w \tag{3.3}$$

Now since we defined $x'$ to be in $D$ then $\langle w, x'\rangle + b = 0$ and $\langle w, x'\rangle = -b$. Therefore, we can simplify the above expression to:

$$\frac{-\langle x_0, w\rangle - b}{||w||^2} w = -\frac{\langle x_0, w\rangle + b}{||w||^2} w \tag{3.4}$$

□

We can decompose the above expression into two parts. The first is the vector $w$ which indicates the direction of the noise. The fact that $w$ is the optimal direction is no surprise since $w$ is the unique vector (up to scaling) that is perpendicular to the decision boundary.

The second part of the expression is the scalar value $-\frac{\langle x_0, w\rangle + b}{||w||^2}$. Without loss of generality, we will assume for the rest of this chapter that $||w|| = 1$.[3] Therefore, scalar value is just $-(\langle x_0, w\rangle + b)$. Since $w$ has length 1, this expression describes the signed distance from the

[3]Intuitively, we only care about the direction of $w$ since we can always rescale things to have norm 1. If a classifier correctly predicts the labels of all points in a set, then the following equation holds for all $i$: $y_i(\langle w, x_i\rangle + b) > 0$. If we multiply both sides by $1/||w||$ then the equation still holds. We can then just let $w' = w/||w||$ and $b' = b/||w||$ and we have the desired property.

point $x_0$ to the decision boundary of the linear predictor. If $y_0 = +1$ then $\langle x_0, w\rangle + b > 0$ and the optimal noise vector moves in the direction opposite of $w$. Geometrically, this conveys the intuitive notion that points that are above the hyperplane are pushed below it. The converse is true for points pertaining to the opposite class. If $y_0 = -1$ then $\langle x_0, w\rangle + b < 0$ and points are pushed in the positive direction.

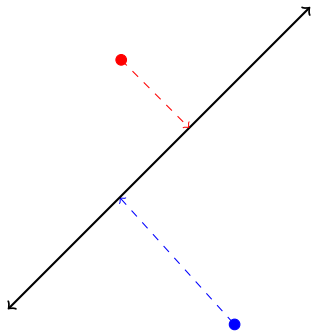


**Figure 3.2:** Optimal noise for linear, binary classifiers

Since the above algorithm is optimal, in the sense that it finds the minimal perturbation needed to induce misclassification, we can now relate the existence of adversarial examples to the noise budget $\alpha$. One immediate consequence of this result is the following:

**Corollary 3.1:** If the learner uses a linear predictor $h_{w,b}$, an adversarial example for a point $(x, y)$ exists if and only if the noise budget $\alpha$ is greater than the minimum distance from $x$ to the decision boundary of $h_{w,b}$.

*Proof.* The proof is simple. Since $\alpha$ is greater than $|\langle x_0, w\rangle + b|$, we let $\varepsilon = \alpha - |\langle x_0, w\rangle + b|$. Adding the vector $v = -(1+\varepsilon)(\langle x, w\rangle + b)w$ to the point $x$ changes the sign of $h_{w,b}(x)$:

$$\begin{aligned} h_{w,b}(x+v) &= \text{sign}(\langle x - (1+\varepsilon)(\langle x, w\rangle + b)w, w\rangle + b) \\ &= \text{sign}(\langle x, w\rangle - (1+\varepsilon)(\langle x, w\rangle + b)||w||^2 + b) \\ &= \text{sign}(\langle x, w\rangle + b - (1+\varepsilon)(\langle x, w\rangle + b)) \\ &\neq \text{sign}(\langle x, w\rangle + b) \\ &= h_{w,b}(x) \end{aligned}$$

Assuming the original prediction of $h_{w,b}$ on $x$ was correct, changing the sign of the input implies that the resulting perturbed point is now incorrectly classified.

Conversely, if an adversarial example exists, then there must be some perturbation $v$ such that $||v|| \leq \alpha$ and $h_{w,b}(x+v) \neq h_{w,b}(x)$. This implies that $x$ and $x+v$ must lie on opposite sides of the hyperplane defined by $w, b$. The distance between $x$ and the hyperplane is less than the distance between $x$ and $x+v$ since the path from $x$ to $x+v$ must intersect the hyperplane at some point. Consequently, the minimum distance from $x$ to the decision boundary is less than $\alpha$.

□

This indicates that the existence of an adversarial example is closely linked to the concept of the margin of a classifier, namely the minimum separation between the decision boundary of a classifier and points in the space. Proposition 3.1 shows that if a classifier has a margin greater than $\alpha$ then it is robust to noise that is generated with a budget of $\alpha$. Therefore, when thinking about ways to defend against noise, a dominant strategy is to train a classifier that has a margin larger than $\alpha$.

However, suppose that this is not possible. On a given data set $S$, no machine learning algorithm can train a classifier that achieves a minimum margin of separation greater than $\alpha$ with regards to points in the set. Is it still the case that the best defense is to make the margin as large as possible? It turns out that the answer to this question is no.

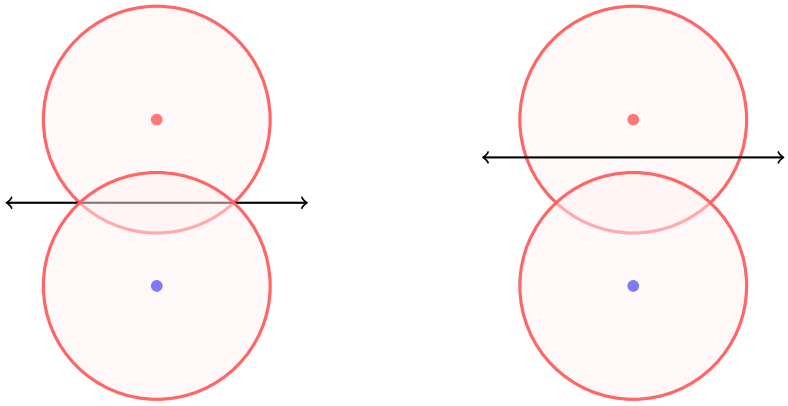


**Figure 3.3:** Increasing the margin of separation isn't always the best defense against noise.

In this pair of figures, the red circles of radius $\alpha$ around each point denote the noise budgets of the adversary for each point. They represent the feasible set of perturbations the adversary can choose from. Notice that in each case, the red and blue points are both correctly classified by the relevant affine classifiers. While the first classifier has a larger minimum margin of separation, both points can be pushed over its boundary. On the other hand, the second classifier one might say does a "worse" job at classifying the points since it

has a smaller margin, but it is more robust to adversarial attacks since adversarial examples for the blue point lie outside the feasible set of perturbations.

### 3.1.1 A Short Remark about Noise Budgets

So far in our model, we have assumed that the adversary is restricted to choosing noise vectors that have $\ell_2$ norm less than $\alpha$. However, in the literature, researchers have adopted a variety of methods to measure the quantity of noise. For example, Goodfellow et al. developed the fast gradient sign method as a way of generating adversarial examples for the case when the adversary is constrained by the $\ell_\infty$ norm of the perturbation. This choice was justified on the grounds that noise should be constrained to be small in each dimension [18]. On the other hand, Papernot et al. argue for the use of the $\ell_0$ norm since, in the context of image recognition, it precisely captures the number of pixels that have been altered in a particular image [33]. However, within the field of deep learning, recent work demonstrates that attacks that are optimized for the $\ell_2$ norm, tend to be stronger than others which use the $\ell_0$ or $\ell_\infty$ norm as the main metric [10]. Given this behavior, we adopt the $\ell_2$ norm as the distance metric with which to measure the magnitude of adversarial perturbations. In addition to this empirical justification, using the $\ell_2$ norm has the added theoretical benefit of simplifying geometric interpretations of different noise models since we can leverage Euclidean geometry.

As a final note, we'd like to point out that the choice of norm is not essential to any of the results we develop in this thesis. In our model, noise budgets are exogenously determined. Therefore, given a noise budget of $\alpha$ in a different norm, we can always relate it a corresponding noise budget $\alpha'$ in the $\ell_2$ norm.

## 3.2 Linear Multiclass Classifiers

Having presented the optimal way to generate noise for the case of binary linear classifiers, we now extend our results to the more general setting of multilabel classification. In order to generate a hypothesis capable of predicting amongst more than two classes, we assume that the learner adopts the "one-vs-all" approach to training multiclass linear classifiers. In addition to this approach, another common method of performing multiclass classification is the "all-pairs" method.[4] While we focus on the "one-vs-all" algorithm, the results developed

[4]The "all-pairs" method is often referred to as the "all-vs-all' method.

within this section can be readily extended to these alternate approaches. We present proofs of these extensions in the Appendix A.1.

Given $k$ classes to predict, the "one-vs-all" method trains $k$ separate linear predictors $h_{w_1,b_1} \dots h_{w_k,b_k}$. Each of the $h_{w_i,b_i}$ is a binary classifier trained to predict $+1$ if $x$ has label $i$ and $-1$ if it does not. Just as before, on input $x$, each hypothesis $h_{w_i,b_i}$ computes the transformation:

$$h_{w_i,b_i}(x) = \langle w_i, x \rangle + b_i \tag{3.5}$$

However, instead of returning an integer as in the binary case, we refrain from applying the sign function and just return a scalar value. On input $x$, the classifier computes the following function in order to determine the label of $x$:

$$h(x) = \arg\max_{i \in [k]} h_{w_i,b_i}(x) \tag{3.6}$$

Due to the number of individual hyperplanes for each class and the dependence on the magnitude of each prediction, understanding the decision boundary of a multiclass classifier is not as intuitive as in the binary case. However, the fact that the classifiers remain linear allows for a useful characterization of the input space as a partition of convex subsets.

**Lemma 3.1:** A "one-vs-all" multiclass linear predictor over $k$ labels partitions the input space into $k$ convex subsets $T_1, \dots, T_k$ where:

1. $T_i \cap T_j = \emptyset \quad \forall i \neq j$.
2. $\mathbb{R}^d \setminus \bigcup_i T_i$ is a set of measure zero.
3. For all $T_i$, if $h(x) = i$ then $x \in T_i$
4. For all $T_i$, if $x_1, x_2 \in T_i$ then $cx_1 + (1-c)x_2 \in T_i \quad \forall c \in [0,1]$

*Proof.* We define $T_i$ to be the region where $h_{w_i,b_i} > h_{w_j,b_j}$ for all $j \neq i$. (3) therefore follows directly from the definition of $T_i$ and a "one-vs-all" linear predictor. (4) follows from the fact that these are all affine functions. Assume $x_1, x_2 \in T_i$, then:

$$\langle w_i, x_1 \rangle + b_i > \langle w_j, x_1 \rangle + b_j \quad \forall j \neq i$$

Similarly:

$$\langle w_i, x_2 \rangle + b_i > \langle w_j, x_2 \rangle + b_j \quad \forall j \neq i$$

Given $x' = cx_1 + (1-c)x_2$, the following is true for all $j \neq i$:

$$\begin{aligned}
h_{w_i,b_i}(x') &= \langle w_i, x' \rangle + b_i \\
&= \langle w_i, cx_1 + (1-c)x_2 \rangle + b_i \\
&= c\langle w_i, x_1 \rangle + cb_i + (1-c)\langle w_i, x_2 \rangle + (1-c)b_i \\
&> c\langle w_j, x_1 \rangle + cb_j + (1-c)\langle w_j, x_2 \rangle + (1-c)b_j \\
&= \langle w_j, cx_1 + (1-c)x_2 \rangle + b_j \\
&= h_{w_j,b_j}(x')
\end{aligned}$$

Therefore, convex combinations of points with a particular label, also share that same label. (1) follows from the fact that if $x \in T_i \cap T_j$ then the relevant inequalities can't be mutually satisfied. If $x$ is in $T_i$ then:

$$\langle w_i, x \rangle + b_i > \langle w_j, x \rangle + b_j \quad \forall j \neq i$$

However, since $x$ is also in $T_j$ then $\langle w_j, x \rangle + b_j > \langle w_i, x \rangle + b_i$ must also be true. Hence, $x$ is not in $T_i \cap T_j$ and $T_i \cap T_j = \emptyset$. Lastly, the set $\mathbb{R}^d \setminus \bigcup_i T_i$ is equal to the set of points $x$ where there are ties for the maximum valued hypothesis:

$$\mathbb{R}^d \setminus \bigcup_i T_i = \{x | \max_i h_{w_i,b_i}(x) = c \ \wedge \ \exists i, j \text{ s.t } h_{w_i,b_i}(x) = h_{w_j,b_j}(x) = c\} \tag{3.7}$$

This set is a subset of the set of points on the intersections of two hyperplanes:

$$\mathbb{R}^d \setminus \bigcup_i T_i \subset \{x | \exists \ i, j \text{ s.t } \langle w_i, x \rangle + b_i = \langle w_j, x \rangle + b_j; \} \tag{3.8}$$

This set has measure zero. For all $\varepsilon > 0$, there exists an $x'$ such that $||x - x'|| < \varepsilon$ and $x' \notin \mathbb{R}^d \setminus \bigcup_i T_i$ since the intersection of two distinct hyperplanes is of dimension 2 less than the overall space. Therefore, $\mathbb{R}^d \setminus \bigcup_i T_i$ must also have measure zero. □

Endowed with this understanding that the domain of a linear multiclass predictor can be decomposed into convex subsets, we can now develop an algorithm for finding adversarial examples.

**Proposition 3.2:** The problem of finding targeted and untargeted adversarial examples for a linear, multiclass predictor using the "one-vs-all" approach reduces to minimizing a quadratic function over a convex set.

*Proof.* As per equation 2.3, given a sample point $(x, y)$, solving for untargeted adversarial examples can be framed as the following optimization problem:

$$\begin{aligned} \min_{v \in \mathbb{R}^d} \quad & ||v|| \\ \text{subject to} \quad & h(x+v) \neq y \end{aligned} \tag{3.9}$$

Instead of solving this directly, we demonstrate how we can reduce the problem above to the problem of finding targeted adversarial examples. Given a target label $j$ if we can solve for:

$$\begin{aligned} \min_{v \in \mathbb{R}^d} \quad & ||v|| \\ \text{subject to} \quad & h(x+v) = j \end{aligned} \tag{3.10}$$

Then we can solve for the untargeted case by iterating through all $j \in [k]$ and choosing the solution with minimal $\ell_2$ norm. In the case of linear multiclass predictors, the constraint $h(x+v) = j$ can be written as a set of $k-1$ linear inequalities:

$$h(x+v) = j \iff \langle w_j, x \rangle + b_j > \langle w_i, x \rangle + b_i \quad \forall i \neq j \tag{3.11}$$

As per Lemma 3.1, the points that satisfy equation 3.11 form a convex set. Additionally, since norms are nonnegative, we can apply a monotonic transformation to the objective and still be guaranteed the same solution. If we square the norm of the vector, we can rewrite equation 3.10 as the minimization of a quadratic function over a convex set:[5]

$$\begin{aligned} \min_{v \in \mathbb{R}^d} \quad & ||v||^2 \\ \text{subject to} \quad & \langle w_j, x \rangle + b_j > \langle w_i, x \rangle + b_i \quad \forall i \neq j \end{aligned} \tag{3.12}$$

□

It turns out that there are several commercially available quadratic program solvers that can be used to efficiently find solutions to these kinds of problems [2, 1, 39]. In Chapter 5, we implement our methods using such solvers. While we do not express the constraint that $||v|| \leq \alpha$ directly, we can simply solve for the minimal $v$ and verify if it has norm less than $\alpha$.

[5] Technically, for this to be a valid optimization we need to make the inequalities weak inequalities so that we optimize a function over a closed set. However, we can employ a series of positive slack variables $\varepsilon_i$ to ensure that these inequalities are strict. For simplicity, we omit these extra variables from the presentation.

## 3.3 Randomization and Distributional Oracles

In the previous two sections, we introduced adversarial noise algorithms that find the optimal perturbation to trick a single affine classifier. Given a noise budget of $\alpha$, if an adversarial example exists within the feasible set of perturbations, they are guaranteed to find the minimal noise vector $v$ such that $x + v$ is misclassified by the learner's hypothesis. Relating this to our model presented earlier, when presented with a data set $S$ and a linear predictor $h_i$, these algorithms solve the following optimization problem for the adversary:

$$\arg\max_{v} M_{S'}(i, v) \tag{3.13}$$

The nature of these algorithms implies that pure strategies for the learner are unlikely to be robust to noise since the adversary can easily maximize the learner's loss. Therefore, it is natural to expect the learner to randomize over classifiers and force the adversary to deal with the resulting uncertainty. However, as we demonstrate in Theorem 3.1, there are adversarial noise algorithms that can maximize the loss of the learner even in cases where he plays distributions over hypothesis. In particular, we demonstrate how one can construct a DISTRIBUTIONAL-ORACLE that solves for the unique perturbation $v$ that maximizes the expected loss of the learner when he plays distributions over linear classifiers.

**Definition 3.1:** Given a data set $S$ of size $m$, a noise budget $\alpha$ and a distribution $\boldsymbol{w}$ over $\mathcal{H}$ a DISTRIBUTIONAL-ORACLE returns the set of $m$ noise vectors $v$ of magnitude less than $\alpha$ which maximizes the expected loss of the learner:

$$\text{DISTRIBUTIONAL-ORACLE } (\boldsymbol{w}, \alpha) = \arg\max_{v} M_{S'}(\boldsymbol{w}, v) \tag{3.14}$$

**Theorem 3.1:** For affine binary classifiers, implementing a DISTRIBUTIONAL-ORACLE to solve for the set of $m$ noise vectors that maximize the expected loss of the learner with respect to a distribution $\boldsymbol{w}$ over classifiers in $\mathcal{H}$, a noise budget $\alpha$, and a data set $S$ reduces to the problem of minimizing a quadratic function over a set of convex polytopes.

Before moving on to presenting the proof of this theorem, we further discuss some of the motivations behind the result and explain how they relate to the shape of the decision boundaries of linear classifiers. As presented in Chapter 2, the problem of universal noise can be neatly modeled as a two-player zero-sum game. According to Nash's famous theorem, every game has a mixed strategy Nash equilibrium. Yet, not all games have pure strategy equilibria. One question we might ask ourselves then is whether the class of games described

in the context of universal noise can always be solved with pure strategies. If this is the case, then we do not need to construct a DISTRIBUTIONAL-ORACLE or apply the Multiplicative Weights algorithm in order to solve for the relevant set of strategies. We can simply iterate through all possible hypothesis for the learner, compute the response of the adversary using one of the algorithms presented in the previous sections, and return the one that achieves the minimum loss. However, such an approach would fail to find optimal strategies for the learner, as mixed strategy equilibria are indeed necessary to solve a vast subset of these games. Consider the following example:

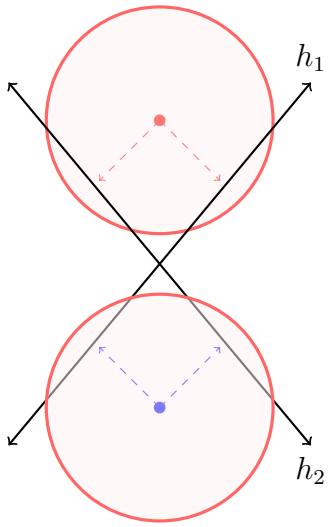



**Figure 3.4:** Randomization is a necessary requirement for universal noise.

In this binary classification example, if played individually, each hypothesis would be completely fooled by the adversary since both lines intersect the feasible noise budget of each point.

$$\max_{v} M_{S'}(i, v) = 1 \quad \forall i$$

However, if we consider mixed strategies for the learner, due to the symmetry of the action space, the unique equilibrium strategy would be to choose the distribution $\boldsymbol{w}$ that randomizes evenly across both classifiers. The optimal strategy $\boldsymbol{q}$ for the adversary would similarly be to randomize the noise across $h_1$ and $h_2$ with equal probability. Now we have that:

$$\max_{\boldsymbol{q}} M_{S'}(\boldsymbol{w}, \boldsymbol{q}) = .5$$

The maximum loss of the learner has gone down by half. This example demonstrates how in the case where a learner is unable to find classifiers that are individually robust to noise, being able to randomize across different hypothesis is a simple and effective defense. Furthermore, in general, randomization is not a only a sufficient condition, it is also a necessary requirement to generate universal noise for a set of classifiers since the learner is likely to randomize across hypothesis.

Having established the need for an approach to universal noise that can cope with distributions over hypothesis, we now move on to present the theory behind our DISTRIBUTIONAL-ORACLE algorithm. Much like our approach for fooling multiclass classifiers, we demonstrate how one can leverage the convex geometry of linear classifiers to find a single noise vector that maximizes the expected loss of the learner. Beginning with the case of linear, binary classifiers, assume that there are $n$ classifiers $h_1, \dots h_n$ each played with weight $w_i$, where $i \in [n]$. Given a data set $S = \{(x_1, y_1), \dots (x_m, y_m)\}$, the objective of the adversary is to find the set of perturbations $v = \{v_1, \dots, v_m\}$ that maximizes the expected loss of the learner:

$$\max_{v} M_{S'}(\boldsymbol{w}, v) = \max_{v} \frac{1}{m} \sum_{i=1}^{n} \sum_{j=1}^{m} w_i \cdot \ell_{0\text{-}1}(h_i, x_j + v_j, y_j) \tag{3.15}$$

The first observation we make is that this objective can be optimized independently for each vector $v_j$. Changing the value of $v_j$ for a particular $j$ only alters the value of terms in the sum where $v_j$ appears. Therefore, we can maximize the loss in equation 3.15 by finding the optimal $v_j$ for $m$ different expressions of the form:

$$\max_{v_j} \sum_{i=1}^{n} w_i \cdot \ell_{0\text{-}1}(h_i, x_j + v_j, y_j) \tag{3.16}$$

Hence, a distributional oracle for the general case of a data set $S$ of $m$ examples simply consists of $m$ calls to a distributional oracle for a data set with a single example. Next, in Lemma 3.2 we point out that the loss of the learner is constant in regions where all the classifiers $h$ predict the same sign for $x + v$. Furthermore, each of these regions is convex:

**Lemma 3.2:** In the case that the learner selects $n$ classifiers, $h_1, \dots h_n$ each with weight $w_i$ where $i \in [n]$, the hyperplanes defined by the hypothesis partition the input space into $2^n$ convex subsets $T_j$ where:

1. $\sum_{i=1}^{n} w_i \cdot \ell_{0\text{-}1}(h_i, x, y) = a_j$ for all $(x, y)$ such that $x \in T_j$
2. $T_i \cap T_j = \emptyset \quad \forall i \neq j$

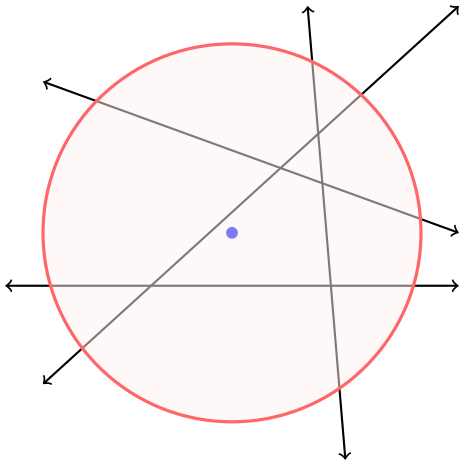


**Figure 3.5:** Hyperplanes partition the set of feasible points into convex subsets

3. $\mathbb{R}^d \setminus \bigcup_j T_j$ is a set of measure zero.

*Proof.* We identify each region $T_j$ with a sign vector $s_j = (s_{j,1}, \ldots, s_{j,n}) \in \{+1, -1\}^n$ where each index of the vector $s_{j,i}$ indicates the sign of hypothesis $h_i$ on points within the region $T_j$. There are $2^n$ such vectors and hence $2^n$ possible regions.[6] Furthermore, each region is convex. If $x_1, x_2 \in T_j$ then convex combinations of $x_1, x_2$ are also in $T_j$.

To show this, choose an arbitrary $h_i$. By the definition of $T_j$, $h_i(x_1) = h_i(x_2)$. Without loss of generality, assume that $h_i$ predicts that both points have positive signs. Consider the point $x' = cx_1 + (1-c)x_2$ where $c \in [0, 1]$:

$$\begin{aligned} h_i(x') &= \text{sign}\left(\langle h_i, cx_1 + (1-c)x_2\rangle + b_i\right) \\ &= \text{sign}\left(c(\langle h_i, x_1\rangle + b_i) + (1-c)(\langle h_i, x_2\rangle + b_i)\right) \\ &= +1 \end{aligned}$$

By our initial assumption both of the sums in parenthesis in the second equation have value greater than zero, therefore the entire expression has value greater than 0. Since $h_i$ was arbitrary, this proves that convex combinations of points in $T_j$ have the same sign with respect to any hypothesis and hence the $T_j$ are convex sets.

[6] While there are at most $2^n$ regions, some regions are often just the empty set since the inequalities corresponding to a set of linear hypothesis predicting a particular sign might be mutually infeasible.

Furthermore, the loss of the learner assumes a constant value in each of the sets. The expression for the loss is defined as:

$$\sum_{i=1}^{n} w_i \cdot \ell_{0\text{-}1}(h_i, x, y)$$

Since the $w_i$ are fixed, the expression only changes value if the 0-1 loss changes value. However, the 0-1 loss is solely determined by the sign of the hypothesis on a particular example. Since these are defined to be the same for all points in a set $T_j$, the value of the 0-1 loss remains the same for all points in $T_j$ and hence the expected loss of the learner can have at most $2^n$ distinct values (one for each region).

Lastly, as in Lemma 3.1, $\mathbb{R}^d \setminus \bigcup_j T_j$ consists of intersections of hyperplanes and therefore has measure zero. And $T_i \cap T_j = \emptyset$ since their corresponding sign vectors $s_i$ and $s_j$ must differ on at least one index. Points cannot belong to the intersection of $T_i$ and $T_j$ since the corresponding inequalities for that index cannot be mutually satisfied. □

With this result, we can now prove Theorem 3.1 and show how maximizing the expected loss of the learner corresponds to minimizing a quadratic function over a convex set.

**Theorem 3.1:** For affine binary classifiers, implementing a DISTRIBUTIONAL-ORACLE to solve for the set of $m$ noise vectors that maximize the expected loss of the learner with respect to a distribution $\boldsymbol{w}$ over classifiers in $\mathcal{H}$, a noise budget $\alpha$, and a data set $S$ reduces to the problem of minimizing a quadratic function over a set of convex polytopes.

*Proof.* Formally, the DISTRIBUTIONAL-ORACLE solves the following problem:

$$\max_{v} \frac{1}{m} \sum_{i=1}^{n} \sum_{j=1}^{m} w_i \cdot \ell_{0\text{-}1}(h_i, x_j + v_j, y_j)$$

However as we discussed previously, this expression can be optimized independently for every $j$ by considering losses of the form:

$$\max_{v_{j'}} \sum_{i=1}^{n} w_i \cdot \ell_{0\text{-}1}(h_i, x_{j'} + v_{j'}, y_{j'})$$

As per Lemma 3.2, the value of this expression is constant within regions $T_j$ that are defined by sign vectors $s_j = (s_{j1}, \ldots, s_{jn})$ where $s_{ji}$ indicates the sign predicted by hypothesis

$h_i$ on points $x' \in T_j$. Since each set $T_j$ is convex, we can solve for the minimum perturbation $v$ such that $x + v$ is in $T_j$ using quadratic programming:

$$\begin{aligned} \min_{v \in \mathbb{R}^d} \quad & ||v||^2 \\ \text{subject to} \quad & \text{sign}(\langle w_1, x + v \rangle + b_1) = s_{j1} \\ & \ldots. \\ & \text{sign}(\langle w_n, x + v \rangle + b_n) = s_{jn} \end{aligned} \tag{3.17}$$

In this equation, $w_i, b_i$ are the parameters for the linear predictor $h_i$. Furthermore, each constraint of the form $\text{sign}(\langle w_i, x + v \rangle + b_i) = s_{ji}$ is just a linear inequality.[7] Having found a feasible solution, $v$ we then evaluate the expression $\sum_{i=1}^{n} w_i \cdot \ell_{0\text{-}1}(h_i, x+v, y)$. By enumerating over all possible regions $T_j$, and then filtering out based on the criterion that $||v|| \leq \alpha$, we are guaranteed to find the single perturbation that maximizes the loss of the learner within the given noise constraints. □

While this result is developed in the context of affine, binary classifiers, the results are easily generalizable to multiclass classification. We provide such an extension in the Appendix A.2.

[7] As in equation 3.12 in order to solve the program, we must express these as weak inequalities. However, we can again use a similar technique of employing slack variables $\varepsilon_i > 0$ in order to convert weak inequalities into strict inequalties.

# Chapter 4

# Algorithms for Universal Noise

In the previous chapters, we have presented how the problem of constructing universal noise models reduces to solving for the set of equilibrium strategies of a zero-sum game and introduced a series of approaches for fooling linear predictors. In this chapter, we present an algorithmic framework for generating universal noise by approximating the minimax value of various classes of games. In Section 4.1, we demonstrate how in the domain of affine classification, we can use the DISTRIBUTIONAL-ORACLE presented in Chapter 3 along with the Multiplicative Weights framework to boost the noise generated by the oracle into a distribution that is guaranteed to be within an arbitrary factor of optimal. In Section 4.2, we generalize our approaches to generate universal noise in cases where the number of hypothesis is large. Lastly, we conclude the chapter in Section 4.3 by presenting how our methods can be extended to also generate noise not just for affine classifiers but also for general deep learning models.

## 4.1 The Multiplicative Weights Algorithm

The Multiplicative Weights algorithm has been used as a efficient method of implementing no-regret dynamics and approximately solving zero-sum games in a variety of contexts [11, 15, 34, 17, 21, 4]. In this section, we prove how the framework can be implemented in the domain of adversarial noise in order to generate a universal noise model that is approximately optimal.

Formally, the goal of the universal noise algorithm is as follows: given an error parameter $\delta$, a data set $S$, a noise budget $\alpha$, and a set $\mathcal{H}$ of hypothesis for the learner, find a distribution $\tilde{\boldsymbol{q}}$ over noise vectors of magnitude less than $\alpha$ that is within a factor $\delta$ of optimal. If we let:

$$\lambda = \max_{\boldsymbol{q}} \min_{i} M_{S'}(i, \boldsymbol{q}) \tag{4.1}$$

Then we want to find a $\tilde{\boldsymbol{q}}$ such that:

$$\min_i M_{S'}(i, \tilde{\boldsymbol{q}}) \geq \lambda - \delta \tag{4.2}$$

For the purposes of our algorithm, we will draw upon our earlier result in Theorem 3.1 that demonstrates how one can construct a DISTRIBUTIONAL-ORACLE that finds the unique set of perturbations (or equivalently the pure strategy) that maximizes the expected loss of the learner for any distribution over classifiers.

$$\text{DISTRIBUTIONAL-ORACLE } (\boldsymbol{w}, \alpha) = \arg\max_v M_{S'}(\boldsymbol{w}, v) \tag{4.3}$$

Using this DISTRIBUTIONAL-ORACLE to model the actions of the adversary, we can run Multiplicative Weights on the set of hypothesis available to the learner in order to boost the quality of the noise into a distribution that is approximately optimal.

**Algorithm 1: MWU for Universal Noise**

**Input:** $\mathcal{H} = \{h_1, \dots h_n\}$, noise budget $\alpha$, $S = \{(x_1, y_1), \dots (x_m, y_m)\}$, $\varepsilon$, $T$

**Output:** Distribution $\tilde{\boldsymbol{q}}$ over $v \in \mathbb{R}^d$, distribution $\tilde{\boldsymbol{w}}$ over $\mathcal{H}$

$w_i^{(1)} \leftarrow 1/n \quad \forall i \in [n]$

**for** $t \in [T]$ **do**

  $v^{(t)} \leftarrow$ DISTRIBUTIONAL-ORACLE $(\boldsymbol{w^{(t)}}, \alpha)$

  **for** $i \in [n]$ **do**

    $w_i^{(t+1)} \leftarrow w_i^{(t)}(1-\varepsilon)^{M_{S'}(i, v^{(t)})}$

  **end**

  Normalize $\boldsymbol{w}^{(t+1)}$

**end**

**return** Distribution $\tilde{\boldsymbol{q}}$ that assigns weight $\frac{\{t | v^{(t)} = v\}}{T}$ to the set of noise vectors $v$,

  $\tilde{\boldsymbol{w}} = \arg\min_{t \in [T]} M_S(\boldsymbol{w}^{(t)}, \boldsymbol{w}^{(t)})$

**Theorem 4.1:** Given an error parameter $\delta$, after $O(\frac{\ln n}{\delta^2})$ iterations, Algorithm 1 returns distributions $\tilde{\boldsymbol{w}}$, $\tilde{\boldsymbol{q}}$ such that:

$$\min_i M(i, \tilde{\boldsymbol{q}}) \geq \lambda - \delta$$

$$\max_v M(\tilde{\boldsymbol{w}}, v) \leq \lambda + \delta$$

*Proof.* We present the proof of the theorem in Section A.3. □

## 4.2 Universal Noise at Scale

While the algorithm presented in the previous section is guaranteed to return a solution whose error goes to zero as the number of iterations goes to infinity, actually running the algorithm for a large number of iterations when the size of the hypothesis set or the number of samples are large is computationally infeasible. Each call to the DISTRIBUTIONAL-ORACLE takes $O(m\,\text{poly}(n)2^n)$ where $n = |\mathcal{H}|$ and $m = |S|$. For an individual point $(x_j, y_j)$, the oracle finds the corresponding noise vector $v_j$ by searching over $2^n$ regions in the input space and solving for the minimal vector that can push the point $x_j$ into that region. Therefore, the DISTRIBUTIONAL-ORACLE computes a total of $2^n$ quadratic programs, where each program takes $\text{poly}(n)$ to solve [2, 1, 39]. This process gets repeated $m$ times, one for each example in $S$. In the case of multilabel classification, the asymptotic complexity is $O(m\,\text{poly}(n)k^n)$ since there are $k^n$ label vectors $s \in [k]^n$, where $k$ is the number of labels.

Despite its exponential running time, our approach remains an effective mechanism of generating universal noise for a wide range of learning tasks. In the literature, most of the problems that have been examined in the context of universal noise suppose that the learner has access only to a small number of hypothesis. This is due to the fact that for relatively complex distributions $\mathcal{D}$ where the number of labels is large or the true decision boundary between different classes is nonlinear, we expect there to only be a small number of linear predictors that can explain the data reasonably well. Having classifiers that achieve low empirical risk is a key feature of the model, since if the learner selects hypothesis that have poor accuracy, the adversary need not perturb the inputs at all in order to induce misclassification. While one could certainly perturb classifiers by $\epsilon$ in order to increase the size of the hypothesis set by an arbitrarily large amount, small changes in the decision boundaries of the classifier should not lead to drastic changes in the quality of the noise distribution.[1]

In cases where the number of hypothesis is large, we can extend our framework to generate universal noise at scale. To do so, we can relax the formulation of the problem to use a convex loss function instead of the 0-1 loss. Replacing nonconvex loss functions by convex alternatives is a standard technique used within machine learning to circumvent problems with high computational complexity. Even simple problems such as linear binary classification are known to be NP-hard in the unrealizable case using the 0-1 loss [37]. Since convex combinations of convex functions are convex, we can maximize the loss of the learner via standard methods of optimizing convex functions such as stochastic gradient descent. In

[1] We currently have no theoretical guarantees of this behavior, however, we conjecture that further exploration of the topic should yield such a result.

the original formulation of the problem, the oracle is tasked with maximizing $m$ different objective functions of the form:

$$\max_{v_j} \sum_{i=1}^{n} w_i \cdot \ell_{0\text{-}1}(h_i, x_j + v_j, y_j) \tag{4.4}$$

Applying stochastic gradient descent to this function would not yield any interesting solutions since the gradient of the 0-1 loss is constantly zero everywhere. However, if we instead apply a convex loss function, then it does become possible to optimize the objective efficiently. In particular, we propose using a variant of the hinge loss $\ell_r$:

**Definition 4.1:** The **reverse hinge loss** is a function $\ell_r : \mathcal{H} \times \mathbb{R}^d \times [k] \to \mathbb{R}_{\geq 0}$ defined on examples $(x, y)$ and linear predictors $h_{w,b}$ as:

$$\ell_r(h_{w,b}, x, y) = \max\{0, y(\langle w, x\rangle + b)\}$$

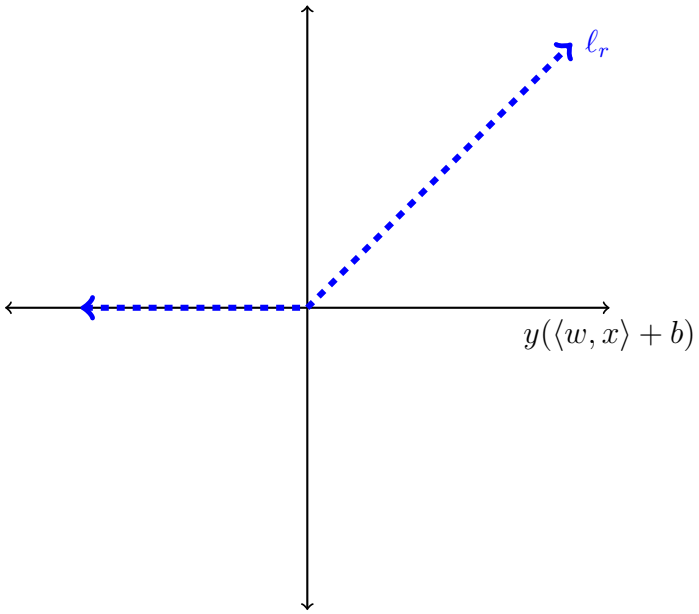



**Figure 4.1:** The Reverse Hinge Loss

From the definition of the reverse hinge loss, we see that $\ell_r(h, x, y) = 0$ if and only if $\ell_{0\text{-}1}(h, x, y) = 1$.[2] Hence minimizing the reverse hinge loss exactly corresponds to maximizing the 0-1 loss. Furthermore, the reverse hinge loss is convex since the gradient of the function

[2]The current presentation of the reverse hinge loss assumes that the learner that the learning task is one of binary classification, however these results hold for multiclass as well. We present an extension of the reverse hinge loss to multilabel classification in Section A.4

is monotonically nondecreasing. If we rewrite the loss of the learner in equation 4.4 using the reverse hinge loss, we arrive at a function that is a weighted sum of convex functions where all the weights are nonnegative. Such functions are guaranteed to be convex [37].

$$\min_{v_j} \sum_{i=1}^{n} w_i \cdot \ell_r(h_i, x_j + v_j, y_j) \tag{4.5}$$

In it's current form, this objective does not take into account the fact that the magnitude of the noise is constrained to have $\ell_2$ norm less than $\alpha$. However, we can use a variant of the stochastic gradient descent algorithm that includes a projection step to ensure that $||v|| \leq \alpha$. Such a variation restricts the optimization to a compact and convex set, yet it does not alter the convergence guarantees of the algorithm [37]. Since convex functions can be efficiently optimized using SGD, in cases where the number of hypothesis is large we can attempt to implement an approximate DISTRIBUTIONAL-ORACLE by using the reverse hinge loss. Even though we provide no guarantees of its performance with regards to the true loss of the learner, if we assume that the result of the gradient descent optimization returns solutions that are a bounded factor $\beta$ from optimal, then we can use our algorithm for universal noise to get a distribution over noise vectors that is within a constant factor $\beta + \delta$ from the equilibrium strategy.

**Definition 4.2:** Given a data set $S$ of size $m$, a noise budget $\alpha$ and a distribution $\boldsymbol{w}$ over $\mathcal{H}$, a $\beta$-DISTRIBUTIONAL-ORACLE returns a set of $m$ noise vectors $\tilde{v}$ of magnitude less than $\alpha$ that are within a constant factor $\beta$ from the optimal solution:

$$\beta\text{-DISTRIBUTIONAL-ORACLE } (\boldsymbol{w}, \alpha) = \tilde{v}$$

$$M_{S'}(\boldsymbol{w}, \tilde{v}) \geq \max_{v} M_{S'}(\boldsymbol{w}, v) - \beta$$

**Theorem 4.2:** Running the universal noise algorithm with parameters $\mathcal{H}$, $S$, $\alpha$ with a $\beta$-DISTRIBUTIONAL-ORACLE is guaranteed to return a distribution over noise vector $\tilde{q}$ with the property that:

$$\min_{i} M_{S'}(i, \tilde{\boldsymbol{q}}) \geq \lambda - \delta - \beta$$

*Proof.* We present the proof in Section A.3. □

## 4.3 Universal Noise in Deep Learning

So far, we've examined the problem of adversarial noise mostly within the context of affine classifiers. However, for many of the problems that we're interested in solving in practice, the underlying data distributions are too complex and the learner requires a richer set of hypothesis in order to achieve reasonable approximations. In this section, we present how some of the approaches to adversarial noise that were presented previously can be extended to the realm of deep learning classifiers.

In order to generalize the algorithmic framework presented in Section 4.1 we need to construct a Distributional-Oracle that can maximize the expected loss of the learner in cases where the hypothesis set $\mathcal{H}$ consists of deep neural networks. Contrary to the case of affine classifiers, the decision boundaries computed by DNNs are highly nonconvex and unsuited to the kinds of convex programming approaches we developed for linear classifiers. Therefore, we instead propose a similar method to the one developed in Section 4.2 in which we rewrite the expected loss of the learner using a different set of loss functions that are more amenable to optimization and move in the opposite direction to the 0-1 loss.[3]

As discussed previously, researchers working to develop adversarial noise algorithms in deep learning have recently shifted their attention to optimization-based techniques. Rather than computing a gradient, these methods attempt to find adversarial examples by minimizing some objective function that captures the accuracy of the network on the altered input. Within this set of approaches, the Carlini $\ell_2$ attack has stood out as being one of the most effective. Several papers have demonstrated how when evaluated on different image classification tasks, it consistently outperforms gradient-based methods at tricking DNNs [10, 27].

The algorithm works by minimizing a weighted combination of the $\ell_2$ distance of the adversarial example and a loss function $\ell$ that captures the prediction error of the network $h$ on the adversarial example.

$$\arg\min_{v} ||v||^2 + c \cdot \ell(h, x + v, y) \tag{4.6}$$

The constant $c$ is a positive real number that is determined empirically during the optimization by a applying a modified version of binary search.[4] It is introduced into the objective in order to balance between minimizing the magnitude of the noise and finding

[3]Like the reverse hinge loss, we want to find a function that is minimized exactly when the 0-1 loss is maximized.

[4]We describe this search algorithm in more detail in Section 5.2

adversarial examples that increase the loss of the network. In practice, the optimization is carried out via the Adam optimizer. In their paper, Carlini and Wagner explore a number of different possibilities for the loss function $\ell$ and compare their performance on a series of experiments [10].[5] Through their work, they determine that the logit loss $\ell_z$ was the most effective at finding adversarial examples that successfully tricked the classifier while only minimally perturbing the original input.

On input $x$, we assume that the neural network $h$ returns a logit vector $Z_h(x) \in \mathbb{R}^k$ where each entry in the vector corresponds to the (unnormalized) likelihood of the each class.[6] The predicted label of the neural network corresponds to the maximum logit value:

$$h(x) = \arg\max_{i \in [k]} Z_h(x)_i \tag{4.7}$$

**Definition 4.3:** The **logit loss** is a function $\ell_z : \mathcal{H} \times \mathbb{R}^d \times [k] \to \mathbb{R}_{\geq 0}$. In the case of untargeted noise, the logit loss is defined as:

$$\ell_z(h, x, y) \stackrel{\text{def}}{=} (Z_h(x)_y - \max_{i \neq y} Z_h(x)_i)^+ \tag{4.8}$$

Given a target label $j \neq y$, we define the logit loss for targeted noise as:

$$\ell_z(h, x, y) \stackrel{\text{def}}{=} (\max_{i \neq y} Z_h(x)_i - Z_h(x)_y)^+ \tag{4.9}$$

It follows from this definition that, for untargeted noise, if $\ell_z(h, x, y) > 0$ then $\ell_{0\text{-}1}(h, x, y) = 0$. Similarly, if $\ell_z(h, x, y) = 0$, then $\ell_{0\text{-}1}(h, x, y) = 1$.

We can now rewrite the expected loss of the learner in the case of deep learning using a similar approach to the one we employed in Section 4.2 for the reverse hinge loss. In the case of a single example $(x, y)$ the loss is:

$$\min_v \sum_{i=1}^{n} w_i \cdot \ell_z(h_i, x + v, y) \tag{4.10}$$

The benefit of reframing the loss in this way is that we can now find adversarial examples using an extension of the Carlini algorithm by applying the Adam optimizer to the expression:

$$\min_v ||v||^2 + c \sum_{i=1}^{n} w_i \cdot \ell_z(h_i, x + v, y) \tag{4.11}$$

[5] A more in depth discussion of the different loss functions can be found in the experiments section of the Carlini Wagner paper [10]

[6] Usually, we apply the softmax function in order to convert the logit vector into a valid probability distribution.

While we again have no guarantees that this extension of the Carlini method will yield optimal solutions, if we assume that it constitutes an $\beta$-Distributional-Oracle then we can use it within the Multiplicative Weights framework in order to find universal noise vectors as per Theorem 4.2. We explore these approaches more in detail in the following chapter.

# Chapter 5

# Experiments

The approaches towards universal noise developed in this thesis are grounded in a well-principled theoretical framework that combines ideas in statistical learning theory, robust optimization, and game theory in order to present a wholistic picture of the universal noise models in machine learning. In addition to having these theoretical properties, the algorithms introduced in this thesis have the added benefit that they can be efficiently implemented in practice to construct universal noise models. In this chapter, we demonstrate the effectiveness of our approaches by applying them to the MNIST dataset.

In Section 5.1, we apply the Multiplicative Weights framework described in Section 4.1 using the Distributional-Oracle for binary and multiclass classifiers to generate noise for linear predictors. In Section 5.2, we examine the behavior of our approaches in the context of deep learning and demonstrate that despite not having guarantees of performance, the approximate Distributional-Oracle described in Section 4.3 is in fact an efficient way of generating noise in nonlinear settings.

## 5.1 Linear Classification

Beginning with the case of binary classification, we train 3 different linear SVM classifiers on a subset of the MNIST dataset. While the MNIST dataset has a total of 10 classes, one for each numerical digit, we subset the data so that we only consider images where the digit is either a 3 or an 8.[1] In order to differentiate amongst hypothesis, we vary the penalty parameter $c$ of the SVM optimization. Furthermore, in order to ensure enough heterogeneity across the different decision boundaries, we train each classifier on a different subset of 1000

[1]The code for all of these experiments is made available on github.

images. Despite the small training size, they achieve fairly good generalization behavior. On a test set of almost 2000 images, the different classifiers achieve accuracies of 94%, 96%, and 97% respectively. To test the quality of our universal noise algorithms, we select a subset $S$ of 300 images from the test set that were correctly classified by all models.

Demonstrating the effectiveness of our approaches to noise relies on carefully selecting the parameter $\alpha$ for the noise budget. If $\alpha$ is too small, then the Distributional-Oracle will fail to find any adversarial examples. In particular, as we proved in Corollary 3.1, if $\alpha$ is smaller than the minimum distance to the decision boundaries of the set of linear predictors, then no perturbation will induce misclassification by part of the adversary. On the other hand, if $\alpha$ is too large, the Distributional-Oracle finds the unique set of $m$ vectors that fools all classifiers regardless of the weight distribution chosen by the learner. In other words, for sufficiently large $\alpha$, there exists a pure strategy equilibrium for the adversary.

The value of $\alpha$ at which this pure strategy equilibrium emerges can be determined by a quadratic program of the form used to test for feasible regions within the Distributional-Oracle algorithm. Without loss of generality, assume that a particular point $x$ has true label $-1$. If the Distributional-Oracle can find a perturbation $v$ such that $x + v$ is in the region defined by the sign vector $(+1, +1, +1)$, then no randomization by part of the learner will diminish his expected loss since $x + v$ constitutes a pure strategy equilibrium for the adversary. Therefore, in order to select the noise budget $\alpha$, we compute the minimum distance from every point $S$ to the decision boundaries of the different classifiers and choose an intermediate value. For our experiments, we select a noise budget of .95 and run Multiplicative Weights for 100 iterations.

Based off the analysis of the algorithm presented in Section A.3, the error parameter $\varepsilon$ and the number of iterations $T$ have the relationship that $T = \lceil \frac{4 \ln n}{\delta^2} \rceil$ where $\delta = \frac{\varepsilon}{2}$. Therefore, by the guarantees of the Multiplicative Weights algorithm, running for 100 iterations guarantees that the solution will be within a factor of .21 from the minimax value. However, from our experiments we see that the algorithm converges to a solution in only a small number of iterations.

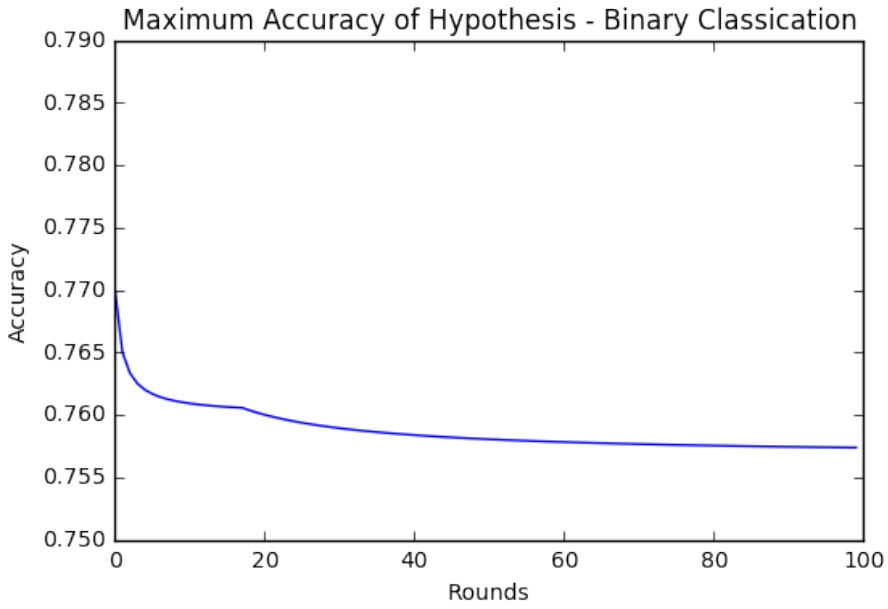



**Figure 5.1:** Maximum accuracy of any binary classifier as function of the noise generated by the algorithm in a given number of rounds.

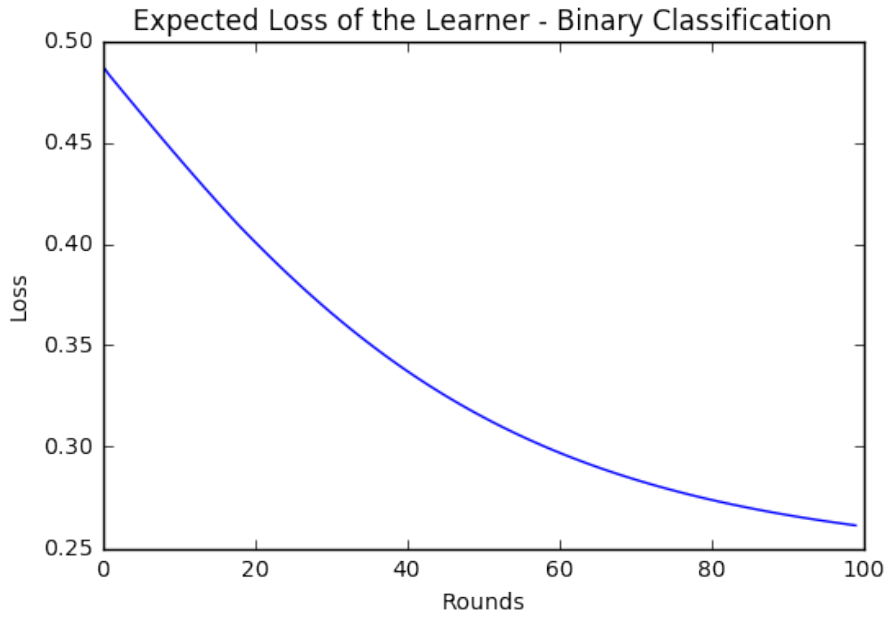



**Figure 5.2:** Expected loss of the learner as a function of the number of rounds.

From Figure 5.1, we see that the quality of the noise generated stops improving after only a small number of rounds. This behavior indicates that while the noise generated is guaranteed to be universal there are actually limited benefits to boosting the quality of

the noise by using the Multiplicative Weights algorithm. Running the DISTRIBUTIONAL-ORACLE once returns a solution that is reasonably close to the minimax value of the game. Conversely, as per Figure 5.2, the loss of the learner continues to decrease as the number of rounds increases. Therefore, from the perspective of the learner, there are significant benefits to using our framework as a way of defending against adversarial noise. By running the algorithm, the learner can find a weight distribution over available hypothesis that drastically reduces his expected loss vis à vis the uniform distribution.

## 5.2 Deep Learning

For the case of deep learning, we train 4 different networks on the full MNIST training set of 55,000 images. The first two networks are convolutional neural networks. In order to distinguish between models, we train the first in the normal manner using stochastic gradient descent. For the second network, we train using defensive distillation. Introduced by Papernot et al, defensive distillation is a way of enhancing the training of neural networks that has been demonstrated to increase their robustness to noise [33]. The second set of networks consists of densely connected feedforward networks with relu activation units in each layer. Again, we train one network with defensive distillation and another one without it.[2] The accuracies of the models on a test set of 10,000 images are 99%, 98%, 99%, and 97%.[3]

In order to generate adversarial noise, we apply the Adam optimizer to the following objective where $w_i$ denotes the current probability weight for hypothesis $h_i$:

$$\arg\min_{v} ||v||^2 + c \sum_{i} w_i \cdot \ell_z(h_i, x + v, y) \tag{5.1}$$

We use a learning rate of .01 and run a binary search on $c$ for 9 steps. In each step we run the Adam optimizer for a maximum of 4000 iterations.[4] We initialize our binary search with $c = .001$. In each step, as we run the optimizer we keep track of the best adversarial example found for that value of $c$. Once we have finished running Adam, we check to see if the example transfers across all models. If it does, then we have found a universal noise

[2]Precise details regarding the architecture of each network can be found in the repository.

[3]The first two accuracies correspond to the convolutional networks, while the second two correspond to the dense networks. Within each pair, the second number indicates the accuracy of the model when trained using defensive distillation.

[4]As per the original implementation by Carlini and Wagner, we employ an early stopping parameter to abort the optimization for unpromising values of $c$. Details of this behavior can again be found in the code.

vector and we decrease the value of $c$. Decreasing $c$ has the effect of giving higher relative weight to the first term in the optimization and therefore yields solutions with a smaller amount of distortion. On the other hand, if we have not yet found a universal perturbation, then we increase $c$ in order to find a solution that further minimizes the weighted sum of logit losses.

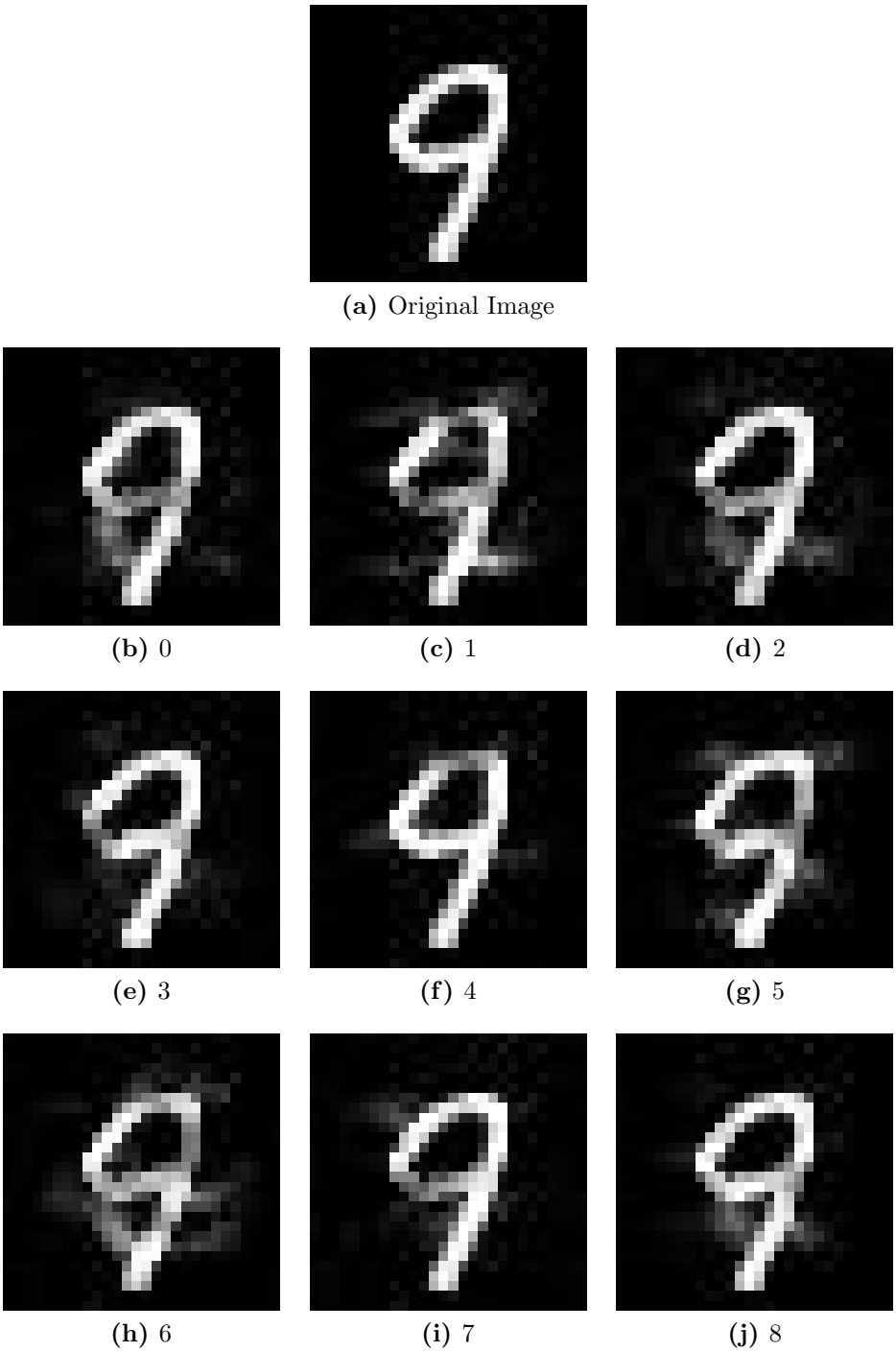
We run two main experiments. For the first, we apply our extension of the DISTRIBUTIONAL-ORACLE for deep learning using the untargeted version of the logit loss to a set of 50 images that are correctly classified by all 4 models. Initially, we implemented this approach in the context of the Multiplicative Weights algorithm. However, the noise vectors found by the algorithm were such that all models were completely fooled by the perturbed inputs.[5] Since each model assumes 0-1 loss equal to 1 on the perturbed inputs, it was not feasible to update the weights since the relative probability distribution remains the same. Each classifier was penalized by the maximum amount. The solution found by the adversary constitutes a pure strategy equilibria and hence there is no benefit in boosting the quality of the noise via the Multiplicative Weights framework.

| $\ell_2$ Norm - Untargeted Noise | |
|---|---|
| Mean | 2.206125107 |
| Max | 3.670451176 |
| Min | 0.527822659 |

For the second experiment we selected 8 images and ran targeted attacks using the targeted version of the logit loss described in Section 4.3. In order to demonstrate the effectiveness of our approach, we attempt to generate noise such that each individual sample is misclassified by the learner for each of the 9 possible labels. For example, if the original image was a 0, we run targeted attacks on the image with target labels 1 through 9. We therefore run a total of 72 targeted attacks. Much like in the untargeted case, the solutions returned by the DISTRIBUTIONAL-ORACLE succeeded on all inputs and on all models. For every digit, our approach was able to generate noise such that every network predicted the desired target label. We illustrate the results of our method in the following figure:

| $\ell_2$ Norm - Targeted Noise | |
|---|---|
| Mean | 1.80 |
| Max | 3.4 |
| Min | 0.69 |

[5]The results of these experiments are available in the git repository.



**(a)** Original Image

**(b)** 0 **(c)** 1 **(d)** 2

**(e)** 3 **(f)** 4 **(g)** 5

**(h)** 6 **(i)** 7 **(j)** 8

**Figure 5.3:** The image at the top is the original unperturbed input. Each subcaption indicates the target label used to generate noise for that image. Every perturbed example was classified with the corresponding target label by every network.

The behavior encapsulated by these results is closely related to our discussion on the relationship between noise budgets and the nature of adversarial noise presented in Section 5.1. If the noise budget is too large, the adversary can find a single perturbation capable of fooling every classifier. In order to verify the benefits of the boosting through Multiplicative Weights, we would need to restrict maximum norm of the noise. However, limiting the magnitude of the noise while ensuring that the weighted logit loss is minimized is a complicated task due to the nature of the binary search on $c$. In particular, it is nontrivial to determine the maximum $c$ such that the norm of the solution $v$ found by the optimizer is less than $\alpha$. Modifying the algorithm to accommodate such a restriction is an interesting direction of future research.[6]

[6]One possibility to consider is whether instead of just using Adam, one could add a projection step to the gradient descent algorithm in order to ensure that the noise perturbation remains within the desired bounds.

# Chapter 6

# Conclusion

In this thesis, we have presented how constructing universal noise models in machine learning can be modeled formally as two-player, zero-sum game between a learner who attempts to classify examples from a distribution and an adversary who limits the learner's progress. By examining the geometry of adversarial noise, we demonstrated how we can construct a DISTRIBUTIONAL-ORACLE that finds single perturbations that maximize the loss of the learner on particular examples. Moreover, we showed how we can apply the Multiplicative Weights robust optimization framework in order to boost the quality of the noise into a distribution that is arbitrarily close to the equilibrium strategy of the adversary in the zero-sum game. The result of this algorithm is a distribution over noise vectors that minimizes the maximum accuracy of any classifier the learner might use for a given data set.

In Chapter 4, we extended our methods to demonstrate how we can generate universal noise even in cases where the number of possible hypothesis functions is very large. To do so, we followed a well-established technique of substituting nonconvex loss functions by convex alternatives in order to make the objective for the oracle more amenable to optimization through stochastic gradient descent. Furthermore, we generalized our approaches developed for affine functions to the domain of deep learning and demonstrated the effectiveness of our methods through a series of experiments on the MNIST dataset.

The results presented in this thesis serve to formalize some of the recent work done within the field of adversarial noise in machine learning. We provide a new set of algorithms grounded in a principled theoretical framework that have strong guarantees of performance. But perhaps more importantly, by providing a theoretical model to reason about universal noise, we open the door for a host of new research directions that could potentially fix some of the current shortcomings of machine learning algorithms.

As presented in this thesis, our algorithms generate noise that is guaranteed to be robust for the set of classifiers that are considered in the optimization. However, we would like to reason about the quality of the noise with regards to classifiers that are outside the examined set. In particular, we would like to demonstrate how the issue of noise can be precisely related to differences in the decision boundaries of different classifiers. We believe that by establishing such a relationship we can then ask the question of what is the optimal decision boundary that a classifier should attempt to compute with regards to particular distribution; and how computing such a boundary relates to the sample complexity of the learning algorithm. Finding the optimal decision boundary would solve the threat of adversarial examples in security sensitive settings and open the door for a host of new applications of machine learning in novel domains. We hope that others will build upon our work to come up with new discoveries in the field.

# Appendix A

# Proofs

## A.1 "All-Pairs" Linear Multiclass Classification

Given $k$ classes to predict, under the "all-pairs" or the "all-vs-all" method, a single hypothesis $h$ is composed of $\binom{k}{2}$ classifiers $h_1 \dots h_{\binom{k}{2}}$. Each classifier $h_{i,j}$ is trained to predict +1 on points that have label $i$ and -1 on points that have label $j$. Notice that $h_{i,j} = -h_{j,i}$. Much like in the "one-vs-all" approach, each individual classifier $h_{i,j}$ is single linear predictor parametrized by a vector $w_{i,j} \in \mathbb{R}^d$ and a bias term $b_{i,j} \in \mathbb{R}$. Furthermore, each predictor $h_{ij}$ is designed to output a scalar value rather than a label in $\{-1, +1\}$.

On a given input $x$, an "all-pairs" classifier predicts a class according to the following expression:

$$h(x) = \arg\max_{i \in [k]} \sum_{j \neq i} h_{i,j}(x) \tag{A.1}$$

Similarly to the "one-vs-all" approach, we are comparing the strengths of the relative predictions of each subclassifier $h_{i,j}$ and somehow combining them to select the label with the highest score. Seeing as how each individual classifier remains linear, it should come as no surprise that the ultimate decision boundary remains linear and that we can leverage the convexity of the hypothesis class to extend some of the results that we had presented earlier on.

**Lemma A.1:** An "all-pairs" multiclass linear predictor over $k$ labels partitions the input space into $k$ convex subsets $T_1, \dots T_k$ where:

1. $T_i \cap T_j = \emptyset \quad \forall i \neq j$.

2. $\mathbb{R}^d \setminus \bigcup_i T_i$ is a set of measure zero.

3. For all $T_i$, if $h(x) = i$ then $x \in T_i$

4. For all $T_i$, if $x_1, x_2 \in T_i$ then $cx_1 + (1-c)x_2 \in T_i \quad \forall c \in [0,1]$

*Proof.* The proof is very similar to Lemma 3.1. We define $T_i$ to be set the of points $x$ for which the following relationship holds:

$$\sum_{j \neq i} \langle w_{i,j}, x \rangle + b_{i,j} > \sum_{j' \neq l} \langle w_{l,j'}, x \rangle + b_{l,j'} \quad \forall l \neq i \tag{A.2}$$

Condition (1) follows directly from this definition since the relevant inequalities for a point $x$ being in sets $T_i$ and $T_l$ can't be mutually satisfied. Similarly to Lemma 3.1 (2) follows from the fact that the set $\mathbb{R}^d \setminus \bigcup_i T_i$ is a subset of the intersection of a series of hyperplanes in $\mathbb{R}^d$ and hence must have measure 0. (3) is determined by the definition of the sets $T_i$ and the mechanism by which the "all-pairs" approach determines the labels of a point. Lastly, (4) follows for them convexity of the decision boundaries of linear predictors.

If $x_1, x_2 \in T_i$ then:

$$\sum_{j \neq i} \langle w_{i,j}, x_1 \rangle + b_{i,j} > \sum_{j' \neq l} \langle w_{l,j'}, x_1 \rangle + b_{l,j'} \quad \forall l \neq i \tag{A.3}$$

$$\sum_{j \neq i} \langle w_{i,j}, x_2 \rangle + b_{i,j} > \sum_{j' \neq l} \langle w_{l,j'}, x_2 \rangle + b_{l,j'} \quad \forall l \neq i \tag{A.4}$$

Let $x' = cx_1 + (1-c)x_2$ for $c \in [0,1]$:

$$\begin{aligned}
\sum_{j \neq i} \langle w_{i,j}, x' \rangle + b_{i,j} &= \sum_{j \neq i} \langle w_{i,j}, cx_1 + (1-c)x_2 \rangle + b_{i,j} \\
&= \sum_{j \neq i} c\langle w_{i,j}, x_1 \rangle + cb_{i,j} + (1-c)\langle w_{i,j}, x_2 \rangle + (1-c)b_{i,j} \\
&> \sum_{j' \neq l} c\langle w_{l,j'}, x_1 \rangle + cb_{l,j'} + (1-c)\langle w_{l,j'}, x_2 \rangle + (1-c)b_{l,j'} \quad \forall l \neq i \\
&= \sum_{j' \neq l} \langle w_{l,j'}, cx_1 + (1-c)x_2 \rangle + b_{l,j'}
\end{aligned}$$

Hence, $x'$ is also in $T_i$.

□

Now we can prove an analogous theorem to 3.2 for the case of "all-pairs" multiclass linear predictors.

**Theorem A.1:** The problem of finding targeted and untargeted adversarial examples for a linear, multiclass predictor using the "all-pairs" approach reduces to minimizing a quadratic function over a convex set.

*Proof.* Again, we follow a similar argument as in Theorem 3.2. We first demonstrate how to find targeted adversarial examples and then use our algorithm for targeted noise to solve for untargeted examples by iterating over the label set.

As per equation 3.10, in order to find targeted adversarial examples for a point $(x, y)$ we attempt to solve the following optimization problem for a particular label $j \in [k]$.

$$\begin{aligned} \min_{v \in \mathbb{R}^d} \quad & ||v|| \\ \text{subject to} \quad & h(x+v) = j \end{aligned} \tag{A.5}$$

Due to the nature of "all-pairs" predictors we can rewrite the constraint in the above equation as:

$$h(x+v) = j \iff \sum_{i \neq j} \langle w_{j,i}, x+v \rangle + b_{j,i} > \sum_{i' \neq l} \langle w_{l,i'}, x+v \rangle + b_{l,i'} \quad \forall l \neq j \tag{A.6}$$

Therefore, the constraint in A.5 is equivalent to a set of $k-1$ linear inequality constraints. As in equation 3.2, we square the objective function knowing that doing so does not alter the underlying optimum (the norm of a vector is nonnegative, therefore squaring the value is a monotonic transformation). We are left with:

$$\begin{aligned} \min_{v \in \mathbb{R}^d} \quad & ||v||^2 \\ \text{subject to} \quad & \sum_{i \neq j} \langle w_{j,i}, x+v \rangle + b_{j,i} > \sum_{i' \neq l} \langle w_{l,i'}, x+v \rangle + b_{l,i'} \quad \forall l \neq j \end{aligned} \tag{A.7}$$

From Lemma A.1, we know that the set described by the constraints in this equation is convex. Therefore, finding targeted examples for linear "all-pairs" classifiers reduces to minimizing a convex function over a convex set. And since we can translate from finding targeted examples to finding untargeted examples, the untargeted case also can be solved by minimizing a convex function over a convex set.

□

## A.2 Distributional Oracle for Multiclass Classification

**Theorem A.2:** For affine, multiclass classifiers over $k$ classes, implementing a DistributionalOracle to solve for the set of $m$ noise vectors that maximizes the expected loss of the learner with respect to a distribution $\boldsymbol{w}$ over classifiers in $\mathcal{H}$, a noise budget $\alpha$, and a data set $S$ reduces to the problem of minimizing a quadratic function over a set of convex polytopes.

*Proof.* The proof is similar to the case of binary classifiers. As per Lemma 3.1, the decision boundary of a single multiclass linear predictor partitions the input space into $k$ convex subsets such that points in each subset are all mapped to the same label by the classifier.

We now define a series of $k^n$ subsets $T_1 \dots T_{k^n}$ where each $T_j$ corresponds to a unique sequence of labels $s_j = (s_{j,1}, \dots, s_{j,n})$ in the set $[k]^n$. The ith entry of vector $s_j$ indicates the label predicted by hypothesis $h_i$ in the set $T_j$. As in the case of binary classification, each $T_k$ is a convex set since the regions on which individual linear predictors output a particular label are convex and intersections of convex sets are convex. The union of these sets cover almost the entire space since points in the complement constitute a set of measure zero. Furthermore, the loss of the learner is constant over all points that lie in a particular region $T_j$ since the value of the loss function is again solely dependent on the 0-1 loss of each hypothesis.

Just as before, the goal of the Distributional-Oracle is to solve for:

$$\max_{v} \frac{1}{m} \sum_{i=1}^{n} \sum_{j=1}^{m} w_i \cdot \ell_{0\text{-}1}(h_i, x_j + v_j, y_j)$$

We can again optimize this expression independently for every $j$ by considering losses of the form:

$$\max_{v_{j'}} \sum_{i=1}^{n} w_i \cdot \ell_{0\text{-}1}(h_i, x_{j'} + v_{j'}, y_{j'})$$

In order to find the $v_{j'}$ that maximizes this expression, we can simply iterate over all the regions $T_j$, find the minimal perturbation $v_{j'}$ such that $x_{j'} + v_{j'}$ is in $T_j$, and compute the associated loss value. By finding feasible points in each subset $T_j$, we can list out all possible values for the loss of the learner and select the vector with $\ell_2$ norm less than $\alpha$ that has the highest associated loss value.

Assume that each region $T_j$ is identified with a label vector $s = (s_1, \ldots, s_n)$ (extra subscripts are removed for clarity), finding a noise vector $v_{j'}$ such that $x_{j'} + v_{j'}$ is in $T_j$ can be done via the following quadratic program:

$$\begin{aligned} \min_{v \in \mathbb{R}^d} \quad & ||v||^2 \\ \text{subject to} \quad & h_1(x+v) = s_1 \\ & \ldots. \\ & h_n(x+v) = s_n \end{aligned} \tag{A.8}$$

If we assume that each $h_i$ computes multiclass predictions via the "one-vs-all" method, then each hypothesis $h_i$ is composed of $k$ linear predictors $h_{i1}, \ldots h_{ik}$. We can write each individual constraint in equation A.8 of the form $h_i(x+v) = s_i,\ s_i \in [k]$ as $k-1$ linear inequalities:[1]

$$h_i(x+v) = s_i$$

$$\iff$$

$$\langle w_{i,s_i}, x+v \rangle + b_{i,s_i} > \langle w_{i,j}, x+v \rangle + b_{i,j} \quad \forall j \neq s_i$$

If we instead assume that multiclass classification is done via the "all-pairs" method then each classifier $h_i$ is composed of $\binom{k}{2}$ classifiers $h_{i,j,l}$ where $j, l \in [k]$. We then rewrite each individual constraint in A.8 as a set of $k-1$ linear inequalities:

$$h_i(x+v) = s_i$$

$$\iff$$

$$\sum_{j \neq s_i} \langle w_{i,s_i,j}, x+v \rangle + b_{i,s_i,j} > \sum_{i' \neq l} \langle w_{i,l,i'}, x+v \rangle + b_{i,l,i'} \quad \forall l \neq s_i$$

Therefore, the result remains the same as in the binary case.

□

[1] In total, the program therefore has $(k-1) \cdot n$ linear inequalities to identify a particular region $T_j$

## A.3 Multiplicative Weight Updates for Universal Noise

**Theorem 4.1:** Given an error parameter $\delta$, after $O(\frac{\ln n}{\delta^2})$ iterations, Algorithm 1 returns distributions $\tilde{\boldsymbol{w}}$, $\tilde{\boldsymbol{q}}$ such that:

$$\min_i M_{S'}(i, \tilde{\boldsymbol{q}}) \geq \lambda - \delta$$
$$\max_v M_{S'}(\tilde{\boldsymbol{w}}, v) \leq \lambda + \delta$$

*Proof.* The following analysis draws heavily upon the work of Freund and Schapire [16], yet the precise analysis follows that of Kale [21].

By guarantees of the Multiplicative Weights algorithm [4, 21], we have that for any distribution $\boldsymbol{w}$ over $[n]$ with losses in $[0, 1]$ the following relationship holds:

$$\sum_{t=1}^{T} M_{S'}(\boldsymbol{w}^{(t)}, v^{(t)}) \leq (1+\varepsilon) \sum_{i=1}^{T} M_{S'}(\boldsymbol{w}, v^{(t)}) + \frac{\ln n}{\varepsilon} \tag{A.9}$$

If we divide by $T$, and note that $M_{S'}(\boldsymbol{w}, v) \leq 1$, and $M_{S'}(\boldsymbol{w}^{(t)}, v^{(t)}) \geq \lambda$ for all $t$, (due to the DISTRIBUTIONAL-ORACLE , the min player can never do better than the minimax value) we now have that for any distribution $\boldsymbol{w}$:

$$\lambda^* \leq \frac{1}{T} \sum_{i=1}^{T} M_{S'}(\boldsymbol{w}^{(t)}, v^{(t)}) \leq \frac{1}{T} \sum_{i=1}^{T} M_{S'}(\boldsymbol{w}, v^{(t)}) + \varepsilon + \frac{\ln n}{\varepsilon T} \tag{A.10}$$

If we let $\boldsymbol{w^*}$ be the optimal strategy for the min player, then $M_{S'}(\boldsymbol{w^*}, v) \leq \lambda^*$ for any $v$. Next, if we set $\epsilon = \frac{\delta}{2}$ and $T = \lceil \frac{4 \ln n}{\delta^2} \rceil$ we arrive at the following:

$$\lambda^* \leq \frac{1}{T} \sum_{i=1}^{T} M_{S'}(\boldsymbol{w}^{(t)}, v^{(t)}) \leq \frac{1}{T} \sum_{i=1}^{T} M_{S'}(\boldsymbol{w^*}, v^{(t)}) + \delta \leq \lambda^* + \delta \tag{A.11}$$

This shows that $\tilde{\boldsymbol{w}}$, the uniform distribution over $\boldsymbol{w}^{(1)}, \ldots, \boldsymbol{w}^{(T)}$ is an approximately optimal solution for the row player. However, if we let:

$$\boldsymbol{w'}, v' = \arg\min_t M_{S'}(\boldsymbol{w}^{(t)}, v^{(t)}) \tag{A.12}$$

Then:

$$M_{S'}(\boldsymbol{w'}, v') \leq \frac{1}{T} \sum_{i=1}^{T} M(\boldsymbol{w}^{(t)}, v^{(t)}) \leq \lambda^* + \delta \tag{A.13}$$

Which means that $\boldsymbol{w'}$ is a single distribution that is approximately optimal.

For the adversary, we know from equation A.11 that the following holds for any strategy $\boldsymbol{w}$ played by the learner:

$$\lambda^* \leq \frac{1}{T}\sum_{i=1}^{T} M(\boldsymbol{w}^{(t)}, v^{(t)}) \leq \frac{1}{T}\sum_{i=1}^{T} M(\boldsymbol{w}, v^{(t)}) + \delta \tag{A.14}$$

If we set $\tilde{\boldsymbol{q}}$ to be the distribution that assigns weight $\frac{\{t:v^{(t)}=v\}}{T}$ to the particular set of noise vectors $v$ then we have that for any distribution $\boldsymbol{w}$:

$$\lambda^* \leq \frac{1}{T}\sum_{i=1}^{T} M(\boldsymbol{w}^{(t)}, v^{(t)}) \leq M(\boldsymbol{w}, \tilde{\boldsymbol{q}}) + \delta \tag{A.15}$$

And $\tilde{\boldsymbol{q}}$ is an approximately optimal strategy for the adversary:

$$\lambda^* - \delta \leq M(\boldsymbol{w}, \tilde{\boldsymbol{q}}) \tag{7}$$

□

**Theorem 4.2:** Running the universal noise algorithm with parameters $\mathcal{H}$, $S$, $\alpha$ with a $\beta$-Distributional-Oracle is guaranteed to return a distribution over noise vector $\tilde{q}$ with the property that:

$$\min_i M_{S'}(i, \tilde{\boldsymbol{q}}) \geq \lambda - \delta - \beta$$

*Proof.* The proof of this result is almost identical to the case of Theorem 4.1. By the guarantees of the Multiplicative Weights algorithm, we have that:

$$\sum_{t=1}^{T} M_{S'}(\boldsymbol{w}^{(t)}, v^{(t)}) \leq (1+\varepsilon)\sum_{i=1}^{T} M_{S'}(\boldsymbol{w}, v^{(t)}) + \frac{\ln n}{\varepsilon} \tag{A.16}$$

Again we divide by $T$ and make use of the fact that $M(\boldsymbol{w}, v) \leq 1$. However, since the oracle is now only $\beta$ approximate, we can only assume that $M_{S'}(\boldsymbol{w}^{(t)}, v^{(t)}) \geq \lambda - \beta$ for all $t$. We then have that:

$$\lambda^* - \beta \leq \frac{1}{T}\sum_{i=1}^{T} M_{S'}(\boldsymbol{w}^{(t)}, v^{(t)}) \leq \frac{1}{T}\sum_{i=1}^{T} M_{S'}(\boldsymbol{w}, v^{(t)}) + \varepsilon + \frac{\ln n}{\varepsilon T} \tag{A.17}$$

Similarly, if we let $\boldsymbol{w}^*$ be the optimal strategy for the min player, then $M_{S'}(\boldsymbol{w}^*, v) \leq \lambda^*$ for any $v$. Next, if we set $\epsilon = \frac{\delta}{2}$ and $T = \lceil \frac{4 \ln n}{\delta^2} \rceil$ we arrive at an analogous inequality to A.11.

$$\lambda^* - \beta \leq \frac{1}{T} \sum_{i=1}^{T} M_{S'}(\boldsymbol{w}^{(t)}, v^{(t)}) \leq \frac{1}{T} \sum_{i=1}^{T} M_{S'}(\boldsymbol{w}^*, v^{(t)}) + \delta \leq \lambda^* + \delta \tag{A.18}$$

Lastly, if we set $\tilde{\boldsymbol{q}}$ to be the distribution that assigns weight $\frac{\{t: v^{(t)} = v\}}{T}$ to the particular set of noise vectors $v$ then we have that for any distribution $\boldsymbol{w}$:

$$\lambda^* - \beta - \delta \leq \frac{1}{T} \sum_{i=1}^{T} M_{S'}(\boldsymbol{w}^{(t)}, v^{(t)}) \leq \frac{1}{T} \sum_{i=1}^{T} M_{S'}(\boldsymbol{w}, \tilde{\boldsymbol{q}}) \tag{A.19}$$

□

## A.4 Reverse Hinge Loss in Multilabel Classification

In the case of multilabel classification, we can extend the reverse hinge loss so that the optimization yields targeted or untargeted noise vectors.

**Definition A.1:** For multilabel classifiers, the **reverse hinge loss** is a function $\ell_r : \mathcal{H} \times \mathbb{R}^d \times [k] \to \mathbb{R}_{\geq 0}$ defined on examples $(x, y)$ and linear predictors $h$.

If $h$ is a "one-vs-all" classifier, given a target label $j$, we define the targeted version of the reverse hinge loss as:

$$\ell_r(h, x, j) \stackrel{\text{def}}{=} \big( \max_{i \neq j} (\langle w_i, x \rangle + b_i) - \langle w_j, x \rangle - b_j \big)^+$$

If $h$ is an "all-pairs" predictor, we define:

$$\ell_r(h, x, j) \stackrel{\text{def}}{=} \big( \max_{i \neq j} (\sum_{l \neq i} \langle w_{i,l}, x \rangle + b_{i,l}) - (\sum_{l \neq j} \langle w_{j,l}, x \rangle + b_{j,l}) \big)^+$$

We see from the definitions that the reverse hinge loss is 0 if and only if the linear predictor $h$ outputs target label $j$. Furthermore, as in the case of binary classification, the reverse hinge loss is convex.[2] Therefore, we can again apply stochastic gradient descent to find an optimal solution. While these functions are only defined for targeted noise, we can always reduce untargeted noise to targeted noise by iterating over all labels in the set $[k]$ and choosing the solution that yields the highest loss amongst those that satisfy the noise constraint.

[2]The maximum of a series of convex functions is convex and we can always add convex functions together to yield new convex functions.We can absorb the negative signs into the vector $w$ so that the differences become sums.

# Bibliography


[1] F. Alizadeh and D. Goldfarb. Second-order cone programming. *Mathematical Programming*, 95:3–51, 2001.

[2] E.D. Andersen, C. Roos, and T. Terlaky. On implementing a primal-dual interior-point method for conic quadratic optimization. *Mathematical Programming*, 95(2):249–277, Feb 2003.

[3] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, Apr 1988.

[4] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[5] Eric Balkanski, Aviad Rubinstein, and Yaron Singer. The limitations of optimization from samples. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1016–1027, 2017.

[6] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pages 1467–1474, USA, 2012. Omnipress.

[7] A. Blum, A. Frieze, R. Kannan, and S. Vempala. A polynomial-time algorithm for learning noisy linear threshold functions. *Algorithmica*, 22(1):35–52, Sep 1998.

[8] Tom Bylander. Learning linear threshold functions in the presence of classification noise. In *Proceedings of the Seventh Annual Conference on Computational Learning Theory*, COLT '94, pages 340–347, New York, NY, USA, 1994. ACM.

[9] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *25th USENIX*

*Security Symposium (USENIX Security 16)*, pages 513–530, Austin, TX, 2016. USENIX Association.

[10] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016.

[11] Robert Chen, Brendan Lucier, Yaron Singer, and Vasilis Syrgkanis. Robust optimization for non-convex objectives. *CoRR*, abs/1707.01047, 2017.

[12] Kenneth L. Clarkson. Las vegas algorithms for linear and integer programming when the dimension is small. *J. ACM*, 42(2):488–499, March 1995.

[13] Edith Cohen. Learning noisy perceptrons by a perceptron in polynomial time. In *Proc. 38th Annu. IEEE Symposium on Foundations of Computer Science*, pages 514–523. IEEE Computer Society Press, Los Alamitos, CA, 1997.

[14] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. *CoRR*, abs/1608.08967, 2016.

[15] Yoav Freund. Boosting a weak learning algorithm by majority. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, COLT '90, pages 202–216, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

[16] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, August 1997.

[17] Drew Fudenberg. *The Theory of Learning in Games*. MIT Press series on economic learning and social evolution ; 2. MIT Press, Cambridge, Mass., 1998.

[18] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014.

[19] Avinatan Hassidim and Yaron Singer. Submodular optimization under noise. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, pages 1069–1122, 2017.

[20] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. Learning with a strong adversary. *CoRR*, abs/1511.03034, 2015.

[21] Satyen Kale. Efficient algorithms using the multiplicative weights update method, January 2007.

[22] Michael Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, November 1998.

[23] Michael Kearns and Ming Li. Learning in the presence of malicious errors. *SIAM J. Comput.*, 22(4):807–837, August 1993.

[24] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[25] Philip D. Laird. *Learning from Good and Bad Data*. Kluwer Academic Publishers, Norwell, MA, USA, 1988.

[26] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, Apr 1988.

[27] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *CoRR*, abs/1611.02770, 2016.

[28] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *CoRR*, abs/1610.08401, 2016.

[29] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, Pascal Frossard, and Stefano Soatto. Analysis of universal adversarial perturbations. *CoRR*, abs/1705.09554, 2017.

[30] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *CoRR*, abs/1511.04599, 2015.

[31] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. *CoRR*, abs/1602.02697, 2016.

[32] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. *CoRR*, abs/1511.07528, 2015.

[33] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *CoRR*, abs/1511.04508, 2015.

[34] Julia Robinson. An iterative method of solving a game. *Annals of Mathematics*, 54(2):296–301, September 1951.

[35] Benjamin I.P. Rubinstein, Blaine Nelson, Ling Huang, Anthony D. Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and J. D. Tygar. Antidote: Understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, IMC '09, pages 1–14, New York, NY, USA, 2009. ACM.

[36] Robert E. Schapire. The strength of weak learnability. *Mach. Learn.*, 5(2):197–227, July 1990.

[37] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, New York, NY, USA, 2014.

[38] Yaron Singer and Jan Vondrak. Information-theoretic lower bounds for convex optimization with erroneous oracles. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3204–3212. Curran Associates, Inc., 2015.

[39] Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright. *Optimization for Machine Learning*. The MIT Press, 2011.

[40] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[41] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick D. McDaniel. Ensemble adversarial training: Attacks and defenses. *CoRR*, abs/1705.07204, 2017.

[42] Florian Tramr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv*, 2017.

[43] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, November 1984.

[44] J. Von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1947.