1.5em

ROBUST LEARNING OF HALFSPACES IN THE PRESENCE OF AGNOSTIC NOISE - STUDY, IMPLEMENTATION, DESIGN AND ANALYSIS OF ALGORITHMS

by

Sesha Phani Deepika Vadlamudi

A THESIS

Submitted to the Faculty of the Stevens Institute of Technology
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE - COMPUTER SCIENCE

_____

Sesha Phani Deepika Vadlamudi, Candidate

ADVISORY COMMITTEE

_____

Eduardo Bonelli, Chairman Date

_____

Hui Wang, Reader Date

STEVENS INSTITUTE OF TECHNOLOGY
Castle Point on Hudson
Hoboken, NJ 07030
2021

ROBUST LEARNING OF HALFSPACES IN THE PRESENCE OF AGNOSTIC NOISE - STUDY, IMPLEMENTATION, DESIGN AND ANALYSIS OF ALGORITHMS

ABSTRACT

Adversarial Robustness is one of the heavily studied topics in Machine Learning these days. Through the course of this work, we study the problem of learning a robust halfspace, the perceptron algorithm and its variants and the problem of learning sparse halfspaces. We have also implemented the algorithms – 'Basic-1 Agnostic Proper Learning Algorithm' [DKM19], 'Near-Optimal (1+δ)-Agnostic Proper Learning Algorithm' [DKM19], and 'Robust and Agnostic Proper Learning Algorithm for lp-margin Halfspaces' [IDM20].While implementing these algorithms we have tuned down the values of parameters suggested in these algorithms. This is because, with the suggested values of the parameters, these algorithms are not practical. So, we proposed modifications to the algorithms given in [DKM19]. This modification brings down the complexity from exponential to polynomial. The performance of the modified algorithms is evaluated and found that they give a decent accuracy. Note that the comparative analysis is carried out by tuning down substantially the suggested values of the parameters. The accuracy of the algorithms given in [DKM19] and [IDM20] may be substantially high if we run the experiments with the suggested values; but, as mentioned, it is not practical.

Author: Sesha Phani Deepika Vadlamudi

Advisor: Eduardo Bonelli

Date: December 13, 2021

Department: Computer Science

Degree: Master of Science - Computer Science

To my amma, naana, and chelli

Acknowledgments

I would like to extend my deepest gratitude to my advisor Dr. Jie Shen for introducing me to this interesting problem of Robust Learning of Halfspaces and for his teachings; based on which I could take up this study. I am extremely grateful to my another advisor Dr. Eduardo Bonelli for his kind consideration and continuous support throughout. I would like to thank Dr. Wendy Hui Wang for accepting to be my committee member.

I would also like to thank (Late) Prof. Charles Suffel, Prof. Philippos Mordohai, and Prof. Shusen Wang. I am forever grateful for their teachings. I would like to express my gratitude to Prof. Negar Tavassolian for her support, letting me assist in her course and giving me a great opportunity to learn. You inspire me. I would also like to express my sincerest thanks to Prof. Erisa Terroli for an amazing opportunity in assisting her course and for her support throughout.

Nobody has been more important to me during this journey than my family. I dedicate this thesis to my father Prof. V. Ch. Venkaiah for being my backbone, my mother Anuradha for helping me stay calm and focused through difficult times, and my sister Harshita for motivating me to push through my limits. You all inspire me to be better every day and are my ultimate role models.I would like to extend my gratitude to my cousins for their love, support, and making this journey away from home comfortable and easier. Lastly, I want to thank my friends for their unconditional love and being my 24x7 hotline.

Table of Contents

Chapter 1
Introduction

The characteristic of Machine Learning that contributes to it's popularity in modern day era is that it allows applications to use historical data to predict outputs of newly observed instances and become more and more accurate over time without being explicitly programmed to do so. Machine Learning is the central part of operations of major companies, FAANG for instance, for rendering powerful and informative insights in customer behavioral trends, operational patterns, and many more that affect important business decisions.

Machine learning is broadly classified as supervised and unsupervised. In supervised learning, algorithms are trained on labeled data; whereas in unsupervised learning, they are trained on unlabeled data. The type of learning one chooses depends on the application and on the data.

Supervised learning algorithms are generally used for (i) Binary classification, (ii) Multi-class classification, and (iii) Regression models. Binary classification algorithms classifies the given data into two classes using a classification rule such as linear and polynomial threshold functions. As the name suggests, the data is classified into more than two classes in multi-class classification. Regression models allow one to predict a continuous outcome.

Linear threshold functions are also known as halfspaces. A fundamental problem in the field of Machine Learning is the problem of learning unknown Halfspaces. A halfspace belonging to the hypothesis class $\mathcal{H}$, $h_w \in \mathcal{H}$, is a boolean function $h_w : \mathbb{R}^d \to \{+1, -1\}$ such that $h_w(x) = sign(\langle w, x \rangle)$ where $sign(z) = 1$ if $z \geq 0$ and $sign(z) = -1$ otherwise, and $w \in \mathbb{R}^d$ is the associated weight vector.

Before proceeding further, we introduce the terminology relevant to this study. Let $\mathcal{C}$ be a concept class on an instant space $\mathcal{X} \subseteq \mathbb{R}^d$ and let $\mathcal{Y} = \{\pm 1\}$ be the space of labels. Proper learning is one in which the concept and the hypothesis classes are one and the same. That is, $\mathcal{C} = \mathcal{H}$ in proper learning; whereas in improper learning $\mathcal{C} \subsetneq \mathcal{H}$. In this study, we confine ourselves only to proper learning. Error or risk of a hypothesis $h_w \in \mathcal{H}$ with respect to a distribution $\mathcal{D}$ is denoted by $err(h_w, \mathcal{D})$ and it is defined as $err(h_w, \mathcal{D}) = Pr_{(x,y)\in\mathcal{D}}(h_w(x) \neq y)$. Best hypothesis is defined as $h^* = \arg\min_{h_w \in \mathcal{H}} err(h_w, \mathcal{D})$.

Learning in the absence of noise is referred to as realizable setting; otherwise it is considered as non-realizable or agnostic setting. The former scenario is when the labels of the instance do not have any noise and when there exists a hyperplane which perfectly distinguishes the data in the hypothesis space $\mathcal{H}$. In this case the problem of learning can be composed as a linear program and solved in polynomial time [ABHU15].

Learning a halfspace refers to determining a hypothesis $h_w \in \mathcal{H}$ that has near optimal error. That is, find $h_w \in \mathcal{H}$ such that $err(h_w, \mathcal{D}) \leq err(h^*, \mathcal{D}) + \epsilon$ for some $\epsilon \in [0, \frac{1}{2})$, where at times ϵ is called excess error. The is called as exact learning. [Kha18]. In realizable setting, this corresponds to determining a hypothesis $h_w \in \mathcal{H}$ such that $err(h_w, \mathcal{D}) \leq \epsilon$. This is because, in realizable setting, $err(h^*, \mathcal{D}) = 0$. There is a relaxed version of exact learning known as approximate learning and it is formulated as find $h_w \in \mathcal{H}$ such that $err(h_w, \mathcal{D}) \leq \alpha \cdot err(h^*, \mathcal{D}) + \epsilon$, where $\alpha > 1$ is known as the approximation ratio and $\epsilon \in [0, \frac{1}{2})$. When $\alpha = 1$, this is same as the exact learning.

In Probably approximately correct (PAC) learning, the goal of a learner is to output a hypothesis $h_w \in \mathcal{H}$ such that with probability atleast $1-\eta$, where $\eta \in [0, \frac{1}{2})$, $err(h_w, \mathcal{D}) \leq err(h^*, \mathcal{D}) + \epsilon$, for some $\epsilon \in [0, \frac{1}{2})$. This is because $\arg\min_{h_w \in \mathcal{C}}(h_w, \mathcal{D}) =$

$\arg\min_{h_w \in \mathcal{H}}(h_w, \mathcal{D})$ in proper learning.

An adversarial example is an instance of input data with small perturbations that cause a machine learning model to misclassify it. So, machine learning models are vulnerable to adversarial attacks. An example of such an attack is that a self driving car misclassifies a stop sign and crashes into another car. Noise is broadly classified into two categories: (i) label noise and (ii) feature noise. Label noise is caused by misclassifying an instance; whereas feature noise is caused due to the errors or perturbations in the attributes of an instance. *Random classification noise, Massart noise, Tsybakov noise,* and *agnostic or adversarial noise* are some of the examples of the label noise. Similarly, *malicious noise* and *nasty noise* are examples of the feature or attribute noise. This study is concerned with the ℓ_p perturbations of the attributes and agnostic or adversarial noise of the labels.

As part of this project work, following tasks are carried out:

1. Some algorithms starting with the perceptron algorithm by Rosenblatt are studied. All these are proper learning algorithms and address the problem of learning a halfspace either in the realizable or non-realizable setting

2. Couple of these most recent PAC learning algorithms are implemented

3. Proposed few algorithms by modifying the algorithms given in [DKM19]

4. Implemented the Fast Gradient Sign Method (FGSM) and generated ℓ_1 attribute perturbed examples.

5. All the implemented algorithms are experimented on the generated datasets along with the agnostic label noise

6. Results are analyzed and the algorithms are compared

The report is organized as follows: Perceptron algorithm and its variants are described in Chapter 2. Algorithms for learning sparse halfspaces are described in Chapter 3. Chapter 4 explains the theory behind the generation of the attribute perturbed examples and the concerned FGSM algorithm. Chapter 5 briefly describes various types of label noise. Chapter 6 talks about the generation of a dataset. Chapter 7 describes the algorithms chosen to be implemented. Section 8 discusses the proposed algorithms and the theory behind these algorithms. Experimental results and comparative analysis of the algorithms are discussed in Chapter 9. Chapter 10 concludes and Chapter 11 suggests some directions for future work.

Chapter 2

Perceptron algorithm - Variants

Perceptron algorithm, one of the early online algorithms for learning linear threshold functions invented by Frank Rosenblatt in 1957, is often used in realizable scenario because of the provable guarantee that given a set of data points D in $\mathbb{R}^d$, each instance labelled as either positive or negative, the algorithm will find a w such that $(w \cdot x) > 0 \ \forall$ positive points x and $(w \cdot x) < 0 \ \forall$ negative instances, if such a vector exists. If the dataset D is not linearly separable then the perceptron algorithm will not terminate.

2.1 Standard Perceptron Algorithm

The standard perceptron algorithm has the following steps:

1. Initialize the weight vector $w = w_0$. (Weights maybe initialized to either some random value but more often than not they are initialized to 0)

2. For each instance x_j in our dataset D, predict the instance as positive, $f(x_j) = 1$ if $w_i \cdot x_j > 0$ and as negative, $f(x_j) = -1$, otherwise.

3. If a mistake has been made in the classification, then

   - $w_{i+1} = w_i + \widehat{x_j}$, if the mistake is on a positive instance
   - $w_{i+1} = w_i - \widehat{x_j}$, if the mistake is on a negative instance

   where $\widehat{x_j} = \frac{x_j}{\|x_j\|_2}$ is the vector x_j normalized to have unit ℓ_2 norm.

4. Repeat until all the instances are perfectly classified

Proof that this algorithm converges in at most $\frac{1}{\sigma^2}$ iterations, where $\sigma = \min_{x \in \mathcal{D}} |w^c \cdot \widehat{x}|$, $\widehat{x} = \frac{x}{\|x\|_2}$, and w^c is a unit vector that correctly classifies our dataset D, is as follows [MP69, ABV98]:

Consider

$$\begin{aligned}(w_{i+1} \cdot w^c) &= (w_i + f(x_j)\widehat{x_j} \cdot w^c) \\ &= (w_i \cdot w^c) + f(x_j)(\widehat{x_j} \cdot w^c) \\ &\geq (w_i \cdot w^c) + \sigma\end{aligned}$$

That is in each update of the algorithm, the quantity $(w_{i+1} \cdot w^c)$ increases by at least σ. So, after t updates, w_0 becomes w_t and $(w_t \cdot w^c)$ increases by at least $t\sigma$.

Also, consider

$$\begin{aligned}\| w_{i+1} \|_2^2 &= (w_{i+1} \cdot w_{i+1}) \\ &= (w_i + f(x_j)\widehat{x_j} \cdot w_i + f(x_j)\widehat{x_j}) \\ &= (w_i \cdot w_i) + 2f(x_j)(w_i \cdot \widehat{x_j}) + 1 \\ &= \| w_i \|_2^2 + 2f(x_j)(w_i \cdot \widehat{x_j}) + 1 \\ &< \| w_i \|_2^2 + 1, \quad \because x_j \text{is a misclassified instance, the cross term is negative}\end{aligned}$$

That is, in each update of the algorithm, the square of the norm of the w_i increases by at most 1. So, after t updates w_0 becomes w_t and the value of $\| w_t \|_2^2$ is more than that of w_0 by at most t.

Now consider the cosine of the angle between w_t and w^c. That is, consider

$$\begin{aligned} \cos(w_t, w^c) &= \frac{(w_t \cdot w^c)}{\| w_t \|_2} \\ &\geq \frac{t\sigma}{\sqrt{t}} \\ \Rightarrow \qquad t &\leq \frac{1}{\sigma^2} \qquad \because \cos\theta \leq 1 \text{ for any } \theta \end{aligned}$$

That is in at most $\frac{1}{\sigma^2}$ updates the algorithm converges to the correct weight vector.

2.2 Modified Perceptron Algorithm - by Blum et al.

Blum et al. [ABV98] modified the standard perceptron algorithm to arrive at the following. Note that the resulting algorithm updates the weight vector by taking the distance of the misclassified instance to the current hypothesis plane. It is mentioned that the algorithm is resistant to the *random classification noise*. The modified algorithm takes σ as input and tries to produce a vector w such that every misclassified $x \in D$ satisfies $|\cos(w, x)| \leq \sigma$.

1. Begin with a random unit vector w
2. If every misclassified $x \in D$ is such $|cos(w, x)| \leq \sigma$ (i.e., if $|w \cdot \widehat{x}| \leq \sigma|w|$) then halt.
3. Otherwise, pick the misclassified $x \in D$ such that $|cos(w, x)|$ is maximum and update w as in the following:

$$w \leftarrow w - (w \cdot \widehat{x})\widehat{x}$$

4. If fewer than $(1/\sigma^2)ln(d)$ updates have been made, repeat from step 2, else repeat from step 1

It is shown in [ABV98] that if the dataset D is linearly separable, then with probability $1-\eta$ the modified perceptron algorithm halts after $\bigcirc((\frac{1}{\sigma^2})\ln(d)\ln(\frac{1}{\eta}))$ iterations, and produces a vector w such that every misclassified $x \in D$ satisfies $|\cos(w,x)| \leq \sigma$.

2.3 Another Modified Perceptron Algorithm - by Sanjoy et al.

Sanjoy et al [SD09] modified the update rule of the perceptron algorithm and arrived at the following algorithm. The algorithm assumes realizable setting and the data, x_t, is drawn uniformly from the unit sphere $\mathcal{S}^{d-1}$ in $\mathbb{R}^d$. That is, x_t is drawn uniformly from the set $\mathcal{S}^{d-1} = \{x \in \mathbb{R}^d : \quad \| x \|_2 = 1\}$.

Algorithm 1 Another Modified Perceptron Algorithm

1: *Input the dimensionality d and the number of updates M*
2: Let $w_1 = \widehat{x_1} y_1$
3: **for** $i = 2, 3, \ldots, M+1$ **do**
4: Let (x_i, y_i) be a misclassified example, i.e., $sign(w_{i-1} \cdot \widehat{x_i}) \neq y_i$
5: $w_i = w_{i-1} - 2(w_{i-1} \cdot \widehat{x_i})\widehat{x_i}$

To briefly analyze this algorithm, let us consider

$$\begin{aligned}(w_i \cdot w^c) &= ((w_{i-1} - 2(w_{i-1} \cdot \widehat{x_i}))\widehat{x_i} \cdot w^c) \\ &= (w_{i-1} \cdot w^c) - 2(w_{i-1} \cdot \widehat{x_i})(\widehat{x_i} \cdot w^c) \\ &= (w_{i-1} \cdot w^c) + 2|(w_{i-1} \cdot \widehat{x_i})||(\widehat{x_i} \cdot w^c)| \quad \because x_i \text{ is a misclassified example}\end{aligned}$$

where w^c is a unit vector that correctly classifies our dataset D. That is the dot product of the computed weight vector w_i and the target weight vector w^c is increasing. That is the angle between these two weight vectors is decreasing. Hence the error, after each update, is decreasing.

2.3.1 Active Modified Perceptron Algorithm - by Sanjoy et al.

By suitably combining the above modified algorithm with a filtering rule for deciding which points to query Sanjoy et al. in [SD09] propose an active learning algorithm. The filtering rule uses a threshold s_i that changes adaptively starting at $s_1 = \frac{1}{\sqrt{d}}$. The labels are requested only for those examples x such that $|(w_i \cdot x)| \leq s_i$. The resulting algorithm is as follows:

Algorithm 2 Active Modified Perceptron Algorithm

Input dimensionality d, maximum number of labels L, and patience R
Set $w_1 = \widehat{x_1} y_1$
Set $s_1 = \frac{1}{\sqrt{d}}$
for $i = 1, 2, \ldots L$ **do**
    wait for the next example x such that $|(\widehat{x} \cdot w_i)| \leq s_i$ and query its label
    Let this labeled example be (x_i, y_i)
    if $(\widehat{x_i} \cdot w_i) y_i < 0$ **then**
        $w_{i+1} = w_i - 2(w_i \cdot \widehat{x_i})\widehat{x_i}$
        $s_{i+1} = s_i$
    else
        $w_{i+1} = w_i$
        If predictions were correct on R consecutive labeled examples
        (i.e., $(\widehat{x_i} \cdot w_i) y_i \geq 0 \; \forall \in \{i - R + 1, i - R + 2, \cdots, i\}$),
        then set $s_{i+1} = \frac{s_i}{2}$
        else $s_{i+1} = s_i$

2.4 Another Modified Perceptron Algorithm - by Songbai et al.

Songbai Yan and Chicheng Zhang [YZ17] modified the update rule of the perceptron algorithm and arrived at an active algorithm that learns homogeneous halfspaces under the uniform distribution over the unit sphere in the non-realizable setting. This is a Probably Approximate Correct (PAC) learning algorithm that works under the bounded and adversarial noise settings. Notation used in this algorithm is as follows: w^c is the underlying concept to be learned, w is the final output of the algorithm, $\mathcal{O}$ is a labeling oracle, failure probability is δ, $\{m_k\}$ is the sample schedule, m is the number of examples in a sample, $\{b_k\}$ or b is the bandwidth, $\mathcal{S}^{d-1} = \{x \in \mathbb{R}^d : \ \| x \|_2 = 1\}$ is the unit sphere in $\mathbb{R}^d$, and ϵ is the target error. That is, the final halfspace w produced by the algorithm should satisfy $\Pr[sign(w^c \cdot x) \neq sign(w \cdot x)] \leq \epsilon$. In other words, the halfspace w should satisfy $err(h_w, \mathcal{D}) \leq err(h_{w^c}, \mathcal{D}) + \epsilon$, where the distribution $\mathcal{D}$ is uniform on the unit sphere $\mathcal{S}^{d-1}$ in $\mathbb{R}^d$. Sample schedule gives the number of examples in each of the samples to be considered. The algorithm consists of two parts and they are as follows:

Algorithm 3 Active Modified Perceptron Algorithm by Songbai et al. - Main Part

1: **Input**: *Labeling oracle* $\mathcal{O}$*, initial halfspace* w_0*, target error* ϵ*, failure probability* δ*, sample schedule* $\{m_k\}$*, band width* $\{b_k\}$

2: **Output**: *learned halfspace* w

3: Let $k_0 = \lceil \log_2 \frac{1}{\epsilon} \rceil$

4: **for** $k = 1, 2, \cdots, k_0$ **do**

5: $\quad w_k \leftarrow$ MODIFIED-PERCEPTRON$(\mathcal{O}, w_{k-1}, \frac{\pi}{2^k}, \frac{\delta}{k(k+1)}, m_k, b_k)$

6: **end for**

7: **Return** $w = w_{k_0}$

Algorithm 4 MODIFIED-PERCEPTRON

1: **Input**: *Labeling oracle $\mathcal{O}$, halfspace w_{k-1}, angle upper bound θ, failure probability δ, number of iterations m, band width b*

2: **Output**: *Improved halfspace w_m*

3: **for** $t = 0, 1, \cdots, m-1$ **do**

- Define region $R_t = \{x \in \mathcal{S}^{d-1} : \frac{b}{2} \leq (w_t \cdot x) \leq b\}$
- Draw x_t from $\mathcal{D}_X$, the marginal distribution of $\mathcal{D}$ over $\mathcal{X}$, until x_t is in R_t. Query $\mathcal{O}$ for its label y_t
- **if** x_t is mis-classified **then**
- $w_{t+1} \leftarrow w_t - 2(w_t \cdot x_t)x_t$

4: **end for**

5: **Return** w_m

The algorithm works in epochs. It can be used under the bounded as well as the adversarial noise models by appropriately setting the sample schedule m_k and b_k parameters. At the beginning of each epoch k, an upper bound on the angle between the underlying halfspace w^c and the current iterate w_{k-1} is assumed. MODIFIED PERCEPTRON is called to find a new iterate w_k, which will have an angle with w^c at most $\frac{\pi}{2^{k+1}}$ with high probability.

Chapter 3

Learning Algorithms for Sparse Halfspaces

Two Probably Approximately Correct (PAC) algorithms for actively learning sparse halfspaces given in [Zha18, ZSA20] are briefly discussed in this section. The algorithms assume that the unlabeled distribution, i.e., $\mathcal{D}_{\mathcal{X}}$, is isotropic log-concave. These algorithms differ from the previously discussed algorithms in that the update step is based on the solution of an optimization problem. One of these is meant to work in a $t-$sparse realizable setting [Zha18] and the other is designed to work in the presence of bounded noise [ZSA20]. A distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ is said to satisfy the $t-$sparse realizable condition, for $t \in \{1, 2, \cdots, d\}$, if there is a $t-$sparse unit vector w^c such that $\Pr(sign(w^c \cdot x) \neq y) = 0$. Or, $err(h_{w^c}, \mathcal{D}) = 0$

3.1 Algorithm for Active Learning of Sparse Halfspaces

The algorithm proceeds in epochs. In each epoch k the algorithm draws a sample S_k consisting of a set of examples from the distribution $\mathcal{D}_{\mathcal{X}|B_k}$, where $\mathcal{D}_{\mathcal{X}|B_k}$ is a marginal distribution $\mathcal{D}$ over $\mathcal{X}$ confined to the sampling region B_k, queries their labels, and updates its iterate w_k. In the algorithm, t denotes the sparsity parameter, ϵ is the target error, δ is the failure probability, P_t is a function that performs hard thresholding step and results in a sparse vector consisting of atmost t non-zero elements, and $\mathcal{L}_\tau(x, y, h_w(x))$ is the τ-hinge loss defined as $\max\{0, (1 - \frac{y(w \cdot x)}{\tau})\}$.

Algorithm 5 Algorithm for Active Learning of Sparse Halfspaces

1: **Input**: *sparsity parameter* t, *target error* ϵ, *failure probability* δ

2: **Output**: *learned halfspace* $\widehat{w}$

3: *Initialization*: $k_0 \leftarrow \lceil \log_2 \frac{1}{C_1 \epsilon} \rceil$, where C_1 is an appropriate constant

4: **for** $k = 0, 1, \cdots, k_0$ **do**

- $S_k \leftarrow$ sample $n_k = c_1 t(\ln d + \ln \frac{1}{\epsilon} + \ln \frac{1}{\delta_k})^3$ examples from $\mathcal{D}_{\mathcal{X}|B_k}$ and query their labels, where

$$B_k = \begin{cases} \mathbb{R}^d, & k = 0, \\ \{x : |(w_{k-1} \cdot x)| \leq b_k\}, & k \geq 1, \end{cases}$$

$$\delta_k = \frac{\delta}{(k+1)(k+2)} \text{ and } b_k = c_2 \cdot 2^{-k}$$

c_1 and c_2 are some other appropriate constants

- Solve the following optimization problem

$$w_k^{'} \leftarrow \text{argmin}_{w \in W_k} \sum_{(x,y) \in S_k} \mathcal{L}_{\tau_k}(x, y, h_w(x))$$

where

$$W_k = \begin{cases} w \in \mathbb{R}^d : \| w \|_2 \leq 1 \text{ and } \| w \|_1 \leq \sqrt{t}, & k = 0, \\ w \in \mathbb{R}^d : \| w - w_{k-1} \|_2 \leq r_k \text{ and } \| w - w_{k-1} \|_1 \leq \rho_k, & k \geq 1 \end{cases}$$

$$r_k = 2^{-k-3}, \rho_k = \sqrt{2t} \cdot 2^{-k-3}, \text{ and } \tau_k = c_3 \cdot 2^{-k}$$

c_3 is an appropriate constant

- Let $w_k \leftarrow \frac{P_t(w_k^{'})}{\|P_t(w_k)^{'}\|_2}$

5: **end for**

6: **Return** w_{k_0}

3.2 Algorithm for Active Learning of Sparse Halfspaces under Bounded Noise

The algorithm uses regret minimization technique and a variant of the perceptron update. The algorithm consists of two stages: an initialization stage *INITIALIZE* and an iterative refinement stage. In the initialization stage, the algorithm finds a vector $\widehat{w}_0$ such that $\theta(\widehat{w}_0, w^c) \leq \frac{\pi}{32}$, where $\theta(\widehat{w}_0, w^c)$ is the angle between $\widehat{w}_0$ and w^c, and w^c is a s-sparse vector such that the data distribution $\mathcal{D}$ satisfies $\Pr((w^c \cdot x) \neq y|x) \leq \eta$ for every $x \in \mathcal{X}$, $\eta \in [0, \frac{1}{2})$. In the iterative refinement stage, it outputs a vector w_k after the k^{th} iteration in such a way that $\theta(\widehat{w}_k, w^c) \leq \frac{\pi}{32 \cdot 2^k}$ with high probability. Let $B_{\widehat{w},b} = \{x \in \mathbb{R}^d : |(\widehat{w} \cdot x)| \leq b\}$, $D_f(w, w') = f(w) - f(w') - (\nabla f(w') \cdot w - w')$, and $\Phi_v(w) = \frac{1}{2(p-1)} \parallel w - v \parallel_p^2$, $p = \frac{\ln(8d)}{\ln(8d)-1}$. It is mentioned that $\nabla\Phi_v$ is a one-to-one mapping from $\mathbb{R}^d$ to $\mathbb{R}^d$ and hence has an inverse, denoted as $\nabla\Phi_v^{-1}$.

Algorithm 6 Active Learning of Halfspaces under Bounded Noise - Main Algorithm

1: **Input**: *target error* ϵ*, failure probability* δ*, bounded noise level* η*, sparsity* s

2: **Output**: *Halfspace* $\widehat{w}$ *in* $\mathbb{R}^d$ such that $err(h_{\widehat{w}}, \mathcal{D}) - err(h_{w^c}, \mathcal{D}) \leq \epsilon$

3: Let $k_0 = \lceil \log \frac{1}{C_1 \epsilon} \rceil$ be the total number of iterations, where C_1 is an appropriate constant

4: Let $\widehat{w}_0 \leftarrow INITIALIZE(\frac{\delta}{2}, \eta, s)$

5: **for** $k = 1, 2, \cdots k_0$ **do**

- $w_{k-1} \leftarrow P_s(\frac{(w_{k-1})}{\|(w_{k-1})\|_2})$
- $\widehat{w}_k \leftarrow REFINE(w_{k-1}, \frac{\delta}{2k(k+1)}, \eta, s, \alpha_k, b_k, \mathcal{K}_k, R_k, T_k)$,

  where

  the step size $\alpha_k = \tilde{\Theta}((1-2\eta)2^{-k})$,

  bandwidth $b_k = \Theta((1-2\eta)2^{-k})$,

  constraint set $\mathcal{K}_k = \{w \in \mathbb{R}^d : \parallel w - w_{k-1} \parallel_2 \leq \pi \cdot 2^{-k-3}, \parallel w \parallel_2 \leq 1\}$,

regularizer $R_k(w) = \Phi_{w_{k-1}}(w) = \frac{1}{2(p-1)} \parallel w - w_{k-1} \parallel_p^2$, $p = \frac{\ln(8d)}{\ln(8d)-1}$

number of iterations $T_k = \bigcirc(\frac{\delta}{(1-2\eta)^2}(\ln \frac{d \cdot k^2 2^k}{\delta(1-2\eta)})^3)$

6: **Return** $\widehat{w}_{k_0} = \frac{w_{k_0}}{\|w_{k_0}\|_2}$

Algorithm 7 Algorithm REFINE

1: **Input**: *initial halfspace* w_1*, failure probability* δ*, bounded noise level* η*, sparsity* s*, learning rate* α*, bandwidth* b*, convex constraint set* $\mathcal{K}$*, regularization function* $R(w)$*, number of iterations* T

2: **Output**: *Refined halfspace* $\widehat{w}$

3: **for** $t = 1, 2, \cdots, T$ **do**

- Sample x_t from $\mathcal{D}_{X|\widehat{w},b}$, the conditional distribution of $\mathcal{D}_X$ on $B_{\widehat{w},b}$, and query $\mathcal{O}$ for its label y_t
- $w_{t+1} \leftarrow \arg\min_{w \in \mathcal{K}} D_R(w, \nabla R^{-1}(\nabla R(w_t) - \alpha g_t))$,

  where

  the gradient $g_t = (-\frac{1}{2}y_t + (\frac{1}{2} - \eta)\widehat{y}_t)x_t$, and $\widehat{y}_t = sign(w_t, x_t)$

4: $\breve{w} \leftarrow \frac{1}{T}\sum_{t=1}^{T} \frac{w_t}{\|w_t\|_2}$

5: **Return** $\widehat{w} \leftarrow \frac{\breve{w}}{\|\breve{w}\|_2}$

Algorithm 8 Algorithm INITIALIZE

1: **Input**: *failure probability* δ*, bounded noise parameter* η*, sparsity parameter* s

2: **Output**: *Halfspace* $\widehat{w}_0$ *such that the angle between* $\widehat{w}_0$ *and* w^c *is less than or equal to* $\frac{\pi}{32}$

3: Draw m *i.i.d* examples $(x_1, y_1), (x_2, y_2), \cdots (x_m, y_m)$ from $\mathcal{D}_X$ and query oracle $\mathcal{O}$ for their labels, where $m = 81 \cdot 2^{51} \cdot \frac{s \ln \frac{8d}{\delta}}{(1-2\eta)^2}$

4: Compute $w_{avg} = \frac{1}{m}\sum_{i=1}^{m} x_i y_i$

5: Let $w^\sharp = \frac{P_{\tilde{s}}(w_{avg})}{\|P_{\tilde{s}}(w_{avg})\|_2}$, where $\tilde{s} = \frac{81 \cdot 2^{38}}{(1-2\eta)^2}s$

6: Find a point w_1 in the set $\mathcal{K} = \{w : \parallel w \parallel_2 \leq 1, \parallel w \parallel_1 \leq \sqrt{s}, (w \cdot w^\sharp) \geq \frac{(1-2\eta)}{9 \cdot 2^{19}}\}$

7: **Return** $\widehat{w}_0 \leftarrow REFINE(w_1, \frac{\delta}{2}, \eta, s, \alpha, b, \mathcal{K}, R, T)$,

where

step size $\alpha = (1 - \tilde{} 2\eta)^2$,

bandwidth $b = \Theta((1-2\eta)^2)$,

constraint set $\mathcal{K}$,

regularizer $R(w) = \Phi_{w_1}(w)$, and

number of iterations $T = \bigcirc(\frac{s}{(1-2\eta)^4}(\ln \frac{d}{\delta(1-2\eta)})^3)$

Chapter 4

Perturbed Examples - Generation

An adversarial example z of an instance x is such that $z \in \mathcal{U}(x)$ and $h_w(z) \neq f(x)$, where $f(x)$ denotes the true label of x. These adversarial examples are constructed by searching for a suitable perturbation δ to be added to x such that $h_w(x+\delta) \neq f(x)$[ACW18]. Before attempting to generate adversarial examples, let us look at its mathematical formulation.

Let $\mathcal{L}$ denote the loss function. Then in the classification setting one tries to maximize the accuracy on the unseen examples [TSE+19]. That is, one tries to train the models that have small expected loss. In other words, one tries to

$$\min_{w} \; E_{(x,f(x))\in\mathcal{D}}[\mathcal{L}(x, f(x), h_w(x))]$$

Whereas in the adversarial setting one tries to solve the following minimax problem:

$$\min_{w} \; E_{(x,f(x))\in\mathcal{D}}[\max_{\delta\in\Delta} \; \mathcal{L}(x+\delta, f(x), h_w(x+\delta))]$$

where Δ denotes the set of allowable perturbations that an adversary can apply to induce misclassification. That is, generation of an adversarial example requires to solve the following inner maximization problem.

$$\max_{\delta\in\Delta} \; \mathcal{L}(x+\delta, f(x), h_w(x+\delta))$$

In this study, we confine ourselves to the case that Δ is the set of all examples that are within γ distance, in ℓ_p metric, from the original example.

Literature suggests three main strategies, namely (i) Local Search, (ii) Combinatorial Optimization, and (iii) Convex Relaxation, to solve this latter optimization problem. Local Search and Convex Relaxation strategies are employed to find the lower and upper bounds on the optimization objective, respectively; whereas the Combinatorial Optimization formulation results in the exact solution of the problem.

Local Search methods can broadly be classified as (i) Methods that find adversarial perturbations which increase the value of the loss function and (ii) Methods that find adversarial perturbations which increase the probability of some target class, where this target class is unlikely to be the true class of the given example. The methods include (i) Fast Gradient [Sign], (ii) One-step Target Class Methods, (iii) Basic Iterative Method, and (iv) Iterative Least Likely Class Method[KGB17]. Here, in this report, we employ the Fast Gradient [Sign] Method (FG[S]M) [TPG+17] to generate the adversarial examples. The equations that govern this method are:

$$\delta = \delta \cdot sign\left(\nabla_x \mathcal{L}(x, f(x))\right) \text{ for } \ell_\infty - norm$$

and

$$\delta = \delta \cdot \frac{(\nabla_x \mathcal{L}(x, f(x)))}{\| \left(\nabla_x \mathcal{L}(x, f(x))\right) \|_p} \text{ for other } \ell_p - norms,$$

Oscar Knagg in [Kna19] used Projected Gradient Descent to generate adversarial examples and he carried out the experiments using the MNIST dataset. In this study, Knagg observed that ℓ_∞ trained model is more robust. Motivated by this observation, we generate ℓ_∞ adversarial examples and train the algorithms on these. As part of this study, we implemented few of the existing halfspace learning algorithms, proposed couple of algorithms, and trained all these algorithms on the ℓ_∞ adversarial examples that we generated using the FGSM. A pseudo-code of the FGSM is as follows:

Algorithm 9 Generating Adversarial Examples using the FGSM

1: *Input δ, the maximum allowable ℓ_p perturbation, and the example z for which a corresponding adversarial example is to be generated. Also input the gradient of the loss function, i.e., input* $\nabla_x(\mathcal{L}(x, f(x), h_w(x)))$

2: Determine the *sign* of the $\nabla_x(\mathcal{L}(x, f(x), h_w(x)))$ at $x = z$. Let this be denoted by sg

3: Output $z + \delta.sg$ as the adversarial example corresponding to z

Chapter 5

Noise - Classification

Noise in the data can significantly affect various data analysis tasks such as classification, clustering, and association analysis. So, machine learning models need to be trained so that the resulting model will be resistant to the noise and produce accurate results. Noise in a machine learning dataset can broadly be classified into two types: (i) attribute noise and (ii) class or label noise [FV14]. Attribute noise is the noise present in the predictive attributes and label noise is the one present in the target attribute. Presence of noise in a dataset can cause increase in the number of training examples and increase in the model complexity. It is mentioned in the literature [FV14, ZW04, SGLH12] that label noise is potentially more harmful than the feature noise. Hence it is important to address this type of noise [FV14]. But it is known that agnostic PAC learning of halfspaces is computationally hard [ABHU15]. So, algorithms specific to a particular noise model are proposed in the literature. Following are some of the noise models considered in the agnostic PAC learning of halfspaces:

1. *Random classification noise*: Here the label of each example x is flipped independently with probability $\eta(x) = \eta \in [0, \frac{1}{2})$. That is each label is flipped with the same probability

2. *Massart noise*: This can be thought of as a generalization of the random classification noise. Here, the label of each example x is flipped independently with probability $\eta(x) \leq \eta \in [0, \frac{1}{2})$. That is, the adversary has control over choosing a noise rate $\eta(x)$ for each example x with the only constraint that $\eta(x) \leq \eta$.

3. *Tsybakov noise*: This is a generalization of the Massart noise. Here also the label of each example x is flipped independently with probability $\eta(x)$. But $\eta(x)$ is a function that satisfies the Tsybakov noise condition with parameters (θ, A). That is, for any $0 < t \leq \frac{1}{2}$, $\eta(x)$ satisfies the condition $\Pr[[\eta(x) \geq \frac{1}{2} - t]] \leq At^{\frac{\theta}{1-\theta}}$ [DKTZ20]

4. *Agnostic or Adversarial noise*: Here the label of each example x is flipped independently with any probability $\eta(x) \in [0, 1]$ [FCG21]

Chapter 6

Dataset - Generation

In order to simulate the environment considered by the algorithms proposed in [IDM20] and [DKM19], we first create an origin centered halfspace $h_w : \mathbb{R}^d \to \{\pm 1\}$ of the form $h_w(x) = sign(\langle w, x \rangle)$ where $w \in \mathbb{R}^d$ and $\| w \|_2 = 1$ using the random function from numpy library in Python. We then use this w to generate our dataset $\mathcal{X} \times \{\pm 1\}$ such that for every $x \in \mathcal{X}$, $\| x \|_2 = 1$ and the dataset is $\gamma-$ margin separated, i.e. $\forall x \in \mathcal{X}$, $|(\langle w, x \rangle)| > \gamma$, where $|c|$ is the absolute value of the real number c. Also the label y corresponding to the instance x is 1 if $(\langle w, x \rangle) \geq 0$ and $y = -1$ otherwise.

After creating the dataset we split it into train and test sets. Before we use the train set to train our algorithm, we randomly pick η number of instances and flip their labels. We feed this dataset with adversarial noise to our algorithm.

For the test dataset which we use to test the performance of our algorithm, we randomly pick η number of instances and perturb them using the methods described in Section 4. This helps us in determining how robust our algorithm actually is when we give it corrupted examples.

Chapter 7

Algorithms - Implemented

In [DKM19] Ilias et al. addressed the problem of learning $\gamma-$margin halfspaces in the agnostic PAC model and proposed couple of algorithms. The notation and the terminology used in these algorithms is as follows: For $\gamma \in (0,1)$, $\gamma-$margin error rate of a halfspace $h_w(x)$ with $\| w \|_2 \leq 1$ is $err_\gamma(h_w, \mathcal{D}) = \Pr_{(x,y)\in\mathcal{D}}[y\langle w, x\rangle \leq \gamma]$. Denote by $OPT_\gamma^{\mathcal{D}} = min_{\|w\|_2\leq 1} err_\gamma(h_w, \mathcal{D})$, the minimum $\gamma-$margin error rate achievable by any halfspace. An algorithm is an $\alpha-$agnostic $\gamma-$margin halfspace proper learner, $\alpha \geq 1$, if it outputs a halfspace h_w, of the hypothesis space $\mathcal{H}$, that with probability atleast $1-\eta$ satisfies $err(h_w, \mathcal{D}) \leq \alpha \cdot OPT_\gamma^{\mathcal{D}} + \epsilon$. That is, the algorithm with probability atleast $1-\eta$ outputs a halfspace h_w whose error rate is competetive with the α times the $\gamma-$margin error rate of the optimal halfspace. The proposed algorithms are $\alpha-$agnostic $\gamma-$margin halfspace proper learning algorithms on the unit ball. That is, the examples (instances) are drawn from the unit ball $\mathbb{B}_d$ in $\mathbb{R}^d$. In other words, the examples are drawn from the set $\mathbb{B}_d = \{x \in \mathbb{R}^d : \| x \|_2 \leq 1\}$. One of these algorithms is meant for small values of approximation ratio, α, such as $\alpha = 1+\delta$ and $\delta \in (0,1)$. Another of these algorithms is meant for large $\alpha > 1$. In this study, only the algorithms for small values of α are considered. Two such algorithms, namely *Basic 1-Agnostic Proper Learning Algorithm* and *Near-Optimal* $(1+\delta)-$*Agnostic Proper Learner* are proposed in their paper. The latter two algorithms employ spectral methods and Empirical Risk Minimization (ERM) Techniques. The authors claim that these algorithms with small modifications to the associated constant factors give a halfspace w that not only works with the zero-one loss but also with the $0.99\gamma-$margin error. That is, the halfspace w that these algorithms give not only

satisfies $err(h_w, \mathcal{D}) \leq \alpha \cdot OPT_{\gamma}^{\mathcal{D}} + \epsilon$ but also satisfies $err_{0.99\gamma}(h_w, \mathcal{D}) \leq \alpha \cdot OPT_{\gamma}^{\mathcal{D}} + \epsilon$.

In another work [IDM20] Ilias et al. addressed the adversarial robust proper learning of halfspaces under ℓ_p perturbations in the distribution-independent agnostic PAC model. Let $\mathcal{U} : \mathcal{X} \to 2^{\mathcal{X}}$ be a function that defines the set of all allowable perturbations, where $2^{\mathcal{X}}$ denotes the power set of $\mathcal{X}$. Here we assume that the perturbations are ℓ_p perturbations for some $p \geq 1$. Let $\mathcal{U}_{p,\gamma}(x) = \{z \in \mathcal{X} :\| z - x \|_p \leq \gamma\}$. That is, $\mathcal{U}_{p,\gamma}(x)$ is the set of all points that are within the γ distance from x, measured using the ℓ_p distance metric. Robust risk of a hypothesis $h \in \mathcal{H}$ with respect to a distribution $\mathcal{D}$ is denoted by $\mathcal{R}_{\mathcal{U}}(h, \mathcal{D})$ and it is defined as $\mathcal{R}_{\mathcal{U}}(h, \mathcal{D}) = \Pr_{(x,y)\in\mathcal{D}}\{\exists z \in \mathcal{U}(x) : h(z) \neq y\}$. Now let $OPT_{p,\gamma}^{\mathcal{D}} = \arg\min_{h_w \in \mathcal{C}} \mathcal{R}_{\mathcal{U}_{p,\gamma}}(h_w, \mathcal{D}) = \arg\min_{h_w \in \mathcal{H}} \mathcal{R}_{\mathcal{U}_{p,\gamma}}(h_w, \mathcal{D})$. This is because $\mathcal{C} = \mathcal{H}$ in proper learning. Then in [IDM20] Ilias et al. designed an algorithm that efficiently outputs a hypothesis h_w that with probability $1-\eta$ satisfies $\mathcal{R}_{\mathcal{U}_{p,(1-\nu)\gamma}} \leq \alpha \cdot OPT_{p,\gamma}^{\mathcal{D}} + \epsilon$, where ν is a small constant close to 0 and $\alpha = 1 + \delta$, $0 < \delta < 1$.

As part of this study, we implemented the two algorithms given in [DKM19] designed for small values of approximation ratio and the algorithm given in [IDM20].

7.1 Basic Algorithm - Agnostic Proper Learning Algorithm by Ilias et al.

Algorithm 10 Basic Algorithm - Agnostic Proper Learning Algorithm

1: Draw a multiset $\mathcal{S} = (x^{(i)}, y(i))$ of i.i.d samples from $\mathcal{D}$,

2: where $m = \Omega(log(1/\tau)/(\epsilon^2\gamma^2))$

3: Let $\hat{\mathcal{D}}_m$ be the empirical distribution on $\mathcal{S}$

4: Let $M^{\hat{\mathcal{D}}_m} = E_{(x,y)\sim\hat{\mathcal{D}}_m}[\mathbf{x}\mathbf{x}^T]$

5: Set $\delta = \epsilon\gamma^2/16$. Use SVD to find a basis of $V_{\geq\delta}^{\hat{\mathcal{D}}_m}$, where $V_{\geq\delta}^{\hat{\mathcal{D}}_m}$ is the space spanned by the eigenvectors of $M^{\hat{\mathcal{D}}_m}$ corresponding to the eigenvalues with magnitude atleast δ

6: Compute a $\delta/2$ cover, $C_{\delta/2}$, in $l_2 - norm$, of $V_{\geq\delta}^{\hat{\mathcal{D}}_m} \cap \mathbb{S}_{d-1}$, where $\mathbb{S}_{d-1} = \{x \in \mathbb{R}^d : \| x \|_2 = 1\}$ is the unit sphere in $\mathbb{R}^d$

7: Let $v \in argmin_{w \in C_{\delta/2}} err_{\gamma/4}^{\hat{\mathcal{D}}_m}(w)$

8: return $h_v(x) = sign(\langle v, x \rangle)$

7.2 Near - optimal $(1+\delta)$ -Agnostic Proper Learner by Ilias et al.

Algorithm 11 Near - optimal $(1+\delta)$ -Agnostic Proper Learner

1: Draw a multiset $\mathcal{S} = (x^{(i)}, y(i))_{i=1}^m$ of i.i.d samples from $\mathcal{D}$,

2: where $m = \Omega(log(1/\tau)/(\epsilon^2\gamma^2))$

3: Let $\hat{\mathcal{D}}_m$ be the empirical distribution on $\mathcal{S}$

4: **for** all sequences $k^{(0)}, k^{(1)}, \ldots, k^{(s-1)}$ of positive integers with sum at most $8/(\delta\gamma^2)+2$ **do**

- Let $\mathbf{w}^{(0)} = 0$
- **for** $i = 0, 1, \ldots, s-1$ **do**
  - Let $\mathcal{D}^{(i)}$ be $\widehat{\mathcal{D}}_m$ conditioned on $y\langle \mathbf{w}^{(i)}, \mathbf{x} \rangle \leq \gamma/2$
  - Let $\mathbf{M}^{(i)} = \mathbf{E}_{(x,y)\sim\mathcal{D}^{(i)}}[\mathbf{x}\mathbf{x}^T]$
  - Use SVD on $\mathbf{M}^{(i)}$ to find basis for $V_{k^{(i)}}$ the span of the top $k^{(i)}$ eigenvectors
  - Let $C^{(i)}$ be a $\delta\gamma^3$-cover, in ℓ_2-norm, of $V_{k^{(i)}} \cap \mathbb{B}_d$ of size $(1/(\delta\gamma))^{O(k^{(i)})}$, where $\mathbb{B}_d = \{x \in \mathbb{R}^d : \| x \|_2 \leq 1\}$ is the unit ball in $\mathbb{R}^d$
  - For each $\mathbf{p}^{(i)} \in C^{(i)}$ repeat the next step of the for loop with $\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} + \mathbf{p}^{(i)}$
- **end for**

5: **end for**

6: Let C denote the set of all $\mathbf{w}^{(i)}$ generated in the above loop

7: Let $\mathbf{v} \in argmin_{\mathbf{w} \in C} err^{\widehat{\mathcal{D}}_m}_{\gamma/2}(\mathbf{w})$

8: **return** $h_v(\mathbf{x}) = sign(\langle \mathbf{v}, \mathbf{x} \rangle)$

7.3 Another Agnostic Proper Learning Algorithm by Ilias et al.

The algorithm uses the concept of online learning with a margin gap as given in the following definition.

Definition: *An online learner A for the class of halfspaces is called an ℓ_p online learner with mistake bound M and (γ, γ') margin gap if it satisfies the following: In each round A returns a vector $w \in \mathbb{B}^d_q$. Moreover, for any sequence of labeled examples (x_i, y_i) such that there exists $w^c \in \mathbb{B}^d_q$ with* $\text{sign}(\langle w^c, x_i \rangle - y_i\gamma) = y_i$ *for all i, there are at most M values of t such that* $\text{sign}(\langle w_t, x_t \rangle - y_t\gamma') \neq y_t$, *where $w_t = A((x_1, y_1)(x_2, y_2), \cdots, (x_{t-1}, y_{t-1}))$,* and q is the dual exponent of p, i.e., it satisfies the equation $\frac{1}{p} + \frac{1}{q} = 1$.

The algorithm is based on the following Lemma and uses the online learning algorithm proposed by Gentile in [Gen01].

Lemma: Assume that there is a polynomial time ℓ_p online learner A for halfspaces with a (γ, γ') margin gap and mistake bound of M. Then there exists an algorithm that given a multiset of labeled examples $S \subseteq \mathbb{B}^d_p \times \{\pm 1\}$ and $\delta \in (0, 1)$, runs in $poly(|S|d)$ time and with probability $2^{-O(Mlog(1/\delta))}$ returns $\mathbf{w} \in \mathbb{B}^d_p$ such that $err^S_{\gamma'}(\mathbf{w}) \leq (1+\delta) \cdot OPT^S_\gamma$, where $\mathbb{B}^d_p$ is the unit ball in $\mathbb{R}$ with ℓ_p distance metric and $err^S_\gamma = \frac{1}{|S|} \cdot |\{(x, y) \in S : \ sign(\langle w, x \rangle - y\gamma) \neq y\}|$, $|S|$ denotes the cardinality of the set S.

The algorithm is as follows:

Algorithm 12 Agnostic proper learning algorithm for ℓ_p -γ-margin halfspace

1: Let Samples $= \emptyset$

2: For $i = 0$ to M

- Let $\mathbf{w} = A(Samples)$, where A is an online mistake bound learning algorithm for halfspaces.
- Let T be the set of $(\mathbf{x}, y) \in S$ so that $sign(\langle \mathbf{w}, \mathbf{x} \rangle - y\gamma') \neq y$
- If $T \neq \emptyset$, and otherwise with 50% probability, return $\mathbf{w}$
- Draw $(\mathbf{x}_i, y_i)$ uniformly at random from T, and add it to Samples

3: Return $\mathbf{w}$

7.3.1 Gentile's Algorithm

Algorithm 13 $ALMA_p(\alpha; B, C)$ with $\alpha \in (0, 1], B, C > 0$

1: **Initialization:** Initial weight vector $w_1 = 0; k = 1$

2: **For** $t = 1, \ldots, T$ **do:**

- Get example $(x_t, y_t) \in \mathcal{X} \times \{\pm 1\}$
  - Set $\gamma_k = B\sqrt{p-1}\frac{1}{\sqrt{k}}$
  - **If** $y_t \langle w_k, \widehat{x}_t \rangle \leq (1 - \alpha)\gamma_k$ **then:**
    - $\eta_k = \frac{C}{\sqrt{p-1}} \frac{1}{\sqrt{k}}$
    - $w'_k = f^{-1}(f(w_k) + \eta_k y_t \widehat{x}_t)$
    - $w_{k+1} = w'_k / max\{1, ||w'_k||_q\}$
    - $k \leftarrow k + 1$

The constants B and C in Gentile's algorithm are taken to be $\frac{\sqrt{8}}{\alpha}$ and $\sqrt{2}$ respectively. Also q is such that $\frac{1}{p} + \frac{1}{q} = 1$.

7.3.2 Agnostic proper learning algorithm for ℓ_p-γ-margin halfspace after incorporating ALMA into it

Algorithm 14 $ALMA$ in Lemma 6

1: **Initialization:** Initial weight vector $w_1 = 0; i = 1$

2: **For** $t = 1, \ldots, T$ **do:**

- $\gamma_i = \frac{\sqrt{8}}{\nu} \cdot \frac{1}{\sqrt{i}}$
- Let T be the set of $(x, y) \in S$ so that $sign(\langle w_i, x \rangle - y(1-\nu)\gamma_i) \neq y$
- If $T = \emptyset$, and otherwise with 50% probability, return w_i
- Draw (x_i, y_i) uniformly at random from T
- $\eta_i = \frac{\sqrt{2}}{\sqrt{i}}$
- $w_i' = w_i + \eta_i y_i \widehat{x}_i$
- $w_{i+1} = w_i' / max\{1, ||w_i||_q'\}$
- $i \leftarrow i + 1$

Chapter 8

Proposed Algorithms

Proposed algorithms are modifications of the *Basic 1-Agnostic Proper Learning Algorithm* and the *Near-Optimal* $(1+\delta)$*-Agnostic Proper Learner* given in [DKM19]. Following are the major steps of these two latter algorithms:

Algorithm 15 Major Steps

1: Compute the covariance matrix of the training examples

2: Compute the SVD of the covariance matrix and determine the eigenvectors corresponding to the largest eigenvalues

3: Determine a linear combination of the selected eigenvectors that give the small misclassification error

First two of these steps require only the polynomial time computation, whereas the third step requires $2^{\tilde{O}(1/(\epsilon\gamma^2))}$. This is because it requires determining a net over unit norm weight vectors by computing a cover of the selected unit norm eigenvectors corresponding to the largest eigenvalues and then the best answer among these is output.

Our modifications focus only on the third step. In the first modification, we replace the third step by an answer to the following related question: What is the optimal linear combination of the chosen eigenvectors that results in the smallest squared γ-margin error? But the answer to this question lies in the least squares and hence results in a polynomial time algorithm. In our second modification, we randomly select sets of d samples each from the training set, determine a linear combination corresponding to each of these sets, and select the one that results in the least classification error. This again can be seen to require only polynomial time as long as the number of sets is bounded by a polynomial.

8.1 Theory Behind the Proposed Modifications

Let $e_1, e_2, \cdots, e_d$ be the eigenvectors of the covariance matrix and let without loss of generality $e_1, e_2, \cdots, e_k,\ k \leq d$, be the eigenvectors corresponding to the largest eigenvalues. Let $E_i = [e_1\ e_2\ \cdots\ e_i]$, for $1 \leq i \leq d$, be a $d \times i$ matrix, whose columns are the first i eigenvectors of the covariance matrix. Now let $C_{d\times n}$ be a matrix such that

$$X = [x_1\ x_2\ \cdots\ x_n] = E_d C_{d\times n}$$

where x_i is the i^{th} training sample (example). That is, the training samples X are expressed as a linear combination of the eigenvectors of the covariance matrix. Then $C_{d\times n} = E_d^T X$. This is because E_d is an orthonormal matrix.

Now let the classifier w be a linear combination of the eigenvectors corresponding to the largest eigenvalues. That is,

$$\begin{aligned} w_{d\times 1} &= \beta_1 e_1 + \beta_2 e_2 + \cdots + \beta_k e_k \\ \text{i.e.,} \quad w &= E_k \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_k \end{bmatrix} \\ \text{or,} \quad w &= E_k \beta \end{aligned}$$

Let $Y = [y_1\ y_2\ \cdots\ y_n]$ be the labels of the corresponding samples $x_1, x_2, \cdots, x_n$. Also

let

$$\begin{aligned} X_Y &= [y_1 x_1 \; y_2 x_2 \; \cdots \; y_n x_n] \\ &= [x_1 \; x_2 \; \cdots \; x_n]_{d \times n} \begin{bmatrix} y_1 & 0 & \dots & 0 \\ 0 & y_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & y_n \end{bmatrix} \\ &= XD, \end{aligned}$$

where

$$\begin{aligned} X &= [x_1 \; x_2 \; \cdots \; x_n], \\ D &= \begin{bmatrix} y_1 & 0 & \dots & 0 \\ 0 & y_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & y_n \end{bmatrix}, \end{aligned}$$

and

$$y_i x_i \qquad \text{for } 1 \leq i \leq n,$$

are scalar vector products of the samples with their corresponding labels.

Consider

$$\begin{aligned} w^T X_Y &= \beta^T E_k^T X D \\ &= \beta^T E_k^T E_d C_{d\times n} D \\ &= \beta^T (I_k|0)_{k\times d} C_{d\times n} D \\ &= (\beta^T|0)_{1\times d} C_{d\times n} D \\ &= \beta^T C_{k\times n} D \end{aligned}$$

We want β such that

$$w^T X_Y \geq \begin{bmatrix} \gamma & \gamma & \cdots & \gamma \end{bmatrix}_{1\times n} * \|w\|$$

This implies

$$(w^T X_Y)^T \geq \begin{bmatrix} \gamma \\ \gamma \\ \vdots \\ \gamma \end{bmatrix}_{n\times 1} * \|w\|$$

$$\Rightarrow D^T C_{k\times n}^T \beta \geq \begin{bmatrix} \gamma \\ \gamma \\ \vdots \\ \gamma \end{bmatrix}_{n\times 1} * \|\beta\| \qquad \because \|w\| = \|\beta\|$$

$$\Rightarrow D^T C_{k\times n}^T \beta = \begin{bmatrix} \gamma+\theta \\ \gamma+\theta \\ \vdots \\ \gamma+\theta \end{bmatrix} * \|\beta\|, \qquad \text{for some } \theta \geq 0 \qquad (A)$$

In order to solve this for β, we set $\|\beta\| = 1$. Then the least squares solution to this problem is

$$\beta = (C_{k\times n} D D^T C_{k\times n}^T)^{-1} C_{k\times n} D \begin{bmatrix} \gamma+\theta \\ \gamma+\theta \\ \vdots \\ \gamma+\theta \end{bmatrix}_{n\times 1}$$

First proposed modification to the algorithms given in [DKM19] is obtained based on this least squares solution. The modified algorithm is as follows:

Algorithm 16 First Proposed Algorithm

1: Draw a multi-set $S = (x_i, y_i)_{i=1}^m$ of i.i.d. samples from $\mathcal{D}$,

2: where $m = (\log(1/\tau)/(\epsilon^2\gamma^2))$

3: Let $\widehat{D}_m$ be the empirical distribution on S.

4: Let $X = [x_1\ x_2\ \cdots\ x_m]$ and let $y_1, y_2, \cdots, y_m$ be the corresponding labels

5: Let $D = diag(y_1, y_2, \cdots, y_m)$

6: Let $M^{(\widehat{D}_m)} = E_{(x,y)\in\widehat{D}_m}[xx^T]$

7: Set $\delta = (\epsilon\gamma^2)/16$. Use SVD to find all the eigenvectors of the matrix $M^{(\widehat{D}_m)}$ and a basis of $V_{\geq\delta}^{\widehat{D}_m}$

8: Let $E = E_d = [e_1\ e_2\ \cdots\ e_d]$ be the eigenvectors of $M^{\widehat{D}_m}$ and let $E_k = [e_1\ \cdots\ e_k]$ be a basis of $V_{\geq\delta}^{\widehat{D}_m}$

9: Express $X = E_d C_{d\times m}$ and let $C_{k\times m}$ be the first k rows of $C_{d\times m}$

10: Compute $w = (C_{k\times m} D D^T C_{k\times m}^T)^{-1} C_{k\times m} D \begin{bmatrix} \gamma+\theta \\ \gamma+\theta \\ \vdots \\ \gamma+\theta \end{bmatrix}_{m\times 1}$

11: return $h_w(x) = sign((w \cdot x))$

Our second proposed modification may be thought of as a randomized algorithm. It

works by repeating the following two steps, say, r times and pick the best solution from these r solutions. One may like to note that this idea of ours has some similarity with the learning algorithms for the Best Separating Hyper-plane and other related problems given in [BDS00].

Algorithm 17 Overview of the Second Proposed Algorithm

1: Randomly choose k equations from the set of n equations given in (A)

2: Solve this system of k equations

Detailed version of this is given in the next following algorithm:

Algorithm 18 Second Proposed Algorithm

1: Draw a multi-set $S = (x_i, y_i)_{i=1}^{m}$ of i.i.d. samples from D,

2: where $m = \log(1/\tau)/(\epsilon^2\gamma^2)$

3: Let $\widehat{D}_m$ be the empirical distribution on S.

4: Let $X = [x_1\ x_2\ \cdots\ x_m]$ and let $y_1, y_2, \cdots, y_m$ be the corresponding labels

5: Let $D = diag(y_1, y_2, \cdots, y_m)$

6: Let $M^{\widehat{D}_m} = E_{(x,y)\in\widehat{D}_m}[xx^T]$

7: Set $\delta = (\epsilon\gamma^2)/16$. Use SVD to find all the eigenvectors of the matrix $M^{\widehat{D}_m}$ and basis of $V_{\geq\delta}^{\widehat{D}_m}$

8: Let $E = E_d = [e_1\ e_2\ \cdots\ e_d]$ be the eigenvectors of $M^{\widehat{D}_m}$ and let $E_k = [e_1\ \cdots\ e_k]$ be a basis of $V_{\geq\delta}^{\widehat{D}_m}$

9: Express $X = E_d C_{d\times m}$ and let $C_{k\times m}$ be the first k rows of $C_{d\times m}$

10: for $i = 1, 2, \cdots r$ do

11: Randomly choose k samples and their corresponding labels; that is randomly select k columns of $C_{k\times m}$ and the corresponding columns of D. Let these be denoted by $C_{k\times k}^{(i)}$ and $D_k^{(i)}$ respectively.

12: Compute $w^{(i)} = (C^{(i)}_{k\times k} D^{(i)}_k D^{(i)T}_k C^{(i)T}_{k\times k})^{-1} C^{(i)}_{k\times k} D^{(i)}_k \begin{bmatrix} \gamma+\theta \\ \gamma+\theta \\ \vdots \\ \gamma+\theta \end{bmatrix}_{k\times 1}$

13: endfor

14: $w = w^{(l)}$, where $w^{(l)}$ is the one that gives best accuracy among $w^{(1)}, \cdots, w^{(r)}$

15: Return $h_w(x) = sign(w \cdot x)$

Chapter 9

Experiments, Results and Comparative Analysis

The first observation was that the algorithm in [DKM19] computes a cover and tries to figure out an appropriate linear combination of the eigenvectors corresponding to the top k eigenvalues and call it as the classifier.

Our experiments are of two types: In the first type we experimented with various linear combinations without having in an ad hoc way; whereas in the second type we experimented with a linear combination suggested by the theory presented in Chapter 8. Some of the combinations that we have tried as part of the first type along with the results are described in the following Experiments 1 to 5.

To measure the performance of the algorithms that we proposed against the algorithms in [DKM19] and [IDM20], we have trained all the algorithms on datasets with similar parameters. We have generated a dataset using the method described in Chapter 6. This dataset has 5 features and has $70,000$ instances with labels being either -1 or 1. This dataset is $\gamma-$ separated with $\gamma = 0.1$ as the $err_{\gamma}^{0-1}(w) = 0$. The accuracy of all algorithms are measured by considering the ground truth to be 0, i.e., $\gamma = 0$. The value of η, the noise rate, is taken to be 0.2. This means that labels of 20% of the instances from the train dataset are flipped and 20% of the instances from the test dataset are perturbed before testing the accuracy of the classifier.

Before we show our results let us denote the algorithm by [DKM19] as IDM19, the agnostic proper learning algorithm by [IDM20] as IDM20, and the results of the modifications specified in each of the experiments are given under the heading EXP.

9.1 Experiment 1

w = sum of product of eigenvalues with eigenvectors.

number of eigenvalues	Index	IDM19		EXP	
		train_acc	test_acc	train_acc	test_acc
1	1	74.37	69.25	74.37	69.25
	2	79.23	77.95	79.23	77.95
	3	77.82	74.68	77.82	74.68
	4	83.20	82.25	83.20	82.25
2	1	75.68	75.68	34.90	30.85
	2	64.52	64.04	40.83	42.08
	3	75.23	77.07	71.48	70.28
	4	81.39	77.31	39.88	41.37
3	1	71.36	70.65	59.65	59.05
	2	65.26	64.19	63.59	64.73
	3	80.57	84.96	46.78	43.65
	4	73.89	71.04	31.18	32.19
4	1	77.53	73.82	72.15	68.41
	2	72.16	73.36	51.04	50.90
	3	78.90	77.11	61.08	63.42
	4	74.27	72.23	55.49	56.26

9.2 Experiment 2

w = sum of products of all eigenvalues with their corresponding eigenvectors.

number of eigenvalues	Index	IDM19		EXP	
		train_acc	test_acc	train_acc	test_acc
1	1	86.73	86.08	52.76	52.71
	2	84.62	85.41	43.44	42.80
	3	87.10	86.86	47.62	49.07
	4	87.43	88.86	61.15	62.39
	5	72.29	70.56	40.39	39.93
2	1	83.54	82.02	42.54	41.94
	2	85.23	85.29	41.01	41.17
	3	76.04	75.66	54.68	54.27
	4	84.24	83.83	52.05	52.58
	5	75.67	72.34	55.87	54.08
3	1	85.90	86.31	42.68	43.74
	2	84.10	84.80	55.47	57.10
	3	87.57	88.087	50.41	50.35
	4	84.17	84.13	42.26	41.97
	5	85.03	84.73	58.99	58.89
4	1	83.63	83.89	46.04	46.02
	2	72.33	69.41	55.92	55.35
	3	89.18	88.78	56.17	55.88
	4	85.47	86.11	43.99	44.07
	5	87.89	87.69	47.37	47.06

9.3 Experiment 3

w = sum of products of inverses of all eigenvalues with their corresponding eigenvectors.

number of eigenvalues	Index	IDM19		EXP	
		train_acc	test_acc	train_acc	test_acc
1	1	86.14	86.56	56.11	55.75
	2	81.28	80.93	55.29	55.74
	3	83.88	83.45	51.53	50.95
	4	88.42	88.19	43.15	42.54
	5	91.57	92.44	55.79	56.10
2	1	86.52	86.33	47.61	47.31
	2	87.27	87.59	44.54	44.70
	3	83.84	83.90	58.68	58.89
	4	86.25	85.65	51.95	50.19
	5	89.24	88.43	44.70	44.69
3	1	84.18	84.65	56.04	54.46
	2	82.40	82.08	55.74	55.64
	3	86.17	86.16	43.52	41.99
	4	79.33	76.54	41.23	43.44
	5	85.35	85.63	47.97	47.96
4	1	86.14	86.61	46.03	46.18
	2	86.94	87.84	48.80	49.34
	3	84.65	84.24	53.97	53.28
	4	83.74	84.35	41.98	43.04
	5	84.35	85.37	44.20	45.12

9.4 Experiment 4

w = sum of product of eigenvalues with their corresponding eigenvectors where eigenvalues can either take their positive or negative multiple. That is, the linear combinations include enumerations of $(+1, -1)$.

number of eigenvalues	Index	IDM19		EXP	
		train_acc	test_acc	train_acc	test_acc
1	1	85.63	85.35	85.63	85.35
	2	91.32	92.02	91.32	92.02
	3	83.81	83.45	83.81	83.45
	4	82.82	82.85	82.82	82.85
	5	85.77	85.20	85.77	85.20
2	1	88.90	88.28	79.77	79.95
	2	89.15	89.38	77.71	77.41
	3	86.04	86.05	78.73	79.17
	4	87.20	87.00	81.00	80.91
	5	84.51	83.23	74.30	74.29
3	1	85.35	86.83	72.78	73.97
	2	89.57	89.33	77.88	78.23
	3	84.00	83.33	70.86	70.45
	4	85.40	85.31	72.04	71.38
	5	86.84	86.28	75.23	74.89
4	1	83.13	83.30	70.08	70.85
	2	82.70	83.08	69.84	69.42
	3	81.01	80.95	69.27	69.29
	4	83.69	83.79	69.52	70.41
	5	80.23	79.72	75.53	76.52

9.5 Experiment 5

w = either eigenvector corresponding to the largest eigenvalue or $(-1)*$eigenvector corresponding to the largest eigenvalue

number of eigenvalues	Index	IDM19		EXP	
		train_acc	test_acc	train_acc	test_acc
1	1	89.52	87.61	89.52	87.61
	2	83.41	81.21	83.41	81.21
	3	80.65	79.06	80.65	79.06
	4	87.89	87.60	87.89	87.60
2	1	90.52	90.61	92.48	92.24
	2	85.27	84.83	86.40	85.66
	3	92.18	92.12	92.18	92.12
	4	84.57	84.42	84.57	84.42
3	1	83.16	83.20	83.84	84.21
	2	86.06	85.63	91.22	91.27
	3	84.09	84.47	88.09	87.36
	4	85.99	85.69	87.73	87.66
4	1	84.54	84.35	86.62	86.55
	2	84.10	84.16	83.99	83.38
	3	83.33	83.86	89.52	90.21
	4	86.68	85.48	91.01	90.01

9.6 Experiment 6 - Testing our first proposed algorithm

This comes under the second type of experiments and hence it is supported by the theory. The accuracy results of our first proposed algorithm versus the Agnostic proper learning algorithm of [IDM20] is given in the following table.

Index	IDM20		First proposed modification	
	train_acc	test_acc	train_acc	test_acc
1	87.00	87.04	86.38	86.52
2	78.98	78.73	84.86	84.68
3	75.90	75.62	81.81	81.74
4	94.10	94.15	82.29	81.67
5	92.23	92.21	98.64	98.62
6	75.72	75.59	91.53	91.51
7	78.89	78.41	93.19	92.96
8	87.35	87.13	83.75	83.09
9	87.39	87.45	98.73	98.95
10	80.71	80.42	83.17	82.82

In the experiments 1 - 5, the basic algorithm from [DKM19] is experimented. The algorithm took about 15 seconds when only one eigenvector was taken into consideration. Given that the parameters were relaxed, the algorithm performed fairly well. Out of all the experiments conducted, experiment 5 yielded the best results where a multiple of the eigenvector corresponding to the largest eigenvalue was taken as the classifier. In detail, if we denote the eigenvector corresponding to the largest eigenvalue as v_1, then the classifier $w = v_1$ or $w = -1 * v_1$. This linear combination gave better results than the original algorithm even when the number of eigenvectors taken into consideration increased. The

highest accuracy of our classifier was 92.48%, while the highest accuracy of their algorithm was 92.18%. The lowest accuracy of their algorithm was recorded as 80.65% and the lowest accuracy of our classifier was 80.65%

Even when the parameters are tuned down, the algorithm from [IDM20] takes roughly about 30 minutes while our algorithm finishes all operations within half a minute. The lowest accuracy of our algorithm observed after running the experiment for 10 times is 81.81% while the lowest accuracy for IDM20 is 75.72%. The highest accuracy for our algorithm is recorded as 98.73%, while IDM20's highest observed accuracy is 94.10%.

Chapter 10
Conclusions

Given the exponential time nature of the algorithms proposed by Ilias et al.we had to relax the parameters considered in their paper in order to get results in a reasonable amount of time. For example the algorithm requires us to have a multiple of 100,000 instances and guarantees to give a robust classifier with a probability of $2^{-O(Mlog(1/\delta))}$. In our case, if we run the algorithm atleast 2^{10000} times, and with a mistake bound of atleast $10,000$ (each time), we would end up with the desired robust classifier with very high accuracy. As one can imagine, the required compute capability is too large to be practical.

Our proposed algorithm can provide a decent robust classifier with much less compute resources and time. Through experiments we found out that the sample complexity of our proposed algorithm is around $50,000$ samples and does better as the number of samples increases. However as all the experiments have been carried out on colab, the VM could not support processing more than $70,000$ samples. But in the case of having more data, we could probably try a randomized approach of picking samples and adding them to the training dataset.

Our algorithms not only produce results within two minutes but also has a decent accuracy of at least 80% with the highest accuracy being 97% on both train and test datasets. We have experimented the algorithms with the tuned parameters such as limiting the number of features to only 5, the maximum number of samples (instances) to only 70000, etc. This is to have fair comparison between the algorithms by Ilias et al. and the proposed algorithms. Also, since we do not have the resources to run the algorithm 2^{15} times with a mistake bound of atleast $10,000$, as required by the Ilias et al. algorithms, in our implementation of their algorithms, we had a mistake bound of 100 and we ran the algorithm for 2^5 times instead. Although this algorithm gave out good results and the

classifier's accuracy is almost always greater than 80% with the least accuracy being 64% and the highest being 94%, it took roughly about 30 minutes for the computation to end. It has to be noted that this is the time taken when the parameters have been modified. With the actual suggested values, the algorithms are expected to take days.

In conclusion, during this course we have studied various different algorithms starting from the standard perceptron algorithm by [Ros58] and the modifications of the perceptron algorithm by [ABV98], [SD09], and [YZ17]. Then we studied algorithms for sparse halfspaces in the realizable setting and in the presence of bounded noise. We learned how to perturb data points and create adversarial examples. We then studied [DKM19] and [IDM20] in detail and tried reproducing the results. In this process we have conducted various experiments and have recorded our observations. Also we have proposed algorithms that have better time and sample complexity. Not only does our algorithm output a classifier with a decent accuracy, it is much more practical in that it outputs the required classifier in a very short time.

Chapter 11

Future Work

We need to conduct more experiments and do a much more comprehensive comparative analysis by considering (i) large data sets, (ii) varying noise levels, etc. Also, our current work can be extended in quite a few ways. Following are some of the ideas that I believe could give some promising results:

- Extend our first proposed algorithm to sparse cases
- Modify the algorithm given in [IDM20] to consider the datapoint closest to the decision boundary of the current hypothesis from the misclassified dataset instead of randomly choosing the datapoint
- We would like to extend our algorithm and the algorithm from [IDM20] to handle other kind of label noise
- See if our algorithm can be modified to an online learning algorithm
- Also check if our algorithm can be modified to an active learning algorithm
- Run our first proposed algorithm for several iterations, each iteration considering a random sample having minimum sample complexity (about $50,000$ samples), and output the classifier with the least error.

Bibliography

[ABHU15] Pranjal Awasthi, Maria-Florina Balcan, Nika Haghtalab, and Ruth Urner. Efficient learning of linear separators under bounded noise. *CoRR*, abs/1503.03594, 2015.

[ABV98] Ravi Kannan Avrim Bum, Alan Frieze and Santosh Vempala. A Polynomial Time Algorithm for Learning Noisy Linear Threshold Functions. *Algorithimica*, 22:35 – 52, 1998.

[ACW18] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. pages 274–283, 2018.

[BDS00] Shai Ben-David and Hans Ulrich Simon. Efficient learning of linear perceptrons. In *NIPS*, pages 189–195, 2000.

[DKM19] Ilias Diakonikolas, Daniel M. Kane, and Pasin Manurangsi. Nearly tight bounds for robust proper learning of halfspaces with a margin. *CoRR*, abs/1908.11335, 2019.

[DKTZ20] Ilias Diakonikolas, Vasilis Kontonis, Christos Tzamos, and Nikos Zarifis. Learning halfspaces with tsybakov noise, 2020.

[FCG21] Spencer Frei, Yuan Cao, and Quanquan Gu. Agnostic learning of halfspaces with gradient descent via soft margins. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3417–3426. PMLR, 18–24 Jul 2021.

[FV14] Benoit Frenay and Michel Verleysen. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, 2014.

[Gen01] Claudio Gentile. A new approximate maximal margin classification algorithm, 2001.

[IDM20] Daniel M Kane Ilias Diakonikolas and Pasin Manurangsi. The Complexity of Adversarially Robust Proper Learning of Halfspaces with Agnostic Noise. *Arxiv*, abs/2007.15220, 2020.

[KGB17] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale, 2017.

[Kha18] Daniel Khashabi. Learning halfspaces; literature review and some recent results. 2018.

[Kna19] Oscar Knagg. Know your enemy: How you can create and defend against adversarial attacks, 2019.

[MP69] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, 1969.

[Ros58] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[SD09] Claire Monteleoni Sanjoy Dasgupta, Adam Tauman Kalai. Analysis of Perceptron-based Active Learning. *Journal of Machine Learning Research*, 10:281 – 299, 2009.

[SGLH12] José A. Sáez, Mikel Galar, Julián Luengo, and Francisco Herrera. Analyzing the presence of noise in multi-class problems: alleviating its influence with the

one-vs-one decomposition. *Knowledge and Information Systems*, 38:179–206, 2012.

[TPG+17] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples, 2017.

[TSE+19] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2019.

[YZ17] Songbai Yan and Chicheng Zhang. Revisiting perceptron: Efficient and label-optimal learning of halfspaces. In *NIPS*, 2017.

[Zha18] Chicheng Zhang. Efficient active learning of sparse halfspaces. *ArXiv*, abs/1805.02350, 2018.

[ZSA20] Chicheng Zhang, Jie Shen, and Pranjal Awasthi. Efficient active learning of sparse halfspaces with arbitrary bounded noise. *ArXiv*, abs/2002.04840, 2020.

[ZW04] Xingquan Zhu and Xindong Wu. Class noise vs. attribute noise: A quantitative study. *Artificial Intelligence Review*, 22:177–210, 2004.