CS 2243 Fall 2024 Scribes: <names>
09/30 Based on notes by: Alexander Cai, Neil Shah

Lecture 8: PAC Learning, Fourier Analysis, Semirandom Noise

1 Introduction to Supervised Learning

So far, we have focused on **distribution learning**, which is when you obtain samples from an unknown distribution and wish to learn about it.

Now, we'll focus on **supervised learning**, which is when samples are of the form (x, y) and we wish to estimate $y \mid x$. These are the questions we'll focus on:

1. What are the most powerful algorithms for such problems, and how do we analyze them? (technical tools)
2. Under what distributional assumptions can we prove they work? (modeling)

In practice, the algorithm of choice is the standard deep learning pipeline: you run SGD to optimize a deep neural network to fit training examples. Outside of some specialized settings, it is unclear why exactly this works.

For the most part, we only know how to analyze these approaches when we know there exists some other algorithm that could solve the problem in polynomial time. So, we might wish to study these other algorithms before analyzing deep learning algorithms.

2 PAC Learning

2.1 Introduction

"Probably Approximately Correct Learning," is a tool proposed by Leslie Valiant in 1984 [Val84] that has the following components:

- **Input domain** Ω_X: For example, could be $\{\pm 1\}^d$ or $\mathbb{R}^d$.
- **Label domain** Ω_X: For example, this could be $\{\pm 1\}$ (binary classification) or $\mathbb{R}$ (regression).
- **Concept class** $\mathcal{C}$: This is some class of functions $\Omega_X \to \Omega_Y$ that we want to work. For example, this could be Boolean formula or neural networks.
- **Loss function** $\ell : \Omega_X \times \Omega_Y \to \mathbb{R}_{\geq 0}$: These functions tell us how good a particular prediction is relative to the ground truth.
- **Data distribution** $\mathcal{D}_{XY}$: A joint distribution over $\Omega_X \times \Omega_Y$.

For most of this unit, we will focus on realizable cases where to sample $(x, y) \sim \mathcal{D}_{XY}$, you do

1. $x \sim \mathcal{D}_X$, where $\mathcal{D}_X$ is the input distribution (a probability distribution over Ω_X.
2. $y = f(x)$ where f is unknown in $\mathcal{C}$.

Now that we've made these definitions, we can define PAC learning:

- **Given**:
  - Examples $(x_1, y_1), \ldots, (x_n, y_n) \sim \mathcal{D}_{XY}$, called the training data.
  - Failure probability δ (this is why we say "probably").

    - Error parameter ϵ (this is why we say "approximately correct").

- **Goal**: Output the function $\tilde{f} : \Omega_X \to \Omega_Y$ such that with probability $\geq 1 - \delta$ over the randomness of $(x_1, y_1), \ldots, (x_n, y_n)$,

$$\underbrace{\mathbb{E}_{(x,y)\sim\mathcal{D}_{XY}}\left[\ell(\tilde{f}(x), y)\right]}_{\text{test loss}} \leq \underbrace{\min_{f\in\mathcal{C}} \mathbb{E}_{(x,y)\sim\mathcal{D}_{XY}}\left[\ell(f(x), y)\right]}_{\text{OPT}} + \epsilon.$$

  (In the realizable setting, we have OPT = 0.)

If we have $\tilde{f} \in \mathcal{C}$, we call this proper (or agnostic) learning.

2.2 Empirical Risk Minimization

Consider the naive algorithm where $\tilde{f}$ is the function in $\mathcal{C}$ that best fits the training data (and thus, minimizes "empirical risk"). This is

$$\tilde{f} = \arg\min_{f\in\mathcal{C}} \frac{1}{n}\sum_i \ell(f(x_i), y_i).$$

There are classical generalization bounds that control the number of samples needed before the test loss achieved by this empirical risk minimizer to be bounded by some ϵ. These bounds typically depend on a notion of the "complexity of $\mathcal{C}$"

The issue is that even for simple concept classes and datasets, empirical risk minimization can be computationally intractable.

3 Boolean Functions and Fourier Analysis

3.1 Introduction

A "textbook" setup is learning Boolean functions over the uniform distribution. Imagine there is a class of functions $\mathcal{C}$ $\{\pm 1\}^n \to \{\pm 1\}$. This could represent scenarios like

- Decision trees.
- Constant-depth Boolean circuits (highly stylized version of a neural network).
- Parities (the "building blocks" for Boolean functions).

In this case, loss is the 0-1 error. Thus, we want to minimize

$$\mathbb{P}_{x\sim\{\pm 1\}^n}[\tilde{f}(x) \neq f(x)].$$

3.2 Fourier Analysis of Boolean Functions

Observe that any Boolean function $f : \{\pm 1\}^n \to \{\pm 1\}$ can be written as a multilinear polynomial

$$f(x) = \sum_{S\subseteq[n]} \tilde{f}[S] \cdot x_S$$

where $\tilde{f}[S]$ is called the Fourier coefficient and x_s is called the parity.

The parity functions $x \to x_s$ are orthonormal with respect to the uniform over $\{\pm 1\}^n$:

$$\mathbb{E}[x_s x_T] = \mathbb{E}[x_{S\setminus T} \cdot x_{T\setminus S}] = \mathbb{E}[x_{S\setminus T}] \cdot \mathbb{E}[x_{T\setminus S}] = \mathbb{1}[S = T].$$

Now, consider the following **Plancherel formula**: the L^2 norm of f equals the L^2 norm of $\tilde{f}$. Equivalently,

$$\mathbb{E}_{x\sim\{\pm1\}^n}[f(x)^2] = \sum_{S\subseteq[n]} \tilde{f}[S]^2.$$

We denote the right hand side of the above as $\|f\|_2^2$. Additionally, we have the corollary that if $g(x) = \Sigma_S \tilde{g}[S] \cdot x_S$,

$$\mathbb{P}[\operatorname{sgn}(g(x)) \neq f(x)] \leq \|f-g\|_2^2 = \sum_{S\subseteq[n]} (\tilde{f}[S] - \tilde{g}[S])^2.$$

This is equivalent to saying that to learn f, it suffices to estimate its Fourier coefficients to small L^2 error.

As the basis is orthonormal, we can extract the coefficients as follows:

$$\mathbb{E}[f(x)\cdot x_T] = \mathbb{E}\left[\sum_{S\subseteq[n]} \tilde{f}[S]\cdot x_S \cdot x_T\right] = \sum_{S\subseteq[n]} \tilde{f}[S]\mathbb{E}[x_S x_T] = \tilde{f}[T].$$

The leftmost term in the equality can be estimated empirically using training data. We can estimate $\tilde{f}[T]$ for any T to error α using $O(1/\alpha^2)$ samples (the bound follows from a Chernoff bound).

However, the issue with the above is that there are exponentially many T's to do this computation for.

3.3 Low-Degree Algorithm

Per a 1993 paper of Linial-Mansour-Nisan [LMN93], for a lot of interesting concept classes, we can approximate the Fourier coefficient using low-degree polynomials. Define the low-degree trunction

$$f^{\leq t} = \sum_{S:|S|\leq t} \tilde{f}[S]\cdot x_S.$$

Then, if $\|f - f^{\leq t}\|_2^2 \leq \frac{\epsilon}{2}$, then to learn f, it suffices to estimate $\tilde{f}[S]$ for all $n^{O(t)}$ subsets S of size at most t, each to error $\left(\frac{\epsilon/2}{n^{O(t)}}\right)^{1/2}$.

From the Chernoff and Union Bound computations, it follows that the runtime of the above is $\frac{n^{O(t)}}{\epsilon}\cdot\log\left(\frac{n^{O(t)}}{\delta}\right)$.

3.4 A Remark on Agnostic Learning

The low-degree algorithm we just discussed also provides guarantees in the agnostic setting. It finds (an approximation to) the best low-degree approximation to the label, i.e.

$$\min_{g:\deg(g)\leq t} \mathbb{E}[(g(x)-y)^2].$$

The optimal g can be found with polynomial regression. Observe that the above is at most the following:

$$\min_{g:\deg(g)\leq t} \mathbb{E}[(g(x)-y)^2] \leq \mathbb{E}[(f^{\leq t}(x)-y)^2]$$

where we think of f as some optimal classifier that we're trying to compete against (and achieves test error OPT) and $f^{\leq t}$ is the low-degree of the optimal f. As we are minimizing over polynomials g and $f^{\leq t} \leq g$ is one such polynomial, this inequality holds.

Now, consider

$$\begin{aligned}\min_{g:\deg(g)\le t} \mathbb{E}[(g(x)-y)^2] &\le \mathbb{E}\left[(f^{\le t}(x)-y)^2\right]\\ &= \mathbb{E}\left[\left((f^{\le t}(x)-f(x))+(f(x)-y)\right)^2\right]\\ &\le 2\mathbb{E}\left[(f^{\le t}(x)-f(x))^2\right]+2\mathbb{E}[(f(x)-y)^2]\\ &\le O(\epsilon)+8\mathbb{P}[f(x)\ne y]=O(\epsilon)+8\mathrm{OPT}.\end{aligned}$$

This calculation tells us that if the optimal classifier we are competing against has a low-degree approximation, then running polynomial regression leads to something with test loss roughly order OPT plus epsilon.

Solving $\min_{g:\deg(g)\le t}\mathbb{E}[|g(x)-y|]$ and taking $\mathrm{sgn}(g(x)-s)$ for optimal s upgrades this to OPT + $O(\epsilon)$ “for free” (based on a work by Kalai, Klivans, Mansour, and Servedio in 2006 [KKMS05]) (“L1 polynomial regression”).

A brief note: while Fourier analysis is no longer relevant beyond the uniform distribution, the perspective of low-degree approximation and polynomial regression is still useful.

For example, if a concept class $\mathcal{C}$ can be represented exactly as $\mathrm{sgn}(p(x))$ for a low-degree polynomial p, we can learn functions from $\mathcal{C}$ over any input distribution by running halfspace learning over “feature space” $(x_s)_{|S|\le t}$ with runtime $n^{O(t)}$.

This is reminiscent of a result from Klivans and Servedio from 2001 [KS01], which is that DNFs can be expressed as degree-t polynomial threshold functions for $t=\tilde{O}(n^{1/3})$.

4 Learning halfspaces under semirandom noise

In previous lectures we discussed the task of **robust statistics**: How can we infer properties of the data generating process when we can only observe some (adversarially) *perturbed* version of the data? We will now examine this question from the perspective of PAC learning, specifically in the context of linear binary classification.

4.1 PAC halfspace learning

In this task:

1. A set of points $x_1,\dots,x_n$ is sampled from some distribution $\mathcal{D}$.
2. Each point is assigned a label $y_i=\mathrm{sgn}(\langle w^*,x_i\rangle)$, where w^* is the normal vector of the “ground truth” separating hyperplane.
3. Given the dataset of points and labels, we would like to construct a “good” estimator $f(x)$ of y. In particular, we would like f to have low generalization error: On an iid *test sample* $x\sim\mathcal{D}$, we would like the **misclassification probability** (a.k.a. the **test error** or **generalization error**)

$$\Pr[f(x)\ne\mathrm{sgn}(\langle w^*,x\rangle)]$$

   to be at most some small ε.

Today we will focus on the **proper learning** case in which $f(x)$ is constrained to have the same form as the ground truth function, i.e. in this case $f(x)=\mathrm{sgn}(\langle w,x\rangle)$ for some $w\in\mathbb{R}^d$. (This was also the case in Pset 0 Problem 2 in which the labels were generated by hyper-rectangles instead of hyperplanes.) Constrast this with the **improper learning** case in which $f(x)$ could belong to some other function class, e.g. the class of neural networks.

There has existed efficient algorithms for this problem since the 50s (eg the perceptron learning algorithm of [Ros58]). In particular, each data point imposes a linear constraint on w, allowing one to phrase this task as a linear programming problem, which can then be efficiently solved. These algorithms are generally **distribution-free**: they do not make any assumptions about the data distribution $\mathcal{D}$.

However, real data is never perfectly separated, and so we will consider three *noisy* versions of this problem, bringing us into the domain of robust statistics.

1. We will first analyze the simple case of **random classification noise** (RCN), in which each label y_i is independently corrupted with some probability η.

2. Then we'll discuss the **agnostic learning** setting, in which the covariates and labels are generated by some arbitrary joint distribution q. We'll see that learning "good" halfspaces in this setting is computationally difficult without making additional assumptions on q.

3. Finally, we'll analyze the **Massart noise** model, which extends the RCN model by allowing the flip probability $\eta(x)$ to depend on the covariates. We'll then present a polynomial-time, distribution-free, proper algorithm for learning halfspaces in this setting.

4.2 Random classification noise

Under the RCN corruption model, the data is first generated according to Section 4.1, and then each label y_i is flipped independently with probability $\eta \in [0, 1/2)$.

Theorem 1 (Halfspace learning under RCN is efficiently solvable [BFKV98])**.** *There exists a polynomial-time distribution-free algorithm for learning halfspaces under RCN.*

This algorithm modifies the original perceptron learning algorithm [Ros58] by intelligent outlier removal. See [Coh97], [DV04], [DV08] for details.

It turns out that, when the data is "nicely separated", halfspace learning under RCN can be formulated as a **convex problem** that can be efficiently solved. Precisely, define the **margin** of the hyperplane orthogonal to w to be the largest τ for which $|\langle w, x\rangle| \geq \tau$ almost surely over $x \sim \mathcal{D}$. Visually speaking, this means that there will be a gap of width 2τ around the decision boundary. We'll now show that for τ large enough, we can simply run stochastic gradient descent against the LeakyReLU over the dataset.

Define the zero-one loss for a hyperplane with normal vector w as

$$\begin{aligned}\operatorname{err}(w) &= \Pr_{x,y}(\operatorname{sgn}(\langle w, x\rangle) \neq y) \\ &= \mathbb{E}_{x,y}[1[-y \cdot \langle w, x\rangle > 0]].\end{aligned}$$

However, the indicator function is difficult to minimize: It is nonconvex and flat everywhere. Let's instead approximate it by a convex surrogate.

We might consider using $\operatorname{ReLU}(z) = \max\{0, z\}$, which is piecewise linear and agrees in sign with the indicator function. But this penalizes large deviations from zero very significantly while entirely ignoring any properly classified points. The LeakyReLU function addresses both of these issues:

$$\operatorname{LeakyReLU}_\lambda(z) = \begin{cases}(1-\lambda)z & z \geq 0 \\ \lambda z & z < 0\end{cases}$$

Typically, we'll set λ to roughly scale with the flip probability η from RCN.

Theorem 2 (Gradient descent on LeakyReLU learns large-margin halfspaces from RCN data [Byl94])**.** *Gradient descent on* $\operatorname{LeakyReLU}(-y \cdot \langle w, x\rangle)$ *learns large-margin halfspaces from RCN data. (This is an implicit consequence in the cited paper.)*

4.2.1 Derivation of LeakyReLU from RCN

It turns out that LeakyReLU is the "canonical" loss function for RCN data in the following sense. Consider the generalized linear model setting:

$$\mathbb{E}[y \mid x] = u(\langle w^*, x\rangle)$$

where u is some **link function** (a.k.a. **activation function** or **transfer function**). Note that setting u to be the identity mapping gives linear regression as a special case. Learning halfspaces under RCN can also be rephrased as such a problem:

$$\begin{aligned}\mathbb{E}[y \mid x] &= \eta \cdot (-\mathrm{sgn}(\langle w^*, x\rangle)) + (1-\eta) \cdot \mathrm{sgn}(\langle w^*, x\rangle) \\ &= u_\eta(\langle w^*, x\rangle) \\ \text{where } u_\eta(z) &= (1-2\eta) \cdot \mathrm{sgn}(z).\end{aligned}$$

Theorem 3 (Canonical loss for monotone link functions [AHW95]). *For a nondecreasing link function u, there exists a "matching" convex loss function $\ell_u(w; x, y)$ in the sense that $\nabla_w \ell_u(w; x, y) = (\hat{y} - y)x$ where $\hat{y} = u(\langle w, x\rangle)$.*

Proof. We can construct such a loss function as follows:

$$\ell_u(w; x, y) := \int_0^{\langle w,x\rangle} (u(r) - y)\, \mathrm{d}r.$$

The gradient result follows from the chain rule and the fundamental theorem of calculus. We skip the proof of convexity. □

It turns out that LeakyReLU is in this sense the matching loss for the u_η obtained from RCN:

$$\begin{aligned}\ell_{u_\eta}(w; x, y) &= \int_0^{\langle w,x\rangle} [(1-2\eta) \cdot \mathrm{sgn}(r) - y]\, \mathrm{d}r \\ &= (1-2\eta)|\langle w, x\rangle| - y\langle w, x\rangle\end{aligned}$$

Note that

$$\begin{aligned}\mathrm{LeakyReLU}_\lambda(z) &= \frac{1}{2}\big[(1-2\lambda)|z| + z\big] \\ \text{and so } \mathrm{LeakyReLU}_\eta(-y \cdot \langle w, x\rangle) &= \frac{1}{2}\ell_{u_\eta}(w; x, y)\end{aligned}$$

after applying $|-y\langle w, x\rangle| = |\langle w, x\rangle|$.

Theorem 4 (Large test loss implies large matching loss [AHW95]). *If the link function is Lipschitz with coefficient L (that is, for all $a, b \in \mathbb{R}$ we have $|u(a) - u(b)| \le L \cdot |a - b|$) then if w achieves large squared test error it will also achieve a large matching loss:*

$$\mathbb{E}_{x,y}[\ell_u(w; x, y) - \ell_u(w^*; x, y)] \ge \frac{1}{2L} \cdot \mathbb{E}_x\left[\Big(u(\langle w, x\rangle) - u(\langle w^*, x\rangle)\Big)^2\right].$$

Proof. This can be derived as follows:

$$\begin{aligned}\mathbb{E}_{x,y}[\ell_u(w; x, y) - \ell_u(w^*; x, y)] &= \mathbb{E}_{x,y}\left[\int_{\langle w^*,x\rangle}^{\langle w,x\rangle} (u(r) - y)\, \mathrm{d}r\right] \\ &= \mathbb{E}_x\left[\int_{\langle w^*,x\rangle}^{\langle w,x\rangle} \Big(u(r) - u(\langle w^*, x\rangle)\Big)\, \mathrm{d}r\right] \\ &\ge \frac{1}{2L}\, \mathbb{E}_x\left[\Big(u(\langle w, x\rangle) - u(\langle w^*, x\rangle)\Big)^2\right]\end{aligned}$$

□

We see that RCN is a relatively weak noise model that is often too simple for practical settings. Let us now consider a more sophisticated corruption model.

4.3 Agnostic learning

We now relax the assumptions on x and y. Rather than assuming some particular relationship between them, e.g. the previous halfspace labelling scheme given by w^*, we let them be drawn from some *arbitrary* joint distribution q over $\mathbb{R}^d \times \pm 1$.

Now, rather than finding a "true" decision hyperplane (which does not generally exist), we would like to compete with the *best possible* hyperplane, that is, the one that achieves the least generalization error:

$$\mathrm{OPT} = \min_{w'} \Pr_{x,y}[\mathrm{sgn}(\langle w', x\rangle) \neq y].$$

We seek to learn a classifer that achieves generalization error $\mathrm{OPT} + \varepsilon$. This is the task of **agnostic learning**.

We can frame agnostic learning in terms of an "adversary" as follows. We imagine that there exists some separating hyperplane orthogonal to w^* that generates the dataset x_i, y_i according to Section 4.1. Then we can imagine an adversary that "corrupts" an $\eta = \mathrm{OPT}$ fraction of the data in order to simulate the true joint distribution over x, y.

Compare this with the **strong contamination** setting in which the adversary is allowed to *arbitrarily* modify an *arbitrary* η fraction of the data. In the agnostic learning setting, our adversary serves primarily as an analytical tool that frames the data generating process in terms of the "corruption" of a labelled dataset.

Can we still design provably accurate and efficient algorithms in this setting?

For certain distributions q, e.g. where x is uniform over the unit sphere or comes from a log-concave distribution, we can prove strong polynomial-time guarantees. But the efficient algorithm for RCN halfspace learning from [BFKV98] makes no assumptions about the covariate distribution. Can we also make distribution-free guarantees in the agnostic learning setting?

Generally speaking, no: there exist counterexamples that demonstrate that even "weak" improper agnostic learning is computationally hard.

Theorem 5 (Agnostic learning is computationally hard [Dan16], [Tie23])**.** *Under standard complexity assumptions, there exist problem instances where, even if there exists an accurate optimal hyperplane, i.e.* $\mathrm{OPT} \approx 1\%$*, it is hard to find* any *classifer, not necessarily a hyperplane, achieving even* 49% *– random guessing!*

You can find other lower bound results on agnostic learning at [KKS06], [KK14], [DV21], [DKPZ21].

We've now seen two extremes of the noise model landscape:

- RCN, for which we can design efficient distribution-free algorithms;
- Agnostic learning, where we require significant assumptions on the data distribution to make any substantial guarantees.

Do there exist rich corruption models for which we can prove efficient *distribution-free* algorithms?

4.4 Massart noise and semirandom noise

One such noise model is **Massart noise**, which is almost identical to RCN (4.2), except that instead of all labels having the same flip probability, we now allow the flip probability to depend on x. That is, to sample from this corrupted distribution, we first sample $x \sim \mathcal{D}$. Then we observe $y = \mathrm{sgn}(\langle w^*, x\rangle)$ with probability $1 - \eta(x)$; otherwise, we observe the opposite (incorrect) label.

At first glance, this seems an easier task than RCN, since as long as $\eta(x)$ is bounded above by some η, the total proportion of flipped labels is in fact *less* than the number we would expect under RCN with flip probability η. (We can imagine a "helpful adversary" who comes and *cleans* some of the corrupted points.) But this **heterogeneous** flip probability creates additional structure that is more difficult to deal with! Composite noise models of this form, where the data is first generated by some simple noise model and then some of the corrupted points get "cleaned", are called **semirandom**, and are useful for testing whether or not algorithms depend on oversimplifying assumptions (such as homogenous noise).

For example, does the LeakyReLU-based algorithm for RCN still work for learning halfspaces of a given margin τ under Massart noise? The answer is *no* in the worst case. The following theorem generalizes this to *any* convex surrogate loss function.

Theorem 6 (Convex surrogate loss functions are not robust to Massart noise [DGT19]). *Given* any *convex surrogate loss function, its minimizer w will achieve at least $\Omega(\eta/\tau)$ misclassification probability in the worst case, i.e.*

$$\Pr_{x,y}[\mathrm{sgn}(\langle w, x\rangle) \neq y] = \Omega(\eta/\tau).$$

Given this, are there any polynomial-time algorithms for this task? This question was posed in and has been studied since [Slo88] (see for example [Coh97]).

4.5 Computational complexity for Massart noise

Theorem 7 (An improper learning algorithm [DGT19]). *There is a* distribution-free improper *algorithm that achieves $\eta + \varepsilon$ misclassification error. That is, the output is not a hyperplane, but rather a* linear threshold circuit*: a decision tree where each node corresponds to a different "band" in input space.*

Can we get the same guarantee with *proper* learning? Yes!

Theorem 8 (A proper learning algorithm [CKMY23]). *There is a polynomial-time, distribution-free,* proper *learning algorithm that also achieves $\eta + \varepsilon$ misclassification error.*

Additionally, if the margin τ is known beforehand, there is also a $O(\frac{d}{\tau^4 \varepsilon^5})$-sample, linear-time*, distribution-free, proper learning algorithm that achieves test loss $\eta + \varepsilon$. Though this takes a linear number of samples in the dimensionality of the data, the polynomial dependence on τ and ε is of quite high degree.*

The rest of these notes will describe and analyze the former proper learning algorithm in detail.

First, however, it is worth asking: is $\eta + \varepsilon$ the correct baseline? This bound only uses the *uniform* noise level η and doesn't take into account the *heterogeneous* noise in the Massart setting. Ideally, we'd want to achieve some OPT closer to $\mathbb{E}_x[\eta(x)]$. However, [CKMY23] demonstrate that achieving OPT + ε is computationally hard.

For other lower bounds on Massart noise halfspace learning, see [DK21], [NT22], and [DKMR22].

4.6 Filtertron as a two-player game (proper halfspace learning)

We can phrase this algorithm in terms of a two-player minimax game between Alice and Bob. Alice is trying to find an optimal w as if the data were corrupted by RCN (i.e. by minimizing the LeakyReLU loss function), while Bob is trying to "disprove" Alice's attempt by finding a reweighting $\phi(x)$ of the data that makes her attempt perform poorly:

$$\min_{\|w\|=1} \max_{\phi:\mathbb{R}^d \to \mathbb{R}} \mathbb{E}[\mathrm{LeakyReLU}_\lambda(-y \cdot \langle w, x\rangle) \cdot \phi(x)]$$

Is this game a good model for our algorithm? That is, does an equilibrium for this game give a good w? It turns out under Massart noise that Alice's best strategy is to play w^*. In fact, minimizing this objective is **equivalent** to achieving optimal test loss.

To make Bob's optimization problem tractable, rather than search over all reweightings of the entire input space, we'll only search over the width of the margin around w to find the rectangular area that judges Alice's boundary most harshly. This is now a practical optimisation over a single scalar variable:

$$\min_{\|w\|=1} \max_{\gamma \in \mathbb{R}} \mathbb{E}_{x,y}[\mathrm{LeakyReLU}_\lambda(-y \cdot \langle w, x\rangle) \mid |\langle w, x\rangle| \leq \gamma] \tag{1}$$

It turns out this simplification is still accurate, that is, that the equilibria for this simplified game also achieve test error $\eta + \varepsilon$.

4.7 Filtertron

The actual optimization of this objective proceeds by alternating between Alice computing projected gradient descent steps on the objective and Bob trying to "disprove" Alice's solution until a satisfactory solution is reached.

Let $L^\gamma(w)$ denote the objective in Eq. (1). We compute the "Filtertron" algorithm as follows:

1. Alice: initialize w to arbitrary unit vector
2. Repeat:
   (a) Bob plays maximizing slab γ
   (b) If zero-one loss of w is less than $\eta + \varepsilon$, return
   (c) Otherwise: update $w \leftarrow \Pi(w - \eta \cdot \nabla L^\gamma(w))$

4.8 Filtertron analysis (optional reading)

The analysis of this algorithm will proceed through two key ideas:

1. Show that the true halfspace w^* achieves small loss regardless of γ.
2. Show that Bob can penalize any bad move by Alice. i.e. for a w with suboptimal error, there exists some γ that causes $L^\gamma(w)$ to be large.

We'll use the following notation:

$$\begin{aligned} L^\gamma(w) &= \mathbb{E}_{x,y}[\ell(w;x,y) \mid |\langle w,x\rangle| \le \gamma] & & \\ \ell(w;x,y) &= \text{LeakyReLU}_\lambda(-y\cdot\langle w,x\rangle) & \text{err}(w;x,y) &= 1[y \ne \text{sgn}(\langle w,x\rangle)] \\ \ell(w;x) &= \mathbb{E}_{y|x}[\ell(w;x,y) \mid x] & \text{err}(w;x) &= \mathbb{E}_{y|x}[\text{err}(w;x,y) \mid x] \\ \ell(w) &= \mathbb{E}_{x,y}[\ell(w;x,y)] & \text{err}(w) &= \mathbb{E}_{x,y}[\text{err}(w;x,y)] \end{aligned}$$

Now consider the error for the true w^*. This is just the probability that the sample gets corrupted:

$$\text{err}(w^*;x) = \eta(x) \le \eta \quad \forall x$$

Intuitively, we aim to show that if Alice plays w^*, then regardless of what γ Bob plays, $L^\gamma(w^*)$ will be small.

Theorem 9 (The true halfspace achieves small loss)**.** *Suppose $\lambda \ge \eta + \varepsilon$ and the margin for w^* is τ. We aim to show that $L^\gamma(w^*) \le -\tau\varepsilon$.*

Proof. For now, we'll consider all points, i.e. assume $\gamma = \infty$. The proof will generalize to whatever finite γ Bob plays. Then we seek to decompose $L^\infty(w^*) = \ell(w^*)$ by casework.

Consider the input to $\text{LeakyReLU}_\lambda(-y\cdot\langle w,x\rangle)$. On a correct classification $y = \text{sgn}(\langle w,x\rangle)$, this is negative, and so the LeakyReLU has slope $1-\lambda$ here. Otherwise, for $y \ne \text{sgn}(\langle w,x\rangle)$, this is positive, and so the LeakyReLU has slope λ here. Decomposing the loss in this way gives

$$\begin{aligned} L^\infty(w^*) &= \ell(w^*) \\ &= \mathbb{E}_{x,y}\Bigg[\underbrace{1[y = \text{sgn}(\langle w^*,x\rangle)]}_{1-\text{err}(w^*;x,y)}\cdot\Big(\lambda\cdot(-|\langle w^*,x\rangle|)\Big) \\ &\qquad + \underbrace{1[y \ne \text{sgn}(\langle w^*,x\rangle)]}_{\text{err}(w^*;x,y)}\cdot\Big((1-\lambda)\cdot|\langle w^*,x\rangle|\Big)\Bigg] \\ &= \mathbb{E}_{x,y}[(\text{err}(w^*;x,y) - \lambda)\cdot|\langle w^*,x\rangle|] \\ &= \mathbb{E}_x\Bigg[\Big(\underbrace{\mathbb{E}_{y|x}[\text{err}(w^*;x,y)]}_{=\text{err}(w^*;x)\le\eta} - \underbrace{\lambda}_{\ge\eta+\varepsilon}\Big)\cdot\underbrace{|\langle w^*,x\rangle|}_{\ge\tau}\Bigg] \\ &\le -\tau\varepsilon \end{aligned}$$

In particular, note that at each step along the way, we can safely condition on events defined on x without affecting the reasoning. In particular, we can condition on $|\langle w, x\rangle| \leq \gamma$ to demonstrate that this bound holds for any γ. □

Let us continue with the second claim.

Theorem 10 (Suboptimal moves are punishable)**.** *If Alice plays a* poor *move w s.t.* $\mathrm{err}(w) \geq \lambda = \eta + \varepsilon$*, then Bob has a good choice of γ that "blames" Alice i.e.* $L^\gamma(w) \geq 0$.

Proof. We'll prove this by contradiction.

We'll begin with the same decomposition from before, with the added conditioning:

$$L^\gamma(w) = \mathbb{E}_x\left[(\mathrm{err}(w;x) - \lambda) \cdot |\langle w, x\rangle| \cdot 1[|\langle w, x\rangle| < \gamma]\right]$$

Now assume for contradiction that this is negative for all γ. Our method of attack is to show that the error $\mathrm{err}(w;x)$ would be too large and surpass η.

The current issue is that the term $|\langle w, x\rangle|$ is difficult to bound. To get around this, consider this simple equality:

$$z = \int_0^\infty 1[s < z]\ \mathrm{d}s$$

for $z \geq 0$. Taking $z = |\langle w, x\rangle|$ and then rearranging the integral outside the expectation gives

$$L^\gamma(w) = \int_0^\infty \mathbb{E}_x\left[(\mathrm{err}(w;x) - \lambda) \cdot 1[s < |\langle w, x\rangle| \leq \gamma]\right] \mathrm{d}s.$$

Since by assumption this quantity is negative, we know for some $s(\gamma)$ that the integrand must be negative. In particular, since negative values are nonzero, the indicated event must hold, and so we must have $s(\gamma) < \gamma$.

Now, by considering $L^{s(\gamma)}$, we can apply the same reasoning again to achieve $s^{(2)}(\gamma) < s^{(1)}(\gamma) < \gamma$ (where the superscript indicates iteration). Repeating this gives a monotonically decreasing sequence $s^{(i)}(\gamma)$ for $i \in \mathbb{N}$. Now consider partitioning the interval $[0, 1]$ according to these points. We know that in each of these sub-intervals $[s^{(i+1)}(\gamma), s^{(i)}(\gamma)]$ that the integrand $\mathbb{E}_x[\mathrm{err}(w;x) - \lambda]$ is negative. But then adding all of these sub-intervals together gives

$$\mathbb{E}_x[\mathrm{err}(w;x) - \lambda] < 0$$

so $\mathrm{err}(w) < \lambda = \eta+\varepsilon$. But this contradicts our original assumption that Alice played a poor move with $\mathrm{err}(w) \geq \eta+\varepsilon$. So Bob must be able to choose some γ such that $L^\gamma(w) \geq 0$. □

Together, these imply that an approximate equilibrium for this game yields a good halfspace w with error $\eta + o(1)$.

References

[AHW95] Peter Auer, Mark Herbster, and Manfred K. K Warmuth. Exponentially many local minima for single neurons. In *Advances in Neural Information Processing Systems*, volume 8. MIT Press, 1995.

[BFKV98] A. Blum, A. Frieze, R. Kannan, and S. Vempala. A Polynomial-Time Algorithm for Learning Noisy Linear Threshold Functions. *Algorithmica*, 22(1):35–52, 1998.

[Byl94] 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. Association for Computing Machinery, 1994.

[CKMY23] Sitan Chen, Frederic Koehler, Ankur Moitra, and Morris Yau. Classification Under Misspecification: Halfspaces, Generalized Linear Models, and Connections to Evolvability. 2023.

[Coh97] E. Cohen. Learning noisy perceptrons by a perceptron in polynomial time. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, FOCS '97, page 514. IEEE Computer Society, 1997.

[Dan16] Amit Daniely. Complexity Theoretic Limitations on Learning Halfspaces. 2016.

[DGT19] Ilias Diakonikolas, Themis Gouleakis, and Christos Tzamos. Distribution-Independent PAC Learning of Halfspaces with Massart Noise. 2019.

[DK21] Ilias Diakonikolas and Daniel M. Kane. Near-Optimal Statistical Query Hardness of Learning Halfspaces with Massart Noise. 2021.

[DKMR22] Ilias Diakonikolas, Daniel M. Kane, Pasin Manurangsi, and Lisheng Ren. Cryptographic Hardness of Learning Halfspaces with Massart Noise. 2022.

[DKPZ21] Ilias Diakonikolas, Daniel M. Kane, Thanasis Pittas, and Nikos Zarifis. The Optimality of Polynomial Regression for Agnostic Learning under Gaussian Marginals. 2021.

[DV04] John Dunagan and Santosh Vempala. Optimal outlier removal in high-dimensional spaces. *Journal of Computer and System Sciences*, 68(2):335–373, 2004.

[DV08] John Dunagan and Santosh Vempala. A simple polynomial-time rescaling algorithm for solving linear programs. *Mathematical Programming*, 114(1):101–114, 2008.

[DV21] Amit Daniely and Gal Vardi. From Local Pseudorandom Generators to Hardness of Learning. In *Proceedings of Thirty Fourth Conference on Learning Theory*, pages 1358–1394. PMLR, 2021.

[KK14] Adam Klivans and Pravesh Kothari. Embedding Hard Learning Problems Into Gaussian Space. In Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*, volume 28 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 793–809. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2014.

[KKMS05] Kalai-Klivans-Mansour-Servedio. Agnostically learning halfspaces. *Proceedings of the 46th Foundations of Computer Science*, 2005.

[KKS06] Adam Tauman Kalai, Adam R Klivans, and Rocco A Servedio. Agnostically Learning Halfspaces. 2006.

[KS01] Klivans-Servedio. Learning DNF in time $2^{\tilde{O}(n^{1/3})}$. *Proceedings of the 26th Annual Symposium on Theory of Computing*, 2001.

[LMN93] Linial-Mansour-Nisan. Constant depth circuits, fourier transform, and learnability. *Journal of the ACM*, 1993.

[NT22] Rajai Nasser and Stefan Tiegel. Optimal SQ Lower Bounds for Learning Halfspaces with Massart Noise. 2022.

[Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[Slo88] Robert H. Sloan. Types of noise in data for concept learning. In David Haussler and Leonard Pitt, editors, *Proceedings of the First Annual Workshop on Computational Learning Theory, COLT '88, Cambridge, MA, USA, August 3-5, 1988*, pages 91–96. ACM/MIT, 1988.

[Tie23] Stefan Tiegel. Hardness of Agnostically Learning Halfspaces from Worst-Case Lattice Problems. 2023.

[Val84] Valiant. A theory of the learnable. *Communications of the ACM*, 1984.