CS 224 Fall 2023 Scribes: Neil Shah
October 18, 2023

Lecture 12: PAC Learning Basics, Fourier Analysis

Today, we continued from the previous class on classification under Massart noise (specifically, the notion of the two-player game) and proved a key lemma mentioned in the last class. Then, we continued by starting the course's Supervised Learning unit.

1 Finishing Lecture 11

1.1 Notational Reference

Recall the notation presented in the previous lecture. For reference, the definition of LeakyRELU is presented below:

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

as well as the following notation:

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

Additionally, $\lambda = \eta + \epsilon$.

1.2 A Brief Recap

Recall the framing of the Filtertron algorithm as a game between two players: Alice and Bob. We sought to find some "good" halfspace w satisfying

$$w = \min_{\|w\|=1} \max_{\gamma \in \mathbb{R}} L^\gamma(w).$$

We can imagine this as some two-player game where Alice is playing some guess for the halfspace and Bob is picking some slab γ such that the loss incurred by Alice over the slab of points selected by Bob is large.

We claimed that if we could show the two lemmas that follow (also in the Lecture 11 scribe notes), we would be done when it comes to showing this.

Lemma 1 (The true halfspace achieves small loss)**.** *Suppose that $\gamma \geq \eta + \epsilon$ and the margin for w^* is τ. Then, $L^\gamma(w^*) = -\tau\epsilon$.*

We proved this at during Lecture 11. Today, we show the latter of the two lemmas.

1.3 Proving Lemma 2

Lemma 2 (Suboptimal moves are punishable)**.** *If Alice plays w such that err$(w) \geq \eta+\epsilon$, there exists some slab $\{x : |\langle w, x\rangle| \leq \gamma|$ for Bob such that $L^\gamma(w) \geq 0$.*

Proof. For the sake of contradiction, assume that no such slab exists (i.e. $L^\gamma(w) < 0$ for all γ). Additionally, observe that

$$\begin{aligned} L^\gamma(w) &= \mathbb{E}_{x,y}[\ell(w; x, y)\mathbb{1}[|\langle w, x\rangle| \leq \gamma]] \\ &= \mathbb{E}_x\left[\Big(\text{err}(w; x) - \lambda\Big) \cdot |\langle w, x\rangle| \cdot \mathbb{1}[|\langle w, x\rangle| < \gamma]\right] \end{aligned}$$

Now, we can use the following representation of $|\langle w, x\rangle|$ as an integral:

$$|\langle w, x\rangle| = \int_0^\infty \mathbb{1}[s < |\langle w, x\rangle|]\, ds.$$

The intuition behind why the above equality holds follows from noting that the area under the curve forms a rectangle with area $|\langle w, x\rangle|$. Applying this to the above,

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

Now, by an averaging argument, as $0 > L^\gamma(w)$ (the integral is negative), there exists some s such that the integrand is negative.

This quantity is nonzero ($< 0 \implies \neq 0$), so $s(\gamma) < \gamma$ (because the indicator $\neq 0$). An analogous argument for $L^{s(\gamma)}$ implies $s^{(2)}(\gamma) < s^{(1)}(\gamma) < \gamma$ and so on. If we partition $[0, 1]$ over these points, we split $[0, 1]$ into intervals over which the integrand is negative. Adding these sub-intervals together yields

$$\mathbb{E}_x[\text{err}(w; x) - \lambda] < 0 \implies \text{err}(w) < \lambda = \eta + \epsilon.$$

However, we assumed that $\text{err}(w) \geq \eta + \epsilon$, so we have a contradiction. Thus, such a slab must exist, as desired. $\square$

A key takeaway from this proof is the trick that for some variable z, we can use

$$z = \int_0^\infty \mathbb{1}[s < z] \, ds$$

to replace a variable with an integral of an indicator function and use this to get rid of this variable.

1.4 Recap of Robustness

This unit revealed ways that robust estimation tasks can be reduced to a min-max optimization over an "estimate" and a "discriminator":

- With **robust mean estimation**, we reduced it to

$$\min_w \max_u u^\top \Sigma_w u.$$

- With **list-decodable mean estimation**, we reduced it to

$$\min_w \max_{u \perp \{u_1, \ldots, u_k\}} u^\top \Sigma_w u.$$

- With **halfspaces with Massart noise**, we reduced it to

$$\min_w \max_\gamma L^\gamma(w).$$

Unlike algorithms in past lectures (i.e. the sum of squares unit), these algorithms are practical, are based on simple iterative updates, and make minimal distributional assumptions.

Sometimes, we have to be careful with the choice of model (especially when making minimal distributional assumptions):

- **Regression:** Does adversary control random or arbitrary function? Is it only the labels that get corrupted?
- **List-decodable learning:** Learn the true parameter or just a list of candidates?
- **Today:** Are label corruptions stochastic, adversarial, or somewhere in between?

And lastly, we can think about these algorithms from an "upstream" and "downstream" perspective:

- **Upstream:** Stress-test existing models to prevent "assumption overfitting."
- **Downstream:** Robustness (and lack thereof) can have unintended consequences even in settings without a real adversary (i.e. fairness).

These algorithms all suggest "general frameworks" that could be used even beyond the realms in which they are developed. [DKKLMS17]

2 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.

3 PAC Learning

3.1 Introduction

PAC Learning, or "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.

3.2 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.

4 Boolean Functions and Fourier Analysis

4.1 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

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

4.2 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,

$$\mathbb{E}_{x\sim\{\pm 1\}^n}[f(x)^2] = \sum_{S\subseteq[n]} \tilde{f}[S]^2.$$

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

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

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.

4.3 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 $\frac{n^{O(t)}}{\epsilon} \cdot \log\left(\frac{n^{O(t)}}{\delta}\right)$.

4.4 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

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

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 $\min_{g:\deg(g)\leq t} \mathbb{E}[|g(x)-y|]$ and taking 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 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" $(x_s)_{|S|\leq t}$ 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 $t=\tilde{O}(n^{1/3})$.

References

[DKKLMS17] Diakonikolas-Kamath-Kane-Li-Moitra-Stewart. Robustly learning a gaussian: Getting optimal error, efficiently. *SODA*, 2017.

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

[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.

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