Mathematical Aspects of Deep Learning

Mathematics of Deep Learning: Lecture 4 – PAC Learning and Deep Nets

Transcribed by Vishesh Jain (edited by Asad Lodhia and Elchanan Mossel)

PAC Learning

We begin by discussing (some variants of) the PAC (Probably Approximately Correct) learning model introduced by Leslie Valiant. Throughout this section, we will deal with a *hypothesis class* or *concept class*, denoted by $\mathcal{C}$; this is a space of functions $\mathcal{X} \to \mathcal{Y}$, where $\mathcal{X} = \{0,1\}^n$ for some $n \in \mathbb{N}$ and $\mathcal{Y}$ is some finite set.

As a concrete example, we could consider formalizing the task of classifying whether a 16×16 bitmap depicts a cat, a dog, or neither, in which case we could take n to be 256, $\mathcal{Y}$ to be $\{c, d, n\}$ (where c denotes "Cat", d denotes "Dog" and n denotes "Neither"), and the concept class $\mathcal{C}$ to consist of a set of "classifiers" i.e. a subset of functions $\{0,1\}^{256} \to \{c, d, e\}$.

Definition. The *proper* PAC learning task is the following: given $\mathcal{X}, \mathcal{Y}, \mathcal{C}$, and a fixed but arbitrary distribution $\mathcal{D}$ on $\mathcal{X}$, suppose we are given samples $\{(x_i, h(x_i))\}_{i=1}^m$, where $h \in \mathcal{C}$ and x_i are i.i.d. samples generated according to the distribution $\mathcal{D}$. Then, we say that an algorithm A, which has access to the samples $\{(x_i, h(x_i)\}_{i=1}^m$, $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{C}$ but no access to $\mathcal{D}$, is a *proper PAC learner for the hypothesis h under the distribution $\mathcal{D}$* if for all $\epsilon, \delta > 0$, the algorithm A (given ϵ, δ as additional inputs) runs in time $\text{poly}(n, \frac{1}{\epsilon}, \frac{1}{\delta})$ and requires $\text{poly}(n, \frac{1}{\epsilon}, \frac{1}{\delta})$ many samples to return a classifier $\tilde{h} \in \mathcal{C}$ such that with probability at least $1 - \delta$ over the examples

$$P_{x\sim\mathcal{D}}[\tilde{h}(x) \neq h(x)] \leq \epsilon$$

Finally, we say that the algorithm A is a *proper PAC learner for the concept class* $\mathcal{C}$ if for every $h \in \mathcal{C}$ and for every distribution $\mathcal{D}$ on $\mathcal{X}$, the above holds. Note that the polynomials in the above definition are *not* allowed to depend on the distribution $\mathcal{D}$ or the "correct" hypothesis h.

Remark. A slight weakening of the above notion is that of an *improper PAC learner*, under which the $\tilde{h}$ returned by the algorithm A need not belong to the concept class $\mathcal{C}$ (but all of the other properties of a PAC learner must still hold).

Before proceeding further, we make a few remarks on the assumptions underlying the PAC learning model. For many applications, the assumption that our samples are i.i.d. samples from a common distribution $\mathcal{D}$ can be very unrealistic. For instance, if our sample consists of images generated by taking snapshots of a video at a fixed frequency, then we expect to have some correlation between samples. On the other hand, the requirement that a PAC learner should perform well under *any* distribution on the domain space can be quite restrictive, and indeed, it is the case that proofs of theorems about non-PAC-learnability sometimes involve carefully constructing a "bad" distribution for a given algorithm. We also emphasize that the definition of PAC learnability that we have presented here seeks efficiency in terms of both time complexity as well as sample complexity. This is in contrast to statistical learning theory, where the focus is typically only on sample complexity.

Another notion, central to the definition of PAC learnability as presented above, is that of "realizability" i.e. the assumption that the data is generated by a function in our hypothesis class $\mathcal{C}$. This can obviously fail to hold in many cases of interest, for instance, when we choose a not-too-big hypothesis class $\mathcal{C}$ to avoid over-fitting. To deal with such situations we introduce the stronger notion of *agnostic PAC learning*.

Definition. The setup is the same as in proper PAC learning, except that instead of assuming that the data is of the form $(x_i, h(x_i))$ for some function $h \in \mathcal{C}$, we assume that the pairs (x_i, y_i) are generated by *some* distribution $\mathcal{D}$ on $\mathcal{X} \times \mathcal{Y}$. Note, in particular, that since the distribution is over $\mathcal{X} \times \mathcal{Y}$ and not just on $\mathcal{X}$, we are also allowing for situations when (x, y) and (x, y') can both have positive

mass for $y \neq y'$ so the data cannot be generated by any function from $\mathcal{X} \to \mathcal{Y}$. We define the "quality of optimal approximation of $\mathcal{D}$ by $\mathcal{C}$" by

$$E(\mathcal{D}, \mathcal{C}) := \inf_{h \in \mathcal{C}} P_{(x,y)\sim\mathcal{D}}[h(x) \neq y],$$

and we say that an algorithm A is an *agnostic PAC learner for a concept class* $\mathcal{C}$ if for all $\epsilon, \delta > 0$, the algorithm consumes $\text{poly}(n, \frac{1}{\epsilon}, \frac{1}{\delta})$ time and samples and, with probability at least $1 - \delta$ over the samples, returns $\tilde{h} \in \mathcal{C}$ such that

$$P_{(x,y)\sim\mathcal{D}}\left[\tilde{h}(x) \neq y\right] \leq E(\mathcal{D}, \mathcal{C}) + \epsilon$$

Remark. As before, we can relax agnostic PAC learning to *improper agnostic PAC learning*. Another relaxation is the notion of *α-agnostic PAC learning* for some $\alpha \geq 1$, where everything is exactly
as above, except we only require that

$$P_{(x,y)\sim\mathcal{D}}\left[\tilde{h}(x) \neq y\right] \leq \alpha E(\mathcal{D}, \mathcal{C}) + \epsilon$$

The general theory of PAC learning goes through with arbitrary domains $\mathcal{X}$ and arbitrary codomains $\mathcal{Y}$. For example, one could talk about the PAC learnability of a concept class $\mathcal{C}$ of functions $\mathbb{R}^n \to \mathbb{R}$. One of the things that changes when going from the discrete to the non-discrete setting is that the "loss" $P_{(x,y)\sim\mathcal{D}}[h(x) \neq y]$ for $h \in \mathcal{C}$ becomes too stringent, and often not useful. Instead, it is much more common to replace it with $\mathbb{E}_{(x,y)\sim\mathcal{D}}[L(h(x), y)]$, where $L\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a choice of "loss function". For instance, when $\mathcal{Y} = \mathbb{R}^k$, common choices of L include the
L^1 loss, $L(y_1, y_2) = ||y_1 - y_2||_{L^1}$ and the L^2 loss, $L(y_1, y_2) = ||y_1 - y_2||_{L^2}^2$.

We conclude this section with an example-sketch of how basic results in PAC learning are proved. Let $\mathcal{X} = \mathbb{S}^{n-1} \subset \mathbb{R}^n$ be the
unit sphere in $\mathbb{R}^n$ and $\mathcal{Y} = [0, 1]$. Consider the concept class $\mathcal{C}$ consisting of functions $\mathbb{S}^{n-1} \to [0, 1]$ of the form

$$h_w(x) = \sum_{i=1}^{n} w_i x_i \qquad \text{with} \qquad w \in \mathbb{S}^{n-1} \subset \mathbb{R}^n.$$

Finally, the loss function L is the L^2 loss (or in this case, the quadratic loss) given by $L(y_1, y_2) = (y_1 - y_2)^2$. To show that $\mathcal{C}$ is agnostically PAC learnable, it

suffices to exhibit an agnostic PAC learning algorithm A for C. We claim that the empirical risk minimizer, given by

$$\tilde{h} = h_{w^*} = \sum_{i=1}^{n} w_i^* x_i$$

for $w^* = \text{argmin}_{w \in \mathbb{S}^{n-1}} \sum_{i=1}^{m} L(h_w(x_i), y_i)$ (where m is the number of samples) is such a learner. There are two steps involved in showing this: first, we note that $w^* = \arg\min_w \sum_{i=1}^{m} (h_w(x_i) - y_i)^2$ has a well-known formula which can be computed in $\text{poly}(m)$ time; second, assuming that $\{(x_i, y_i)\}_{i=1}^{m}$ are i.i.d. samples from some distribution $\mathcal{D}$ on $\mathbb{S}^{n-1} \times [0, 1]$, it follows from a standard concentration of measure argument that for $m = \text{poly}(n, \frac{1}{\epsilon}, \frac{1}{\delta})$ samples, the empirical loss $\frac{1}{m} \sum_{i=1}^{m} (\tilde{h}(x_i) - y_i)^2$ will be $\frac{\epsilon}{4}$ close to the true loss $\mathbb{E}_{(x,y)\sim\mathcal{D}}[L(\tilde{h}(x), y)]$ with probability at least $1 - \delta$ over the samples, and one can use this, along with the compactness of $\mathbb{S}^{n-1} \times [0, 1]$ to show that $\mathbb{E}_{(x,y)\sim\mathcal{D}}[L(\tilde{h}(x), y)] \leq \inf_{h \in C} \mathbb{E}_{(x,y)\sim\mathcal{D}}[L(h(x), y)] + \epsilon$.

1. PAC Learnable Net Models

In this section, we intend to give a flavour of PAC learning results for feedforward neural networks. This is a very young and active area of research, and most theoretical results are negative. However, in this section, our focus will be on positive results, following Livni, Shalev-Shwartz and Shamir. Using the same notation as in the previous section, we work in the setting where $\mathcal{X} = \{0, 1\}^d$ and $\mathcal{Y} = \{\pm 1\}$. Our concept classes will be feedforward neural networks depending on a few parameters, and we will denote them by $C_{t,n,\sigma,L}$: t denotes the depth of the network, n denotes its size, σ denotes the activation function, and L denotes a constant such that for each neuron in the network, we have $||w||_1 + |b| \leq L$, where w denotes the vector of weights input to the neuron, and b denotes the bias added by the neuron. Note that the function computed by such a network is real valued—we convert it to a $\{\pm 1\}$ valued function by taking its sign. Moreover, in the learning theorems below, we will make both the "realisability" assumption as well as the "margin" assumption i.e. we assume that our learner receives samples of the form $(x, \text{sign}(h^*(x)))$,

where h^* is some (real-valued) function belonging to the class $\mathcal{C}$, and further, $|h^*(x)| \geq 1$ for all $x \in \mathcal{X}$. Finally, we will consider improper learners.

Proposition. *[LSS, '14] Constant-depth polynomial networks (i.e. the activation function is $\sigma(y) = y^2$) are learnable in $\mathrm{poly}(d)$ time under the above assumptions.*

Indeed, a polynomial network of depth ℓ always computes a polynomial function in d variables of degree at most 2^ℓ. We can view such polynomials as living in a linear space of dimension k, where $k \leq (d+1)^{2^\ell}$ denotes the number of monomials of degree at most 2^ℓ in d variables. Since ℓ is constant, 2^ℓ is also constant. Hence, we are reduced to learning a linear function on $\mathrm{poly}(d)$ variables, which can be done in $\mathrm{poly}(d)$ time (for instance, when a convex loss function is used) using standard learning algorithms.

By an argument involving the approximation of the sigmoid function by Chebyshev polynomials, the previous proposition can be used to show the following:

Theorem. *[LSS, '14] The concept class $\mathcal{C}_{2,n,\sigma_{\mathrm{sigmoid}},L}$ can be learned to accuracy ϵ with probability at least 0.99 over the samples in time $\mathrm{poly}(T)$, where $T = \frac{1}{\epsilon} O(d^{4L\ln(11L^2+1)})$.*

We conclude this brief section by mentioning a recent learning result due to Goel, Kanade, Klivans and Thaler for feedforward neural networks with ReLU gates. In this theorem, we take $\mathcal{X} = \mathbb{S}^{d-1}$ and $\mathcal{Y} = [0,1]$. We mention that the theorem actually holds for a stronger notion called *reliable learnability*. We also note that the $2^{O(1/\epsilon)}$ factor appearing below precludes this from being a statement about (efficient) PAC-learnability, according to the definition in this lecture.

Theorem. *[GKKT, '16] Let $\mathcal{C} = \{x \mapsto \max(0, w \cdot x), ||w||_2 \leq 1\}$. Then, there is a learning algorithm A that reliably learns $\mathcal{C}$ in time $2^{O(1/\epsilon)} \cdot d^{O(1)}$.*

2. Hardness of Learning

We now discuss results related to the hardness of learning. In general, hardness results state that it is *algorithmically* hard to find a solution to a problem, even

though it is information theoretically possible to do so. Hardness of learning results are often based on hardness assumptions coming from cryptography or computational complexity. For instance, a typical hardness result might state that if one can learn a class of functions, then one can break a cryptosystem, or one can solve some canonical algorithmic problem (which is believed to be hard). An early result in this direction is the following proposition due to Kearns and Valiant:

Proposition. *[KV, '94] Consider a public-key cryptosystem for encrypting individual bits by n-bit strings. Let $\mathcal{C}$ be a concept class that contains all the decryption functions $d_K : \{0,1\}^n \to \{0,1\}$ of the cryptosystem, one for each choice of key $K = (K_{priv}, K_{pub})$. Let $\epsilon(n) = P_{K,r}[d_K(e_{K,r}(0)) \neq 0 \text{ or } d_K(e_{K,r}(1)) \neq 1]$ be the probability of decryption error (over the choice of keys and randomization in the encryption). If $\mathcal{C}$ is weakly PAC-learnable in time $t(n)$ with $t(n)\epsilon(n) = 1/n^{\omega(1)}$, then there is a distinguisher between the encryptions of 0 and 1 that runs in time $O(t(n))$.*

In the above result, we are considering public-key cryptography with key $K = (K_{priv}, K_{pub})$, randomized encryption (using the key K, and random bits r), and deterministic decryption (using the key K). Weak PAC-learnability relaxes the requirement that $P_{x\sim\mathcal{D}}[\tilde{h}(x) \neq h(x)] \leq \epsilon$ to $P_{x\sim\mathcal{D}}[\tilde{h}(x) \neq h(x)] < \frac{1}{2} - \frac{1}{n^c}$ for some constant c i.e. the learner performs non-negligibly better than random guessing. Finally, an algorithm A is said to distinguish between the encryptions of 0 and 1 if some some universal constant c

$$|P_{K,r}[A(K_{pub}, e_{K,r}(1)) = 1] - P_{K,r}[A(K_{pub}, e_{K,r}(0)) = 1]| \geq \frac{1}{n^c}$$

The intuition behind the Kearns-Valiant theorem is that, since encryptions are easy to generate, one can view a collection of encryptions as training examples for learning the decryption function. But since learning the decryption function to a non-negligible advantage amounts to breaking the cryptosystem, it follows that if the cryptosystem is secure, we cannot learn the decryption function. Our focus in this section will be a result of Klivans and Sherstrov which states that PAC learning the concept class $\mathcal{C}$ of 2-layer feedforward neural networks with the threshold activation function $\sigma(x) = \mathbf{1}_{x\geq 0}$ is hard under a certain cryptographic assumption. More precisely, we have the following:

Theorem. *[KS, '06] Assume that depth-2 polynomial-size circuits of threshold gates are PAC learnable. Then there is a polynomial-time solution to $\tilde{O}(n^{1.5}) - uSVP$.*

Here, $f(n) - uSVP$ refers to the unique shortest vector problem on an n-dimensional lattice, in which the goal is to find the shortest non-zero vector in a given lattice, provided that it is shorter by a factor of at least $f(n)$ than any other non-parallel vector. We take $f(n) = \tilde{O}(n^{1.5})$, an approximation factor for which this problem is believed to be hard. The theorem follows from the following result about the hardness of learning the intersection of half-spaces.

Theorem. *[KS, '06] Assume that intersections of n^λ halfspaces in n dimensions are PAC learnable for some constant $\lambda > o$. Then there is a polynomial-time solution to $\tilde{O}(n^{1.5}) - uSVP$.*

Indeed, this result implies the previous one, since half-spaces i.e. subsets of the form $a_1 x_1 + \cdots + a_n x_n - \theta \geq 0$ are easily implemented by threshold gates, and so are intersections. Moreover, by an argument due to Regev, the theorem continues to hold even when all the half-spaces involved are "light" i.e. $\sum_{i=1}^{n} |a_i| + |\theta|$ is bounded by some polynomial in n.

To prove the latter theorem, one first uses a simple linearization argument (like the one we used in the previous section) to show that if the intersections of n^λ halfspaces in n-dimensions are weakly PAC-learnable, then for any constant $c > 0$, the intersections of n^c degree-2 polynomial threshold functions (PTFs) i.e. functions of the form $\mathbf{1}_{p(x)\geq 0}$ where $p(x)$ is is a polynomial of degree at most 2, is also weakly PAC-learnable. Next, we show that the class of such intersections of degree-2 PTFs contains the decryption function of a cryptosystem whose security is based on the hardness $\tilde{O}(n^{1.5}) - uSVP$. By the Kearns-Valiant result, it follows that if this class is weakly PAC-learnable, the cryptosystem is insecure, and hence $\tilde{O}(n^{1.5}) - uSVP$ is solvable in polynomial time.

We now provide a description of this cryptosystem due to Regev. We begin by taking $N = 2^{8n^2}$ and $m = cn^2$ where c is a universal constant. The private key consists of a real number $H \in [\sqrt{N}, 2\sqrt{N}]$, and the public key consists of a vector $(A_1, \ldots, A_m, i_0)$ where $i_0 \in \{1, \ldots, m\}$, and each $A_i \in \{0, \ldots, N-1\}$. Moreover, the $A_i's$ are "almost" integer multiples of N/H and A_{i_0} is "almost" an odd integer multiple of N/H. To encrypt 0, we pick a random subset $S \subset [m]$ and output $\sum_{i\in S} A_i \mod N$; to encrypt 1, we pick a random subset $S \subset [m]$ and output $\lfloor \frac{A_{i_0}}{2} \rfloor + \sum_{i\in S} A_i \mod N$. The idea is that 0 will be encrypted to

something which is close to an integer multiple of N/H, whereas 1 will be encrypted to something which is close to an odd half-integer multiple of N/H. Accordingly, on receiving $W \in \{0, \dots, N-1\}$, the decryption system outputs 0 if $\{WH/N\} < \frac{1}{4}$ and outputs 1 otherwise. Here, $\{a\}$ for $a \in \mathbb{R}$ denotes the minimum distance between a and an integer, where the distance between two real numbers is the absolute value of their difference. Moreover, by a standard argument, we may use the decryption system $\{AW\}$, where A is a representation of H/N to within $\mathrm{poly}(n)$ fractional bits. The only thing that remains to be shown is that the decryption system for this cryptosystem can be implemented using the intersection of n^β degree-2 PTFs for some constant $\beta > 0$. We have already remarked that we can use a decryption function of the form $\{AW\}$ where A has $k = \mathrm{poly}(n)$ fractional bits. Moreover, since W is integral, we can ignore the integral part of A. Denoting $A =. b_1 b_2 \dots b_k$ and $W = \sum_{j=0}^{\ell-1} x_j 2^j$ with $\ell \leq 8n^2$, we have

$$\begin{aligned}\{AW\} &= \{\sum_{i=1}^{k}\sum_{j=0}^{\ell-1} b_i x_j 2^{j-i}\} \\ &= \{\sum_{i=1}^{k}\sum_{j=0}^{\min\{\ell-1,i-1\}} b_i x_j 2^{j-i}\}\end{aligned}$$

since we can discard those terms $b_i x_j 2^{j-i}$ which are integers. Denoting $S(x) = \sum_{i=1}^{k}\sum_{j=0}^{\min\{\ell-1,i-1\}} b_i x_j 2^{j-i}$, we observe that $S(x)$ is an integral multiple of 2^{-k} which lies between 0 and k. Let $\{[a_u, b_u]\}$ denote the collections of intervals in $[0, k]$ which are not within $1/4$ of an integer. Note that there are k such intervals, and that each interval $[a, b]$ can be recognized by the PTF $\left(S(x) - \frac{a+b}{2}\right)^2 \leq \left(\frac{b-a}{2}\right)^2$. Hence, the negation of such an interval can be recognized by $\left(S(x) - \frac{a+b}{2}\right)^2 > \left(\frac{b-a}{2}\right)^2$, and by taking the intersection of these k degree-2 PTFs, we obtain the region of $[0, k]$ which is within $1/4$ of an integer, which is precisely the region where the decryption system outputs 0.

To conclude, we mention some other hardness of learning results in the literature. Assuming that there is no efficient algorithm that never refutes a satisfiable random 3-SAT formula with $f(n) = n \log^2 n$ clauses, but refutes most such random formulae, Daniely and Shalev-Shwartz proved that learning the intersection of $\log^3 n$ half-spaces is hard. Daniely also proved that

agnostically learning a single half-space is hard, assuming that a related computational problem is hard on average. Along the lines of our comments about PAC learning, we note that results all show that there are *some* distributions and *some* functions which are hard to learn. It would be interesting to obtain hardness results for an *a priori* fixed, natural choice of distribution.

elmos / May 8, 2017