11/13/23

Lecture 19: Cryptography + Learning

① Hardness of LPN → Hardness of agnostically learning halfspaces

② Daniely-Vardi lifting: Crypto hardness for learning MLP's.

---

① Recall agnostic learning: for function class $\mathcal{C}$,

Given: $(x_1, y_1), \ldots, (x_n, y_n) \sim \mathcal{D}$ over $\mathbb{R}^d \times \{\pm 1\}$

Goal: output $f \in \mathcal{C}$ s.t.

$$\Pr_{x,y}[f(x) \neq y] \leq \min_{f^* \in \mathcal{C}} \Pr_{x,y}[f^*(x) \neq y]$$

When $\mathcal{D}_x = \mathrm{Unif}(\{0,1\}^d)$ and $\mathcal{C}$ is {halfspaces}, how hard is this task?

[Kalai-Klivans-Mansour-Nisan '06]:

A) Halfspaces approx'd by deg poly$(1/\varepsilon)$ polynomials, so can learn in time $d^{\mathrm{poly}(1/\varepsilon)}$.

B) This is qualitatively tight, assuming hardness of LPN

Pf of B) :

Given $S \subseteq [d]$ of even size, note that

$$\mathrm{Maj}_S(x) = \begin{cases} 1 & \text{if } \sum_{i\in S} x_i \ge \frac{|S|}{2} \\ -1 & \text{o.w} \end{cases}$$

agrees with $\mathrm{parity}_S(x) = \begin{cases} 1 & \text{if } \sum_{i\in S} x_i \text{ even} \\ -1 & \text{o.w.} \end{cases}$

w.p. $\approx \frac{1}{2} \pm \frac{1}{\sqrt{|S|}}$

$\Pr\left[\sum_{i\in S} x_i = \frac{|S|}{2}\right] \approx \frac{1}{\sqrt{|S|}}$

	agree $\frac{1}{2}$ the time	agree	agree $\frac{1}{2}$ the time
Maj	-1 .... -1 -1 -1	1	1 1 1 ---- 1
$(-1)^{\frac{\lvert S\rvert}{2}}$ Parity	-1 1 -1	1	-1 1 -1
Hamming weight $\sum_{i\in S} x_i$	$0, 2,$	$\frac{\lvert S\rvert}{2}$	........, $d-2, d$

So majority and parity have correlation $\frac{1}{\sqrt{|S|}}$.

$\Rightarrow$ noise in parity reduces this correlation to $\frac{1-2\eta}{\sqrt{|S|}} \geq \Omega(1/\sqrt{n})$,

So if we could agnostically learn majorities (special case of halfspaces) to error $\varepsilon = \frac{1}{\sqrt{n}}$, would get alg. for noisy parity.

e.g. if alg for the former ran in time $n^{O(1/\varepsilon^{2-\beta})}$, would imply $2^{O(n^{\gamma})}$ alg for noisy parity.

---

② <u>Daniely - Vardi lifting</u>:

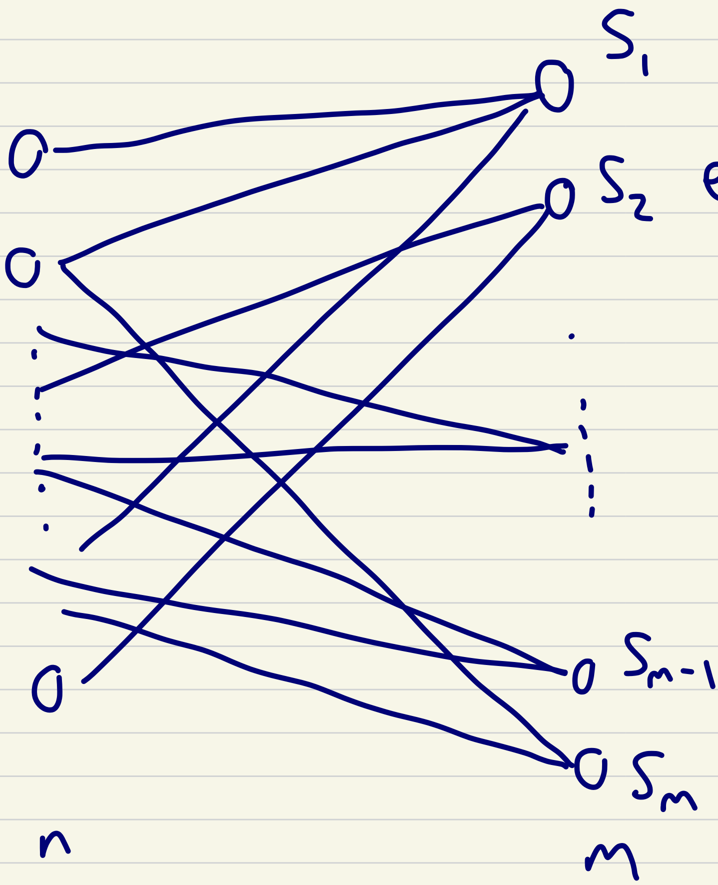
For $n << m$, a function $F: \{0,1\}^n \to \{0,1\}^m$ is a pseudorandom generator if no poly-time adversary can distinguish a sample from $\mathrm{Unif}(\{0,1\}^m)$ from a sample from $F(\mathrm{Unif}(\{0,1\}^n))$ with non-negl. success prob.

Goldreich's PRG:

Let $P: \{0,1\}^k \to \{0,1\}$ be a predicate, e.g.

$$P(z) = \mathrm{XOR\text{-}MAJ}_{a,b}(z) = (z_1 \oplus \cdots \oplus z_a) \oplus \mathrm{Maj}(z_{a+1}, \ldots, z_{a+b}).$$

For some constant k, sample m random subsets $S_1, \ldots, S_m$ of $[n]$ of size k



each vertex on left connected to a random subset

Define

$$F(z) = \Big(P(\underbrace{z|_{S_1}}_{\substack{\text{restriction} \\ \text{of } x \in \{0,1\}^n \\ \text{to bits in } S_1}}), \ldots\ldots, P(z|_{S_m}) \Big)$$

"Goldreich's PRG" / "Local PRG"

Crypto assumption: For every constant $s > 1$, there is a constant k and predicate $P: \{0,1\}^k \to \{0,1\}$ s.t. that Goldreich's construction is a valid pseudorandom generator.

We will use this to prove hardness of learning MLP's.

Strategy:

(I) Hardness over $\{0,1\}^n$

(II) Naive lifting

(III) Daniely-Vardi gadget

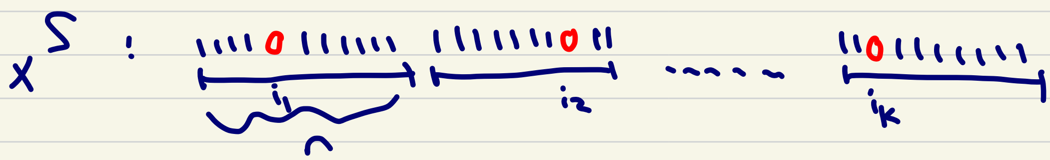
(I) Let's first show that under the above assumption, MLPs are hard to learn over $\{0,1\}^n$.

If we have a sample $F(z)$ from PRG, can regard it as a dataset of pairs

$$(S_i, P(z|_{S_i}))$$

each S_i is a random subset of size k, and we want to encode this into a sample from $\{0,1\}^n$

Given $S=\{i_1,\dots,i_k\}$, define $x^S \in \{0,1\}^{kn}$ via:



i.e. jth block of ^{S} is e_{i_j}

Claim: $\exists$ MLP $N:\{0,1\}^{kn}\to\{0,1\}$ s.t.

$$N(x^S) = P(z|_S).$$

Pf: (tedious, included for completeness):

the function $x^S \mapsto P(z|_S)$ can be implemented as a DNF:

$$\bigvee_{\substack{b\in\{0,1\}^k \\ \text{s.t. } P(b)=1}} \underbrace{\bigwedge_{j\in[k]} \bigwedge_{\ell: z_\ell \neq b_j}}_{} x_{j,\ell}$$

Can implement as a relu: if there are M literals in this conjunction, then take

$$\mathrm{ReLU}\Big(\sum_{j\in[k]} \sum_{\ell : z_\ell \neq b_j} x_{j,\ell} - (M-1) \Big)$$

note: at most one conjunction satisfied, so can implement by simply summing the neurons:

$$\sum_{\substack{b\in\{0,1\}^k : \\ P(b)=1}} \mathrm{ReLU}(\cdots\cdots) \qquad \square$$

Implies learning one-hidden-layer MLPs over the distribution over strings

$$x^S : \underbrace{\underbrace{1111\,0\,111111}_{i_1}}_{n}\ \ \underbrace{11111111\,0\,11}_{i_2}\ \cdots\cdots\ \underbrace{11\,0\,11111111}_{i_k} \in \{0,1\}^{kn}$$

is hard (if learner achieves nontrivial test error we know that we are in the pseudorandom scenario and can distinguish).

(II) Naive lifting

Now want to show hardness over Gaussian inputs.

Initial idea:

$$\text{dist. over } \underbrace{\underbrace{1111\,0\,111111}_{i}}_{n} \approx \mathrm{Ber}\Big(\frac{n-1}{n}\Big)^n$$

So given sample x^S, can "Gaussianize" as follows:

if in block j, ith entry of x^S is

0, then draw $g^S_{j,i} \sim N(0,1) \mid \geq t$

1, then draw $g^S_{j,i} \sim N(0,1) \mid < t$

for t s.t. $\Pr_{g \sim N(0,1)}[g \geq t] = \frac{1}{n}$, so that

if we apply $\boxed{\mathrm{thres}(g) \triangleq \mathbb{1}[g \geq t]}$ to $g \sim N(0, \mathrm{Id}_{kn})$, we get a sample from $\mathrm{Ber}\left(\frac{n-1}{n}\right)^{\otimes kn}$.

Naive attempt: define N' by

$$N'(g) = N(\mathrm{thres}(g))$$

(applied entrywise)

Fails for several reasons, one of which is that g need not encode a subset

Given $(S_1, P(z|_{S_1})), \ldots, (S_m, P(z|_{S_m}))$, define

Gaussian dataset: for every $i \in [m]$,

- sample $g \sim N(0, \mathrm{Id}_{kn})$
- if $\mathrm{thres}(g)$ is valid encoding of a subset S

(i.e. has exactly one 0 in each block);

- permute its entries so that thres(g) encodes S_i
- add example $(g, P(z|_{S_1}))$ to the dataset

- otherwise, add $(g, 0)$ to the dataset

<u>Claim</u>: $\exists$ MLP N_{naive} that labels this dataset perfectly (caveat, requires Sign activations)

<u>Pf</u>: First, note $\exists$ MLP N_{encode} s.t.

$$N_{encode}(g) = \begin{cases} 0 & \text{if thres}(g) \text{ is not valid encoding}, \\ \text{large} & \text{o.w.} \end{cases}$$

Take

$$N_{encode}(g) \triangleq \underset{\substack{\uparrow \\ \text{some big factor}}}{C} \sum_{j \in [k]} \sum_{i \in \text{block } j} \left(\text{thres}(g)_{j,i} - (n-2) \right).$$

(this solves problem of g not necessarily encoding a subset)

So $N_{naive}(g) \triangleq \text{ReLU}\left(N(\text{thres}(g)) - N_{encode}(g) \right)$

correctly labels the Gaussian dataset. $\square$

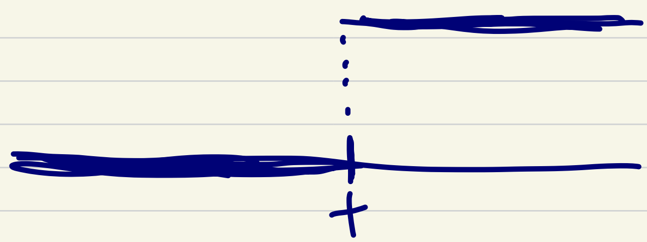
One more (major) issue: thres is a discontinuous function!

would need infinite weights to implement w/ ReLUs, or super-poly-sized weights to approximate sufficiently well...

III Daniely-Vardi lifting:

Instead of

thres:



,

Consider

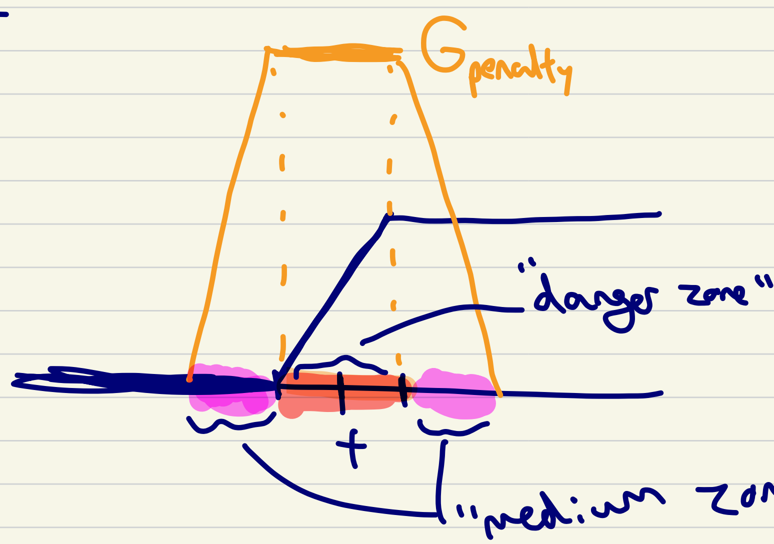
ramp:

Let N'_{encode} be N_{encode} with thres $\to$ ramp.

because we can only afford to use

poly-sized weights, the intermediate interval is $\geq \frac{1}{\text{poly}(n)}$ wide and we will see some g's w/ coordinates landing in the interval, and replacing thres w/ ramp in naive lifting would fail.

Key idea: consider



$G_{\text{penalty}}(x)$ is large whenever g has a coordinate landing in "danger zone".

Consider

$$N_{\text{final}} = \text{ReLU}\Big(N(\text{ramp}(g)) - N'_{\text{encode}}(g) - \sum_{j \in [k], i \in [n]} G_{\text{penalty}}(g_{j,i}) \Big)$$

What happens if g falls in "medium zone"?

Idea: in our Gaussian dataset, modify the labels as follows:

- if all coordinates of g outside of danger and medium zones, keep label as before, i.e. $P(z|_{s_i})$
- if $\exists$ coordinate in danger zone, set label to 0
- if no coordinate in danger zone but some in medium zone, set label to

$$\mathrm{ReLU}\left(P(z|_{s_i}) - \sum_{j,i} G_{penalty}(g_{j,i})\right)$$