Lecture 10: Computational complexity of Deep Learning

From last lecture, we saw that using ERM estimator, we can learn any Lipschitz low-dim model efficiently.
However, when restricting to estimators that are efficiently learned by SGD, we saw that some low-dim fct are harder to learn and require more steps/sample (for online SGD)

⇒ there seem to be a computational bottleneck for learning large-leap fcts. Is it fundamental?

This lecture is a very short overview of computational hardness in DL.

* Present computational/learning hardness for NNs
* Conversely, what can NNs learn efficiently?

E.g. by changing hyperparameters, SGD-trained NNs can emulate

→ kernel methods: any problem efficiently learnable by kernel methods can be learned by SGD-trained NNs (with right hyperparam)

→ support recovery for small leap low-dim fcts

↳ What other problems can be learned by SGD-trained efficiently? Are there "easy problems" on which they fail?

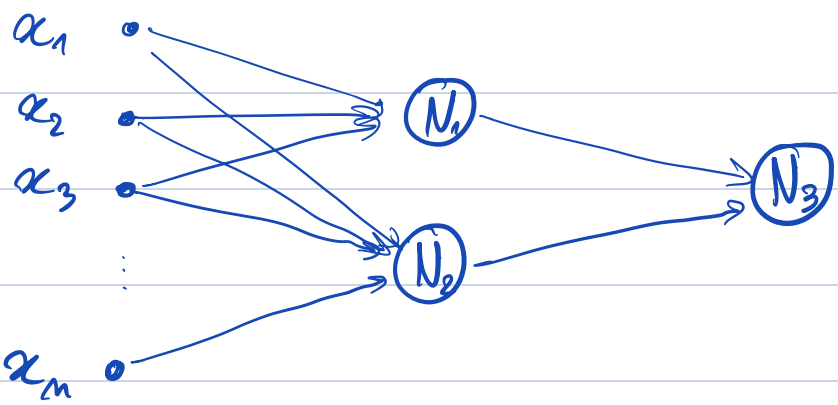
Computational Hardness of NNs

(A) Empirical risk minimization over NNs is NP-complete

First result: hardness result for training NNs

→ "we can't hope to have an algo that unconditionally find an (approximate) minimizer of ERM over NNs"
(under general hardness conjecture)

Training 3-Node network [Blum, Rivest, 92]
(with many follow-up work)



"2 hidden layer with 3 neurons"

$$N_i: \quad \varphi_i(x) = \text{sign}\left(\langle w_i, x \rangle + b_i\right)$$

Q*: Given a set of $O(d)$ examples (x_i, y_i) with $x_i \in \{0,1\}^d$ and $y_i \in \{\pm 1\}$. Does there exist a 3-neuron network that has 0 training error?

Thm [Blum, Rivest, '92] Training 3-neuron NN is NP-complete

This means that if $P \neq NP$, no polynomial time algo solve Q^* for any (x_i, y_i) in poly(d) runtime.

Rmk: [Some quick definitions]

- "P": problems that can be solved by a deterministic turing machine in polynomial time
- "NP": problems that can be solved by a nondeterministic Turing machine (transition fct is not deterministic) in polynomial time
- "Polynomial-time reduction": We say that a problem A is as hard a problem B, if there exists a polynomial reduction from B to A. Hence, if I can solve A in polynomial time, then I can solve B in polynomial time. Conversely, if we expect B to be hard to solve, then we expect A to be hard too.

"NP complete": a problem A in NP such that any problem in NP is reducible to A. This means that A is as hard as any problem in NP.

If $P \neq NP$, NP-complete problems cannot be solved in polynomial time. ($P \stackrel{?}{=} NP$ is a major unsolved problem)

Proof idea: We can reduce Q^* to: given a set of $O(d)$ samples does there exist either

- an hyperplane separating ± 1
- two hyperplanes that split the space in 4 quadrants s.t. one quadrant contains all positive (resp negative) labels and no negative (resp positive) labels

$\tilde{Q}^*$

The second case is the "hard" case. We can show that we can reduce a NP complete problem to this problem $\longrightarrow$ making it NP-complete

Set-Splitting: Given S, a collection of subsets $\{C_i \mid C_i \subset S\}$

Does there exists S_1, S_2 with $S_1 \cap S_2 = \emptyset$ and $S_1 \cup S_2 = S$ such that $C_i \not\subset S_1$ and $C_i \not\subset S_2$ for all i

Thm [Lovasz (Garey & Johnson, '79)] Set-splitting is NP-complete.

We need to construct a reduction from set-splitting to $\mathcal{Q}^*$

Given an instance of set-splitting:

$S = \{s_1, \dots, s_d\}$ $\quad C = (C_1, C_2, \dots)$ $\quad C_j \subseteq S$

we convert it to the following instance of $\mathcal{Q}^*$

- Let the origin $(\overbrace{0, 0 \dots 0}^{d})$ have label $+1$
- for each s_i, we make a point $x_i = (0, \dots, \overset{i\text{-th}}{1}, \dots 0)$ and label it $y_i = -1$
- for all $C_j = (s_{j1}, \dots, s_{jk})$, we put a point $x_{C_j} = (0 \dots 0\ 1, 0\ 1 \dots 1\ 0)$ $\quad y_{C_j} = +1$ (1's at the s_{ji})

E.g.

$S = (s_1, s_2, s_3)$ $\quad C_1 = (s_1, s_2)$ $\quad C_2 = (s_2, s_3)$

-1 at	$(1\,0\,0)$	$(0\,1\,0)$	$(0\,0\,1)$
$+1$ at	$(0\,0\,0)$	$(1\,1\,0)$	$(0\,1\,1)$

(Rmk: we can always append 0's s.t. # pts is $O(d)$)

Prop: This set of pts is separable by a quadrant iff set-splitting has a solution

Proof: $\Leftarrow$ If set-splitting has $S_1 \cup S_2 = S$, $S_1 \cap S_2 = \emptyset$ s.t. $c_i \notin S_1$ and $c_i \notin S_2$ for all i:

Then

consider P_1, P_2 s.t. $P_j : w_{j1} x_1 + \dots + w_{jd} x_d + \frac{1}{2} \geq 0$

s.t. $$w_{ji} = \begin{cases} -1 & \text{if } s_i \in S_j \\ 0 & \text{o.w.} \end{cases}$$

Then:

P_j predicts
- $+$ for $(0, \dots, 0)$
- $+$ for training pts with $+$
- $-$ for x_i if $s_i \in S_j$

Intersection $P_1 \geq 0$ $P_2 \geq 0$ (quadrant) contains all the pts with $+1$ and none of -1 labels.

$\Rightarrow$ Let S_1 be the set of points -1 that are separated from $(0,\dots,0)$ by P_1.
Let S_2 be the points -1 not in S_1 that are separated from $(0,\dots,0)$ by P_2.

By assumption $P_1 \geq 0 \cap P_2 \geq 0$ contain all $+1$ labels and none of -1

so $S_1 \cup S_2 = S$ & $S_1 \cap S_2 = \emptyset$ by construction.

It remains to show that $C_j \not\subseteq S_1, S_2$.

$C_j = \{s_{j1}, \dots, s_{jk}\}$ and points $x_{j1}, \dots, x_{jk}$ label -1

$x_{C_j} = x_{j1} + \dots + x_{jk}$ label $+1$

If $C_j \subseteq S_1$ then P_1 separate $(0,\dots,0)$ from $x_{j1}, \dots, x_{jk}$

and in particular it separates $(0,\dots,0)$ from x_{C_j}

contradiction

Same for $C_j \subseteq S_2$. $\square$

$\rightarrow$ Hence if can solve $\widetilde{Q}^*$ we can solve set splitting!

Is this a compelling lower bound?

(should we give up on Deep Learning? :))

For this result:
* "worst case" over training data points (need to solve given any collect° (x_i, y_i))
* The architecture is fixed: it is NP hard to decide whether there exist weights that perfectly fit the data.

In Deep Learning (and machine learning in general), we care instead about: given training data, can we construct a model that predicts well new data point.

In learning theory, this is described as the difference between

proper vs improper learning

Hypothesis class: $\mathcal{H} \subseteq \{h : \mathcal{X} \to \{-1, +1\}$

e.g. the set of all 3-neuron neural networks.

def: [Proper learning] Given data $(x_i, y_i) \sim \mathcal{D}$ where $y_i = h(x_i)$ with $h \in \mathcal{H}$, the algorithm outputs $\hat{h} \in \mathcal{H}$ s.t.

$$\text{Test error}_{\mathcal{D}}(\hat{h}) = \mathbb{P}(\hat{h}(x) \neq h(x)) \leq \varepsilon$$

def [Improper learning] The algorithm is allowed to output $\hat{h} \notin \mathcal{H}$ (e.g. in a much broader class of models)

s.t. $\text{Test error}_{\mathcal{D}}(\hat{h}) \leq \varepsilon$

Improper learners are allowed to use a much richer / more expressive hypothesis class even if data comes from a simpler class

> Hardness of proper learning does not imply hardness of improper learning

Here is an old counterexample:

Def: [k-term DNF] (Disjunctive normal form)

It is the set of boolean formula

$$\bigvee_{i=1}^{k} \left(\bigwedge_{j=1}^{n_k} L_{ij} \right)$$

(L_{ij} is either X_j or $\overline{X_j}$ $\in \{0,1\}$)

Thm: [Pitt, Valliant] Under a general computational hardness assumption, there is no polynomial time algo for learning properly k-term DNFs for $k \geq 3$

Conversly:

Thm [Valliant] There is a polynomial time algo that learns improperly k-term DNFs.

Here, use k-CNF (conjunctive normal form)

$$\bigwedge_{i=1}^{n} (L_{i_1} \vee L_{i_2} \vee \dots \vee L_{i_k})$$

We have k-CNFs $\neq$ k-DNFs

E.g. 3 DNFs can always be written a 3 CNFs

$$T_1 \vee T_2 \vee T_3 = \bigwedge_{\substack{L_1 \in T_1 \\ L_2 \in T_2 \; L_3 \in T_3}} (L_1 \vee L_2 \vee L_3)$$

(Convince yourself)

there are 3 CNFs not of this form.

→ 3 CNFs can be learned improperly efficiently

↳ algo that always outputs a 3 CNFs that perfectly fits the training data points:

* start will all 3-sized conjunctions ($\binom{2d}{3}$ of them) and remove all that don't agree with training data
* it generalizes because $|H_{3\text{-CNF}}| \leq 2^{8d^3}$ and

standard generalization error bound:

$$|\text{error}_D(\hat{h}) - \text{empirical error}(\hat{h})| \leq \varepsilon \quad \text{w.p } 1-\delta$$

when sample size $n \geq \frac{1}{2\varepsilon^2}\left(\ln(2|H|) + \ln\left(\frac{1}{\delta}\right)\right)$

[This is "Uniform convergence" bound simply using tail bound + union bound]

Summary: don't trust hardness results for proper learning!

→ one can enrich the model class and learn improperly efficiently

→ this is what we do in practice with NNs: we take highly overparametrized NNs (much richer than needed to fit the data) which seems to greatly simplify the ERM problem.

Caveat that we saw in lecture 2: by taking a much richer class of functions, we make generalization much harder

→ we might require way more samples

There is a fundamental "computational vs statistical trade-off" for proper vs improper learning.

Question: if proper learning hardness is not interesting, can we show hardness of improperly learning NNs?

(B) Hardness of improperly learning NNs

Klivans and Sherstov (2006) showed that for every $\varepsilon > 0$, it is hard to learn improperly 2-hidden layer neural networks with d^{ε} neurons, under a cryptographic hardness assumption.

[This was improved by [Daniely, Linial, Shalev-Shwartz 19] to $M = \omega_d(1)$ neurons under a different hardness assumpt°]

What does it mean "cryptographically hard": it's a problem that is not NP-hard, but people have built cryptosystems assuming the problem is hard (e.g. factoring)

Here: shortest vector problem (SVP) in a lattice

Lattice $= \{a_1 v_1 + \dots + a_n v_n : a_1, \dots, a_n \in \mathbb{Z}\}$

$f(n)$-SVP: approximate the length of the shortest non-zero vector within a factor $f(n)$.

Def: [Intersection of k-halfspaces] $x \in \{\pm 1\}^d$

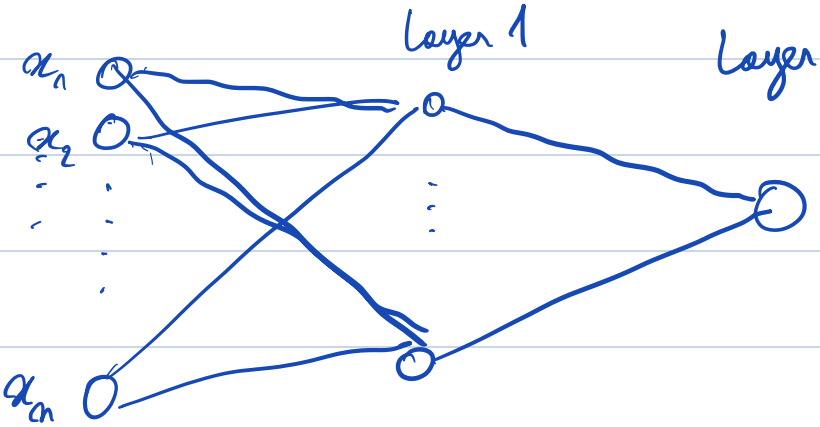
$w_j \in \{\pm 1\}^d$

$$h(x) = \prod_{i=1}^{k} \mathbb{1}[\langle w_j, x \rangle > 0]$$

$\mathcal{H}_{k\text{-halfspaces}}$ = the set of these fcts

Thm [Klivans, Sherstov, '06] Assuming $d^{\frac{3}{2}}$-SVP is hard, then for every $\varepsilon > 0$, $\mathcal{H}_{d^\varepsilon\text{-halfspaces}}$ cannot be learned efficiently, even improperly.

intersection of halfspaces can be written as a 2-hidden-layer NN with d^ε neurons and Relu non-linearity



For each half space $w_j \to 2$ neurons on layer 1

$\langle w_i, x \rangle_+$ and $(\langle w_i, x \rangle - 1)_+$

Then $\langle w_i, x \rangle_+ - (\langle w_i, x \rangle - 1)_+ = \begin{cases} 1 & \text{if } \langle w_i, x \rangle > 0 \\ 0 & \text{o.w.} \end{cases}$

Then in layer 2:

$$\left(\sum_{i=1}^{k} \langle w_i, x \rangle_+ - (\langle w_i, x \rangle - 1)_+ + 1 - k \right)_+$$

$$= \begin{cases} 1 & \text{iff } x \text{ is in the intersection} \\ 0 & \text{o.w.} \end{cases}$$

Summary: There exists distributions that can be fitted with d^ε neurons, that cannot be learned efficiently even if we take much larger neural networks

→ ERM will learn efficiently (approx = 0, gen $\leq \frac{\text{polylog}(d)}{\sqrt{n}}$)

→ No algo on neural nets will succeed in polynomial time

Is it a compelling hardness result?

→ better than before but still unsatisfying

To prove hardness in this case, they use a weird data distribution. Here is the high level idea:

In public-key encryption system:

take a message $m \in \{0,1\}^p \longrightarrow enc(m) \in \{0,1\}^q$

↳ public key: use to encrypt message

→ easy

given $x = enc(m)$ $\quad dec(x) = m$

↳ private key to decrypt the encrypted message

↳ computationally easy if you have private key

hard otherwise

Klivans & Sherstov used a cryptosystem (Regev) based on hardness of SVP. The decoding fct can be implemented as an intersection of d^2 halfspaces.

Idea: If hypothesis class H contains "dec" function, then it must be hard to improperly learn.

Otherwise an attacker can generate data

$$(x = \underbrace{enc(m)}_{\text{public key}}, \quad y = m = dec(m))$$

so can compute $enc(m)$ efficiently ↑ they sample messages

and if he can learn the decryption function even improperly, then it is able to predict the decryption better than random on a new encrypted message

↳ can break the security of this cryptoscheme.

So.... basically the above hardness result is about learning neural nets from data points that are encrypted messages ... not exactly a natural data distribution!!!

Does this hardness hold for more natural data distributions?

- $\mathrm{Unif}(\{\pm 1\}^d)$: there is $d^{O\left(\frac{\log k}{\varepsilon^2}\right)}$ time algo to learn intersection of half-spaces within error ε.
  ↳ in the following, a large class of algo fail to do better.

- Gaussian dist: [Vempala] in $\mathrm{poly}(d, k, \varepsilon) + k^{O\left(\frac{\log k}{\varepsilon^4}\right)}$

  so poly time even when $k = \frac{\log d}{\log \log d}$

[while Amit et al., $\exists$ dist such that impossible for $k = \omega_d(1)$]

How to prove computational lower bounds for "nice" distributions?

→ it is too much to ask for lower bounds based on NP-hardness or cryptographic hardness
→ instead, prove lower bounds for some "large" classes of algorithm

In this lecture, we will consider a powerful framework that has proven succesful to capture hardness in many settings

(C) Statistical Queries

Kearns (1998) introduced SQ as a framework to design algo that are tolerant to noise.
↳ it restricts algo access to data

In standard ("PAC") learning, an algo has access to n samples
In SQ model, algo only has access to (approximate) oracles on the data, and not directly the samples.

Specifically the algo specifies a query $\phi : \mathcal{Y} \times \mathcal{X} \to [-1,1]$ and a tolerance τ and it gets a response r s.t.

$$|r - \mathbb{E}_D[\phi(y,x)]| \leq \tau$$

(D: data dist)

$\Rightarrow$ this is a restriction of what algo can do.

Rmk: From n samples, then $\left|\hat{\mathbb{E}}_n[\phi(y,x)] - \mathbb{E}_D[\phi(y,x)]\right| \leq \frac{1}{\sqrt{n}}$

SQ algo with q queries of tolerance τ:

* proceeds in q rounds, at each round $t \in [q]$, algo issues a query ϕ_t and receive a response

$$|r_t - \mathbb{E}_D[\phi_t]| \leq \tau$$

ϕ_t is chosen adaptively, that is, can depend on past responses $v_1, \dots, v_{t-1}$

* After q rounds, algo output a model.

Rmk: any SQ algo with q queries of tolerance τ can be implemented using $O\left(\frac{\log q}{\tau^2}\right)$ samples

In particular, if a problem can be solved by an SQ algo

with 1) at most polynomially many queries
2) each tolerance at least inverse polynomially big
3) Can compute next query ϕ_t from $v_1, \dots, v_{t-1}$ and evaluate ϕ_t efficiently

Then there exists a learning algo that succeeds in polynomial time

In summary, SQ algo don't directly look at data but instead use approximate statistics of the data.

⚠ Lower bound: SQ "succeeds" if it succeeds for any compatible responses $v_1, \dots, v_q$

↳ hence lower bound is against "worst-case" noise
"adversarial"

↳ much more pessimistic than statistical noise

Nonetheless:
- Easy to prove lower bounds in this model
- Correctly capture lower bound in many settings

⇒ We can prove many tight lower/upper bounds within this framework.

Rmk: If a problem is efficiently SQ learnable, then it is efficiently PAC learnable even with a lot of noise in the labels
↳ SQ framework give us a way to design noise-tolerant algo

Correlation Statistical Queries (CSQ)

A restriction of SQ introduced by Bshouty and Feldman (2002)

$$\phi(y,x) = y\,\tilde{\phi}(x) \qquad \tilde{\phi}: \mathcal{X} \to [-1,1]$$

Rmk: If $y \in \{\pm 1\}$, then any statistical query can be decomposed as

$$\phi(y,x) = \phi(1,x)\frac{1+y}{2} + \phi(-1,x)\frac{1-y}{2}$$
$$= \underbrace{\frac{1}{2}\,y\,[\phi(1,x) - \phi(-1,x)]}_{\text{CSQ query}} + \underbrace{\frac{\phi(1,x)+\phi(-1,x)}{2}}_{\text{this does not depend on label}}$$

Therefore, if we know distribution over x (meaning P_X fixed), SQ = CSQ

However $|Y| > 2$, then SQ might be much more powerful than CSQ

[**Note:** In the distribution free setting SQ >> CSQ even if $|Y| = 2$]

For gradient descent w.r.t. squared loss

$$\hat{R}_n(w) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - f(x_i, w))^2$$

GD algo: $w \in \mathbb{R}^p$

$$w^{(k+1)} = w^{(k)} - \nabla \hat{R}_n(w^{(k)})$$

$$= w^{(k)} + \underbrace{\frac{1}{n} \sum_{i=1}^{n} y_i \nabla_w f(x_i, w^{(k)})}_{\text{vector of } p \text{ CSQ}} - \underbrace{\frac{1}{n} \sum_{i=1}^{n} f(x_i, w^{(k)}) \nabla_w f(x_i, w^{(k)})}_{\text{does not depend on the labels}}$$

GD with T steps can be implemented by CSQ with pT queries and tolerance $\approx \frac{1}{\sqrt{n}}$

⚠ Careful: if reusing the same data, the neural network can implement "sample extraction" and go beyond SQ (more later)

But if online SGD on squared loss, seems to agree with CSQ

→ if loss ≠ squared loss, not CSQ but still SQ

(See later for more discussion about online SGD)

CSQ "captures" GD on squared loss + is very convenient to prove lower bounds

Def [Statistical Query dimension] For a class of functions $\mathcal{H} \subseteq \{h : \mathcal{X} \to \mathbb{R}, \ \|h\|_{L^2} \leq 1\}$

The SQ-dim of $\mathcal{H}$ w.r.t. P_x is the largest integer N s.t. $\mathcal{H}$ contains N functions h_j s.t.

$$\|h_j\|_{L^2} = 1 \quad \text{and} \quad |\langle h_i, h_j \rangle| \leq \frac{1}{N}$$

High level: CSQ needs $\Omega(N\varepsilon^2)$ queries to learn $\mathcal{H}$ with accuracy ε
Let's prove a slightly more general result:

Thm: Let $\mathcal{H}$ be a class of fcts that contains N h_j s.t. $\|h_j\|_{L^2} = 1$ and $|\langle h_i, h_j \rangle| \leq \varepsilon$. Then any CSQ with tolerance ε, number of queries $q \leq N\frac{\varepsilon^2 - \varepsilon}{4}$ and $N \geq \frac{2}{\varepsilon}$ has

$$\sup_{h \in \mathcal{H}} \|\hat{h} - h\|_{L^2} \geq 1 - 2\varepsilon$$

($\hat{h}$: output of CSQ algo when true fct is h.)

Proof: Idea: with $q \leq N\frac{\varepsilon^2-\varepsilon}{4}$, we are not able to distinguish between all h_j

that is, each query allow to rule out some h_j's whose correlation is not ε-close. After q queries, we are not able to rule out a large fraction of them. Specifically, we will show that after $q \leq N\frac{\varepsilon^2-\varepsilon}{4}$ there are at least $k \geq \frac{N}{2}$ fits compatible with the answers.

$$\sup_{h \in H} \|\hat{h} - h\|_{L^2}^2 \geq \min_{\tilde{h}} \max_{i \in [k]} \|\tilde{h} - h_i\|_{L^2}^2$$

$$\geq \min_{\tilde{h}} \frac{1}{k} \sum_{i=1}^{k} \|\tilde{h} - h_i\|_{L^2}^2$$

$$= \frac{1}{k} \sum_{i=1}^{k} \|h_i - \frac{1}{k}\sum_{i} h_i\|_{L^2}^2$$

$$= \frac{1}{k} \sum_{i=1}^{k} \|h_i\|_{L^2}^2 - \frac{1}{k^2} \sum_{i=j}^{k} \langle h_i, h_j \rangle$$

$$= 1 - \frac{1}{k} - \frac{1}{k^2} \sum_{i \neq j} \langle h_i, h_j \rangle$$

$$\geq 1 - \frac{1}{k} - \varepsilon \geq 1 - \frac{2}{N} - \varepsilon$$

$$\geq 1 - 2\varepsilon$$

Now let's show $\geq \frac{N}{2}$ fcts are compatible with the responses.

Noise is adversarial, so it is enough to consider a deterministic sequence of answers that are compatible with $\geq \frac{N}{2}$ fcts

Consider $a_1 = \dots = a_q = 0$

and the associated sequence of queries $\phi_1, \dots \phi_q$

(fully determined by the responses)

For each $t \in [q]$, define

$$S_t^+ = \{ i \in [N] : \mathbb{E}[h_i(x)\phi_t(x)] > \tau \}$$

$$S_t^- = \{ i \in [N] : \mathbb{E}[h_i(x)\phi_t(x)] < -\tau \}$$

$S_t^+ \cup S_t^-$ are all the h_j that we are able to rule out at step t

$$N_t^+ = |S_t^+| \qquad N_t^- = |S_t^-|$$

By definition,

$$N_t^+ \tau \leq \sum_{j \in S_t^+} \langle h_j, \phi_t \rangle \leq \overbrace{\|\phi_t\|_{L^2}}^{\leq 1} \left\| \sum_{j \in S_t^+} h_j \right\|_{L^2}$$

$$\leq \sqrt{\sum_{j \in S_t^+} \|h_j\|_{L^2}^2 + \sum_{\substack{i \neq j \\ \in S_t^+}} \langle h_j, h_i \rangle}$$

So that $\left(N_t^+ z\right)^2 \leq N_t^+ + \left(N_t^+\right)^2 \varepsilon$

$$\Rightarrow N_t^+ \leq \frac{1}{z^2 - \varepsilon}$$

Some results holds for $N_t^- \leq \frac{1}{z^2-\varepsilon}$ so that

$$\sum_{t=1}^{q} N_t \leq \frac{2q}{z^2-\varepsilon}$$

Hence if $q \leq N \frac{z^2-\varepsilon}{2}$, then we rule out at most $\leq \frac{N}{2}$ fcts □

Applications

From above, to give lower bound, it is sufficient to find a family of nearly orthogonal functions.

(I) Hardness of learning parities

$$\mathcal{H} = \{ h(x) = \chi_S(x) : S \subseteq [d] \}$$

$$\chi_S(x) = \prod_{i \in S} x_i$$

$\mathcal{H}$ contains 2^d orthogonal fcts. They are all binary valued so that $CSQ = SQ$

Applying the above general lower bound, we get

Thm: [Blum, Furst, Kearns, Mansour, ...]

Any SQ algo for learning parities must make at least $2^{\Omega(d)}$ queries or have $\tau = 2^{-\Omega(d)}$

Rmk: Is this lower bound correctly capturing hardness of learning parities?

(1) There is a polynomial time algo for learning parities: it suffices to solve a linear system

in $\mathbb{F}_2 = \{\pm 1\}$ $\quad A c = y$

$A \in \mathbb{R}^{n \times d}$ $\quad \begin{bmatrix} - x_1 - \\ \vdots \\ - x_n - \end{bmatrix}$ $\quad c = 1_S$

But this algo is highly non-robust and will fail if there is noise in the label

$\neq$ SQ algo that is robust to noise.

(2) SQ correctly captures lower bound for noisy problem?

→ not quite, but almost?

[Blum, Kalai, Wasserman] There is a $2^{\tilde{O}(\frac{d}{\log d})}$ time algo for learning noisy parities.

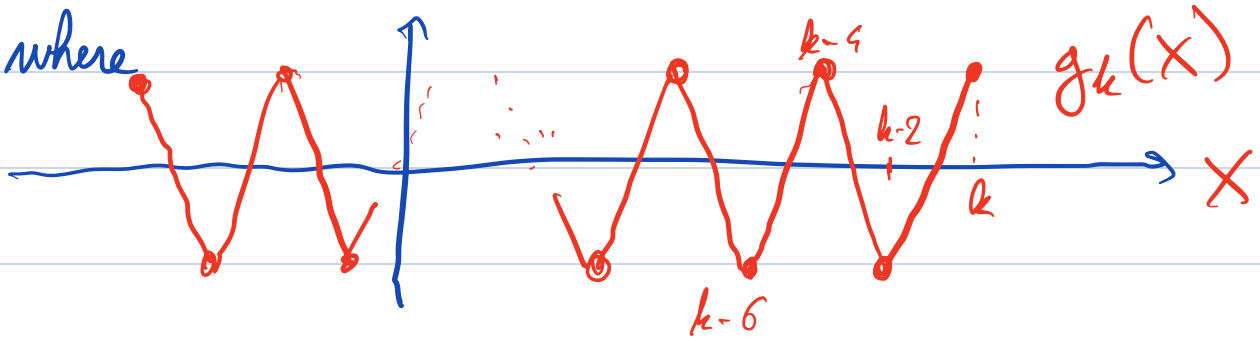
(↳ noise: flip label with prob $\frac{1}{3}$)

→ this is slightly better than $2^{\Omega(d)}$ in the lower bound

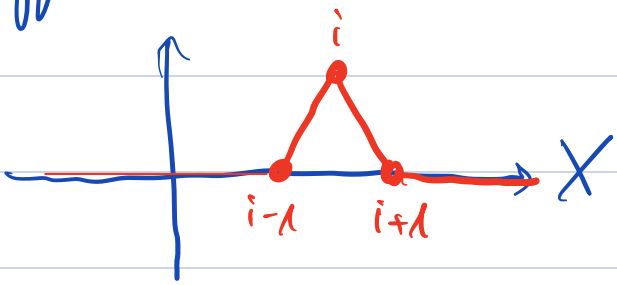
Lower bound for NNs:

Lemma: Any parity fct $\chi_S(x)$ $|S| = k$ can be easily implemented as a 2-layer NN with Relu activation and $\Theta(k)$ neurons.

Proof: $\chi_S(x) = \prod_{i \in S} x_i = g_k\left(\sum_{i \in S} x_i\right)$



It suffices to show that we can implement



→ and then add and substract these fcts

$$\downarrow$$

$$(X - i + 1)_+ - 2(X - i)_+ + (X - i - 1)_+$$

↳ Hence taking $X = \langle 1_S, x \rangle$, and summing them indeed implement the parity

□

Hence • ERM learns parity fcts:

approx error $= 0$

gen error $\lesssim \frac{\text{poly}(d)}{\sqrt{n}}$ → weights bounded by d and at most $O(d)$ nonzero

• However GD fail (if we trust we cannot efficiently learn noisy parities)

II Leap complexity

The set $\{\chi_S(x) : |S| = k\}$ has $\binom{d}{k} = \Theta(d^k)$ orthogonal fct

SQ lower bound: $\frac{q}{\tau^2} \geq c d^k$

For more general sparse fct:

$$g(x) = x_1 + x_1 x_2 x_3 x_4 \qquad \|g\|_{L^2}^2 = 2$$

If we want to learn $\|\hat{f} - g\|_{L^2} < 1$, we need to learn $x_2 x_3 x_4$. Conditional on knowing $x_1 \equiv$ learning 3-parity

and we can prove CSQ lower bound → actually can be made SQ lower bound for this specific fct.

$$\frac{q}{z^2} \geq c\, d^3$$

More generally: k-leap fcts,

CSQ requires $\frac{q}{z^2} \geq c\, d^k$

For online SGD with batch 1, if we trust this lower bound, that means we need

$T d \gtrsim d^k$ step size of batch 1 $(z \simeq 1$

that is $T \gtrsim d^{k-1}$

It agrees with numerical simulations!!

III Ridge functions

Take $x \sim N(0, I_d)$ and consider the set

$$\mathcal{H} = \{ h(x) = He_k(\langle \omega_*, x \rangle) : \omega_* \in S^{d-1} \}$$

where He_k is the degree-k Hermite polynomial (normalized)

[recall $\mathbb{E}_G[He_k(G) He_j(G)] = \mathbb{1}[j=k]$ $G \sim N(0,1)$]

Let's construct nearly orthogonal fcts with $\{\omega_j\}_{j=1}^N$

Note that $\langle h_i, h_j \rangle = \mathbb{E}[He_k(\langle \omega_i, x \rangle) He_k(\langle \omega_j, x \rangle)]$

$$= \langle \omega_i, \omega_j \rangle^k$$

Take a family s.t. $|\langle \omega_i, \omega_j \rangle| \le \delta$ for all $i \neq j$

↳ we can choose $N \ge \exp(c d \delta^2)$ of them

Then setting $\varepsilon = \delta^k$ and using general lower bound from above,

CSQ requires $\boxed{q \ge \frac{e^{cd\delta^2}}{q} (2^2 - \delta^k)}$

Fixing $\delta = \left(\frac{2^2}{2}\right)^{\frac{1}{k}} \rightarrow \log(8q/2^2) \ge c' d\, 2^{4/k}$

so we need $\frac{1}{\tau^2} \gtrsim d^{k/2}$ for $\frac{q}{\tau^2}$ to not be exponential in d. If we heuristically set

$\tau = \frac{1}{\sqrt{n}}$ n = # of samples (to implement CSQ algo)

we need $\boxed{n \gtrsim d^{\frac{k}{2}}}$ samples in order to be efficiently solvable

Rmk: ⚠ This lower bound is incorrect for Me_k if we reuse samples / learn with another loss function. In fact $n = \tilde{\Theta}(d)$ (info-theoretic) is sufficient.

↳ however, it seems to be correct for some functions whose first non-zero hermite coeff is k.

[Damian et al., 24]

Noisy Gradient Descent and CSQ

Population GD + gaussian noise

For $t = 1, \dots, T$

$$w^{t+1} = w^t - \eta \nabla_w R(w^t) + \eta \underbrace{z^t}_{\sim}$$

where $R(w) = \mathbb{E}\left[(y - f(x, w))^2\right]$ $\quad \sim_{iid} N(0, \sigma^2 I_p)$

Rmk: [Online SGD]

$$z^t = \underbrace{\nabla_w \left(y^t - f(x^t, w^t)\right)^2 - \nabla_w R(w^t)}$$

mean-0 iid noise but not Gaussian

Noisy GD can be approximated by a CSQ algo

$$-\nabla_w R(w) = \underbrace{2\mathbb{E}\left[y \nabla_w f(x, w)\right]}_{p \text{ CSQ queries}} - \underbrace{2\mathbb{E}\left[f(x; w) \nabla_w f(x, w)\right]}_{\text{marginal over } x \text{ is fixed}}$$

Let's couple noisy GD with CSQ with tolerance τ

$$\tilde{w}^{k+1} = \tilde{w}^{k} - \eta\, n^{k} + \eta\, z^{t}$$ (some Gaussian noise)

$$\hookrightarrow \quad \| n^{k} - \nabla_{w} R(\tilde{w}^{k}) \|_{\infty} \leq \varepsilon$$

We compare the law of $w^0, w^1, \dots, w^T \equiv w^{\leq T}$

to the law of $\tilde{w}^0, \tilde{w}^1, \dots, \tilde{w}^T \equiv \tilde{w}^{\leq T}$

We consider the KL divergence between the two

$$D_{KL}(P \| Q) = \int_{s \in S} \log\left(\frac{dP}{dQ}(s)\right) dP(s)$$

We have

$$D_{KL}(w^{\leq T} \| \tilde{w}^{\leq T}) = \underbrace{D_{KL}(w^0 \| \tilde{w}^0)}_{=0} \quad \text{(same initialization)}$$

chain rule for KL

$$+ \sum_{t=1}^{T} D_{KL}\left(w^t \mid w^{\leq t-1} \,\|\, \tilde{w}^t \mid \tilde{w}^{\leq t-1}\right)$$

$$= \sum_{t=1}^{T} \underbrace{D_{KL}\left(w^t \mid w^{t-1} \,\|\, \tilde{w}^t \mid \tilde{w}^{t-1}\right)}_{\equiv C_t}$$ Markov property

Using the update equation and def of conditional KL

$$C_t = \mathbb{E}_{w \sim \mathcal{A}(w^{t-1})} \left[D_{KL}\left(\underbrace{w - \eta \nabla R(w) + \eta z}_{N(w - \eta \nabla R(w), \eta^2 \sigma^2 I)} \,\|\, \underbrace{w - \eta n + \eta z}_{N(w - \eta n, \eta^2 \sigma^2 I)} \right) \right]$$

$$D_{KL}\left(N(\mu, \alpha^2 I) \,\|\, N(\mu', \alpha^2 I) \right) = \frac{1}{2\alpha^2} \|\mu - \mu'\|$$

$$= \frac{1}{2\sigma^2} \|\nabla R(w) - n\|_2^2$$

$$\leq \frac{1}{2\sigma^2} \rho^2 z^2$$

Therefore: $D_{KL}(w^{\leq T} \,\|\, \tilde{w}^{\leq T}) \leq \frac{T \rho z^2}{2\sigma^2}$

"total variation distance"

Consider $\widehat{TV}(P \| Q) = \sup_A |P(A) - Q(A)|$

$$TV(w^T \| \tilde{w}^T) \leq TV(w^{\leq T} \| \tilde{w}^{\leq T})$$

data processing inequality

$$\leq \sqrt{\frac{1}{2} D_{KL}(w^{\leq T}, \tilde{w}^{\leq T})}$$

Pinsker's inequality

$$\leq \frac{\sqrt{\rho T}}{2} \frac{z}{\sigma}$$

We consider the optimal coupling between w^T and $\tilde{w}^T$

$$\sup_{\text{joint probs}} \mathbb{P}(w^T = \tilde{w}^T) = 1 - TV(w^T \| \tilde{w}^T)$$

$$\geq 1 - \frac{\sqrt{p^T}}{2} \frac{\varepsilon}{\sigma}$$

Take $\varepsilon = \frac{2\delta\sigma}{\sqrt{pT}}$:

with probability $\geq 1-\delta$, we have $w^T = \tilde{w}^T$

Hence, if $\tilde{w}^T$ fails with Tp queries of tolerance $\frac{2\delta\sigma}{\sqrt{pT}}$

then, so does noisy GD with proba $1-\delta$

Corollary: Noisy GD on polynomial sized NN must take either $e^{\Omega(d)}$ many steps or have noise $\sigma = e^{-\Omega(d)}$ in order to learn parities.

Rmk: Note that it is still not a lower bound directly on GD or SGD. However still very informative!

(it is hard to go beyond SQ with noise-tolerant algo)

Rmk: Similar argument: instead of coupling noisy GD to a CSQ algorithm, one can couple to a dynamics with label y independent of x (a "junk flow" where there is nothing to learn)

→ the guarantees obtained are very similar to the ones obtained with CSQ

[Abbe, Sandon, '20] [Abbe, Boix-Adsera, '22]

Rmk: Can we prove unconditional lower bounds for SGD directly?

Results below indicates that no (if we don't constrain hyperparameter) as it would imply $P \neq NP$

① Poly-time inversality of DL

Did we learn anything about deep learning from the previous lower bounds?

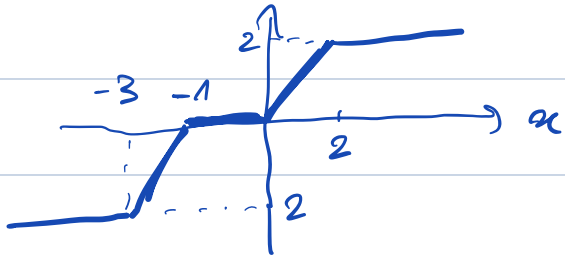
So far, we only used that there exist some problems that are hard to learn with ~~any~~ algo (or any SQ algo) and therefore SGD-trained NNs cannot succeed too...

Instead, we would like to get some insights on why neural networks specifically are so successful and what are their limitations

e.g. is there a problem that is efficiently solvable in polynomial time but where SGD on NNs fail?

Abbe and Sandon (2020) answered this question negatively in the following sense.

For the following activation function



such that the following holds.

Thm [Abbe, Sandon (2020)] If a problem can be learned in polynomial time, then there exists a neural network with the stage-wise ramp activation and of polysize such that running SGD on it learn the problem in poly time.

SGD on NNs is P-complete

Proof idea: it is an emulation proof. One can implement "bit extraction" that allows the NN to extract samples using SGD and then poly-computable function can be implemented as a polynomial-sized circuit.

The proof is quite involved and not that insightful.

E.g. SGD on NNs can learn noiseless parities in polytime by emulating Gaussian elimination.

Rmk: For the above emulation, one needs enough "precision" at each step. If the precision is too small, SGD on NNs can only implement poly-time SQ algo.
E.g. will fail to learn parities.

Is it an interesting resulting?

Yes: proving an unconditional lower bound on SGD-trained NNs would imply $P \neq NP$.
→ so we cannot show such a result without solving one of the major open problem of our time

No: These emulation NNs are highly non-standard and require complex architectures and initialization
↳ these hyperparameters depend on the problem.
↳ This is very different than what NNs do in practice (i.e, they are not emulating Gaussian elimination...)

Instead, NNs used in practice have regular architectures and generic initializations

What we really want: understand what SGD on regular NNs can learn / fail to learn.

Some work in that direction (e.g. learning leap fct).

The challenge of regular NNs vs emulation NNs is that it requires to understand SGD dynamics from generic initialization on regular architectures...

→ high-dimensional non-convex dynamics

This is hard!

Next lecture: high-dimensional non-convex dynamics