Optimization of Neural Nets

Instructor: Lei Wu [1]

Topics in Deep Learning Theory (Spring 2025)

Peking University

[1]School of Mathematical Sciences and Center for Machine Learning Research

Outline

1. Harndess of Optimizing Two-layer Nets (i.e., Learning Barron Functions)

2. Trainability in Neural Tangent Kernel Regime (i.e., Learning RKHS functions)

Motivating Questions

- Different from traditional ML, NNs are **non-convex** even for two-layer ones.

Motivating Questions

- Different from traditional ML, NNs are **non-convex** even for two-layer ones.
- The trainability of NNs highly depend on the target function f^*. Therefore, we need to carefully specify **target functions** when studying the trainability.

Motivating Questions

- Different from traditional ML, NNs are **non-convex** even for two-layer ones.
- The trainability of NNs highly depend on the target function f^*. Therefore, we need to carefully specify **target functions** when studying the trainability.
- We already show Barron functions can be learned efficiently in terms of approximation and estimation. Then, a very nature questions is:

  Can Barron functions be learned in terms of optimization ?

Motivating Questions

- Different from traditional ML, NNs are **non-convex** even for two-layer ones.
- The trainability of NNs highly depend on the target function f^*. Therefore, we need to carefully specify **target functions** when studying the trainability.
- We already show Barron functions can be learned efficiently in terms of approximation and estimation. Then, a very nature questions is:

  Can Barron functions be learned in terms of optimization ?

Motivating Questions

- Different from traditional ML, NNs are **non-convex** even for two-layer ones.
- The trainability of NNs highly depend on the target function f^*. Therefore, we need to carefully specify **target functions** when studying the trainability.
- We already show Barron functions can be learned efficiently in terms of approximation and estimation. Then, a very nature questions is:

  Can Barron functions be learned in terms of optimization ?

Keep in mind that we in fact have two subproblems:

- How do commonly-used algorithms, e.g., 2LNNs+SGD, perform?
- We can also design new algorithm! This addresses the question: if there exists an algorithm such that 2LNNs can be learned efficiently?

Main result

Definition 1

We say an algorithm $\mathcal{A}$ is efficient in learning a function class $\mathcal{F}$, if for every $\varepsilon > 0$, $f^* \in \mathcal{F}$, the time complexity for $\mathcal{A}$ returning an solution $\hat{f}$ such that $\|\hat{f} - f^*\| \leq \varepsilon$ satisfies:

$$\text{Time complexity} = \mathrm{poly}(1/\varepsilon, d).$$

Otherwise, we say $\mathcal{A}$ suffers the curse of dimensionality (CoD).

Main result

Definition 1

We say an algorithm $\mathcal{A}$ is efficient in learning a function class $\mathcal{F}$, if for every $\varepsilon > 0$, $f^* \in \mathcal{F}$, the time complexity for $\mathcal{A}$ returning an solution $\hat{f}$ such that $\|\hat{f} - f^*\| \leq \varepsilon$ satisfies:

$$\text{Time complexity} = \mathrm{poly}(1/\varepsilon, d).$$

Otherwise, we say $\mathcal{A}$ suffers the curse of dimensionality (CoD).

- The "Barron structure" is **insufficient** to ensure an efficient learning no matter which algorithm is used.
- We need to identify more refined structures that can guarantee an efficient learning. We will discuss this later.

- **The cryptographic perspective.**

Learning the intersection of halfspaces

- Let $\mathcal{X} = \{-1, 1\}^d$ be the Hamming cube and consider the binary classification problem, i.e., $f^* : \mathcal{X} \mapsto \{-1, 1\}$.
- Let σ_{step} be the Heaviside step function.

Learning the intersection of halfspaces

- Let $\mathcal{X} = \{-1, 1\}^d$ be the Hamming cube and consider the binary classification problem, i.e., $f^* : \mathcal{X} \mapsto \{-1, 1\}$.
- Let σ_{step} be the Heaviside step function.

We will need the following hardness result for learning the intersection of halfspaces (LIH).

Theorem 2 (Theorem 1.2, Kalai, Klivans, 2008)

Let $\mathcal{H} = \{x \mapsto \sigma_{\text{step}}(w^\top x - b - 1/2) : b \in \mathbb{N}, w \in \mathbb{N}^d, |b| + \|w\|_1 \leq \text{poly}(d)\}$. Define

$$\mathcal{H}_K = \{x \mapsto h_1(x) \wedge h_2(x) \wedge \cdots \wedge h_K(x) \ : \ h_i \in \mathcal{H}\}.$$

Assume $K \geq d^\rho$ with $\rho > 0$. Then, under a certain **cryptographic** *assumption, $\mathcal{H}_K$ is not efficiently learnable.*

Learning the intersection of halfspaces (cont'd)

- The proof essentially reduces the LIH problem to some classical hard problems, e.g., k-coloring.

Learning the intersection of halfspaces (cont'd)

- The proof essentially reduces the LIH problem to some classical hard problems, e.g., k-coloring.
- The **cryptographic** assumption means that we assume that these hard problems are indeed hard in certain sense. If this assumption does not hold, then the modern cryptosystem can be broken in a polynomial time.

Learning the intersection of halfspaces (cont'd)

- The proof essentially reduces the LIH problem to some classical hard problems, e.g., k-coloring.
- The **cryptographic** assumption means that we assume that these hard problems are indeed hard in certain sense. If this assumption does not hold, then the modern cryptosystem can be broken in a polynomial time.
- We shall show two-layer neural nets can simulate the functions in $\mathcal{H}_K$.

Hardness of learning two-layer ReLU nets

Theorem 3 (Livni, et al, 2014)

Let $\mathcal{X} = \{-1, 1\}^d$*, and* $\mathcal{G} = \{f \in \mathcal{B} : \|f\|_{\mathcal{B}} \leq \mathrm{poly}(d)\}$*. Then,* $\mathcal{G}$ *is not efficiently learnable.*

Hardness of learning two-layer ReLU nets

Theorem 3 (Livni, et al, 2014)

Let $\mathcal{X} = \{-1, 1\}^d$, *and* $\mathcal{G} = \{f \in \mathcal{B} : \|f\|_{\mathcal{B}} \leq \mathrm{poly}(d)\}$. *Then,* $\mathcal{G}$ *is not efficiently learnable.*

- The intuition is that 2-layer neural network can simulate the intersection of hyperspaces.

Hardness of learning two-layer ReLU nets

Theorem 3 (Livni, et al, 2014)

Let $\mathcal{X} = \{-1, 1\}^d$, *and* $\mathcal{G} = \{f \in \mathcal{B} : \|f\|_{\mathcal{B}} \leq \mathrm{poly}(d)\}$. *Then,* $\mathcal{G}$ *is not efficiently learnable.*

- The intuition is that 2-layer neural network can simulate the intersection of hyperspaces.
- The step function can be approximated by two ReLU functions very well:

$$\sigma_{\text{step}}(t) = \lim_{a \to \infty} \left(\texttt{ReLU}(at) - \texttt{ReLU}(at - 1)\right).$$

Hardness of learning two-layer neural nets

Proof:

- Let $c(x) = w^\top x - b - 1/2$. Since $w \in \mathbb{N}^d, x \in \{-1, 1\}^d, b \in \mathbb{N}$, we have $|c(x)| \geq 1/2$. Assume $\|w\|_1 + |b| \leq \mathrm{poly}(d)$.

Hardness of learning two-layer neural nets

Proof:

- Let $c(x) = w^\top x - b - 1/2$. Since $w \in \mathbb{N}^d, x \in \{-1, 1\}^d, b \in \mathbb{N}$, we have $|c(x)| \geq 1/2$. Assume $\|w\|_1 + |b| \leq \mathrm{poly}(d)$.
- Consider k hyperplanes $\{c_i\}_{i=1}^k$. Let $h_i(x) = \sigma_{\text{step}}(c_i(x)) \in \mathcal{H}$.

Hardness of learning two-layer neural nets

Proof:

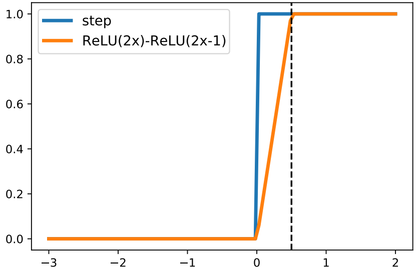
- Let $c(x) = w^\top x - b - 1/2$. Since $w \in \mathbb{N}^d, x \in \{-1,1\}^d, b \in \mathbb{N}$, we have $|c(x)| \geq 1/2$. Assume $\|w\|_1 + |b| \leq \mathrm{poly}(d)$.
- Consider k hyperplanes $\{c_i\}_{i=1}^k$. Let $h_i(x) = \sigma_{\mathrm{step}}(c_i(x)) \in \mathcal{H}$.
- Let

$$
\begin{aligned}
g(x) &= \frac{1}{2k}\left(\sum_{i=1}^k \sigma_{\mathrm{step}}(c_i(x)) - k + \frac{1}{3}\right) \\
&= \frac{1}{2k}\left(\sum_{i=1}^k \left(\mathtt{ReLU}(2c_i(x)) - \mathtt{ReLU}(2c_i(x)-1)\right) - k + \frac{1}{3}\right).
\end{aligned}
$$

Obviously, g is a 2-layer ReLU network with the path norm bounded by

$$
\frac{1}{2k}\left(k + \frac{1}{3} + \sum_{i=1}^k (2(2\|w_i\|_1 + |b_i| + 1/2) + 1\right) = \mathrm{poly}(d).
$$

- The blue part is equal to $\sigma_{\text{step}}(c_i(x))$ due to $\sigma_{\text{step}}(z) = \mathtt{ReLU}(2z) - \mathtt{ReLU}(2z-1)$ for $|z| \geq 1/2$.



- We can verify that

$$\text{sign}(g(x)) = h_1(x) \wedge h_2(x) \wedge \cdots \wedge h_k(x), \ \forall x \in \{-1, 1\}^d.$$

Remarks

- Note that similar results also hold for two-layer nets with the sigmoid activation function, since the sigmoid function can approximate the step function as well. See [Livni, et al, 2014] for more details.

[2]We use it in establishing a lower bound of the linear approximation of Barron spaces.

Remarks

- Note that similar results also hold for two-layer nets with the sigmoid activation function, since the sigmoid function can approximate the step function as well. See [Livni, et al, 2014] for more details.
- The above results rely on the hardness of certain classical hard problems.

[2]We use it in establishing a lower bound of the linear approximation of Barron spaces.

Remarks

- Note that similar results also hold for two-layer nets with the sigmoid activation function, since the sigmoid function can approximate the step function as well. See [Livni, et al, 2014] for more details.
- The above results rely on the hardness of certain classical hard problems.
    - Pros: It implies that the hardness holds for any algorithms.

[2]We use it in establishing a lower bound of the linear approximation of Barron spaces.

Remarks

- Note that similar results also hold for two-layer nets with the sigmoid activation function, since the sigmoid function can approximate the step function as well. See [Livni, et al, 2014] for more details.
- The above results rely on the hardness of certain classical hard problems.
  - Pros: It implies that the hardness holds for any algorithms.
  - Cons: This perspective is too abstract. It does not provide any concrete examples and intuitions behind the hardness of training.

[2]We use it in establishing a lower bound of the linear approximation of Barron spaces.

Remarks

- Note that similar results also hold for two-layer nets with the sigmoid activation function, since the sigmoid function can approximate the step function as well. See [Livni, et al, 2014] for more details.
- The above results rely on the hardness of certain classical hard problems.
  - Pros: It implies that the hardness holds for any algorithms.
  - Cons: This perspective is too abstract. It does not provide any concrete examples and intuitions behind the hardness of training.
- In the following, we will provide a more intutive understanding from the orthogonal function perspective [2].)

[2]We use it in establishing a lower bound of the linear approximation of Barron spaces.

Orthonormal classes

Let $\rho \in \mathcal{P}(\mathcal{X})$. For any $f_1, f_2 \in L^2(\rho)$, let $\langle f_1, f_2 \rangle = \mathbb{E}_{x\sim\rho}[f_1(x) f_2(x)]$.

Definition 4 (Orthonormal class)

Let $\mathcal{F}$ be a function class. We say that it is an orthonormal class, if $\langle f_i, f_j \rangle = \delta_{i,j}$ for any $f_i, f_j \in \mathcal{F}$.

Orthonormal classes

Let $\rho \in \mathcal{P}(\mathcal{X})$. For any $f_1, f_2 \in L^2(\rho)$, let $\langle f_1, f_2 \rangle = \mathbb{E}_{x\sim\rho}[f_1(x) f_2(x)]$.

Definition 4 (Orthonormal class)

Let $\mathcal{F}$ be a function class. We say that it is an orthonormal class, if $\langle f_i, f_j \rangle = \delta_{i,j}$ for any $f_i, f_j \in \mathcal{F}$.

- Let $\mathcal{B}_d = \{f \in \mathcal{B} : \|f\|_{\mathcal{B}} \lesssim d^2\}$. We will show that $\mathcal{B}_d$ contains an orthonormal subset $\mathcal{F} = \{f_1, \ldots, f_m\}$ with $m = \exp(d)$.
- We will show that learning the orthonormal class $\mathcal{F}$ is hard if $|\mathcal{F}| = \exp(d)$.

Parity functions

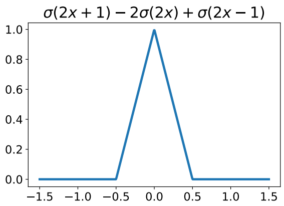
- $\mathcal{F}_1 = \{f_v(x) = (-1)^{\langle v,x\rangle} : v \in \{0,1\}^d\}$, where $x \in \{0,1\}^d$.
- Consider $\rho = \text{Unif}(\{0,1\}^d)$. Then, we have

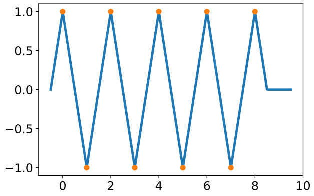
$$\langle f_v, f_{v'}\rangle = \mathbb{E}_x[(-1)^{(v+v')^\top x}] = \mathbb{E}_x[\prod_{i=1}^{d}(-1)^{(v_i+v_i')x_i}] \tag{1}$$

$$= \prod_{i=1}^{d}\mathbb{E}_{x_i}[(-1)^{(v_i+v_i')x_i}] = \delta_{v,v'}. \tag{2}$$

Hence, $\mathcal{F}_1$ is an orthonormal class with $|\mathcal{F}_1| = 2^d$.

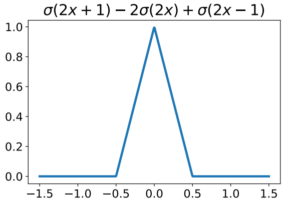
Parity functions as two-layer neural nets

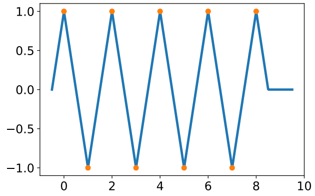




- Observation:
  - The function $(-1)^s$ for $s \in \mathbb{N}$ can be implemented using the triangle wave.
  - The triangle wave can be written as a linear combination of the hat function, which is a linear combination of ReLU function (left figure).

Parity functions as two-layer neural nets





- Observation:
  - The function $(-1)^s$ for $s \in \mathbb{N}$ can be implemented using the triangle wave.
  - The triangle wave can be written as a linear combination of the hat function, which is a linear combination of ReLU function (left figure).
- Note that $v^\top x \in \{0, 1, \ldots, d\}$. Let σ be the ReLU function. Then,

$$(-1)^{v^\top x} = \sigma_{\text{tri}}(v^\top x) = \sum_{i=0}^{d} (-1)^i \sigma_{\text{hat}}(v^\top x - i)$$

$$= \sum_{i=0}^{d} (-1)^i \Big(\sigma(2v^\top x - 2i + 1) - 2\sigma(2v^\top x - 2i) + \sigma(2v^\top x - 2i - 1) \Big)$$

Parity functions as two-layer neural nets (Cont'd)

- It is easy to show that the path norm of this network is bounded by Cd^2.

Cosine neurons

Consider the domain $\mathcal{X} = [0, 2\pi]^d$.

- Let $S_d = \{x \mapsto \cos(w^\top x) \, : \, w \in \mathbb{N}^d, \sum_{i=1}^d w_i \leq d\}$.
- In the previous lecture, we have shown that

$$|S_d| \gtrsim 2^d \tag{3}$$
$$\|f\|_{\mathcal{B}} \lesssim d^2, \quad \forall f \in S_d. \tag{4}$$

- Hence, for this continuous case, S_d contains exponential many orthonormal functions.

Remark

- $\mathcal{F}_\phi := \{\phi(w^\top x) : \|w\| = \sqrt{d}\}$. $\mathcal{D} = \mathcal{N}(0, I_d)$. [Shamir, 2017] shows that as long as ϕ is periodic, under some mild condition, $\mathcal{F}_\phi$ contains an orthonormal subset $\mathcal{F}_1 = \{f_1, \dots, f_m\}$ with $m = \exp(d)$.

Gradients for an orthonormal class

Let $h(\cdot;\theta)$ be any parametric model. Denote by $R^f(\theta) = \mathbb{E}_x[(h(x;\theta) - f(x))^2]$ the risk. Then, we have the following theorem.

Theorem 5

Let $\mathcal{F}$ be an orthonormal class. Let P denote the uniform distribution over the space of $\mathcal{F}$ and $g(\theta) = \mathbb{E}_{f\sim P}[\nabla R^f(\theta)]$. We have

$$\mathbb{E}_{f\sim P}[(\nabla R^f(\theta) - g(\theta))^2] \leq \frac{\mathbb{E}_x[\|\nabla_\theta h(x;\theta)\|^2]}{|\mathcal{F}|}. \tag{5}$$

Gradients for an orthonormal class

Let $h(\cdot;\theta)$ be any parametric model. Denote by $R^f(\theta) = \mathbb{E}_x[(h(x;\theta) - f(x))^2]$ the risk. Then, we have the following theorem.

Theorem 5

Let $\mathcal{F}$ be an orthonormal class. Let P denote the uniform distribution over the space of $\mathcal{F}$ and $g(\theta) = \mathbb{E}_{f\sim P}[\nabla R^f(\theta)]$. We have

$$\mathbb{E}_{f\sim P}[(\nabla R^f(\theta) - g(\theta))^2] \le \frac{\mathbb{E}_x[\|\nabla_\theta h(x;\theta)\|^2]}{|\mathcal{F}|}. \tag{5}$$

- If $|\mathcal{F}|$ is exponentially in d, e.g., the parity functions. The variance of gradients is exponentially small.

Gradients for an orthonormal class

Let $h(\cdot;\theta)$ be any parametric model. Denote by $R^f(\theta) = \mathbb{E}_x[(h(x;\theta) - f(x))^2]$ the risk. Then, we have the following theorem.

Theorem 5

Let $\mathcal{F}$ be an orthonormal class. Let P denote the uniform distribution over the space of $\mathcal{F}$ and $g(\theta) = \mathbb{E}_{f\sim P}[\nabla R^f(\theta)]$. We have

$$\mathbb{E}_{f\sim P}[(\nabla R^f(\theta) - g(\theta))^2] \leq \frac{\mathbb{E}_x[\|\nabla_\theta h(x;\theta)\|^2]}{|\mathcal{F}|}. \tag{5}$$

- If $|\mathcal{F}|$ is exponentially in d, e.g., the parity functions. The variance of gradients is exponentially small.
- This theorem implies that the "information" about the target function contained in the gradient is exponentially small.
- Therefore, one would expect that gradient-based methods will be unlikely to learn the function class $\mathcal{F}$. For instance, this information might be distroyed by the rounding error of a finite-precision machine.

Proof:

- First, the gradient can be written as follows

$$\nabla_\theta R^f = \mathbb{E}_x[(h(x;\theta) - f)\nabla_\theta h(x;\theta)] = C_\theta - \langle f, \nabla_\theta h(x;\theta)\rangle,$$

where C_θ is independent of the target function f.

Proof:

- First, the gradient can be written as follows

$$\nabla_\theta R^f = \mathbb{E}_x[(h(x;\theta) - f)\nabla_\theta h(x;\theta)] = C_\theta - \langle f, \nabla_\theta h(x;\theta)\rangle,$$

  where C_θ is independent of the target function f.

- Hence,

$$\begin{aligned}
\mathbb{E}_f[(\nabla_\theta R^f - g(\theta))^2] &\leq \mathbb{E}_f[\langle f, \nabla_\theta h(x;\theta)\rangle^2] && (6)\\
&\leq \frac{1}{|\mathcal{F}|}\sum_f \langle f, \nabla_\theta h(x;\theta)\rangle^2 && (7)\\
&\leq \frac{\mathbb{E}_x[\|\nabla_\theta h(x;\theta)\|^2]}{|\mathcal{F}|}. && (8)
\end{aligned}$$

Hardness of learning with GD: Setup

Setup:

- Assume $\mathcal{F}$ to be an orthonormal class with $|\mathcal{F}| = 2^d$. Consider the binary classification with the hinge loss. The risk is given by

$$R^f(\theta) := \mathbb{E}_x[\max(0, 1 - h(x;\theta)f(x))]. \tag{9}$$

- Assume $|h(x;\theta)| \leq 1$ and $|f(x)| \leq 1$ for any $x \in \mathcal{X}$. Then we have

$$\begin{aligned}\mathbb{E}_f[\|\nabla_\theta R^f(\theta)\|^2] &= \mathbb{E}_f\left(\mathbb{E}_x[f(x)\nabla_\theta h(x;\theta)]\right)^2 \\ &= \frac{1}{|\mathcal{F}|}\sum_i \langle f_i, \nabla h(\cdot;\theta)\rangle^2 \leq \frac{\|\nabla_\theta h\|^2}{|\mathcal{F}|} \leq \frac{G_\theta}{2^d}.\end{aligned}$$

Remark: the above assumption holds for parity functions.

Hardness of learning with GD

Theorem 6

Assume the model satisfies that $\sup_{x \in X} |h(x;\theta)| \leq 1$ *and* $\mathbb{E}_x[\|\nabla_\theta h(x;\theta_1) - \nabla_\theta h(x;\theta_2)\|^2] \leq L\|\theta_1 - \theta_2\|^2$. *Let* θ_0, θ_t^f *be the GD solution at time* 0 *and time* t*, respectively. Then, there exist* C_1, C_2 *such that*

$$\mathbb{E}_f[\|\theta_t^f - \theta_0\|^2] \leq C_1(e^{\frac{C_2 t}{2^{d/2}}} - 1), \tag{10}$$

where C_1, C_2 *only depend on* L *and* θ_0.

Hardness of learning with GD

Theorem 6

Assume the model satisfies that $\sup_{x\in X}|h(x;\theta)| \le 1$ *and* $\mathbb{E}_x[\|\nabla_\theta h(x;\theta_1) - \nabla_\theta h(x;\theta_2)\|^2] \le L\|\theta_1 - \theta_2\|^2$. *Let* θ_0, θ_t^f *be the GD solution at time* 0 *and time* t, *respectively. Then, there exist* C_1, C_2 *such that*

$$\mathbb{E}_f[\|\theta_t^f - \theta_0\|^2] \le C_1(e^{\frac{C_2 t}{2^{d/2}}} - 1), \tag{10}$$

where C_1, C_2 *only depend on* L *and* θ_0.

The above theorem implies that GD solution is exponentially close to the initialization in polynomial time.

Hardness of learning with GD (Cont'd)

More rigorously, we have the following corollary.

Corollary 7

For any $T = poly(d)$*, there exists a* $f \in \mathcal{F}$ *such that*

$$\|\theta_t^f - \theta_0\| \leq C\frac{poly(d)}{2^d}, \ \forall t \in [0, T]$$

where C *only depends on* L *and* θ_0*.*

Hardness of learning with GD

Proof:

- $G(\theta) = \mathbb{E}_x[\|\nabla_\theta h(x;\theta)\|^2]$ satisfies

$$G(\theta) \leq G(\theta_0) + 2L\|\theta - \theta_0\|^2. \tag{11}$$

- Therefore,

$$\begin{aligned}
\frac{d\,\mathbb{E}_f[\|\theta_t^f - \theta_0\|^2]}{dt} &= 2\,\mathbb{E}_f[\langle \theta_t^f - \theta_0, -\nabla_\theta R^f(\theta_t^f)\rangle] && (12)\\
&\leq \frac{1}{2^{\frac{d}{2}-1}}\sqrt{\mathbb{E}_f[\|\theta_t^f - \theta_0\|^2]\,\mathbb{E}_f[G(\theta_t^f)]} && (13)\\
&\leq \frac{1}{2^{\frac{d}{2}-1}}\sqrt{\mathbb{E}_f[\|\theta_t^f - \theta_0\|^2]\,\mathbb{E}_f[G(\theta_0) + 2L\|\theta_t^f - \theta_0\|^2]}. && (14)
\end{aligned}$$

Hardness of learning with GD

Proof: Let $\delta_t = \sqrt{\mathbb{E}_f[\|\theta_t^f - \theta_0\|^2]}$. Then, we have

$$\dot{\delta}_t \leq 2^{2-\frac{d}{2}}(\sqrt{2L}\delta_t + \sqrt{G(\theta_0)}). \tag{15}$$

By Gronwall's inequality, we obtain

$$\delta_t \leq \sqrt{\frac{G(\theta_0)}{2L}}(e^{2^{2-\frac{d}{2}}\sqrt{L}t} - 1).$$

Numerical evidence

Consider learning parity functions with online SGD. Fig. 1 shows the convergence of SGD. Here, the model is two-layer neural nets with width being 2000. The hinge loss $\ell(y, y') = \max(0, 1 - yy')$ is used, batch size is 2000 and learning rate is 0.002. We see clearly that when $d = 20$, the training process does not show any improvement in a reasonable time.

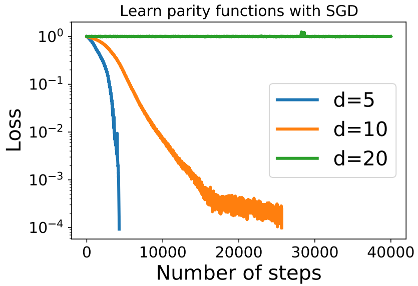


Figure 1: Learning party functions with SGD and two-layer neural nets.

Summary

- Learning a subset of two-layer neural nets, whose path norms are bounded by $\mathrm{poly}(d)$, can be reduced to certain classical hard problems, whose hardness is assumed to be true. Otherwise, the modern cryptosystem can be broken in polynomial time. This type of hardness results hold for any algorithms.

Summary

- Learning a subset of two-layer neural nets, whose path norms are bounded by $\mathrm{poly}(d)$, can be reduced to certain classical hard problems, whose hardness is assumed to be true. Otherwise, the modern cryptosystem can be broken in polynomial time. This type of hardness results hold for any algorithms.
- For orthonormal classes, we show that the gradient variance (wrt the target function) is exponentially small. Hence, *gradient-based* algorithms are unlikely to succeed. This observation hold for any parametric model as long as they satisfy certain Lipschitz condition.

Summary

- Learning a subset of two-layer neural nets, whose path norms are bounded by $\mathrm{poly}(d)$, can be reduced to certain classical hard problems, whose hardness is assumed to be true. Otherwise, the modern cryptosystem can be broken in polynomial time. This type of hardness results hold for any algorithms.
- For orthonormal classes, we show that the gradient variance (wrt the target function) is exponentially small. Hence, *gradient-based* algorithms are unlikely to succeed. This observation hold for any parametric model as long as they satisfy certain Lipschitz condition.
- Typical examples include the parity function and the cosine neuron: $f(x) = \cos(w^\top x)$. The Barron norms of these functions are not greater than $O(d^2)$.

Summary

- Learning a subset of two-layer neural nets, whose path norms are bounded by $\mathrm{poly}(d)$, can be reduced to certain classical hard problems, whose hardness is assumed to be true. Otherwise, the modern cryptosystem can be broken in polynomial time. This type of hardness results hold for any algorithms.
- For orthonormal classes, we show that the gradient variance (wrt the target function) is exponentially small. Hence, *gradient-based* algorithms are unlikely to succeed. This observation hold for any parametric model as long as they satisfy certain Lipschitz condition.
- Typical examples include the parity function and the cosine neuron: $f(x) = \cos(w^\top x)$. The Barron norms of these functions are not greater than $O(d^2)$.
- These hardness results suggest that the Barron space is very likely too large to study the training of two-layer neural nets.

More Remarks

- More intuitions behind the hardess of learning $f(x) = \phi(w^\top x)$ with ϕ being a "randomized function".
- The analogy with cryptography system?
- Frequency perspective?
- Symmetry perspective?

Convergence of GD in NTK regime

Setup

- Consider the two-layer neural network (2LNN):

$$f_m(x;\theta) = \sum_{j=1}^{m} a_j \sigma(b_j^\top x), \tag{16}$$

where $\theta = (a, B)$ be the parameters, and $\sigma(z) = \max(0, z)$. The results can be extended to general Lipschitz activation functions with small modifications.

- The empirical risk with the square loss is given by

$$\hat{\mathcal{R}}(\theta) = \frac{1}{2n} \sum_{i=1}^{n} (f_m(x_i;\theta) - y_i)^2. \tag{17}$$

- Let $e_i = f_m(x_i;\theta) - y_i$. The GD flow is given by

$$\begin{aligned} \dot{a}_j &= -\sum_{i=1}^{n} e_i \sigma(b_j^\top x_i) \\ \dot{b}_j &= -\sum_{i=1}^{n} e_i a_j \sigma'(b_j^\top x_i) x_i. \end{aligned} \tag{18}$$

Setup (Cont'd)

- Let $\pi_0 = \text{Unif}(\mathbb{S}^{d-1})$. We will mainly focus on the initialization:

$$a_j = 0, \quad b_j \sim \pi_0, \text{ for } j = 1, \dots, m. \tag{19}$$

Convergence results

Define an associate kernel

$$k(x, x') = \mathbb{E}_{b\sim\pi_0}[\sigma(b^\top x)\sigma(b^\top x')].$$

Let $K = (k(x_i, x_j)) \in \mathbb{R}^{n\times n}$ be the kernel matrix, and $\lambda_n(K)$ be the smallest eigenvalue of K.

Theorem 8

Let $\theta(t)$ be the GD solution at time t. For any $\delta \in (0,1)$, assume that $m \geq \frac{10n\log(2n^2/\delta)}{\lambda_n^2(K)}$. Then, w.p. $1-\delta$ over the initialization, we have

$$\hat{\mathcal{R}}(\theta(t)) \leq e^{-m\lambda_n t}\hat{\mathcal{R}}(\theta_0).$$

A general observation: Neural tangent kernel

The proof relies on the following observation.

- The GD flow can be written as

$$\dot{\theta}(t) = -\frac{1}{n}\sum_{i=1}^{n}(f(x_i;\theta(t)) - y_i)\nabla_\theta f(x_i;\theta(t))$$

- Let $e_i(t) = f(x_i;\theta(t)) - y_i$. Then, we have

$$\frac{de_i(t)}{dt} = \langle \nabla f(x_i;\theta(t)), \dot{\theta}(t)\rangle = -\sum_{i'=1}^{n} \langle \nabla f(x_i;\theta(t)), \nabla f(x_{i'};\theta(t))\rangle e_{i'}(t).$$

- Let $G = (G_{i,i'}) \in \mathbb{R}^{n\times n}$ with $G_{i,i'} = \langle \nabla f(x_i;\theta), \nabla f(x_{i'};\theta)\rangle$ be the Gram matrix, and $e = (e_1,\ldots,e_n)^\top \in \mathbb{R}^n$. Then,

$$\frac{de(t)}{dt} = -G(\theta(t))e(t). \tag{20}$$

- If $\lambda_n(G(\theta_t))$ is bounded away from zero for any $t \geq 0$, then the empirical risk converges to zero exponentially fast, since

$$\frac{d\|e(t)\|^2}{dt} = -2e(t)^\top G(\theta_t)e(t) \leq -2\lambda_n(G(\theta_t))\|e(t)\|^2. \tag{21}$$

The Gram matrix for a 2LNN

- In this case, $G(\theta) = m\hat{K}(\theta)$

$$\hat{K}_{i,i'}(\theta) = \frac{1}{m}\sum_{j=1}^{m} \sigma(b_j^\top x_i)\sigma(b_j^\top x_{i'}) + a_j^2 \sigma'(b_j^\top x_i)\sigma'(b_j^\top x_{i'}) x_i^\top x_{i'}.$$

- As $m \to \infty$, we have

$$\hat{K}_{i,i'} \to K_{i,i'} = k(x_i, x_{i'}).$$

where

$$k(x, x') = \mathbb{E}_{a,b}[\sigma(b^\top x)\sigma(b^\top x') + a^2 \sigma'(b^\top x)\sigma'(b^\top x') x^\top x']. \tag{22}$$

- For the initialization considered here,

$$k(x, x') = \mathbb{E}_{b\sim\pi_0}[\sigma(b^\top x)\sigma(b^\top x')]. \tag{23}$$

Here, only the gradients wrt a contribute to the kernel.

Positivity of the Gram matrix at initialization

Lemma 9

Assume $\lambda_n(K) > 0$. *For any* $\delta \in (0,1)$, *if* $m \geq \frac{\log(n^2/\delta)}{2\lambda_n^2(K)}$, *with probability* $1-\delta$ *over the random initialization, we have*

$$\lambda_n(G) \geq \frac{m}{2}\lambda_n(K).$$

Remark:

- The condition: $\lambda_n(K) > 0$ does not allow two samples x_i and x_j to align with each other for $i \neq j$.
- If $\{x_i\}_{i=1}^n$ are independently drawn from Unif($\mathbb{S}^{d-1}$), [Braun, 2006] proved that with high probability, $\lambda_n(K) > n\lambda_n/2$, where λ_n is the n-th largest eigenvalue of the kernel function $k(\cdot,\cdot)$.
- For the ReLU activation function, one can show that $\lambda_n \geq \frac{C_d}{n^{1+1/d}}$. So, $\lambda_n(K) \geq C_d/n^{1/d}$ (see the appendix of (Ma et al, MSML2020)).
- We will leave $\lambda_n(K) > 0$ as a basic assumption.

Positivity of the Gram matrix at initialization

Proof:

- By Hoeffding's inequality, we have for any $i, j \in [n]$

$$\mathbb{P}\{|\hat{K}_{i,j} - K_{i,j}| \geq \varepsilon\} = \mathbb{P}\left\{\left|\frac{1}{m}\sum_{s=1}^{m} \sigma(b_s^\top x_i)\sigma(b_s^\top x_j) - \mathbb{E}[\sigma(b_s^\top x_i)\sigma(b_s^\top x_j)]\right| \geq \varepsilon\right\} \leq e^{-2m\varepsilon}$$

- Taking the union bound leads to

$$\mathbb{P}\{\|\hat{K} - K\|_F \leq \varepsilon\} \geq 1 - \sum_{i,j=1}^{n} \mathbb{P}\{|\hat{K}_{i,j} - K_{i,j}| \geq \varepsilon\} \geq 1 - n^2 e^{-2m\varepsilon^2}.$$

- Using the Weyl's inequality, we have

$$\lambda_n(\hat{K}) \geq \lambda_n(K) - \|\hat{K} - K\|_F \geq \lambda_n(K) - \varepsilon.$$

- Take $\varepsilon = \lambda_n(K)/2$ and let the failure prob. $n^2 e^{-2m\varepsilon^2} \leq \delta$. This leads to $m \geq \frac{\log(n^2/\delta)}{2\lambda_n^2(K)}$.

Gradient descent near the initialization

- Define a neighbor of the initialization by

$$\mathcal{I}(\theta_0) := \left\{\theta : \|\hat{K}(\theta) - \hat{K}(\theta_0)\|_F \le \frac{\lambda_n(K)}{4}\right\}$$

- Using Lemma 9, for any $\delta \in (0,1)$ with probability $1-\delta$, we have for any $\theta \in \mathcal{I}(\theta_0)$ that

$$\lambda_n(\hat{K}(\theta)) \ge \lambda_n(\hat{K}(\theta_0)) - \|\hat{K}(\theta_0) - \hat{K}(\theta)\|_F \ge \frac{\lambda_n(K)}{2} - \frac{\lambda_n(K)}{4} = \frac{\lambda_n(K)}{4}.$$

Lemma 10

Let $t_0 = \inf\{t : \theta(t) \notin I(\theta_0)\}$. *For any* $\delta \in (0,1)$, *assume* $m \ge \frac{\log(n^2/\delta)}{2\lambda_n^2(K)}$. *For any* $t \in [0, t_0]$,

$$\hat{\mathcal{R}}(\theta(t)) \le e^{-\frac{m\lambda_n(K)}{2}t}\hat{\mathcal{R}}(\theta_0).$$

Proof:

$$\frac{d}{dt}\hat{\mathcal{R}}(\theta(t)) = \frac{1}{2n}\frac{d\|e(t)\|^2}{dt} = \frac{-m}{n}e^\top \hat{K} e \le \frac{-m}{n}\frac{\lambda_n(K)}{4}\|e(t)\|^2 = \frac{-m\lambda_n(K)}{2}\hat{\mathcal{R}}(\theta_t)$$

Long-time convergence of GD

Proof of Theorem 8:

- We only need to prove that $t_0 = \infty$. Otherwise, assume that $t_0 < \infty$.
- First, the empirical risk is *smooth* in the sense that

$$\|\nabla \hat{\mathcal{R}}(\theta)\|^2 \leq \|\theta\|^2 \hat{\mathcal{R}}(\theta).$$

- Then,

$$\begin{aligned}\|\theta(t) - \theta_0\| &\leq \int_0^{t_0} \|\nabla \hat{\mathcal{R}}(\theta(t))\| dt \leq \max_{t\in[0,t_0]} \|\theta(t)\| \int_0^{t_0} \sqrt{\hat{\mathcal{R}}(\theta)} dt \\ &\leq \max_{t\in[0,t_0]} \|\theta(t)\| \int_0^{t_0} e^{-\frac{m\lambda_n(K)}{4}t} \sqrt{\hat{\mathcal{R}}(\theta_0)} dt \lesssim \frac{\max_{t\in[0,t_0]} \|\theta(t)\|}{m\lambda_n(K)}.\end{aligned}$$

Let $\gamma = \max_{t\in[0,t_0]} \|\theta(t) - \theta_0\|$. Using the fact that $\|\theta_0\| = \sqrt{\sum_{s=1}^m \|b_s(0)\|^2} = \sqrt{m}$, we have

$$\gamma \lesssim \frac{\gamma + \sqrt{m}}{m\lambda_n(K)},$$

which leads to

$$\gamma \lesssim \frac{1}{\sqrt{m}\lambda_n(K)}.$$

Long-time convergence of GD (Cont'd)

Proof of Theorem 8:

- Since σ is 1-Lipschitz continuous, we have for any $t \in [0, t_0]$,

$$\begin{aligned}
\|\hat{K}(\theta(t)) - K(\theta_0)\|_F^2 &= \sum_{i,j} |\frac{1}{m}\sum_{s=1}^{m} \sigma(b_s(t)^\top x_i)\sigma(b_s(t)^\top x_j) - \frac{1}{m}\sum_{s=1}^{m} \sigma(b_s(0)^\top x_i)\sigma(b_s(0) \\
&\lesssim \frac{n^2}{m^2}(\|\theta(t) - \theta_0\| + \|\theta(t) - \theta_0\|^2) \\
&\lesssim \frac{n^2}{m^2}(\gamma + \gamma^2).
\end{aligned}$$

- By the assumption, $\gamma \leq 1$. Hence, $m \geq 20n/\lambda_n(K)$ leads to

$$\|\hat{K}(\theta(t)) - K(\theta_0)\|_F \leq \frac{\lambda_n(K)}{8},$$

which contradicts the definition of t_0. Therefore, $t_0 = \infty$.

Remarks

- In the above analysis, the main ingredient is the positivity of the Gram matrix G, which relies on the positivity of the **tangent kernel**:

$$k(x, x') = \lim_{m\to\infty} m^{-\alpha}\langle \nabla f(x;\theta), \nabla f(x';\theta)\rangle,$$

  where α is a specific factor related to the initialization such that the limit exists.
- The key observation is that $b_j(t) - b_j(0) \sim \frac{1}{m}$. The parameters of the convergent solution is close to the initialization.
- The results can be extended to general initializations. For instance, consider the balanced initialization: $a_j \sim \mathcal{N}(0, 1/m), b_j \sim \mathcal{N}(0, I_d/(md))$, for which the Gram matrix

$$G_{i,i'} = \sum_{j=1}^{m} \sigma(b_j^\top x_i)\sigma(b_j^\top x_{i'}) + a_j^2\sigma'(b_j^\top x_i)\sigma'(b_j^\top x_{i'})x_i^\top x_{i'}$$

$$\to k(x_i, x_{i'}) := \mathbb{E}_{b\sim\mathcal{N}(0,I_d/d)}[\sigma(b^\top x_i)\sigma(b^\top x_{i'}) + \sigma'(b^\top x_i)\sigma'(b^\top x_{i'})x_i^\top x_{i'}] \text{ as } m \to$$

  We only need to show that smallest eigenvalue of the kernel matrix: $K = (k(x_i, x_{i'})) \in \mathbb{R}^{n\times n}$ is away from zero.

Characterization of the whole GD trajectory

The following theorem concerns the function class that the GD solutions can represent. Let $f_m(x; a, B_0) = \sum_{j=1}^m a_j \sigma(b_j(0)^\top x)$ be the random feature model (RFM).

Theorem 11 (E, Ma, Wu, 2019)

Let $\theta_t = \{a(t), B(t)\}$ be the GD solution at time t, and $\tilde{a}(t)$ be the GD solution of RFM with zero initialization. For any $\delta \in (0, 1)$, assume that $m \gtrsim \frac{n^4}{\lambda_n^2(K)} \log(\frac{n^2}{\delta})$. Then, with probability $1 - \delta$ over the random initialization, we have

$$\sup_{t \in [0,\infty], x \in \mathbb{S}^{d-1}} |f_m(x; a(t), B(t)) - f_m(x; \tilde{a}(t); B_0)| \leq \frac{1 + \sqrt{\log(1/\delta)}}{\lambda_n(K)\sqrt{m}}.$$

Remark:

- The theorem implies that the GD trajectory of a wide 2LNN is **uniformly** close to that of the associate RFM.
- The result is implicit in the proof of convergence result: $\theta(t) - \theta_0 \ll 1$.

Proof sketch

$$\begin{aligned}
|f_m(x;a(t),B(t)) - f_m(x;a(t),B_0)| &\le \sum_{s=1}^m a_s(t)|\sigma(b_s(t)^\top x) - \sigma(b_s(0)^\top x)| \\
&\le \sum_{s=1}^m a_s(t)\|b_s(t) - b_s(0)\| \\
&\le \frac{1}{2}\sum_{s=1}^m (a_s^2(t) + \|b_s(t) - b_s(0)\|^2) \\
&= \frac{1}{2}\|\theta(t) - \theta_0\|^2 \lesssim \frac{1}{m\lambda_n^2(K)}
\end{aligned}$$

The closeness of $a(t)$ and $\tilde{a}(t)$ is a consequence of the closeness of $B(t)$ and B_0. The proof is lengthy but straightforward.

Compare the NN and RFM under GD dynamics

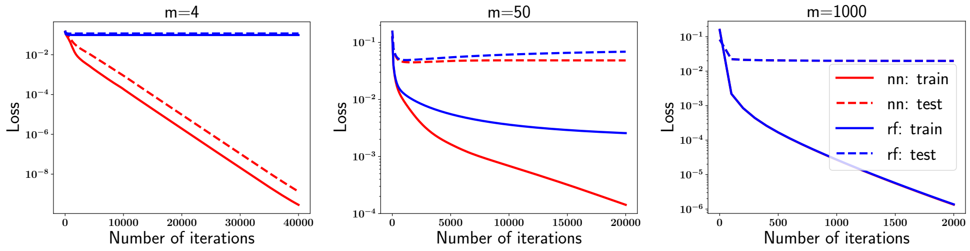


Figure 2: GD dynamics for fitting a single neuron $f^*(x) = \sigma(x_1)$ where $x \in \mathbb{S}^{d-1}$. Here $d = 20, n = 50$. **Left:** $m = 4$; **Middle:** $m = 50$; **Right:** $m = 1000$.

Comparison between the implicit and explicit regularization

Consider the explicit regularization:

$$\min_{\theta} \hat{\mathcal{R}}(\theta) + \lambda \sqrt{\frac{\log(d)}{n}} \sum_{j=1}^{m} |a_j| \|b_j\|_2.$$

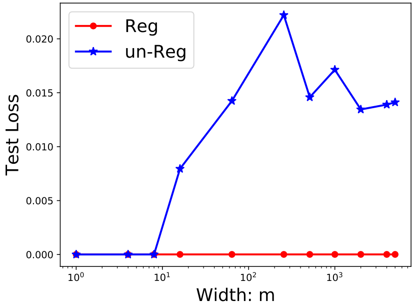


Figure 3: Fitting a single neuron $f^*(x) = \sigma(x_1)$ where $x \in \mathbb{S}^{d-1}$. Here $d = 20, n = 50$. The nn-Reg solution is the GD solution without any regularization.

Lazy training

- In the previous analysis, the main insight is that for the highly over-parameterized setting, the perturbation satisfies $\|\theta - \theta_0\| \ll 1$.
- Why is a small deviation enough? Consider the expansion around the initialization:

$$f_m(x;\theta) = f_m(x;\theta_0) + \langle \theta - \theta_0, \nabla f_m(x;\theta_0)\rangle + o(\|\theta - \theta_0\|). \tag{24}$$

- Note that the each entry of $\theta - \theta_0$ is in the order of $O(1/m)$ is enough to ensure the change of output: $f_m(x;\theta) - f_m(x;\theta_0) \sim 1$. Meanwhile, $\|\theta - \theta_0\| = O(1/\sqrt{m}) \ll 1$.
- So essentially, only the linear part contributes to the final model. In our case, the linear part is a RFM.
- In the literature, training methods that essentially only explore the linear part of a nonlinear model to fit data are called **lazy training** (Chizat, Oyallon, Bach, 2018).

Neural tangent kernel

- In the lazy training regime, the model essentially performs kernel method with the kernel given by:
$$k_m(x, x') = \langle \nabla f_m(x; \theta_0), \nabla f_m(x; \theta_0) \rangle,$$
which is called tangent kernel.
- Large width limit: For neural network models $f_m(\cdot; \theta)$, k_m often has a limit with a proper rescaling:
$$k(x, x') = \lim_{m \to \infty} m^{-\alpha} k_m(x, x').$$
$k(\cdot, \cdot)$ is called the **neural tangent kernel** (NTK) (Jacot, Gabriel and Hongler, 2018).

Multi-layer fully-connected nets

The observation that GD only performs lazy training can be extended to general wide neural nets. The proof is similar to the case of 2LNN and can be summarized as follows.

- Recall that
$$\frac{d\|\mathbf{e}(t)\|^2}{dt} = -2\mathbf{e}(t)^\top G(\theta(t))\mathbf{e}(t). \tag{25}$$
- First show that if m is sufficiently large, at initialization $\lambda_n(G(\theta_0)) \geq m\lambda_n(K)$, where K is the kernel matrix of NTK. Assume that $\lambda_n(K) > 0$ (Justify it).
- Let $I(\theta_0)$ be the ball around the initialization such that the $\lambda_n(G(\theta)) \geq m\lambda_n(K)/2$. Let t_0 be the time that $\theta(t)$ first leaves the ball. Then, for any $t \in [0, t_0]$, we have $\hat{R}_n(\theta(t)) \leq e^{-cm\lambda_n(K)t}\hat{R}_n(\theta_0)$ for a constant $c > 0$.
- The combination of exponential convergence and continuity of $\hat{\mathcal{R}}$ implies
$$\|\theta_t - \theta_0\| \leq \int_0^{t_0} \|\nabla\hat{\mathcal{R}}(\theta_t')\|dt' \leq \int_0^{t_0} C(\|\theta(t)\|)\sqrt{\hat{\mathcal{R}}(\theta(t'))}dt' \leq \frac{poly(n, \lambda_n(K))}{m}.$$
- When m is sufficiently large, we must have $\theta_t \in \mathcal{I}(\theta_0)$ for any $t \geq 0$.

We refer to (Arora et al, 2019) for a detailed proof for multi-layer fully-connected nets.

Summary

- Under conventional setting, neural nets trained by GD converges to kernel method. Moreover, the convergence is uniform in time. It means only when f^* lies in the appropriate RKHS, the GD solution can generalize well.
- What happens when the network is less over-parameterized?
- Can we still learn larger class of target functions in the over-parameterized regime?