Analysis of Neural Networks

Mathematical Analysis of Machine Learning Algorithms
(Chapter 11)

Two-layer Neural Networks

Two-Layer Neural Networks with Real-Valued Output

Observe d-dimensional input vector $x \in \mathbb{R}^d$:

$$f_m(w, x) = \sum_{j=1}^{m} u_j h(\theta_j^\top x + b_j), \tag{1}$$

where $x \in \mathbb{R}^d$, $\theta_j \in \mathbb{R}^d$, $b_j \in \mathbb{R}$, $u_j \in \mathbb{R}$, and

$$w = \{[u_j, \theta_j, b_j] : j = 1, \ldots, m\}.$$

Function $h(\cdot)$: activation function, and popular choices include

- rectified linear unit (ReLU) $h(z) = \max(0, z)$
- sigmoid $h(z) = 1/(1 + e^{-z})$

Deep Neural Networks

K-layer fully-connected DNN with Real-Valued Output

- Let $m^{(0)} = d$ and $m^{(K)} = 1$
- Define recursively

$$
\begin{aligned}
x_j^{(0)} =& x_j \quad (j = 1, \ldots, m^{(0)}), \\
x_j^{(k)} =& h\left(\sum_{j'=1}^{m^{(k-1)}} \theta_{j,j'}^{(k)} x_{j'}^{(k-1)} + b_j^{(k)}\right) \begin{cases} j = 1, \ldots, m^{(k)}, \\ k = 1, 2, \ldots, K-1 \end{cases} \\
f(x) =& x_1^{(K)} = \sum_{j=1}^{m^{(K-1)}} u_j x_j^{(K-1)}.
\end{aligned}
$$

- $w = \{[u_j, \theta_{j,j'}^{(k)}, b_j^{(k)}] : j, j', k\}$: model parameters
- $m^{(k)}$: the number of hidden units at layer k.

Universal Approximation

For two-layer neural networks, we have the following result.

Theorem 1 (Universal Approximation Thm 11.1)

If h is a non-polynomial continuous function, then the two-layer neural network function class in (1) is dense in $C^0(K)$ for all compact subsets K of $\mathbb{R}^d$, where $C^0(K)$ denotes the set of continuous functions on K.

M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken (1993). "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function". In: *Neural networks* 6.6, pp. 861–867 .

Barron Class

A more refined result can be obtained for functions with a certain smoothness property in Fourier representation.

Definition 2

Consider a real valued function $f(x) : \mathbb{R}^d \to \mathbb{R}$. Assume that $f \in L_1(\mathbb{R}^d)$ has the following Fourier representation:

$$f(x) = \int_{\mathbb{R}^d} e^{i\omega^\top x} \tilde{f}(\omega) \, d\omega,$$

where $\tilde{f}(\omega)$ is the Fourier transform of $f(x)$ that may be a complex function. Define

$$C(f) = \int_{\mathbb{R}^d} \|\omega\|_2 |\tilde{f}(\omega)| d\omega.$$

Refined Universal Approximation

Theorem 3 (Thm 11.3)

If $h(z)$ is a bounded measurable function on the real line for which $\lim_{z\to-\infty} h(z) = 0$ and $\lim_{z\to\infty} h(z) = 1$. Consider $B_r = \{x \in \mathbb{R}^d : \|x\|_2 \le r\}$, and let f be a real-valued function defined on B_r such that $C(f) < \infty$. Then there exists a neural network (1) such that

$$\int (f(x) - f(0) - f_m(w, x))^2 \, d\mu(x) \le \frac{(2rC(f))^2}{m},$$

where μ is an arbitrary probability measure on B_r.

Note that we can take any b so that $h(b) \neq 0$, and

$$f(0) = (f(0)/h(b))h(0^\top x + b).$$

It follows that if $f(x) - f(0)$ can be represented by a two-layer neural network (1) with m neurons, then $f(x)$ can be represented by a two-layer neural network (1) with $m + 1$ neurons.

From Shallow to Deep Representation

Two Layer Neural Networks

- A function f can be efficiently represented by a two-layer neural network if $C(f)$ is small [Barron, 1993].
- Target functions with large $C(f)$: exponentially many nodes to represent with two layer neural networks.

Deep Representation

- Reduce the number of nodes needed to represent complexity functions.
- Related to the ability of deep neural networks to form more complex composite features from simpler features.

Benefit of Deep Representation

Theorem 4 (Thm 11.4)

Consider any integer $k \geq 3$. There exists $f(x) : [0,1] \to [0,1]$ computed by a $2k^2$-layer neural network with standard ReLU activation function, with no more than 2 neurons per layer so that

$$\int_0^1 |f(x) - g(x)| dx \geq \frac{1}{16}, \tag{2}$$

where g is any function of a ReLU network with no more than k layers and $\leq 2^{k-2}$ nodes per layer.

Proof of Theorem 4 (I/III)

We will briefly explain the high-level intuition of Telgarsky 2016, which indicates what kind of functions are difficult to approximate with shallow neural networks. Consider the case of $d = 1$, a specific construction of a hard function $f(x)$ for shallow neural networks is via the function composition of the triangle function $f_0(x) = \max(0, \min(2x, 2(1 - x)))$ on $[0, 1]$. We may define $f_k(x) = f_0(f_{k-1}(x))$ with $k \geq 1$, as illustrated in Figure 1.

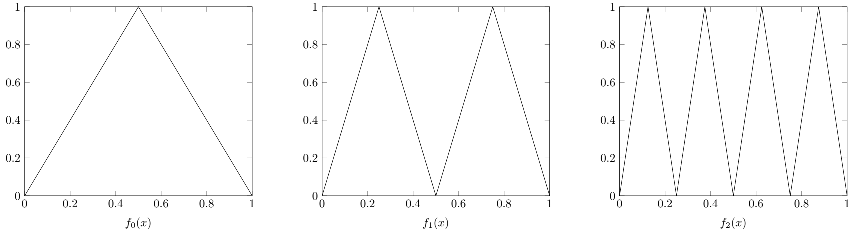

Figure: Plot of $f_k(x)$ with $k = 0, 1, 2$

Proof of Theorem 4 (II/III)

Since $f_0(x)$ can be represented by a two-layer neural ReLU network with no more than two nodes per layer as $f_0(x) = 2\max(0, x) - 4\max(0, x - 0.5)$ in $[0, 1]$, $f_k(x)$ can be represented by a $2k$-layer neural network with no more than two nodes per layer.
It can be seen that $f_k(x)$ contains 2^k points that reach values of 1, and $2^k + 1$ points that reach values of 0. It follows that the number of solution segments of $f_k(x) = 0.5$, referred to as its crossing number, is 2^{k+1}.
It is easy to show that $f(x) - f_k(x)$ cannot be approximated well by a function $g(x)$ with crossing number $< 2^k$ in that the approximation error is lower bounded by (2).

Proof of Theorem 4 (III/III)

Therefore in order to show that $f_k(x)$ cannot be approximated efficiently by shallow neural networks, we only need to show that the function of a shallow neural network cannot have many crossings unless it contains exponentially many nodes.

Specifically, it can be shown (left as Exercise 11.2) that an ℓ-layer ReLu network with no more than m ReLU nodes per layer has a crossing number of no larger than $2(2m)^{\ell}$.

It follows that if a neural network can approximate $f_{k^2}(x)$ well, then $(2m) \geq 2^{(k^2-1)/\ell}$. Therefore the node number $m > 2^{k-2}$if $\ell \leq k$.

Random Feature Method

Random Feature Model

We assume that $\{\theta_j : j = 1, \ldots, m\}$ are m independent samples drawn from a distribution μ on $\mathbb{R}^d$. A typical example is to take μ as a Gaussian distribution.

The two-layer neural network in (1) can be written as

$$f_m(u, x) = \frac{1}{m} \sum_{j=1}^{m} u_j h(\theta_j^\top x). \tag{3}$$

Equivalent to a two-layer neural network, where we do not train parameter θ.

It follows from Theorem 1 that (3) is universal with appropriate *h*.

Overparameterized: Infinity Width Network

As $m \to \infty$, the law of large numbers implies

$$f_\infty(x) = \mathbb{E}_{\theta \sim \mu} u(\theta) h(\theta^\top x), \tag{4}$$

where $u(\theta)$ is a weight function.

- Treat both $u(\theta)$ and $h(\theta^\top x)$ as infinite dimensional vectors indexed by θ.
- May regard the limiting function class (4) as a linear system, in which we learn the infinite dimensional linear weight $u(\theta)$.

Random Feature Kernel

Consider the L_2 regularization for random feature method, where the function class is given by

$$\mathbb{E}_{\theta\sim\mu}|u(\theta)|^2 \leq A^2.$$

This function class induces a kernel class.

Proposition 5 (Prop 11.5)

Consider any probability measure μ on $\mathbb{R}^d$. The function class (4) *with L_2 regularization*

$$\|f\|_2 = \left(\mathbb{E}_{\theta\sim\mu}|u(\theta)|^2\right)^{1/2}$$

is equivalent to the RKHS function class defined in Definition 9.4 with kernel

$$k_\infty(x,x') = \mathbb{E}_{\theta\sim\mu} h(\theta,x)h(\theta,x').$$

Rademacher Complexity

Corollary 6 (Infinite-Width Network, Cor 11.7)

$$\textit{Let} \quad \mathcal{F}_A^2 = \{\mathbb{E}_{\theta\sim\mu} u(\theta) h(\theta^\top x) : \mathbb{E}_{\theta\sim\mu} u(\theta)^2 \le A^2\},$$

$$\textit{then} \quad R_n(\mathcal{F}_A^2, \mathcal{D}) \le A\sqrt{\frac{\mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{\theta\sim\mu} h(\theta^\top x)^2}{n}}.$$

Corollary 7 (Finite-Width Network, Cor 11.8)

$$\textit{Let} \quad \mathcal{F}_{A,m}^2 = \left\{ \frac{1}{m}\sum_{j=1}^{m} u_j h(\theta_j^\top x) : \|u\|_2 \le \sqrt{m}A \right\},$$

$$\textit{then} \quad R_n(\mathcal{F}_{A,m}^2, \mathcal{D}) \le A\sqrt{\frac{\mathbb{E}_{x\sim\mathcal{D}} \sum_{j=1}^m h(\theta_j^\top x)^2}{mn}}.$$

L_1 Regularization: Infinity-Width Network

Proposition 8 (Infinity-Width Network, Prop 11.9)

Let $\mathcal{F} = \{h(\theta^\top x) : \theta \in \mathbb{R}^d\}$*. Let*

$$\mathcal{F}_A^1 = \{\mathbb{E}_{\theta\sim\mu} u(\theta) h(\theta^\top x) : \mathbb{E}_{\theta\sim\mu}|u(\theta)| \leq A\}.$$

Then for all monotone function $h(\cdot)$ *with* $h(\cdot) \in [-M, M]$*:*

$$R(\mathcal{F}_A^1, \mathcal{S}_n) \leq AM \frac{32\sqrt{d+1}}{\sqrt{n}}.$$

Proof of Proposition 8

Since $h(\cdot)$ is monotone, we know that $\mathcal{F}$ is a VC-subgraph class with VC dimension $d + 1$ (see Chapter 5).
Therefore from Theorem 5.11 and the calculation in Example 6.26, we obtain

$$R(\mathcal{F}, \mathcal{S}_n) \leq M \frac{16\sqrt{d+1}}{\sqrt{n}}.$$

The desired bound now follows from Theorem 10.8:

$$R(\mathcal{F}_A^1, \mathcal{S}_n) \leq AR(\mathcal{F}_\pm, \mathcal{S}_n) \leq 2AR(\mathcal{F}, \mathcal{S}_n).$$

L_1 Regularization: Finite-Width Network

Corollary 9 (Finite Width Network, Cor 11.10)

Let

$$\mathcal{F}^1_{A,m} = \left\{ \frac{1}{m} \sum_{j=1}^{m} u_j h(\theta_j^\top x) : \|u\|_1 \le mA \right\}.$$

Assume that $h(\cdot) \in [-M, M]$ *and* $h(\cdot)$ *is monotone. Then*

$$R(\mathcal{F}^1_{A,m}, \mathcal{S}_n) \le AM \frac{32\sqrt{d+1}}{\sqrt{n}}.$$

The proof is the same as that of Proposition 8.

Example

Example 10

Consider a target function which is represented by a single neuron

$$f_*(x) = u_1 h(\theta_1^\top x),$$

with $|u_1| \leq 1$ and $h(\cdot) \in [-1, 1]$. Then $f_* \in \mathcal{F}^1_{1,m}$ for all $m \geq 1$.
Corollary 9 implies that the Rademacher complexity using L_1 regularization is $R_n(\mathcal{F}^1_{1,m}) = O(\sqrt{d/n})$, which is well-behaved when $m \to \infty$.
However, if we employ L_2 regularization, then $f_* \in \mathcal{F}^2_{\sqrt{m},m}$. The corresponding Rademacher complexity bound becomes $R_n(\mathcal{F}^2_{\sqrt{m},m}) \leq \sqrt{m/n}$, which becomes infinity when $m \to \infty$.

Neural Tangent Kernel

In the random feature approach

- The bottom layer model parameter θ is fixed, and only the top layer model parameter u is trained.

Practical applications of neural networks

- both model parameter θ and parameter u are trained jointly.

It is possible to generalize the kernel view to handle this case, which leads to the concept of neural tangent kernel (NTK).

The key idea of NTK is to linearly approximate the two-layer NN in both θ and u around a small neighborhood around the initialization.

NTK Initialization: Finite Width

To derive NTK, we start with a random initialization of the neural network (1) (again, for simplicity, we assume $b_j = 0$) at $[u, \theta] = [\tilde{u}, \tilde{\theta}]$, which we refer to as the NTK initialization.

Here we independently draw m $d+1$ dimensional model parameters $[\tilde{u}_j, \tilde{\theta}_j] \in \mathbb{R}^{d+1}$ from a probability distribution μ on $\mathbb{R}^{d+1}$. The probability distribution is often chosen as an iid normal distribution.

The resulting initial neural network is given below.

NTK Initialization

With NTK Initialization, we have

$$\tilde{f}_m^{\mathrm{NTK}}(x) = \frac{1}{\sqrt{m}} \sum_{j=1}^{m} \tilde{u}_j h(\tilde{\theta}_j^\top x), \qquad (5)$$

which is similar to the random feature method (3), but $\tilde{u}_j$ is also drawn randomly.

NTK Initialization: Infinity Width

We often choose μ to be a normal distribution with a diagonal covariance matrix. This implies that

$$\mathbb{E}[\tilde{u}_j|\tilde{\theta}_j] = 0.$$

Expected Function Value and Variance of NTK Initialization

Given $x \in \mathbb{R}^d$, the expected value and variance of $\tilde{f}_m^{\mathrm{NTK}}(x)$ are

$$\begin{aligned}\mathbb{E}_\mu[\tilde{f}_m^{\mathrm{NTK}}(x)] =& 0, \\ \mathrm{Var}_\mu[\tilde{f}_m^{\mathrm{NTK}}(x)] =& \mathbb{E}_{[\tilde{u}_0,\tilde{\theta}_0]\sim\mu} \tilde{u}_0^2 h(\tilde{\theta}_0^\top x)^2.\end{aligned}$$

- Variance is finite when the right hand side is finite.
- Variance is independent of m as $m \to \infty$.

This is why we divide by $\sqrt{m}$ instead of by m in NTK.

Properties of NTK Initialization (I/II)

Proposition 11 (Prop 11.12)

Assume that the central limit theorem holds for (5) *(uniformly for all x) as $m \to \infty$. Then as $m \to \infty$, $\tilde{f}_m^{\mathrm{NTK}}(x)$ converges to a Gaussian process $\tilde{f}_\infty^{\mathrm{NTK}}(x)$ with zero-mean and covariance matrix*

$$k(x, x') = \mathbb{E}_{[\tilde{u}_0, \tilde{\theta}_0] \sim \mu} \tilde{u}_0^2 h(\tilde{\theta}_0^\top x) h(\tilde{\theta}_0^\top x').$$

Properties of NTK Initialization (II/II)

Proposition 12 (Prop 11.13)

Consider $\tilde{f}_m^{\mathrm{NTK}}(x)$ defined in (5). Let $h'(z)$ be the derivative of $h(\cdot)$. We have for all x and j:

$$\mathbb{E}\|\nabla_{\tilde{u}_j}\tilde{f}_m^{\mathrm{NTK}}(x)\|_2^2 = \frac{1}{m}\mathbb{E}_{\tilde{\theta}_0\sim\mu} h(\tilde{\theta}_0^\top x)^2,$$
$$\mathbb{E}\|\nabla_{\tilde{\theta}_j}\tilde{f}_m^{\mathrm{NTK}}(x)\|_2^2 = \frac{1}{m}\mathbb{E}_{[\tilde{u}_0,\tilde{\theta}_0]\sim\mu}\tilde{u}_0^2 h'(\tilde{\theta}_0^\top x)^2\|x\|_2^2,$$

where the expectation is with respect to the random initialization. Moreover, for any x, as $m\to\infty$:

$$\|\nabla_{\tilde{u}}\tilde{f}_m^{\mathrm{NTK}}(x)\|_2^2 \xrightarrow{p} \mathbb{E}_{\tilde{\theta}_0\sim\mu} h(\tilde{\theta}_0^\top x)^2,$$
$$\|\nabla_{\tilde{\theta}}\tilde{f}_m^{\mathrm{NTK}}(x)\|_2^2 \xrightarrow{p} \mathbb{E}_{[\tilde{u}_0,\tilde{\theta}_0]\sim\mu}\tilde{u}_0^2 h'(\tilde{\theta}_0^\top x)^2\|x\|_2^2,$$

where the probability is with respect to the random initialization.

NTK Approximation

Let $w = [u, \theta]$ and $\tilde{w} = [\tilde{u}, \tilde{\theta}]$. Let $B_\infty(\tilde{w}, r) = \{w : \|w - \tilde{w}\|_\infty \leq r\}$. Let

$$f_{\text{nn}}(w, x) = \frac{1}{\sqrt{m}} \sum_{j=1}^{m} u_j h(\theta_j^\top x),$$

and we can define its NTK approximation as follows.

NTK Approximation of $f_{\text{nn}}(w, x)$

We define

$$\begin{aligned} f_m^{\text{NTK}}(w, x) =& \tilde{f}_m^{\text{NTK}}(x) + \frac{1}{\sqrt{m}} \sum_{j=1}^{m} \Big[(u_j - \tilde{u}_j) h(\tilde{\theta}_j^\top x) \\ & + \tilde{u}_j h'(\tilde{\theta}_j^\top x)(\theta_j - \tilde{\theta}_j)^\top x \Big]. \end{aligned} \tag{6}$$

When $w \in B_\infty(\tilde{w}, r)$ for a sufficiently small r, we have $f_m^{\text{NTK}}(w, x) \approx f_{\text{nn}}(w, x)$ and $\nabla_w f_m^{\text{NTK}}(w, x) \approx \nabla_w f_{\text{nn}}(w, x)$.

Technical Conditions for NTK Approximation

Assumption 13 (Asm 11.14)

For any x, $\delta \in (0,1)$ and $\epsilon > 0$, there exist $A_0 > 0$, $r_0 > 0$ and $m_0 > 0$ such that when $m > m_0$, with probability at least $1-\delta$ over random initialization, the following events hold uniformly for $w \in B_\infty(\tilde{w}, r_0)$:

- $|\tilde{f}_m^{\mathrm{NTK}}(x)| \le A_0$
- $\|\nabla_w f_m^{\mathrm{NTK}}(w,x)\|_2 + \|\nabla_w f_{\mathrm{nn}}(w,x)\|_2 \le A_0$
- $|f_m^{\mathrm{NTK}}(w,x) - f_{\mathrm{nn}}(w,x)| \le \epsilon$
- $\|\nabla_w f_m^{\mathrm{NTK}}(w,x) - \nabla_w f_{\mathrm{nn}}(w,x)\|_2 \le \epsilon$
- $\|\nabla_w f_m^{\mathrm{NTK}}(w,x)\|_\infty + \|\nabla_w f_{\mathrm{nn}}(w,x)\|_\infty \le m^{-1/4}$

Proposition 14 (Prop 11.15)

Assumption 13 holds for both ReLU and for sigmoid activation functions with Gaussian initialization $\tilde{w} \sim N(0, \sigma^2 I)$.

Proposition 15 (Limiting Kernel, Prop 11.16)

Consider the feature space NTK formulation (6)*. Then* $f_m^{\mathrm{NTK}}(w, x) - \tilde{f}_m^{\mathrm{NTK}}(x)$ *belongs to the RKHS with kernel*

$$k_m^{\mathrm{NTK}}(x, x') = k_{m,1}^{\mathrm{NTK}}(x, x') + k_{m,2}^{\mathrm{NTK}}(x, x'), \qquad \textit{where}$$

$$k_{m,1}^{\mathrm{NTK}}(x, x') = \frac{1}{m} \sum_{j=1}^{m} h(\tilde{\theta}_j^\top x) h(\tilde{\theta}_j^\top x'),$$
$$k_{m,2}^{\mathrm{NTK}}(x, x') = \frac{1}{m} \sum_{j=1}^{m} \tilde{u}_j^2 h'(\tilde{\theta}_j^\top x) h'(\tilde{\theta}_j^\top x') x^\top x'.$$

Moreover, for any x, x'*, as* $m \to \infty$*, we have*

$$k_{m,1}^{\mathrm{NTK}}(x, x') \xrightarrow{p} k_{\infty,1}^{\mathrm{NTK}}(x, x') = \mathbb{E}_{\tilde{\theta}_0 \sim \mu} h(\tilde{\theta}_0^\top x) h(\tilde{\theta}_0^\top x'),$$
$$k_{m,2}^{\mathrm{NTK}}(x, x') \xrightarrow{p} k_{\infty,2}^{\mathrm{NTK}}(x, x') = \mathbb{E}_{[\tilde{u}_0, \tilde{\theta}_0] \sim \mu} \tilde{u}_0^2 h'(\tilde{\theta}_0^\top x) h'(\tilde{\theta}_0^\top x') x^\top x',$$

where the probability is with respect to the random initialization.

NTK Approximation for Arbitrary Function

Theorem 16 (Thm 11.17)

Consider an arbitrary function $f(x)$, and n distinct points $\{X_1, \ldots, X_n\}$. Assume that the limiting NTK kernel $k_\infty^{\mathrm{NTK}}(x, x')$ in Proposition 15 is universal. Consider a two-layer neural network with initialization (5). *Given any $\epsilon > 0$ and $\delta \in (0, 1)$, there exist $A > 0$ and m_0 such that when $m > m_0$, with probability at least $1 - \delta$, there exists $w \in B_\infty(\tilde{w}, r_m)$ that satisfy:*

- *$r_m = A/m^{1/4}$ and $\|w - \tilde{w}\|_2 \leq A$.*
- *$|f_m^{\mathrm{NTK}}(w, X_i) - f(X_i)| \leq \epsilon$ for all $i = 1, \ldots, n$.*

SGD Convergence in NTK Regime

Corollary 17 (Cor 11.18)

Assume that the NTK kernel $k_\infty^{\mathrm{NTK}}(x, x')$ in Proposition 15 is universal. Let $f(x)$ be an arbitrary function, and $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ be n distinct points. Consider a convex loss function $L(f(x), y)$ which is Lipschitz in $f(x)$. There exists $A > 0$ so that the following holds. For any $T > 0$, assume we run SGD from the NTK initialization (5) for T steps with constant learning rate $1/\sqrt{T}$, and return $f_{\mathrm{nn}}(w, x)$ with w chosen uniformly at randomly from the SGD iterates. Then as $m \to \infty$, $\|w - \tilde{w}\|_\infty \xrightarrow{p} 0$ and

$$\mathbb{E}_w \frac{1}{n} \sum_{i=1}^{n} L(f_{\mathrm{nn}}(w, X_i), Y_i) \leq \frac{1}{n} \sum_{i=1}^{n} L(f(X_i), Y_i) + \frac{A}{\sqrt{T}} + o_p(1),$$

where $\mathbb{E}_w$ indicates the randomness from the SGD iterates, and the convergence in probability is with respect to the randomness in the initialization.

NTK Kernel: Rademacher Complexity

Corollary 18 (Cor 11.19)

Let

$$\mathcal{F}_A^{\mathrm{NTK}}(\tilde{w}) = \{f_m^{\mathrm{NTK}}(w, x) : \|w - \tilde{w}\|_2^2 \le A^2\},$$

then

$$R_n(\mathcal{F}_A^{\mathrm{NTK}}(\tilde{w}, \mathcal{D}) \le A\sqrt{\frac{\mathbb{E}_{x\sim\mathcal{D}} k_m^{\mathrm{NTK}}(x, x)}{n}}.$$

This result is a special case of Theorem 9.20.

Mean Field Formulation

Mean Field Formulation

We consider a different normalization of (1) (still ignoring b_j) as

$$f_m(x) = \frac{\alpha}{m} \sum_{j=1}^{m} u_j h(\theta_j^\top x), \tag{7}$$

with a scaling constant $\alpha > 0$. We assume that $\theta_j \in \mathbb{R}^d$ and $u_j \in \mathbb{R}$.

- For mean filed formulation, as $m \to \infty$:

$$f_{\mathrm{mf}}(q, x) = \alpha \mathbb{E}_{[u,\theta]\sim q}\, u h(\theta^\top x). \tag{8}$$

- The NTK approximation of two-layer NN does not have a continuous integral formulation due to the $1/\sqrt{m}$ normalization.

Continuous Mean-Field Optimization

Consider ERM with entropy regularization for mean-field:

$$\min_q \left[\frac{1}{n} \sum_{i=1}^{n} L(f_{\mathrm{mf}}(q, X_i), Y_i) + \mathbb{E}_{[u,\theta]\sim q} r(u, \theta) + \lambda \mathbb{E}_{[u,\theta]\sim q} \ln q([u, \theta]) \right],$$

where $f_{\mathrm{mf}}(q, x)$ is given by (8), and $r([u, \theta])$ is an appropriately chosen regularization term such as L_2 regularization.

Theorem 19 (Informal Version of Thm 11.20)

Assume that $L(f, y)$ is convex in f. Under suitable conditions, noisy GD to solve entropy regularized ERM with continuous mean-field formulation converges to the global optimal solution.

Rademacher Complexity

Proposition 20 (Prop 11.23)

$$\textit{Let} \qquad \mathcal{F}_A^{\mathrm{mf}} = \{\mathbb{E}_{[u,\theta]\sim q}\, u h(\theta^\top x) : \mathbb{E}_{u\sim q}|u| \le A\}, \qquad \textit{then}$$

$$R(\mathcal{F}_A^{\mathrm{mf}}, \mathcal{S}_n) \le AM\frac{32\sqrt{d+1}}{\sqrt{n}}.$$

Corollary 21 (Cor 11.25)

$$\textit{Let} \qquad \mathcal{F}_{A,m}^{\mathrm{mf}} = \left\{ \frac{1}{m}\sum_{j=1}^{m} u_j h(\theta_j^\top x) : \|u\|_1 \le mA \right\}.$$

Assume that $h(\cdot) \in [-M, M]$ *and* $h(\cdot)$ *is monotone. Then*

$$R(\mathcal{F}_{A,m}^{\mathrm{mf}}, \mathcal{S}_n) \le AM\frac{32\sqrt{d+1}}{\sqrt{n}}.$$

L_1 Regularized Deep Neural Networks

We define a deep function class deep neural networks recursively:

$$\mathcal{F}^{(1)} = \{\theta^\top x : \theta \in \Theta\},$$

and for $k = 2, \ldots, K$, using the notation of Section 10.2, we define

$$\begin{aligned} \mathcal{F}^{(k)} =& \mathcal{F}_{A_k, L_1}(h \circ \mathcal{F}^{(k-1)}) \\ =& \left\{ \sum_{j=1}^{m} w_j h(f_j(x)) : \|w\|_1 \leq A_k,\ f_j \in \mathcal{F}^{(k-1)}, m > 0 \right\}. \end{aligned} \tag{9}$$

When $A_1, \ldots, A_k$ are sufficiently large, then it is clear that any function $f(x)$ that can be represented by a deep K layer neural network that belongs to $\mathcal{F}^{(K)}$.

The representation allows an arbitrary large m, and thus can handle continuous deep neural networks.

Rademacher Complexity of L_1 Regularized DNN

Theorem 22 (Thm 11.26)

Consider K-layer neural networks defined by (9). Assume that $h(\theta^\top x) \in [-M, M]$ for all $\theta \in \Theta$ and $x \in \mathcal{X}$. Assume also that h is 1-Lipschitz and monotone. Then there exists a constant C, such that for all distribution D on $\mathcal{X}$, we have

$$R(\mathcal{F}^{(k)}, \mathcal{S}_n) \leq AM\frac{32\sqrt{d+1}}{\sqrt{n}},$$

where $A = 2^{k-2}\prod_{\ell=2}^{k} A_\ell$.

Proof of Theorem 22

We prove the statement by induction. The case $k = 2$ follows from Corollary 9.

Assume the statement holds at layer $k - 1$, then at layer k. We have

$$\begin{aligned} R(\mathcal{F}^{(k)}, \mathcal{S}_n) \leq & A_k[R(h \circ \mathcal{F}^{(k-1)}, \mathcal{S}_n) + R(-h \circ \mathcal{F}^{(k-1)}, \mathcal{S}_n)] \\ \leq & 2A_k R(\mathcal{F}^{(k-1)}, \mathcal{S}_n). \end{aligned}$$

The first inequality used Theorem 10.8. The second inequality used Theorem 6.28 (Rademacher comparison) with $\gamma_i = 1$ and $h = 0$. Now by using induction, we obtain the desired bound.

Consequences

- For the ReLu function, the L_1 regularization can be moved to the top layer, which simplifies the theory.
- The learning complexity for the function composition example considered in Theorem 4 is still high if we measure it by L_1 regularization. This is not surprising because functions with exponentially many crossing numbers are complex.
- The generalization bound in Theorem 22 is that the generalization performance does not depend on the number of neurons. Therefore the more neurons we use, the better. This is consistent with the practical observations.

Double Descent

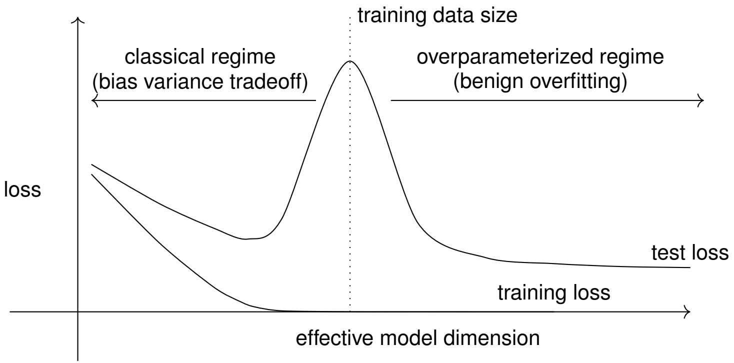

Figure: Double descent curve

A Simple Model

We consider the following simple problem in the Bayesian setting with $(X, Y) \in \mathbb{R}^d \times \mathbb{R}$:

$$\begin{cases} Y & = w_*^\top X + \epsilon \\ X & \sim N(0, I_{d\times d}) \\ \epsilon & \sim N(0, \sigma^2) \\ w_* & \sim N\left(0, \frac{\tau^2}{d} I_{d\times d}\right) \end{cases} \tag{10}$$

We consider least squares regression in the overparameterized setting, where we observe $n \ll d$ training data $\{(X_i, Y_i) : i = 1, \ldots, n\}$.

Ridge Regression Method

The optimal Bayes estimator that minimizes the quantity is given by ridge regression:

$$\hat{w} = \arg\min_{w} \left[\sum_{i=1}^{n} (w^\top X_i - Y_i)^2 + \lambda d \|w\|_2^2 \right] \tag{11}$$

with $\lambda = \sigma^2/\tau^2$.

We consider general ridge regression with different λ, and we want to achieve small test loss

$$\mathbb{E}_{(X,Y)\sim\mathcal{D}}(\hat{w}^\top X - Y)^2 = \|\hat{w} - w_*\|_2^2 + \sigma^2.$$

Theorem 23 (Overparameterized Regime of Thm 11.27)

Consider (10) *with fixed* τ, σ *and* $n \ll d$.
Let $X = [X_1, \ldots, X_n]$ *be the* $d \times n$ *data matrix with response* $Y = [Y_1, \ldots, Y_n]^\top$. *Given any* $\delta \in (0,1)$, *let*

$$c = \sqrt{\frac{n}{d}} + \sqrt{\frac{2\ln(2/\delta)}{d}} < 1.$$

Then with probability $1 - \delta$ *over the random choice of* X, *the following statements hold for the ridge regression estimator* (11). *There exists* $c', c'' \in [-c, c]$ *such that*

$$\mathbb{E}\left[\frac{1}{n}\|X^\top \hat{w} - Y\|_2^2 \middle| X\right] = \frac{\lambda^2(\tau^2(1+c')^2 + \sigma^2)}{((1+c')^2 + \lambda)^2},$$

$$\mathbb{E}\left[\|\hat{w} - w_*\|_2^2 + \sigma^2 \middle| X\right] = \sigma^2 + \left(1 - \frac{n}{d}\right)\tau^2 + \frac{n}{d} \cdot \frac{(\lambda^2\tau^2 + (1+c'')^2\sigma^2)}{((1+c'')^2 + \lambda)^2}.$$

The conditional expectation is with respect to both w_* *and* $\{Y_i\}$.

Consequence: Optimal λ

Consider the overparameterized regime, where $n/d \ll 1$ and $c \approx 0$. The expected test loss is minimized with $\lambda = \sigma^2/\tau^2$, which corresponds to the choice of the optimal Bayes estimator. With $\lambda = \sigma^2/\tau^2$ and $c \approx 0$, we have

$$\text{training loss} \approx \frac{\sigma^4}{\tau^2 + \sigma^2},$$
$$\text{test loss} \approx \sigma^2 + \frac{n/d}{1 + \sigma^2/\tau^2}\sigma^2 + \left(1 - \frac{n}{d}\right)\tau^2.$$

If $\sigma \ll \tau$, then the training loss is approximately $\sigma^2(\sigma/\tau)^2$. It is significantly smaller than the test loss, which is approximately $(1 + n/d)\sigma^2 + (1 - n/d)\tau^2$.

- Significant overfitting is necessary to achieve optimal performance.
- Variance term becomes $(n/d)\sigma^2$ in the overparameterized case.

Compare to Minimum Norm Estimator

Let $\lambda = 0$, corresponding to the minimum norm estimator, which is the focus in the recent literature.

Training loss is zero, while while test loss decreases as d increases.

$$\text{training loss} = 0,$$
$$\text{test loss} = \sigma^2 + \frac{n/d}{(1+c'')^2}\sigma^2 + \left(1 - \frac{n}{d}\right)\tau^2.$$

When σ/τ is small, the test loss achieved at $\lambda = 0$ is close to the optimal test loss achieved at $\lambda = \sigma^2/\tau^2$ up to a difference of $O((n/d)\sigma^2)$.

The phenomenon that overfitting the noise is required to achieve near optimal test performance is often referred to as *benign overfitting*.

Summary (Chapter 11)

- Neural Network Formulations
- Universal Approximation: shallow
- Benefit of Deep networks: more compact approximation
- Random Feature Approach: L_2 versus L_1 Regularization
- NTK Formulation (L_2 and kernel analysis)
- Mean Field Formulation (L_1 and entropy analysis)
- Deep Neural Networks with L_1 analysis
- Double Descent and Benign Overfitting