Approximation-based Statistical Guarantees for Deep Learning

김지수 (Jisu KIM)

딥러닝의 통계적 이해 (Deep Learning: Statistical Perspective), 2024년 2학기

This lecture note is a combination of Prof. Joong-Ho Won's "Deep Learning: Statistical Perspective" with other lecture notes. Main references are:

Tong Zhang, Mathematical Analysis of Machine Learning Algorithms, https://tongzhang-ml.org/lt-book.html

Matus Telgarsky, Deep learning theory lecture notes, https://mjt.cs.illinois.edu/dlt/

1 Review

1.1 Basic Model for Supervised Learning

- Input(입력) / Covariate(설명 변수) : $x \in \mathbb{R}^d$, so $x = (x_1, \ldots, x_d)$.
- Output(출력) / Response(반응 변수): $y \in \mathcal{Y}$. If y is categorical, then supervised learning is "classification", and if y is continuous, then supervised learning is "regression".
- Model(모형) :

  $$y \approx f(x).$$

  If we include the error ϵ to the model, then it can be also written as

  $$y = \phi(f(x), \epsilon).$$

  For many cases, we assume additive noise, so

  $$y = f(x) + \epsilon.$$

- Assumption(가정): f belongs to a family of functions $\mathcal{M}$. This is the assumption of a model: a model can be still used when the corresponding assumption is not satisfied in your data.
- Loss function(손실 함수): $\ell(y, a)$. A loss function measures the difference between estimated and true values for an instance of data.
- Training data(학습 자료): $\mathcal{T} = \{(y_i, x_i), i = 1, \ldots, n\}$, where (y_i, x_i) is a sample from a probability distribution P_i. For many cases we assume i.i.d., or x_i's are fixed and y_i's are i.i.d..
- Goal(목적): we want to find f that minimizes the expected prediction error,

  $$f^0 = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{(Y,X)\sim P} \left[\ell(Y, f(X))\right].$$

  Here, $\mathcal{F}$ can be different from $\mathcal{M}$; $\mathcal{F}$ can be smaller then $\mathcal{M}$.
- Prediction model(예측 모형): f^0 is unknown, so we estimate f^0 by $\hat{f}$ using data. For many cases we minimizes on the empirical prediction error, that is taking the expectation on the empirical distribution $P_n = \frac{1}{n}\sum_{i=1}^n \delta_{(Y_i,X_i)}$.

  $$\hat{f} = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{P_n} \left[\ell(Y, f(X))\right] = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n \ell(Y_i, f(X_i)).$$

- Prediction(예측): if $\hat{f}$ is a predicted function, and x is a new input, then we predict unknown y by $\hat{f}(x)$.

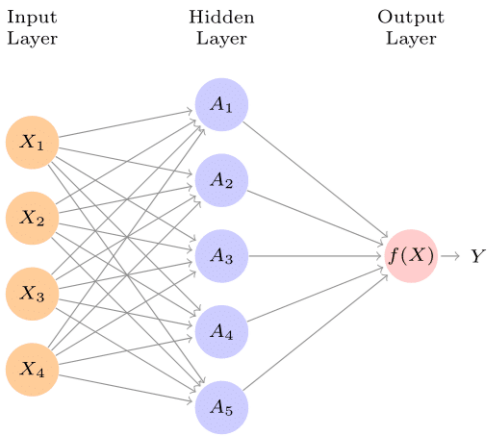

Figure 1: Neural network with a single hidden layer. The hidden layer computes activations $A_j = \sigma_j(x)$ that are nonlinear transformations of linear combinations of the inputs $x_1, \ldots, x_d$. Hence these A_j are not directly observed. The functions σ_j are not fixed in advance, but are learned during the training of the network. The output layer is a linear model that uses these activations A_j as inputs, resulting in a function $f(x)$. Figure 10.1 from [3].

1.2 Two Layer Neural Networks

A two-layer neural network takes an input vector of d variables $x = (x_1, x_2, \ldots, x_d)$ and builds a nonlinear function $f(x)$ to predict the response $y \in \mathbb{R}^D$. What distinguishes neural networks from other nonlinear methods is the particular structure of the model:

$$f(x) = f_\theta(x) = g\left(\beta_0 + \sum_{j=1}^{m} \beta_j \sigma(b_j + w_j^\top x)\right),$$

where $x \in \mathbb{R}^d, b_j \in \mathbb{R}, w_j \in \mathbb{R}^d, \beta_0 \in \mathbb{R}^D, \beta_j \in \mathbb{R}^D$. See Figure 1.

- $\theta = \{[\beta, a_j, b_j, w_j] : j = 1, \ldots, m\}$ denotes the set of model parameters.
- $x_1, \ldots, x_d$ together is called an input layer.
- $A_j \coloneqq \sigma_j(x) = \sigma(b_j + w_j^\top x)$ is called an activation.
- $A_1, \ldots, A_m$ together is called a hidden layer or hidden unit; m is the number of hidden nodes.
- $f(x)$ is called an output layer.
- g is an output function. Examples are:
  - softmax $g_i(x) = \exp(x_i)/\sum_{l=1}^{D} \exp(x_l)$ for classification. The softmax function estimates the conditional probability $g_i(x) = P(y = i|x)$.
  - identity/linear $g(x) = x$ for regression.
  - threshold $g_i(x) = I(x_i > 0)$
- σ is called an activation function. Examples are:
  - sigmoid $\sigma(x) = 1/(1 + e^{-x})$ (see Figure 2)
  - rectified linear (ReLU) $\sigma(x) = \max\{0, x\}$ (see Figure 2)
  - identity/linear $\sigma(x) = x$
  - threshold $\sigma(x) = I(x > 0)$, threshold gives a direct multi-layer extension of the perceptron (as considered by Rosenblatt).

Activation functions in hidden layers are typically nonlinear, otherwise the model collapses to a linear model. So the activations are like derived features - nonlinear transformations of linear combinations of the features.

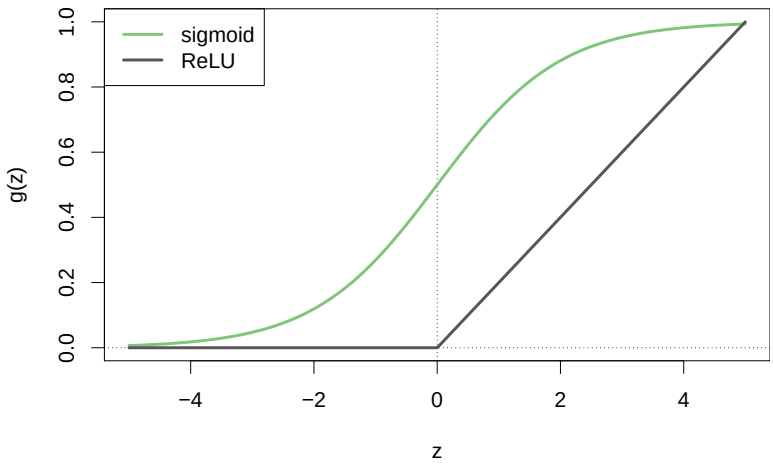

Figure 2: Activation functions. The piecewise-linear ReLU function is popular for its efficiency and computability. We have scaled it down by a factor of five for ease of comparison. Figure 10.2 from [3].

2 Notation and Goal

From here, we only consider regression problem, so $g(x) = x$. We assume $\beta_0 = 0$. Hence, for the two-layer neural network with the width of the hidden layer m and activation function σ, the function space we consider is

$$\mathcal{F}_{m,\sigma} = \left\{ f_\theta : f_\theta(x) = \sum_{j=1}^{m} \beta_j \sigma(b_j + w_j^\top x) \right\},$$

and if we consider all two-layer neural network with arbitrary width, then

$$\mathcal{F}_\sigma = \bigcup_{m=1}^{\infty} \mathcal{F}_{m,h} = \left\{ f_\theta : f_\theta(x) = \sum_{j=1}^{m} \beta_j \sigma(b_j + w_j^\top x),\ m \in \mathbb{N} \right\}.$$

Suppose the true regression function f_* is in a function class $\mathcal{M}$, so

$$y \approx f_*(x), \qquad f_* \in \mathcal{M}.$$

Suppose are using the ℓ_2-loss, so we find f among deep neural network class $\mathcal{F}$ that minimizes the expected risk (평균위험),

$$f^0 = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{(Y,X)\sim P} \left[(y - f(x))^2 \right].$$

f_0 is the expected risk mimizing function (평균위험최소함수). And we estimate f^0 by $\hat{f}$ using data by minimizes on the empirical risk (경험위험) on training dataset, so

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2.$$

$\hat{f}$ is the empirical risk mimizing function (경험위험최소함수). And we set $\tilde{f}$ be the approximation of $\hat{f}$ by optimization(최적화); $\tilde{f}$ is the learned function (학습된 함수).

So there are three sources of errors: approximation error, generalization error, and optimization error.

$$f_* - \tilde{f} = \underbrace{f_* - f^0}_{\text{approximation error}} + \underbrace{f^0 - \hat{f}}_{\text{generalization error}} + \underbrace{\hat{f} - \tilde{f}}_{\text{optimization error}}.$$

We focus on approximation error and generalization error. What we would like to achieve is that:

For the approximation error: we would like to control $\left\|f_* - f^0\right\|_{L^2(P)}$ appropriately in terms of the width of the neural network m. Ideally, we would like to restrict the function class $\mathcal{M}$ where f_*comes from, and define an appropriate norm $\|f_*\|_*$, so that

$$\inf_{f \in \mathcal{F}_{m,h}} \left\|f_* - f^0\right\|^2_{L^2(P)} \lesssim \frac{\|f_*\|^2_*}{m}.$$

For the generalization error: we have seen from the concentration lecture note that with probability at least $1-\delta$,

$$\sup_{f \in \mathcal{F}_{m,h}} \left| \frac{1}{n} \sum_{i=1}^n f(x_i) - \mathbb{E}[f] \right| \leq 2\mathrm{Rad}(\mathcal{F}_{m,h}) + \sqrt{\frac{1}{2n} \log\left(\frac{2}{\delta}\right)}.$$

Hence we would like to see that, with appropriate norm $\|f\|_{**}$ for $f \in \mathcal{F}_{m,h}$, define $\mathcal{F}_{m,h,Q} \coloneqq \{f \in \mathcal{F}_{m,h} : \|f\|_{**} \leq Q\}$, and then

$$\mathrm{Rad}(\mathcal{F}_{m,h,Q}) \lesssim \frac{Q}{\sqrt{n}}.$$

If both holds, then

$$\left\|f_* - \hat{f}\right\|^2_{L^2(P)} = O_P\left(\frac{\|f_*\|^2_*}{m} + \frac{Q}{\sqrt{n}}\right).$$

3 Approximation error: Classical Universal Approximation

Definition. For spaces X and Y, let $\mathcal{C}(X, Y)$ be the set of continuous functions $f : X \to Y$.

Definition. For $X \subset \mathbb{R}^d$, A class of functions $\mathcal{F} \subset \mathcal{C}(X, \mathbb{R}^D)$ is a universal approximator if for every continuous function $g \in \mathcal{C}(X, \mathbb{R}^D)$, every compact set S, and target accuracy $\epsilon > 0$, there exists $f \in \mathcal{F}$ with

$$\sup_{x \in S} |f(x) - g(x)| < \epsilon.$$

Remark. Typically we will take $S = [0,1]^d$; we can then reduce arbitrary compact sets to this case by defining a new function which re-scales the input. Also, universal approximation is often stated more succinctly as some class being dense in all continuous functions over compact sets.

The classical Weierstrass theorem establishes that polynomials are universal approximators (Weierstrass 1885), and its generalization, the Stone-Weierstrass theorem, says that any family of functions satisfying some of the same properties as polynomials will also be a universal approximator. Stone-Weistrass theorem is a fairly standard way to prove universal approximation; this approach was first suggested in [2].

Theorem (Stone-Weierstrass)**.** *Let functions $\mathcal{F}$ be given as follows.*

- *Each $f \in \mathcal{F}$ is continuous.*
- *For every $x \in S$, there exists $f \in \mathcal{F}$ with $f(x) \neq 0$.*
- *For every $x \neq y \in S$, there exists $f \in \mathcal{F}$ with $f(x) \neq f(y)$ ($\mathcal{F}$ separates points).*
- *$\mathcal{F}$ is closed under multiplication and vector space operations ($\mathcal{F}$ is an algebra).*

Then $\mathcal{F}$ is a universal approximator.

Remark.

- This is a heavyweight tool, but a convenient way to quickly check universal approximation.
- Weierstrass theorem itself has interesting proofs:
  - The modern standard one is due to Bernstein; it picks a fine grid and then a convenient set of interpolating polynomials which behave stably off the grid.
  - Weierstrass's original proof convolved the target with a Gaussian, which makes it analytic, and also leads to good polynomial approximation.

- The second and third conditions in Stone-Weierstrass are necessary; if there exists x so that $f(x) = 0$ for all $f \in \mathcal{F}$, then we can't approximate g with with $g(x) \neq 0$; if we can't separate points $x \neq x'$, then we can't approximate functions with $g(x) \neq g(x')$.

We first go with activation function $\sigma = \cos$, which was the original choice in [2]; we can then handle arbitrary activations by univariate approximation of cos, without increasing the depth (but increasing the width).

Lemma ([2]). *$\mathcal{F}_{\cos}$ is universal.*

Proof. Let's check the Stone-Weierstrass conditions: □

- Each $f \in \mathcal{F}_{\cos}$ is continuous.
- For each x, $\cos(0 + 0^\top x) = 1 \neq 0$.
- For each $x \neq x'$, $f(z) \coloneqq \cos\left(0 + (z - x')^\top (x - x')/\left\|x - x'\right\|_2^2\right) \in \mathcal{F}_\sigma$ satisfies

$$f(x) = \cos(1) \neq \cos(0) = f(x').$$

- $\mathcal{F}_{\cos}$ is closed under products and vector space operations: since $2\cos y \cos z = \cos(y+z) + \cos(y-z)$,

$$2\left[\sum_{i=1}^n \alpha_i \cos(u_i + v_i^\top x)\right] \cdot \left[\sum_{j=1}^m \beta_j \cos(b_j + w_j^\top x)\right] =$$
$$\sum_{i=1}^n \sum_{j=1}^m \alpha_i \beta_j \left(\cos\left((u_i + b_j) + (v_i + w_j)^\top x\right) + \cos\left((u_i - b_j) + (v_i - w_j)^\top x\right)\right),$$

  hence $f, g \in \mathcal{F}_{\cos} \implies fg \in \mathcal{F}_{\cos}$. And closed under vector space operations as well.

Now the two-layer neural network is a universal approximator.

Theorem ([2]). *Suppose $\sigma : \mathbb{R} \to \mathbb{R}$ is sigmoidal: it is continuous, and*

$$\lim_{z \to -\infty} \sigma(z) = 0, \qquad \lim_{z \to \infty} \sigma(z) = 1.$$

Then $\mathcal{F}_\sigma$ is a universal approximator.

Sketch of Proof. GIven $\epsilon > 0$ and continuous g, use above Lemmas to obtain $h \in \mathcal{F}_{\cos}$ (or $\mathcal{F}_{\exp}$) with $\sup_{x \in S} |h(x) - g(x)| \leq \epsilon/2$. To finish, replace all appearances of cos with an element of $\mathcal{F}_\sigma$ so that the total additional error is $\epsilon/2$. □

$\sigma =$ReLU is fine: use $z \mapsto \sigma(z) - \sigma(z-1)$ and split nodes. In fact, the weakest conditions on σ is as follows:

Theorem ([4]). *If $\sigma : \mathbb{R} \to \mathbb{R}$ is continuous and non-polynomial, then $\mathcal{F}_\sigma$ is a universal approximator.*

Carefully accounting within the proof seems to indicate that m needs to satisfy $m = \Omega\left(\frac{1}{\epsilon^d}\right)$, hence indicating curse of dimension again.

4 Universal Approximation on Baron Class

Classical universal approximation guarantees an approximation of arbitrary small error. But it has a drawback: we consider the two-layer neural network with arbitrary width, and to achieve a given accuracy we don't know how wide the hidden layer should be. If we fix the width of the neural network as m, then for f_* in Sobolev space,

$$\inf_{f^0 \in \mathcal{F}_{m,h}} \left\|f^0 - f_*\right\|_{L^p(S)} = O\left(\frac{1}{m^{\alpha/d}}\right),$$

so the approximation error suffers from the curse of dimensionality.

One fundamental reason is that maybe setting the regression function class $\mathcal{M}$ as the Sobolev space is too large; to overcome the curse of dimensionality, one approch would be to restrict $\mathcal{M}$.

We first recall the Fourier transform:

$$\tilde{f}(\omega) \coloneqq \int \exp(-2\pi i \omega^\top x) f(x) dx.$$

Then we have the Fourier inversion: if $f, \tilde{f} \in L^1$,

$$f(x) = \int \exp(2\pi i \omega^\top x) \tilde{f}(\omega) d\omega.$$

Barron class is a function that the Fourier transform of its gradient is integrable: note that $\widetilde{\nabla f}(\omega) = 2\pi i \omega \tilde{f}(\omega)$.

Definition. The Barron class is

$$\mathcal{M}_{\mathcal{B}} = \left\{ f : \mathbb{R}^d \to \mathbb{R} : \int \|\omega\|_2 \left| \tilde{f}(\omega) \right| d\omega < \infty \right\}.$$

Theorem ([1, Theorem 11]). *For any $f_* \in \mathcal{M}_{\mathcal{B}}$ and $m \in \mathbb{N}$, there exists a two-layer neural network $f^0 \in \mathcal{F}_{m,h}$ with m neurons such that*

$$\left\| f_* - f^0 \right\|_{L_2(P)} \lesssim \frac{\int \|\omega\|_2 \left| \tilde{f}(\omega) \right| d\omega}{\sqrt{m}}.$$

Theorem ([1, Theorem 12]). *For any $f_* \in \mathcal{M}_{\mathcal{B}}$ and $m \in \mathbb{N}$, there exists a two-layer neural network $f^0 \in \mathcal{F}_{m,h}$ with m neurons such that*

$$\left\| f_* - f^0 \right\|_{L^\infty([0,1]^d)} \lesssim \sqrt{\frac{d+1}{m}} \int \|\omega\|_2 \left| \tilde{f}(\omega) \right| d\omega.$$

5 Generalization error

For $f_w \in \mathcal{F}_{m,h}$, we can write $f_w(x) = \sum_{j=1}^m a_j h(\theta_j^\top x)$. Define the 1-norm of w as

$$\|w\|_1 \coloneqq \frac{1}{m} \sum_{j=1}^m |a_j| \|\theta_j\|_1.$$

Theorem ([1, Theorem 15]). *Let $\mathcal{F}_{m,h,Q} \coloneqq \{f_w \in \mathcal{F}_{m,h} : \|w\|_1 \le Q\}$. Then we have*

$$\mathrm{Rad}(\mathcal{F}_{m,h,Q}; Z^n) \le 2Q \sqrt{\frac{2\log(2d)}{n}}.$$

Instead of minimizing the training error, we can also consider the regularized term as

$$\mathcal{L}(w) = \frac{1}{n} \sum_{i=1}^n (y_i - f_w(x_i))^2 + \lambda \sqrt{\frac{\log(2d)}{n}} \|w\|_1,$$

and let $\hat{w}^{(1)}$ be its minimizer.

Theorem ([1, Theorem 16]). *Suppose $\mathcal{X} \subset \mathbb{R}^d$ is compact, and assume $f_* : \mathcal{X} \to [0,1]$. There exists some $\lambda_0 > 0$ such that for $\lambda \ge \lambda_0$, with probability $1-\delta$,*

$$\frac{1}{n} \sum_{i=1}^n (y_i - f_{\hat{w}^{(1)}}(x_i))^2 \lesssim \frac{\|f_*\|_{\mathcal{B}}^2}{m} + \lambda \|f_*\|_{\mathcal{B}} \sqrt{\frac{\log(2d)}{n}} + \sqrt{\frac{\log(n/\delta)}{n}}.$$

References

[1] Weinan E, Chao Ma, Stephan Wojtowytsch, and Lei Wu. Towards a mathematical understanding of neural network-based machine learning: what we know and what we don't. *CoRR*, abs/2009.10713, 2020.

[2] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[3] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning—with applications in R.* Springer Texts in Statistics. Springer, New York, [2021] ©2021. Second edition [of 3100153].

[4] Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993.