Optimization error for Deep Learning: Mean Field

김지수 (Jisu KIM)

딥러닝의 통계적 이해 (Deep Learning: Statistical Perspective), 2025 2nd Semester (fall)

This lecture note is mainly from:
Boris Hanin, Mathematics Aspects of Deep Learning Theory, 2024.
https://boris-hanin.github.io/Univ__Luxembourg_Lectures_June_2024_notes.pdf

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 $\hat{f}(x)$.

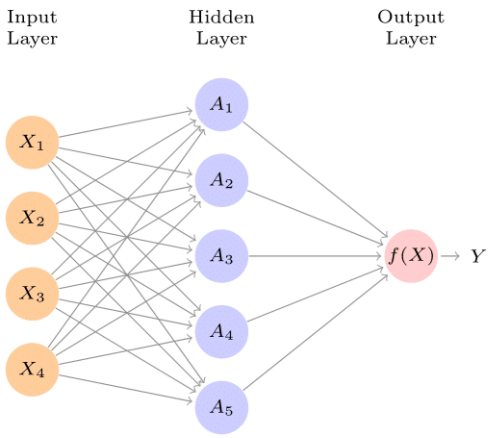

Figure 1: Neural network with a single hidden layer. The hidden layer computes activations $A_j = \sigma_j(x)$ that are nonlinear transformations of linear combinations of the inputs $x_1, \ldots, x_d$. Hence these A_j are not directly observed. The functions σ_j are not fixed in advance, but are learned 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 [?].

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.

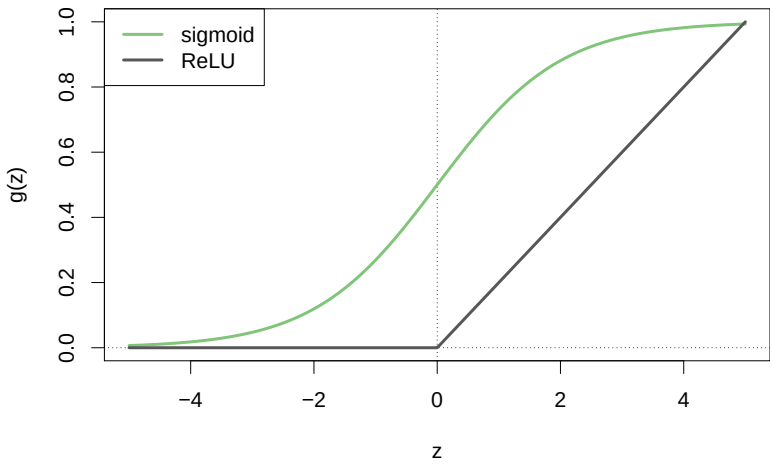

Figure 2: Activation functions. The piecewise-linear ReLU function is popular for its efficiency and computability. We have scaled it down by a factor of five for ease of comparison. Figure 10.2 from [**?**].

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

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

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

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

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

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

$\hat{f}$ 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. See Figure 3.

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

Here, we focus on optimization error.

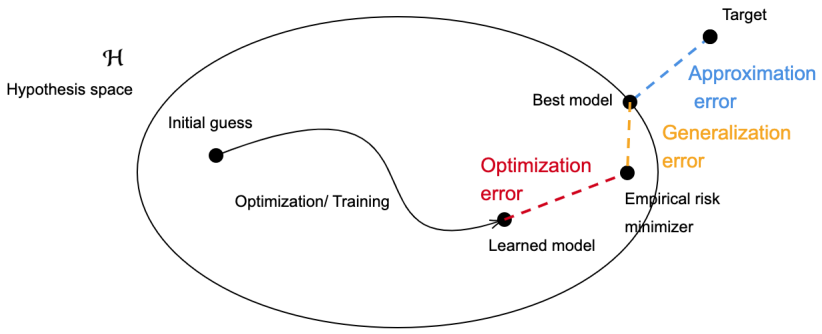

Figure 3: Diagram representing the learning procedure, the three main paradigms and their corresponding errors. Figure 2 from https://dcn.nat.fau.eu/breaking-the-curse-of-dimensionality-with-barron-spaces/.

3 Training dynamics-based statistical guarantees

In the approximation and generalization lecture notes, we assumed that the global minimizer of the empirical risk, $\hat{f}$ in (1), is obtainable. However, due to the non-convex nature of the loss function, neural networks estimated using commonly employed gradient-based methods lack guarantees of finding $\hat{f}$, which leads to the following natural question:

Does the neural network estimated by gradient-based methods generalize well?

We will try to answer to the above question. Due to the complex nature of the problem, most work have considered the suggested shallow neural network: (i.e., networks with one hidden layer). Assume $b_j = 0$, and with a slightly different parameterrization, we consider the following functions:

$$f_\theta(x) = \frac{\alpha}{m} \sum_{j=1}^{m} a_j \sigma(w_j^\top x), \tag{2}$$

and the corresponding function class becomes

$$\mathcal{F}_{m,\sigma} = \left\{ f_\theta : f_\theta(x) = \frac{\alpha}{m} \sum_{j=1}^{m} a_j \sigma(w_j^\top x) \right\}.$$

The network dynamic is scaled with the factor $\frac{\alpha}{m}$. If the network width (i.e., m) is small, the scaling factor has negligible effects on the network dynamics. But for the wide enough networks (i.e., overparametrized setting), the scaling difference yields completely different behaviors in the dynamics. Given the m is large enough, we will focus on two specific regimes:

1. Neural Tangent Kernel regime: with $\alpha = \sqrt{m}$.
2. Mean Field regime: with $\alpha = 1$.

We consider the ℓ_2-loss function: $\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - f_\theta(x_i))^2$. The model parameter pairs θ are updated through the gradient-based methods. Let $\theta_{(0)}$ be the initialized weight pairs. Then, we have the following gradient descent (GD) update rule with step-size $\eta > 0$ and $k \geq 1$:

$$\text{GD}: \qquad \theta_{(k)} = \theta_{(k-1)} - \eta \nabla_\theta \mathcal{L}(\theta)|_{\theta=\theta_{(k)}}. \tag{3}$$

Another celebrated gradient-based method is Stochastic Gradient Descent (SGD). The algorithm takes a randomly sampled subset of the data, computes the gradient with the selected samples, and this significantly reduces the computational burdens in GD. Another frequently adopted algorithm in practice is Noisy Gradient Descent (NGD), which adds the centered Gaussian noise to the gradient of loss function in (3). It is known that adding noises to gradient helps training and generalization of neural networks.

4 Mean Field Regime

In the mean-field regime, a one layer neural network takes the form

$$f(x;\theta) = \frac{1}{m}\sum_{j=1}^{m} a_j \sigma(w_j^\top x). \tag{4}$$

This corresponds to setting $\alpha = 1$ in (2). While the choice of α may only appear to be a matter of parameterization, we will see later that it has a significant effect on the nature of first order optimization with one layer networks. Before explaining this, we take a brief but useful detour to explain how to view any one layer neural network in the mean field regime as a probability measure.

5 One Layer Networks as Empirical Measures

The neural network (4) can naturally be rewritten in terms of a measure on $\mathbb{R}^{m+1}$:

$$f(x;\theta) = \int_{\mathbb{R}^{m+1}} a\sigma\left(w^\top x\right) d\rho_\theta(a,w),$$

where

$$\rho_\theta(a,w) := \frac{1}{m}\sum_{j=1}^{m} \delta_{(a_j,w_j)},$$

is simply the empirical measure of incoming and outgoing weights across all neurons. Thus, there is a map from the space of one layer networks to the space of probability measures $\mathcal{P}(\mathbb{R}^{m+1})$ on $\mathbb{R}^{m+1}$. Except on a measure 0, the image of the networks with hidden layer width m is the collection of empirical measures with m atoms in $\mathcal{P}(\mathbb{R}^{m+1})$.

Any question about one hidden layer networks can, using the mapping from one layer networks to probability measures, be translated into a question about the linear space $\mathcal{P}(\mathbb{R}^{m+1})$. This is often useful since one can now reason about networks of all widths $m \geq 1$ in a uniform way.

Consider for example, the classical question of which functions $f : \mathbb{R}^m \to \mathbb{R}$ can be well-approximated by one hidden layer networks with non-linearity σ and width m? The identification between one layer networks and measures naturally decomposes this into two questions:

1. Approximation at infinite width. Which functions $f : \mathbb{R}^m \to \mathbb{R}$ can be written

$$f(x) = \int_{\mathbb{R}^{m+1}} a\sigma\left(w^\top x\right) d\mu(a,w) \tag{5}$$

for some $\mu \in \mathcal{P}(\mathbb{R}^{m+1})$?

2. Rate of approximation. Given a measure $\mu \in \mathcal{P}(\mathbb{R}^{m+1})$, how well can it be approximated by an empirical measure with m atoms?

The answer to question 1 is that for any "reasonable" non-linearity σ (e.g. not a polynomial on some interval), every "reasonable" function (say continuous) can be written in the form (5). This is essentially the point of view taken by [Hornik et al.(1989)Hornik, Stinchcombe, and White]. Obtaining sharp answers to the second question is more subtle (see [Barron(1994), Mhaskar and Poggio(2016)]). One simple approach is to note that every $\mu \in \mathcal{P}(\mathbb{R}^m)$ can be approximated in the infinite width limit by such measures by the law of large numbers:

$$\frac{1}{m}\sum_{j=1}^{m} \delta_{(a_j,w_j)} \to \mu, \qquad (a_j,w_j) \overset{i.i.d.}{\sim} \mu, \tag{6}$$

where the convergence is in the weak sense and occurs almost surely. For example using a uniform central limit theorem to quantify the rate of convergence in (6) one may obtain as in [Barron(1994)], an $m^{-1/2}$ upper bound on the rate of convergence for certain function spaces.

6 Optimization of One Layer Networks in the Mean Field Regime

The identification between one layer networks and empirical measures has been used by a variety of authors to understand not just approximation with one layer networks but also optimization. The high-level takeaways from these articles are as follows:

- **Optimal transport.** Suppose that our training data is generated by

$$y = f(x),$$

for some fixed $f : \mathbb{R}^d \to \mathbb{R}$, which admits a representation

$$f(x) = \int_{\mathbb{R}^{m+1}} a\sigma(w^\top x) d\mu_f(a, w).$$

Suppose further the empirical loss of the training dataset is replaced by the population loss

$$\mathcal{L}(\theta) = \mathbb{E}_{(x,y)} \left[(f(x;\theta) - y)^2 \right].$$

The trajectory of the empirical measures

$$\rho_t := \rho_{\theta_t} = \frac{1}{m} \sum_{j=1}^{m} \delta_{(a_j, w_j)},$$

is that of ℓ^2-based optimal transport. The initial condition ρ_0 is determined by the distribution of weights at initialization and the final condition is μ_f. This point of view was developed in [Mei et al.(2018)Mei, Montanari, and Nguyen], which describes also many extensions, such as how gradient flow on an empirical loss approximates gradient on the population loss.

- **Particles with Mean-Field Interactions.** Define

$$X_j(t) := \left(a_j, w_j(t)^\top x_\alpha \right)_{\alpha=1}^{n}, \quad j = 1, \ldots, m$$

and write

$$\bar{\rho}_t := \frac{1}{m} \sum_{j=1}^{m} \delta_{X_j(t)}.$$

A direct computation shows that, even at finite width, the dynamics of $X_i(t)$ form a closed system of mean-field type in the sense that the time-derivative of the "state" of each "particle" $X_i(t)$ depends on the other particles only through their empirical measure, i.e.

$$\frac{d}{dt} X_i(t) = F(X_i(t), \bar{\rho}_t),$$

for an explicit function F. In this way, the dynamics at any finite width of a one hidden layer network trained by gradient flow over a fixed dataset is the discretization of the Vlasov equation with random initial conditions. This is, with some variations, the point of view taken in [Bordelon and Pehlevan(2022)].

7 Comparison Between the Mean Field and NTK Regimes

Aspect	NTK regime	Mean-Field Regime
Scaling	$\alpha = \sqrt{m}$	$\alpha = 1$
Parameter movement	Very small ("lazy training") — parameters stay near initialization	Large movement — parameters evolve significantly
Model behavior during training	Network is effectively linearized around initialization	Network remains non-linear in parameters
Theoretical tools	kernel method	mean-field limit, PDE, distributional dynamics
Strength	Mathematical simplicity; global convergence; closed-form kernel limit	Captures representation learning; realistic dynamics; expressive
Weakness	No representation learning; unrealistic for finite width	Harder to analyze; fewer global guarantees

References

[Barron(1994)] Andrew R. Barron. Approximation and estimation bounds for artificial neural networks. *Mach. Learn.*, 14(1):115–133, 1994. doi: 10.1007/BF00993164. URL https://doi.org/10.1007/BF00993164.

[Bordelon and Pehlevan(2022)] Blake Bordelon and Cengiz Pehlevan. Self-consistent dynamical field theory of kernel evolution in wide neural networks. *CoRR*, abs/2205.09653, 2022. doi: 10.48550/ARXIV.2205.09653. URL https://doi.org/10.48550/arXiv.2205.09653.

[Hornik et al.(1989)Hornik, Stinchcombe, and White] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. doi: 10.1016/0893-6080(89)90020-8. URL https://doi.org/10.1016/0893-6080(89)90020-8.

[James et al.([2021] 2021)James, Witten, Hastie, and Tibshirani]JamesWHT2021 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. ISBN 978-1-0716-1417-4; 978-1-0716-1418-1. doi: 10.1007/978-1-0716-1418-1. URL https://doi.org/10.1007/978-1-0716-1418-1. Second edition [of 3100153].

[Mei et al.(2018)Mei, Montanari, and Nguyen] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layers neural networks. *CoRR*, abs/1804.06561, 2018. URL http://arxiv.org/abs/1804.06561.

[Mhaskar and Poggio(2016)] Hrushikesh N. Mhaskar and Tomaso A. Poggio. Deep vs. shallow networks : An approximation theory perspective. *CoRR*, abs/1608.03287, 2016. URL http://arxiv.org/abs/1608.03287.