CS7140 Lecture Notes on Statistical Learning and Its Applications (Spring 2026)

Hongyang R. Zhang, Northeastern University, Boston

February 11, 2026

Contents

1 Overview

Background. Machine learning has been increasingly used in technology platforms and products, affecting our daily lives. Machine learning involves a collection of models, algorithms, and engineering frameworks:

- Regression and classification: least squares estimation, logistic regression, ℓ_1/ℓ_2-regularization, bias-variance tradeoff, cross-validation.
- Neural networks and deep learning: convolutional neural networks, backpropagation, foundation models, language modeling.
- Unsupervised learning: dimension reduction such as principal component analysis, clustering, contrastive learning.
- Reinforcement learning and sequential decision-making: robotics, reinforcement learning from human feedback.
- Generative AI: large language models, diffusion models, multi-modal data.
- Causal machine learning: study the cause-and-effect to estimate the counterfactual.
- Machine learning libraries: numpy, sklearn, pytorch, tensorflow, huggingface…

1.1 What is this course about? (Lecture 1)

This course aims to uncover the common mathematical and statistical principles underlying the diverse array of machine learning models and algorithms. This class primarily focuses on the theoretical analysis of learning algorithms and models. Many of the techniques introduced in this course—which involve a beautiful blend of probability, linear algebra, and optimization—are separate fields in their respective discipline with independent interests outside of machine learning. For example, we will study the supremum of a complex random variable corresponding to the outcome of a learning algorithm applied to train a neural network model. We will show how to design estimation algorithms when working under distribution shifts between the training and test datasets.

From a practical point of view, studying the underlying working mechanisms of a learning algorithm can deepen our understanding of how machine learning models work. For example, suppose we want to design a neural network classifier to predict the sentiment of a document. We train a regression model using word frequencies as features and achieve 100% training accuracy on 1000 training documents and 85% test accuracy on 1000 test documents. How can we reduce the gap between training and test accuracy? Further, what happens if the word frequencies between the training corpus and the test corpus are different? It is possible to answer these questions from an engineering perspective; instead, this course will mostly focus on the mathematical analysis underlying these procedures, although we will provide computational examples from time to time to help you understand the theoretical concepts.

There is a clear gap between theoretical analysis and an algorithm's practical performance. For instance, theoretical analysis is usually conducted under standard technical assumptions that are often made to simplify the analysis. The goal, instead, is to *build a deeper understanding of the underlying key concepts through mathematical modeling and theoretical analysis*. We will see if we succeed in accomplishing this objective by the end of this semester.

The course materials are divided into three parts: *fundamental concepts of statistical learning theory* (January), *generalization and optimization of neural networks and deep learning* (February), and *statistical modeling of emerging learning paradigms* (March).[1]

1.2 Supervised prediction (Lecture 1)

Central questions: *Does minimizing training error lead to low test error? How does the generalization ability depend on the model architecture and the training algorithm?* It turns out that answering these questions is highly non-trivial as it also depends on the underlying data distribution.

To formally study these questions, let us first describe the mathematical setup:

- Let $\mathcal{X}$ denote the feature space. Let $\mathcal{Y}$ denote the space of all possible outcomes. Binary classification example: $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{+1, -1\}$
- Consider the problem of predicting an output $y \in \mathcal{Y}$ given an input $x \in \mathcal{X}$.
- Let $\mathcal{H}$ be a set of hypotheses. Linear model example:

$$\mathcal{H} = \left\{ x \to \beta^\top x + \epsilon : \forall \beta \in \mathbb{R}^d, \epsilon \in \mathbb{R} \right\}$$

- Let $\ell : (\mathcal{X}, \mathcal{Y}) \times \mathcal{H} \to \mathbb{R}$ be a loss function. For example, the mean squared error (MSE) applied to linear models is

$$\ell((x, y), \epsilon) = \left(\beta^\top x + \epsilon - y \right)^2, \forall \beta \in \mathbb{R}^d, \forall \epsilon \in \mathbb{R}$$

- Given n training data samples, denoted by $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, the training loss (or empirical risk) of a hypothesis $h \in \mathcal{H}$ is defined as

$$\hat{L}(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i), y_i), \forall h \in \mathcal{H} \tag{1}$$

We make a critical assumption about the data-generating process. We assume that every x_i, y_i pair is drawn independently and identically from an unknown distribution $\mathbb{P}^\star$, supported on $\mathcal{X} \times \mathcal{Y}$.

The test loss (or expected risk) of a hypothesis $h \in \mathcal{H}$ is then given by

$$L(h) = \mathop{\mathbb{E}}_{(x,y) \sim \mathbb{P}^\star} \left[\ell(h(x), y) \right]. \tag{2}$$

Example 1.1 (Linear regression)**.** *To make the above setup more concrete, perhaps the best example would be linear regression. There are many standard texts on this topic; see, e.g., Wainwright, 2019. In a standard parametric regression setup, we have n samples $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, where every $x_i \in \mathbb{R}^p$ is a p-dimensional feature vector drawn from some unknown distribution $\mathcal{D}$, and $y_i \in \mathbb{R}$, for every $i = 1, 2, \ldots, n$. In addition, suppose that there exists an unknown $\beta \in \mathbb{R}^p$ such that*

$$y_i = x_i^\top \beta + \varepsilon_i, \text{ for every } i = 1, 2, \ldots, n, \tag{3}$$

[1] April will be dedicated to course project presentations.

where $x_i^\top \beta = \sum_{j=1}^p x_{i,j}\beta_j$, and $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ is a noise random variable with mean zero and variance σ^2.

Given n samples, the goal of this problem is to learn a linear model parameterized by $\hat{\beta}$ that achieves the lowest mean-squared error (MSE) on an unseen sample.

Remark 1.2. *We have assumed the training and test distributions are the same. While this assumption does not hold exactly in practice, morally, the training and test distributions must be related.*

Formulating what it means to be related and not related, and addressing the discrepancy between training and test data, are studied in the area of domain adaptation *or* transfer learning.

The independence assumption, which also does not hold exactly in practice, ensures that more training data gives us more information.

1.2.1 Empirical risk minimization

Consider minimizing the training loss

$$\hat{h}_{\text{ERM}} \leftarrow \operatorname*{argmin}_{h \in \mathcal{H}} \hat{L}(h). \tag{4}$$

What can say that the relationship between $\hat{L}(\hat{h}_{\text{ERM}})$ and $L(\hat{h}_{\text{ERM}})$? A key challenge is that the randomness of $\hat{h}_{\text{ERM}}$ now depends on $\hat{L}$. Thus, $\hat{L}(\hat{h}_{\text{ERM}})$ involves a correlation between the training data samples and the minimizing hypothesis. A central aspect we will tackle in the first part of the course is developing the machinery to address this challenge.

Example 1.3 (Pretraining and fine-tuning). *An emerging learning paradigm that has emerged over the past few years follows a two-stage procedure involving pretraining on a large amount of unlabeled data, followed by fine-tuning on a small amount of labeled data.*

The pretraining stage usually follows some masked prediction procedure on unlabeled data. The supervised fine-tuning (SFT) procedure can be formulated with the above ERM setup.

- *Suppose we have some model like a neural network, f_{W_0}, parameterized by some initialization W_0.*
- *There is a small amount of training dataset, S, from which we compute the training loss $\hat{L}(f_{W_0})$.*
- *SFT corresponds to minimizing $\hat{L}(f_{W_0})$, usually via a stochastic gradient optimization algorithm.*
- *An important consideration in SFT is overfitting, since the model is pretrained on a large amount of unlabeled data. The size of the model is usually much larger than the size of the training dataset S.*

1.2.2 Uniform convergence and generalization gap

In the first part of this course, we will show various "uniform convergence" statements of the following flavor:

With probability at least $1-\delta$, the gap between test loss and training loss of any hypothesis is upper bounded by some small ϵ, that is, $L(h) - \hat{L}(h) \le \epsilon$, where the ϵ is generally a function that depends on δ and other aspects of the learning algorithm/model

More rigorously, we would like to show statements of the following:

$$\Pr\left[\underbrace{L(h) - \hat{L}(h)}_{\text{Generalization gap}} > \epsilon\right] \le 1-\delta, \tag{5}$$

where the randomness is on the training data samples drawn from $\mathbb{P}^\star$. This statement essentially quantifies the generalization gap between the training and test losses of the machine learning model.

Equipped with such a statement, we will then apply the statement to the empirical risk minimizer $\hat{h}_{\text{ERM}}$, since the result essentially holds for any $h \in \mathcal{H}$, which also subsumes $\hat{h}_{\text{ERM}}$ as a special case.

1.3 Multi-layer neural networks and generative models (Lecture 2)

Consider the case of a basic one-layer network:

$$f_{a,W,b}(x) := x \to \sum_{i=1}^{m} a_i \sigma(w_i^\top x + b_i), \text{ where} \tag{6}$$

- σ is the nonlinear activation function. Typical choices of σ: ReLU $x \to \max(0,x)$, sigmoid $x \to \frac{1}{1+\exp(-x)}$. Key property: Lipschitz-continuity: A function $f : \mathbb{R} \to \mathbb{R}$ is said to be *C-Lipschitz-continuous* if the following is true:

$$|f(x) - f(y)| \le C \cdot |x-y|.$$

- $Z = \{\alpha = (a_i, w_i, b_i)\}_{i=1}^m$ are trainable parameters of the network. By varying α, we define the function class $\mathcal{H}$ as

$$\mathcal{H} = \{f_\alpha : \forall \alpha \in Z\}.$$

- $\mathcal{H}$ essentially represents a set of one-hidden-layer neural networks with m neurons.

- Let $W = [w_1, w_2, \ldots, w_m]$, and $b = [b_1, b_2, \ldots, b_m]$. We may write $f_{a,W,b}$ equation (6) as $x \to a^\top \sigma(Wx+b)$.

By extending the above setup, we may write a deep network as

$$f_\alpha(x) = \sigma_L\left(W_L \sigma_{L-1}\left(\cdots \sigma_2\left(W_2 \sigma_1\left(W_1 x + b_1\right) + b_2\right)\cdots\right)\right), \tag{7}$$

where α now encodes all the parameters of the network. The depth of the network is given by L. The width is given by $\max(m_1, m_2, \ldots, m_l)$, i.e., the layer with the most neurons in the layer.

Motivating questions: *How could we analyze the training and test losses of a deep network? How well does a deep network generalize, and how does it depend on its depth and width?*

How does this ability to learn and to generalize rely on the data distributions, and what is the role of optimization algorithms used to train the network?

A *language model* specifies a conditional probability distribution $\Pr_\theta(\cdot \mid P)$, given a prompt sequence P, produces the next-token according to underlying probability masses.

Example 1.4 (In-context learning)**.** *To illustrate the concept of a language model, let us consider a few-shot meta-learning problem (Garg et al., 2022). In this problem, each prompt P_{θ_i} involves a sequence of examples or demonstrations*

$$P_{\theta_i} := (x_1, y_1, x_2, y_2, \dots, x_{t-1}, y_{t-1}, x_t),$$

ended with a query example x_t. The goal is to predict the correct output y_t corresponding to the query x_t.

To make this more concrete, suppose that $y_j = \theta_i^\top x_j$, for every $j = 1, 2, \dots, t-1$. The desired output $y_t = \theta_i^\top x_j$.

- *At training time, the model sees a sequence of prompt-answer pairs (P_{θ_i}, y_t).*
- *At test time, the model sees a new prompt P_θ parameterized by some unknown θ. The model is asked to first "solve" the linear regression from the in-context examples given in P_θ, and then use the "learned" regression model to output the correct answer corresponding to the query.*

1.4 Transfer learning and minimax estimation (Lecture 2)

An important learning paradigm that has emerged in the past few years is transfer learning—transferring the knowledge from one task to help solve another task. How could we develop a more rigorous statistical modeling of transfer learning? A better understanding of this question has applications in language modeling, computer vision, robotics, to name a few.

1.4.1 Transfer learning setup

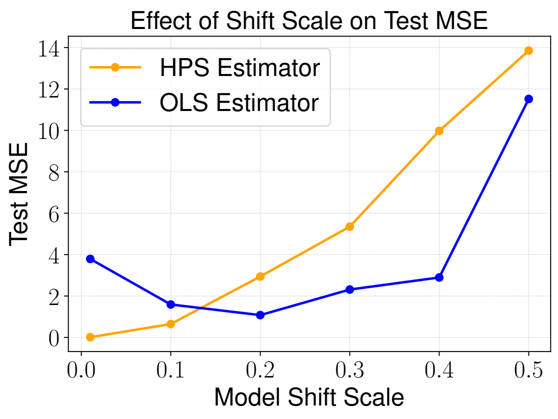
Perhaps the simplest modeling framework is to examine transfer learning in linear regression tasks. For example, we may consider the case of two linear regression tasks, one called the source task and the other called the target task.

Suppose we have n_1 samples from the source task. We have n_2 samples from the target task. How could we use the samples from the source task to help estimate the target task? Concretely, let the samples of the source task be denoted by $(x_1^{(1)}, y_1^{(1)}), (x_2^{(1)}, y_2^{(1)}), \dots, (x_{n_1}^{(1)}, y_{n_1}^{(1)})$, where every $x_i^{(1)}$ is a p-dimensional vector and $y_i^{(1)}$ is a real-valued outcome. Similarly, we denote the samples of the target task as $(x_1^{(2)}, y_1^{(2)}), (x_2^{(2)}, y_2^{(2)}), \dots, (x_{n_2}^{(2)}, y_{n_2}^{(2)})$.

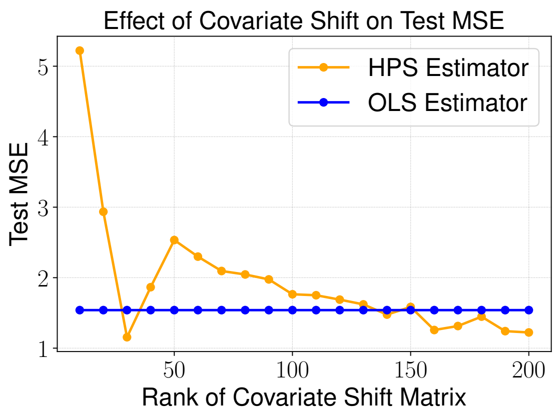
Now we can ask a few more concrete questions:

- How does the difference between the $x \to y$ mappings affect transfer learning performance?
- How does the covariance between the feature vectors of source and target tasks affect transfer learning?

More generally, we may say that the source task and the target task involve a distribution shift between their them. In the area of domain adaptation (Kouw and Loog, 2018):



(a) Model shift



(b) Covariate shift

Figure 1: Illustrating the effects of model shift and covariate shift in transfer learning linear regression.

- Covariate shift refers to scenarios where both tasks follow the same model conditioned on the features, but they have different feature distributions.
- Model shift refers to scenarios where the two tasks follow different models conditioned on the same features.

We may now ask, how does covariate shift and model shift affect transfer learning performance?

1.4.2 Transfer learning estimators

Typically, there are two strategies for transfer learning, one called hard transfer, where we hard-code the shared component across tasks, the other called soft transfer, where we use separate components for task, and encourage the separate components to be close to each other (Ruder, 2017; Dhifallah and Lu, 2021).

Example 1.5 (Illustration of model and covariate shifts in linear regression)**.** *We shall assume that the source task follows a linear relation specified by an unknown parameter* $\beta^{(1)} \in \mathbb{R}^p$*:*

$$y_i^{(1)} = {x_i^{(1)}}^\top \beta^{(1)} + \epsilon_i^{(1)}, \text{ for all } i = 1, 2, \ldots, n_1, \tag{8}$$

where $\epsilon_i^{(1)}$ *is a white noise with mean zero and variance* σ_1^2*.*

We further assume that the target task follow another linear relation specified by an unknown parameter $\beta^{(2)} \in \mathbb{R}^p$*:*

$$y_i^{(2)} = {x_i^{(2)}}^\top \beta^{(2)} + \epsilon_i^{(2)}, \text{ for all } i = 1, 2, \ldots, n_2, \tag{9}$$

where $\epsilon_i^{(2)}$ *is a white noise with mean zero and variance* σ_2^2*.*

In the context of linear regression, we can define an hard parameter sharing estimator as follows:

$$\hat{L}^{\text{HPS}}(\beta) = \frac{1}{n_1 + n_2} \left(\sum_{i=1}^{n_1} \left({x_i^{(1)}}^\top \beta - y_i^{(1)} \right)^2 + \sum_{j=1}^{n_2} \left({x_j^{(2)}}^\top \beta - y_j^{(2)} \right)^2 \right) \tag{10}$$

We may also elect to use a soft parameter sharing estimator instead:

$$\hat{L}^{\text{SPS}}(\beta, z) = \frac{1}{n_1 + n_2} \left(\sum_{i=1}^{n_1} \left({x_i^{(1)}}^\top (\beta + z) - y_i^{(1)} \right)^2 + \sum_{j=1}^{n_2} \left({x_j^{(2)}}^\top \beta - y_j^{(2)} \right)^2 \right) + \lambda \|z\|^2 \tag{11}$$

Essentially, by adjusting λ, we can adjust the magnitude of z, which then determines how far (and how close) the source and target task models are.

A natural baseline is when we do not use the source task data at all. That is, perform least squares regression using target task data alone.

In Figure 1, we illustrate the effects incurred from model shifts and covariate shifts in transfer learning linear regression, comparing between HPS and OLS. In particular, we capture model shift as the distance error between $\beta^{(1)}$ and $\beta^{(2)}$, and we capture covariate shift as the difference in the population covariance matrix between task one and task two. To generate the condition matrix, we use a rank-r matrix whose trace is equal to p, and set its nonzero eigenvalues as p/r.

1.4.3 Optimality of the estimator

The above estimation algorithms are based on the best practices of practitioner (see the surveys above). Suppose we analyze their performances. However, how can we know that there are no better estimators out there? How could we understand the fundamental limits of estimation and optimization procedures? These are often called *minimax lower bounds* on the performance of estimators, and it usually falls into the area of information theory (Duchi, 2019). In particular, we will touch on the framework of minimax lower bounds for transfer learning (though the scope of this is much broader than we'll cover in our lectures).

2 Uniform convergence in supervised prediction

Recall that we have introduced the empirical loss and expected loss of a hypothesis (denoted by $L(h)$ and $\hat{L}(h)$) for some h in a hypothesis class $\mathcal{H}$. Suppose we minimize the empirical risk to get $\hat{h}_{\text{ERM}}$. Two questions:

- Generalization gap: how does the expected and empirical risks compare for ERM, i.e., $L(\hat{h}_{\text{ERM}}) - \hat{L}(\hat{h}_{\text{ERM}})$? This is called the generalization gap.
- Excess risk: how well does ERM do with respect to the best possible hypothesis in the hypothesis class, i.e., $L(\hat{h}_{\text{ERM}}) - \min_{h\in\mathcal{H}} L(h)$? This is also called the excess risk.

A particularly fruitful framework for analyzing learning algorithms is the probably approximately correct (PAC) framework (Valiant, 1984):

A learning algorithm A PAC learns a hypothesis class $\mathcal{H}$ if

a) For any distribution $\mathbb{P}^\star$ supported over $\mathcal{X} \times \mathcal{Y}$, and any $\epsilon > 0$, $\delta > 0$

b) Upon taking n I.I.D. samples from $\mathbb{P}^\star$, A produces an output $\hat{h} \in \mathcal{H}$ such that with probability at least $1 - \delta$ (over the randomness of the samples)

$$L(\hat{h}) - \min_{h\in\mathcal{H}} L(h) \le \epsilon$$

c) Further, n is a polynomial function of $\epsilon^{-1}, \delta^{-1}, d, |\mathcal{H}|$, and A runs in time polynomial in $n, d, \epsilon^{-1}, \delta^{-1}$ (where d is the dimension of the input)

Remark: Notice that the running time complexity places a bound on the sample complexity as well. We will assume that the empirical risk minimizer can be computed efficiently. For instance, think of a large neural network whose training loss can be efficiently reduced to reach zero using stochastic gradient descent.

Example 2.1 (Policy learning)**.** *Even though the above definition was proposed three decades ago, it remains a very fundamental concept, and applies broadly beyond supervised prediction.*

Consider a finite Markov decision process (MDP) with state space $\mathcal{S}$, action space $\mathcal{A}$, horizon H, and unknown transition and reward dynamics. Let Π be a finite class of deterministic policies $\pi : \mathcal{S} \to \mathcal{A}$. For any policy $\pi \in \Pi$, we can define the expected loss as

$$L(\pi) := \mathbb{E}\left[\sum_{t=1}^{H} \ell(s_t, a_t)\right],$$

where the trajectory (s_t, a_t) is generated by executing π in the MDP, and ℓ is the loss measured at each step.

Once we collect n trajectories (e.g., via an exploration policy),[2] *we can write down the empirical loss averaged over the n sampled trajectories. Therefore, a policy learning algorithm*

[2]For instance, a uniform random exploration picks an action $a \in \mathcal{A}$ uniformly at random at any state. This guarantees unbiased coverage of all actions. An ε-greedy exploration instead follows a fixed deterministic policy with probability $1 - \varepsilon$, while choosing a random action with probability ε.

A PAC-learns the policy space Π if A can produce a near-optimal policy with excess loss at most ϵ with probability at least $1-\delta$, and n only depends polynomially on $|\mathcal{S}|, |\mathcal{A}|, \delta^{-1}$.

2.1 Learning a realizable, finite hypothesis class (Lecture 3)

Next, we give a concrete example to illustrate learnability in a finite, realizable hypothesis class.

Assumptions (realizable, finite hypothesis): i) The size of the hypothesis space, $\mathcal{H}$, is finite; ii) There exists a hypothesis $h^\star \in \mathcal{H}$ such that $h^\star$ achieves perfect performance, i.e.,

$$L(h^\star) = \mathop{\mathbb{E}}_{(x,y)\in\mathbb{P}^\star}[\ell(h^\star(x), y)] = 0.$$

Under these assumptions, we shall prove the following property of ERM:

Under the above assumptions, with probability $1-\delta$,

$$L(\hat{h}_{\text{ERM}}) \leq \frac{\log(|\mathcal{H}|) + \log(\delta^{-1})}{n} \tag{12}$$

In particular, to reduce the expected risk below ϵ, we want

$$n \geq \frac{\log(|\mathcal{H}|) + \log(\delta^{-1})}{\epsilon}.$$

Example 2.2. *Imagine a clinic designing a digital diagnostic tool to determine if a patient should be referred to a specialist based on three binary (Yes/No) symptoms: 1. Fever? 2. Cough? 3. Shortness of breath?*

Since there are only 3 binary features, there are only 8 possible patient profiles. A hypothesis in this setup is a rule that assigns a "Refer" or "Not Refer" label to each profile. The number of possible ways to assign binary labels to these 8 profiles is $2^8 = 256$, which is finite. A hypothesis is any rule that assigns a binary labeling to the 8 profiles. Thus, the size of the hypothesis space is 256 in total.

If a medical board has already decided a specific protocol regarding the referral, then there exists a true target function in the 256 options. Thus, suppose you train the model on historical data from a clinic, you are guaranteed that at least one hypothesis in your class can achieve zero error, because the data was generated by a specific rule.

Proof. We'd like to upper bound the probability of the bad event that $L(\hat{h}_{\text{ERM}}) > \epsilon$: Let $B \subseteq \mathcal{H}$ be the set of bad hypotheses: $\{B \in \mathcal{H} : L(h) > \epsilon\}$

We can rewrite our goal as upper bounding the probability of selecting a bad hypothesis $\Pr[L(\hat{h}_{\text{ERM}}) > \epsilon] = \Pr[\hat{h}_{\text{ERM}} \in B]$. Recall the empirical risk of ERM is always zero because at least $\hat{L}(h^\star) = 0$. Hence for any "bad" hypothesis in B, they must have zero empirical risk

$$\Pr[\hat{h}_{\text{ERM}} \in B] \leq \Pr[\exists h \in B : \hat{L}(h) = 0]$$

Now we shall deal with the above in two steps. First, bound $\Pr[\hat{L}(h) = 0]$ for a fixed $h \in B$. Notice that on a random example from $\mathcal{P}^\star$, the accuracy of h should be $1 - L(h)$. Since the training data is drawn IID from $\mathcal{P}^\star$, and $L(h) \geq \epsilon$ for any $h \in B$, we get that

$$\Pr[\hat{L}(h) = 0] \leq (1 - L(h))^n \leq (1-\epsilon)^n \leq \exp^{-\epsilon n},$$

where we use the fact that $1 - x \le \exp(-x)$.

Second, we want the above to hold simultaneously for all $h \in B$. We can apply the union bound to bound the probability of all bad events:

$$\begin{aligned}\Pr[\exists h \in B : \hat{L}(h) = 0] &\le \sum_{h \in B} \Pr[\hat{L}(h) = 0] \\ &\le |B| \exp(-\epsilon n) \\ &\le |\mathcal{H}| \exp(-\epsilon n)\end{aligned}$$

By setting the above at most δ, we conclude that ϵ must be at least $\frac{\log(|\mathcal{H}|)+\log(\delta^{-1})}{n}$. This concludes the proof for learning finite, realizable hypothesis spaces.

Two remarks:

- The excess risk only grows logarithmically with the size of the hypothesis class, so we can afford to use an exponentially large hypothesis space.
- The result is independent of $\mathbb{P}^\star$. This is nice because typically we don't know the true distribution.

The proof of this result is elementary but illustrates an important pattern that will recur in more complex scenarios. We are interested in the expected risk, but only have access to empirical risk to choose the ERM:

- Step 1 (convergence): for a fixed h, show that $\hat{L}(h)$ is close to $L(h)$ with high probability.
- Step 2 (uniform convergence): show that the above holds simultaneously for all hypotheses $h \in \mathcal{H}$.

However, the assumptions are restrictive. There exists a perfect hypothesis (realizability). What happens when the problem is not realizable? To answer this, we introduce the tools of concentration estimates.

Second, the hypothesis class is finite. What happens when the number of hypotheses is infinite? To answer this, we need better ways of measuring the "size" of a set – leading to Rademacher complexity, VC dimension, and PAC-Bayes bounds (to name a few).

2.2 Concentration estimates (Lecture 4)

Concentration inequalities are powerful tools from probability theory that show the average of independent random variables will be concentrated around its expectation. Concentration estimates are the basis of a large branch of learning theory (Bach, 2024) and high-dimensional statistics (Wainwright, 2019). They are one of the most basic tools for studying supervised learning algorithms and models (Zhang, 2023), primarily because much of supervised learning deals with in-distribution samples.

Example 2.3 (Mean estimation). *Let $X_1, X_2, \ldots, X_n$ be independent and identically distributed random variables with mean $\mu = \mathbb{E}[X_i]$, for all $i = 1, 2, \ldots, n$. Define the empirical mean as*

$$\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$$

How does $\hat{\mu}_n - \mu$ behave? There are at least three types of statements one could make:

- *Consistency: by law of large numbers, $\hat{\mu}_n - \mu \to 0$.*
- *Asymptotic normality: let the variance of X_i (for all i) be equal to σ^2, by the central limit theorem, we have $\sqrt{n}(\hat{\mu}_n - \mu) \sim \mathcal{N}(0, \sigma^2)$.*
- *Tail estimates: for our purpose, we would like to draw a statement of the following type $\Pr[|\hat{\mu}_n - \mu| \geq \epsilon] \leq \delta$. For getting such tail estimates, we typically need to study the tail of a distribution, for instance, the tail of a Gaussian distribution, etc.*

Markov's inequality: Let $Z \geq 0$ be a non-negative random variable. Then

$$\Pr[Z \geq t] \leq \frac{\mathbb{E}[Z]}{t}$$

Proof. Since Z is a non-negative quantity, we always have the condition that

$$t\mathbb{1}_{Z \geq t} \leq Z$$

To see this, notice that if $Z \geq t$, then the above is true. On the other hand, if $Z < t$, then the left-hand side above is zero, whereas $Z \geq 0$. Next, take the expectation over Z on both sides, we get

$$\mathbb{E}[t\mathbb{1}_{Z \geq t}] \leq \mathbb{E}[Z] \Rightarrow \mathbb{E}[\mathbb{1}_{Z \geq t}] \leq \frac{\mathbb{E}[Z]}{t}$$

Notice that $\mathbb{E}[\mathbb{1}_{Z \geq t}] = \Pr[Z \geq t]$. Thus, we have shown that Markov's inequality is true.

Chebyshev's inequality: Let X be a random variable with mean equal to μ. Then

$$\Pr[|X - \mu| \geq \epsilon] \leq \frac{\mathrm{Var}[X]}{\epsilon^2}$$

Proof. We will use Markov's inequality to get this result. Let $Z = (X - \mu)^2$ and let $t = \epsilon^2$. Notice that $Z \geq 0$. Thus, based on Markov's inequality

$$\Pr[Z \geq t] \leq \frac{\mathbb{E}[Z]}{t} = \frac{\mathbb{E}\left[(Z - \mu)^2\right]}{t} = \frac{\mathrm{Var}[Z]}{t},$$

which completes the proof.

Hoeffding's inequality: Let $Z_1, Z_2, \ldots, Z_n$ be n independent and identically distributed random variables drawn from a distribution with expectation μ and whose values are bounded from above by one.

Let $\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^n Z_i$ denote the mean of the n random variables. Then, for any $\epsilon \in (0, 1)$, we have

$$\Pr[|\hat{\mu}_n - \mu| > \epsilon] \leq 2\exp(-2\epsilon^2 n). \tag{13}$$

The Hoeffding's inequality is a very powerful result when we work with the average of n random variables. Variants of this inequality (which is restricted to bounded random variables) are also called Chernoff bound.[3] Next, we shall see a proof through the use of moment generating functions (MGF).

[3] https://en.wikipedia.org/wiki/Chernoff_bound

2.2.1 Moment generating functions

Definition 2.4 (Moment generating functions)**.** *For a random variable X, its MGF is given by*

$$M_X(t) := \mathbb{E}\left[\exp(tX)\right]$$

One can also think of the MGF in terms of Taylor's expansion of $\exp(tX)$ *as*

$$M_X(t) = 1 + t\,\mathbb{E}\left[X\right] + \frac{t^2}{2}\,\mathbb{E}\left[X^2\right] + \frac{t^3}{6}\,\mathbb{E}\left[X^3\right] + \cdots$$

Thus, the first moment is the mean of X. The second moment is the variance of X (assuming the mean of X is zero). And so on.

The MGF of the sum of two independent random variables X and Y is the product of the MGF of X and Y, respectively.

- To see this, notice that

$$M_{X+Y}(t) = \mathbb{E}\left[e^{t(X+Y)}\right] = \mathbb{E}\left[e^{tX}e^{tY}\right] = \mathbb{E}\left[e^{tX}\right]\mathbb{E}\left[e^{tY}\right] = M_X(t)M_Y(t)$$

- Here we have used that X and Y are independent, and hence e^{tX} and e^{tY} are independent, to conclude that $\mathbb{E}\left[e^{tX}e^{tY}\right] = \mathbb{E}\left[e^{tX}\right]\mathbb{E}\left[e^{tY}\right]$

Example 2.5 (Gaussian random variables)**.** *In this example, we shall work through the above Definition 2.4 in the case of a Gaussian random variable. Let $X \sim \mathcal{N}(0, \sigma^2)$. Then, we will show that $M_X(t) = e^{\sigma^2 t^2/2}$. To derive this, we use the definition of the probability density of Gaussian random variables:*

$$\begin{aligned} M_X(t) = \mathbb{E}\left[e^{tX}\right] &= \int \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2 - 2\sigma^2 t x}{2\sigma^2}\right) dx \\ &= \int \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\sigma^2 t)^2 - \sigma^4 t^2}{2\sigma^2}\right) dx \\ &= \exp\left(\frac{\sigma^2 t^2}{2}\right) \end{aligned}$$

The high-level idea for showing the Hoeffding's inequality is obtained by applying Markov's inequality to e^{tX} for some well-chosen value t. From Markov's inequality, we can derive the following useful inequality: for any $t > 0$,

$$\Pr[X \ge a] = \Pr\left[e^{tX} \ge e^{ta}\right] \le \frac{\mathbb{E}\left[e^{tX}\right]}{e^{ta}}$$

In particular,

$$\Pr[X \ge a] \le \min_{t>0} \frac{\mathbb{E}\left[e^{tX}\right]}{e^{ta}} \tag{14}$$

Similarly, for any $t < 0$,

$$\Pr[X \le a] = \Pr[e^{tX} \ge e^{ta}] \le \min_{t<0} \frac{\mathbb{E}\left[e^{tX}\right]}{e^{ta}}$$

Bounds for specific distributions are obtained by choosing appropriate values for t. See Chapter 4 in Mitzenmacher and Upfal, 2017 for further references.

2.2.2 Chernoff bounds for the sum of Poisson trials

We now develop the most commonly used version of the Chernoff bound for the tail distribution of a sum of independent 0-1 random variables, which are also known as Poisson trials.[4]

Let $X_1, \ldots, X_n$ be a sequence of independent Poisson trials with $\Pr[X_i = 1] = p_i$. Let $X = \sum_{i=1}^{n} X_i$, and let

$$\mu = \mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathbb{E}[X_i] = \sum_{i=1}^{n} p_i$$

For a given $\delta > 0$, we are interested in bounds on $\Pr[X \geq (1+\delta)\mu]$ and $\Pr[X \leq (1-\delta)\mu]$, that is, the probability that X deviates from its expectation μ by $\delta\mu$ or more. To develop a Chernoff bound, we need to compute the moment generating function of X. We start with the MGF of each X_i:

$$\begin{aligned} M_{X_i}(t) = \mathbb{E}\left[e^{tX_i}\right] = p_i e^t + (1-p_i) &= 1 + p_i(e^t - 1) \\ &\leq e^{p_i(e^t-1)}, \end{aligned}$$

where in the last step, we have used the fact that, for any y, $1 + y \leq e^y$. Since the X_i's are independent from each other, we take the product of the n MGF to obtain

$$\begin{aligned} M_X(t) = \prod_{i=1}^{n} M_{X_i}(t) &\leq \prod_{i=1}^{n} e^{p_i(e^t-1)} \\ &= \exp\left(\sum_{i=1}^{n} p_i(e^t - 1)\right) = e^{(e^t-1)\mu} \end{aligned}$$

Based on this result, we now apply Markov's inequality: for any $t > 0$, we have

$$\begin{aligned} \Pr[X \geq (1+\delta)\mu] &= \Pr[e^{tX} \geq e^{t(1+\delta)\mu}] \\ &\leq \frac{\mathbb{E}\left[e^{tX}\right]}{e^{t(1+\delta)\mu}} \\ &\leq \frac{e^{(e^t-1)\mu}}{e^{t(1+\delta)\mu}} \end{aligned}$$

For any $\delta > 0$, we can set $t = \ln(1+\delta) > 0$ to get

$$\Pr[X \geq (1+\delta)\mu] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}$$

For $0 < \delta \leq 1$, we need to show that

$$\frac{e^{\delta}}{(1+\delta)^{1+\delta}} \leq e^{-\delta^2/3}$$

Taking the logarithm of both sides, we obtain

$$f(\delta) = \delta - (1+\delta)\ln(1+\delta) + \frac{\delta^2}{3} \leq 0$$

[4] Poisson trials differ from Poisson random variables.

Computing the derivatives of $f(\delta)$, we have:

$$f'(\delta) = 1 - \frac{1+\delta}{1+\delta} - \ln(1+\delta) + \frac{2\delta}{3}, f''(\delta) = -\frac{1}{1+\delta} + \frac{2}{3}$$

We see that the second derivative is less than zero if $\delta < 1/2$, and it is positive otherwise. Hence, $f'(\delta)$ first decreases and then increases in the interval $[0, 1]$. Since $f'(0) = 0$ and $f'(1) < 0$, we conclude that $f'(\delta) \le 0$ in the interval $[0, 1]$. Since $f(0) = 0$, it follows that $f(\delta) \le 0$, which shows that for any δ between 0 and 1, we have

$$\Pr[X \ge (1+\delta)\mu] \le e^{-\mu\delta^2/3}$$

Next, let us look at an example of Chernoff bounds.

Example 2.6 (Coin flips)**.** *Let X be the number of heads in a sequence of n independent fair coin flips. That is $X = X_1 + X_2 + \cdots + X_n$, where every X_i is a Bernoulli random variable yielding one with probability $1/2$. Applying the Chernoff bound, we have*

$$\Pr\left[\left|X - \frac{n}{2}\right| \ge \frac{1}{2}\sqrt{6n\ln n}\right] \le 2\exp\left(-\frac{1}{3}\frac{n}{2}\frac{6\ln n}{n}\right) = \frac{2}{n}$$

This demonstrates that the concentration of the number of heads around the mean $n/2$ is very tight. Most of the time, the deviations from the mean are on the order of $O(\sqrt{n\ln n})$.[5]

Example 2.7 (Gaussian random variables)**.** *Recall that the MGF of a standard Gaussian random variable is $M_X(t) = e^{\sigma^2 t^2/2}$. We derive a tail bound by plugging in $M_X(t)$ to equation* (14)*:*

$$\Pr[X \ge \epsilon] \le \inf_t \exp\left(\frac{\sigma^2 t^2}{2} - t\epsilon\right)$$

The minimum of the RHS is attained by setting $t = \frac{\epsilon}{\sigma^2}$, yielding:

$$\Pr[X \ge \epsilon] \le \exp\left(-\frac{\epsilon^2}{2\sigma^2}\right)$$

What about non-Gaussian random variables? Notice that the bound still holds if we replace $M_X(t)$ with an upper bound. This motivates the following definition.

Definition 2.8 (Sub-Gaussian)**.** *A mean zero random variable X is sub-Gaussian with parameter σ^2 if its MGF is bounded as follows:*

$$M_X(t) \le \exp\left(\frac{\sigma^2 t^2}{2}\right) \tag{15}$$

It follows that for sub-Gaussian X, we again have that $\Pr[X \ge \epsilon] \le \exp(-\epsilon^2/(2\sigma^2))$.

[5]Another use case of Chernoff bound: Suppose we are interested in evaluating the probability that a particular gene mutation occurs in the population. Given a DNA sample, a lab test can determine if it carries the mutation. However, the lab test is expensive and we would like to obtain a relatively reliable estimate from a small number of tests.

Example 2.9 (Revisiting ERM in the lens of sub-Gaussian random variables)**.** *Imagine you are training a classifier. You want to know if the training error is a good proxy for the test error. For most ML loss functions, their loss values are bounded (e.g., between* 0 *and* 1 *or another small constant* $C > 0$*). Next, we shall verify that any bounded random variable is sub-Gaussian. For any* $t > 0$*, suppose that* X *is a bounded random variable between* a *and* b*. First, the variance of* X *is at most* $(b-a)^2/4$*. To see this,*

$$\begin{aligned}
\mathrm{Var}[X] &= \inf_t \mathbb{E}\left[(X-t)^2\right] && \text{(because the minimum is achieved when } t = \mathbb{E}[X]) \\
&\leq \mathbb{E}\left[\left(X - \frac{b+a}{2}\right)^2\right] && \text{(one can verify that } \mathbb{E}\left[(t - \tfrac{b+a}{2})(\tfrac{b+a}{2} - X)\right] \leq 0) \\
&\leq \max\left((b-(b+a)/2)^2, (a-(b+a)/2)^2\right),
\end{aligned}$$

which is $(b-a)^2/4$*.*

Next, let $\varphi(\lambda) = \log \mathbb{E}_P\left[e^{\lambda X}\right]$*. Let* $Q_\lambda(x)$ *be the distribution of* x *defined by*

$$dQ_\lambda(x) = \frac{e^{\lambda x}}{\mathbb{E}_{X\sim P}\left[e^{\lambda X}\right]} dP(x)$$

Next, let us look at $\varphi'(\lambda)$ *and* $\varphi''(\lambda)$*.*

$$\begin{aligned}
\varphi'(\lambda) &= \frac{\mathbb{E}_P\left[Xe^{\lambda X}\right]}{\mathbb{E}_P\left[e^{\lambda X}\right]} = \mathop{\mathbb{E}}_{Q_\lambda}[X], \\
\varphi''(\lambda) &= \frac{\mathbb{E}_P\left[X^2 e^{\lambda X}\right]}{\mathbb{E}_P\left[e^{\lambda X}\right]} - \frac{(\mathbb{E}_P\left[Xe^{\lambda X}\right])^2}{(\mathbb{E}_P\left[e^{\lambda X}\right])^2} = \mathrm{Var}_{Q_\lambda}[X] \leq \frac{(b-a)^2}{4}
\end{aligned}$$

Lastly, by Taylor's theorem, there exists λ' *between* 0 *and* λ *(when* $\lambda > 0$*; The case when* $\lambda < 0$ *is similar) such that*

$$\varphi(\lambda) = \varphi(0) + \varphi'(0)\lambda + \frac{1}{2}\varphi''(\lambda')\lambda^2 \leq \frac{\lambda^2(b-a)^2}{8}$$

Plugging this back to the expression of $\varphi(\lambda)$ *completes the proof.*

2.3 Using uniform convergence to reason about generalization (Lecture 5)

We now give a high-level picture of the logic behind how we can use uniform convergence to reason about generalization in the context of ERM. This will lead us to the concept of Rademacher complexity, which is a crucial notion in statistical learning theory. We would like to show that the excess risk of ERM is small:

$$\Pr\left[L(\hat{h}_{\mathrm{ERM}}) - L(h^\star) \geq \epsilon\right] \leq \delta \tag{16}$$

We can expand the excess risk as

$$L(\hat{h}) - L(h^\star) = \underbrace{\left(L(\hat{h}) - \hat{L}(\hat{h})\right)}_{\text{Uniform convergence}} + \underbrace{\left(\hat{L}(\hat{h}) - \hat{L}(h^\star)\right)}_{\leq 0} + \underbrace{\left(\hat{L}(h^\star) - L(h^\star)\right)}_{\text{Concentration}} \tag{17}$$

We'll see how concentration estimates can be used to control this difference in the third part. However, the same reasoning does not apply to the first part because the ERM $\hat{h}_{\mathrm{ERM}}$ depends on the training examples $\hat{L}$. In particular,

$$\hat{L}(\hat{h}_{\mathrm{ERM}}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\hat{h}_{\mathrm{ERM}}(x_i), y_i). \tag{18}$$

Example 2.10 (Stochastic optimization). *In practice, $\hat{h}_{\text{ERM}}$ is often computed using* stochastic optimization*, such as stochastic gradient descent (SGD), which updates:*

$$h^{(t+1)} = h^{(t)} - \eta_t \nabla \ell(h^{(t)}(x_{i_t}, y_{i_t})),$$

where i_t is sampled uniformly from $1, 2, \ldots, n$.

Each SGD step is an unbiased estimate of the gradient of $\hat{L}(h)$, but the final iterate $\hat{h}_{\text{ERM}}$ depends on the entire dataset through the optimization trajectory.

Due to the correlation, $\hat{L}$ is not a sum of independent random variables. The central thesis of uniform convergence is that if we could ensure that $L(h)$ and $\hat{L}(h)$ are close for all $h \in \mathcal{H}$, then $L(\hat{h}_{\text{ERM}})$ must be close to $\hat{L}(\hat{h}_{\text{ERM}})$ as well. In summary, the goal of uniform convergence can be stated as

$$\Pr[L(\hat{h}_{\text{ERM}}) - L(h^\star) \geq \epsilon] \leq \Pr\left[\left(\sup_{h \in \mathcal{H}} \left|L(h) - \hat{L}(h)\right|\right) \geq \frac{\epsilon}{2}\right] \leq \delta \tag{19}$$

In particular, the 1/2 above comes from combining the error terms from the first and third parts together. Put it in words, we'd like to upper bound the probability that the largest difference between the empirical risk and the expected risk is larger than $\epsilon/2$.

2.4 Rademacher complexity (Lecture 6)

Both of our generalization bounds require finite hypothesis classes. What about infinite hypothesis classes? Note that we cannot directly apply the union bound to infinite hypothesis classes. Instead, we need a more sophisticated way to measure complexity of a hypothesis class. With Rademacher complexity, the key idea is symmetrization. Along the way, we need an extension of the Hoeffding's inequality to some function of bounded random variables, which is known as McDiarmid's inequality. Let us first start by motivating why we need these tools.

2.4.1 Motivation

Recall that, within the uniform convergence framework, we want to get a statement of the following

$$\Pr\left[L(\hat{h}) - L(h^\star) \geq \epsilon\right] \leq \Pr\left[\sup_{h \in \mathcal{H}} \left|L(h) - \hat{L}(h)\right| \geq \frac{\epsilon}{2}\right] \leq \delta \tag{20}$$

Since there are two cases here, let us denote

$$G_n := \sup_{h \in \mathcal{H}} L(h) - \hat{L}(h), \quad G'_n := \sup_{h \in \mathcal{H}} \hat{L}(h) - L(h)$$

Then we have that

$$\Pr\left[\sup_{h \in \mathcal{H}} \left|L(h) - \hat{L}(h)\right| \geq \frac{\epsilon}{2}\right] \leq \Pr\left[G_n \geq \frac{\epsilon}{2}\right] + \Pr\left[G'_n \geq \frac{\epsilon}{2}\right]$$

Let us focus on the first case, since the second case will be similar.

Now, our main object becomes G_n: This is a rather non-trivial function, because of taking the supremum over a sum of random variables. Instead, we will look at its expectation (or first moment).

2.4.2 Definition of Rademacher complexity

Our main object of interest is now

$$\mathbb{E}[G_n] = \mathbb{E}\left[\sup_{h \in \mathcal{H}} (L(h) - \hat{L}(h))\right].$$

Recall that $\hat{L}(h)$ is the empirical risk, and $L(h)$ is the expected risk.

This quantity is still quite difficult because it depends on the expected risk, an expectation over the unknown data distribution. The key idea of symmetrization is to remove this expected risk term with a simpler term. Let us imagine n data points $Z_1' = (x_1', y_1'), Z_2' = (x_2', y_2'), \ldots, Z_n' = (x_n', y_n')$, sampled from the true data distribution. Then, clearly $L(h) = \mathbb{E}\left[\hat{L}'(h)\right]$, where $\hat{L}'(h)$ is the empirical risk on this copy dataset. Hence,

$$\begin{aligned}
\mathbb{E}[G_n] &= \mathop{\mathbb{E}}_{Z_{1:n}}\left[\sup_{h \in \mathcal{H}} (L(h) - \hat{L}(h))\right] \\
&= \mathop{\mathbb{E}}_{Z_{1:n}}\left[\sup_{h \in \mathcal{H}} \mathop{\mathbb{E}}_{Z'_{1:n}}\left[\hat{L}'(h) - \hat{L}(h)\right]\right] \\
&= \mathop{\mathbb{E}}_{Z_{1:n}}\left[\mathop{\mathbb{E}}_{Z'_{1:n}}\left[\sup_{h \in \mathcal{H}} (\hat{L}'(h) - \hat{L}(h))\right]\right]
\end{aligned}$$

Let us remove the dependence on the copy dataset. To that end, we introduce independent Rademacher random variables $\sigma_1, \sigma_2, \ldots, \sigma_n$ sampled uniformly from $\{+1, -1\}$. Notice that

$$\hat{L}'(h) - \hat{L}(h) = \sum_{i=1}^{n} (\ell(h(x_i'), y_i') - \ell(h(x_i), y_i)),$$

is symmetric around zero. Hence, multiplying each individual term by σ_i does not change its distribution. Thus,

$$\begin{aligned}
\mathbb{E}[G_n] &\le \mathop{\mathbb{E}}_{Z_{1:n}, Z'_{1:n}}\left[\sup_{h \in \mathcal{H}} (\hat{L}'(h) - \hat{L}(h))\right] \\
&= \mathop{\mathbb{E}}_{Z_{1:n}, Z'_{1:n}, \sigma_{1:n}}\left[\sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i (\ell(h(x_i'), y_i') - \ell(h(x_i), y_i))\right] \\
&\le 2 \mathop{\mathbb{E}}_{Z_{1:n}, \sigma_{1:n}}\left[\sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \ell(h(x_i), y_i)\right] \qquad (21)
\end{aligned}$$

The last line (without the factor of 2) is the Rademacher complexity of $\mathcal{H}$.

In summary, let $\mathcal{H}$ be a hypothesis class consisting of a class of real-valued functions. Example: two-layer neural nets, linear models. Define the Rademacher complexity (or Rademacher average) of $\mathcal{H}$ to be

$$R_n(\mathcal{H}) := \mathop{\mathbb{E}}_{Z_{1:n}, \sigma_{1:n}}\left[\sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \ell(h(x_i), y_i)\right],$$

where $Z_i = (x_i, y_i)$ is a random sample from the underlying data distribution, and σ_i is sampled uniformly from $\{+1, -1\}$.

Example 2.11 (Rademacher complexity of ℓ_2-norm bounded linear predictors)**.** *Let $\mathcal{H} = \{z \to \langle w, z\rangle \mid w \in \mathcal{W}\}$ be a set of linear functions such that $\|w\| \le B$ for any $w \in \mathcal{W}$. Furthermore, assume the data distribution has a bounded second moment:*

$$\mathop{\mathbb{E}}_{z\sim\mathbb{P}}\left[\|z\|^2\right] \le C^2$$

Then we have

$$R_n(\mathcal{H}) = \mathbb{E}\left[\sup_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i h(z_i)\right] \le \frac{B \cdot C}{\sqrt{n}}$$

Example 2.12 (Rademacher complexity of two-layer neural networks with bounded layer norms)**.** *Consider a two-layer ReLU neural net with m hidden units. Let $\Phi = (w \in \mathbb{R}^m, U \in \mathbb{R}^{m\times d})$ denote the parameters for the first and second layers, respectively.*

Given an input vector $x \in \mathbb{R}^d$, let

$$f_\Phi(x) = w^\top \phi(Ux) \in \mathbb{R},$$

denote the output of the network, where $\phi(\cdot)$ is the activation function that is 1-Lipschitz continuous. Let us assume the training set $\{x_i\}_{i=1}^n \subseteq \mathbb{R}^d$ lies in some bounded ℓ_2-norm ball: $\|x_i\| \le 1$ for all $i = 1, 2, \ldots, n$.

Let $\mathcal{H} = \{f_\Phi \mid \|w\| \le B', \|U\|_2 \le B\}$. Then the Rademacher complexity of $\mathcal{H}$ is bounded by

$$R_n(\mathcal{H}) \le 2BB'\sqrt{\frac{m}{n}}$$

Imagine we use a two-layer neural network to classify MNIST digits. If we use a network with m neurons. This result says that the Rademacher complexity scales with the operator norm of the first layer weights and the second layer weights.

2.4.3 Generalization bounds

To see the power of the concept we just introduced, here is a very general statement where we can always rely on when we work with supervised learning algorithms and models. Define

$$\mathcal{A} = \{(x, y) \to \ell(h(x), y) : h \in \mathcal{H}\}$$

as the loss function composed with the hypothesis space. Let $\mathcal{R}_n(\mathcal{A})$ denote the Rademacher complexity of the function class $\mathcal{A}$. With probability at least $1 - \delta$,

$$L(\hat{h}_{\text{ERM}}) - L(h^\star) \le 4R_n(\mathcal{A}) + \sqrt{\frac{2\log(2\delta^{-1})}{n}} \tag{22}$$

Recall that $\hat{h}_{\text{ERM}}$ is the empirical risk minimizer (ERM), $h^\star$ is the expected risk minimizer, and n is the size of the training set.

Example 2.13 (Interpreting G_n for linear predictors)**.** *Let $\mathcal{H} = \{x \mapsto \langle w, x\rangle : \|w\|_2 \le B\}$ and assume the loss $\ell(h(x), y)$ is bounded. For a fixed hypothesis h, the quantity $L(h) - \hat{L}(h)$ measures how much the empirical risk underestimates the true risk. The supremum in G_n searches for the worst-case hypothesis whose empirical performance looks overly optimistic. Geometrically, G_n quantifies how well the training data fits noise. If there exists a direction w such that the correlation between w and data align on the sample but not in expectation,* *then G_n will be large. Using symmetrization, we showed that $\mathbb{E}[G_n] \leq 2R_n(\mathcal{A})$, where $\mathcal{A} = (x,y) \mapsto \ell(h(x),y) \in \mathcal{H}$. For linear predictors with bounded norm, this implies $\mathbb{E}[G_n] \lesssim \frac{BC}{\sqrt{n}}$, where $C^2 = \mathbb{E}\left[\|x\|^2\right]$. Thus, as the sample size grows, even the most overfit hypothesis cannot exploit randomness in the data beyond $O(n^{-1/2})$ scaling.*

2.4.4 Properties of Rademacher complexity

Boundedness:

$$R_n(\mathcal{A}) \leq \max_{h\in\mathcal{H}} \max_{x,y} \ell(h(x),y)$$

This only shows that the Rademacher complexity is bounded by some constant. Usually, we'd like to show it goes to zero as n goes to infinity.

Monotonicity: If $\mathcal{H}_1 \subseteq \mathcal{H}_2$, then $R_n(\mathcal{H}_1) \leq R_n(\mathcal{H}_2)$.
This is because we now take the supreme over a larger set, hence R_n increases.

Scaling: $R_n(c \cdot \mathcal{H}) = c \cdot R_n(\mathcal{H})$. Multiplying the functions within $\mathcal{H}$ by a fixed constant $c > 0$ increases the Rademacher complexity of $\mathcal{H}$ by a factor of c.

Lipschitz composition: Suppose ϕ is a Lipschitz-continuous function, bounded by some constant c_ϕ. Recall a function is Lipschitz-continuous if changing x only changes the function value by c_ϕ times. In addition, $\phi(0) = 0$.
Let $\phi \circ \mathcal{H} = \{(x,y) \to \phi(h(x),y) : h \in \mathcal{H}\}$, i.e., compose ϕ with h. Then, we have that

$$R_n(\phi \circ \mathcal{H}) \leq c_\phi \cdot R_n(\mathcal{H}).$$

The proof essentially uses the contraction property of the sum of Rademacher random variables. For details, see Corollary 3.17 of Ledoux and Talagrand, 2013. This property is useful because we can start by studying a simpler hypothesis class, and then compose more functions with the class without going through the calculation again.

Convex hull: The convex hull of a set $\mathcal{H}$ is when we take all possible linear combinations of the hypotheses in $\mathcal{H}$. One useful property is that taking the convex of $\mathcal{H}$ preserves the Rademacher complexity: $R_n(\mathcal{H}) = R_n(\text{convex-hull}(\mathcal{H}))$.
To see this, notice that the supreme over the convex hull must be attained at a vertex of the convex hull. This property is quite useful if we want to calculate the Rademacher complexity of a polytope. Though it is an infinite set, it suffices to examine the vertices of the polytope. We'll use this property when we examine the ℓ_1-regularization.

2.4.5 Proof of Rademacher complexity generalization bounds (optional)

The proof involves three steps:

- Show that empirical Rademacher complexity is close to the expectation.
- Use McDiarmid's inequality, which is essentially a concentration result for functions of independent random variables.

- Show that the Rademacher complexity upper bounds the excess risk. We've seen this step from Section 2.4.1.

In more detail, recall that

$$G_n = \sup_{h \in \mathcal{H}} L(h) - \hat{L}(h).$$

Our first claim is to show that G_n is close to $\mathbb{E}[G_n]$:

$$\Pr[G_n \geq \mathbb{E}[G_n] + \epsilon] \leq \exp(-2n\epsilon^2) \tag{23}$$

This shows that G_n is indeed close to its expectation plus a small error. Hence it suffices to upper bound its expectation.

Our second claim is to show that $\mathbb{E}[G_n]$ is upper bounded by the Rademacher complexity.

$$\mathbb{E}[G_n] \leq 2R_n(\mathcal{A}).$$

We have already seen the proof of this claim. Combined together, we can derive (22).

Recall that (23) that we'd like to show G_n and its expectation $\mathbb{E}[G_n]$ are close to each other. In order to complete this result, we'll use a tool called McDiarmid's inequality.

McDiarmid's inequality (The bounded differences inequality): Let f be a function satisfying the following bounded differences condition for all $i = 1, 2, \ldots, n$ and all $x_1, x_2, \ldots, x_n$, and any x_i':

$$\left| f(x_1, x_2, \ldots, x_n) - f(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_n) \right| \leq c_i$$

In other words, modifying one coordinate does not change the function value too much.

Let $X_1, X_2, \ldots, X_n$ be independent random variables. Then

$$\Pr[|f(X_1, X_2, \ldots, X_n) - \mathbb{E}[f(X_1, X_2, \ldots, X_n)]| \geq \epsilon] \leq \exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^n c_i^2}\right) \tag{24}$$

Remark 2.14 (McDiarmid's inequality generalizes Hoeffding's inequality)**.** *Let the function*

$$f(x_1, x_2, \ldots, x_n) = \frac{1}{n} \sum_{i=1}^n x_i$$

be the empirical mean. Suppose every x_i and x_i' are bounded from above by one. Then, changing x_i to x_i' changes the function value by at most $\frac{2}{n}$, leading to $c_i = \frac{2}{n}$ for all $i = 1, 2, \ldots, n$. Hence, the sum of c_i^2 is equal to $\frac{4}{n}$. Plugging this to (24) recovers the Hoeffding's inequality.

This result is quite powerful, since it holds for any independent random variables, and f can be complex (say a neural net). As long as this function is not too sensitive to perturbations in one coordinate, we get good concentration.

Proof of equation (24). Recall the definition of a martingale sequence A sequence of random variables $Z_0, Z_1, \ldots, Z_n$ is a martingale sequence with respect to another sequence of random variables $X_1, X_2, \ldots, X_n$ if and only if Z_i only depends on $X_{1:i}$ and

$$\mathbb{E}[Z_i | X_{1:i-1}] = Z_{i-1}.$$

For a martingale sequence, we have the concentration because of Azuma-Hoeffding's inequality: Suppose $Z_{1:t}$ is a martingale sequence such that

$$\mathbb{E}\left[Z_i | Z_{<i}\right] = 0, \text{ and } |Z_i| \le c_i.$$

Then with probability at least $1-\delta$,

$$\sum_{i=1}^{n} Z_i \le \sqrt{2\Big(\sum_{i=1}^{n} c_i^2\Big)\log(\delta^{-1})} \tag{25}$$

For reference, see Chapter 12.4 of Mitzenmacher and Upfal (2017).[6]

We're going to construct a sequence

$$Z_i = \mathbb{E}\left[f(X_1, X_2, \ldots, X_n) \mid X_{1:i}\right] \tag{26}$$

Let's check the following properties

- Clearly, Z_i only depends on the values of $X_{1:i}$, but not of the remaining sequence
- Besides,
$$\mathbb{E}\left[Z_i \mid X_{1:i-1}\right] = \mathbb{E}\left[\mathbb{E}\left[f(X_1, X_2, \ldots, X_n) \mid X_{1:i}\right] \mid X_{1:i-1}\right] = Z_{i-1}$$
- Note that for $i = 0$,
$$Z_0 = \mathbb{E}\left[f(X_1, X_2, \ldots, X_n)\right]$$
- For $i = n$,
$$Z_n = \mathbb{E}\left[f(X_1, \ldots, X_n) \mid X_{1:n}\right] = f(X_1, \ldots, X_n)$$

Using these notations, we'd like to bound the following

$$\Pr[Z_n - Z_0 \ge \epsilon]$$

Let $D_i = Z_i - Z_{i-1}$. Clearly, $Z_n - Z_0 = \sum_{i=1}^{n} D_i$. We're going to show that $|D_i|$ is bounded by c_i. Therefore, using Azuma-Hoeffding's inequality, we can get an exponentially decreasing tail bound for $Z_n - Z_0$. We write down both Z_i and Z_{i-1} as follows:

$$\begin{aligned} Z_{i-1} &= \mathbb{E}\left[f(X_1, \ldots, X_n) \mid X_{1:i-1}\right] \\ Z_i &= \mathbb{E}\left[f(X_1, \ldots, X_n) \mid X_{1:i}\right] \end{aligned}$$

The key difference is whether we condition on X_i or not above. Imagine the set of possible realizations of X_i. We'll consider the maximum and minimum achieving realizations

$$\begin{aligned} L_i &= \inf_x \mathbb{E}\left[f(X_1, \ldots, X_{i-1}, X_i = x, \ldots, X_n) \mid X_{1:i-1}, X_i = x\right] - Z_{i-1} \\ U_i &= \sup_x \mathbb{E}\left[f(X_1, \ldots, X_{i-1}, X_i = x, \ldots, X_n) \mid X_{1:i-1}, X_i = x\right] - Z_{i-1} \end{aligned}$$

Clearly, D_i is somewhere between L_i and U_i. Suppose L_i is achieved by $X_i = x_L$ and U_i is achieved by $X_i = x_U$.

[6] See also https://en.wikipedia.org/wiki/Azuma%27s_inequality

By the bounded difference assumption, for any fixed values $X_{1:n}$ except X_i (being x_L or x_U), they are at most c_i apart. Taking expectation over $X_{1:i-1}$ and note that they are independent of X_i (key!), we have that

$$\begin{aligned}|U_i| &= |\mathbb{E}\left[f(X_1,\ldots,X_{i-1},X_i=x_U,\ldots,X_n) - f(X_1,\ldots,X_{i-1},X_i,\ldots,X_n) \mid X_{1:i-1}\right]| \\ &\le c_i,\end{aligned}$$

by the bounded difference condition on X_i. And similarly, $|L_i| \le c_i$. Thus, we have verified that the sequence $D_1, D_2, \ldots, D_n$ is bounded. In addition, we can check that they are a Martingale sequence based on the properties listed above.

Having checked through McDiarmid's inequality, it remains to show that G_n satisfies the bounded difference condition. Recall that

$$G_n = \sup_{h\in\mathcal{H}} L(h) - \hat{L}(h).$$

This quantity naturally arises as we derive a uniform convergence claim. Recall that $\hat{L}$ is the empirical risk. Consider changing Z_i to Z_i' in G_n, we'd like to bound the difference

$$\left|\left(\sup_{h\in\mathcal{H}} L(h) - \hat{L}(h)\right) - \left(\sup_{h\in\mathcal{H}}\left(L(h) - \hat{L}(h) + \frac{1}{n}(\ell(Z_i,h) - \ell(Z_i',h))\right)\right)\right| \le \frac{1}{n}$$

This is because the loss values are within 0 and 1. Taking supreme cannot increase the difference.

We now put all the pieces together to complete the proof of equation (22).

Proof of equation (22). We'd like to show that, for

$$\epsilon = 4R_n(\mathcal{A}) + \sqrt{\frac{2\log(2\delta^{-1})}{n}},$$

we have

$$\Pr[L(\hat{h}_{\mathrm{ERM}}) - L(h^\star) \ge \epsilon] \le \Pr\left[\sup_{h\in\mathcal{H}}\left|L(h) - \hat{L}(h)\right| \ge \frac{\epsilon}{2}\right] \le \delta$$

The latter reduces to showing that

$$\Pr\left[\sup_{h\in\mathcal{H}}\left|L(h) - \hat{L}(h)\right| \ge \frac{\epsilon}{2}\right] \le \Pr[G_n \ge \frac{\epsilon}{2}] + \Pr[G_n' \ge \frac{\epsilon}{2}]$$

By the steps above, we have

$$\begin{aligned}\Pr[G_n \ge \frac{\epsilon}{2}] &\le \Pr\left[G_n - \mathbb{E}\left[G_n\right] \ge \sqrt{\frac{\log(2\delta^{-1})}{2n}}\right] \\ &\le \exp\left(-2n\left(\frac{\log(2\delta^{-1})}{2n}\right)\right) = \frac{\delta}{2}\end{aligned}$$

The first line is because by equation (21),

$$\mathbb{E}\left[G_n\right] \le 2R_n(\mathcal{A}).$$

A similar proof applies to show that $\Pr[G_n' \ge \frac{\epsilon}{2}] \le \frac{\delta}{2}$.

2.5 Examples of using Rademacher complexity (Lectures 7-8)

So far, we have set up Rademacher complexity for bounding the complexity of an infinite hypothesis class. Now, let us apply Rademacher complexity for the case when the function class is finite. Then, we will go through several examples where the function class is infinite.

2.5.1 Learning finite hypothesis classes

Massart's finite lemma: Let $\mathcal{H}$ be a set of functions/hypotheses. Let M^2 be a bound on the second moment of functions of $\mathcal{H}$:

$$\sup_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (f(Z_i))^2 \leq M^2,$$

where $Z_i = (x_i, y_i)$ denotes an input data sample. Then, the empirical Rademacher complexity is bounded by:

$$\hat{R}_n(\mathcal{H}) \leq \sqrt{\frac{2M^2 \log(|\mathcal{H}|)}{n}} \tag{27}$$

Example 2.15 (Model Selection in AutoML)**.** *In modern AutoML pipelines, we often select a model from a finite library $\mathcal{H}$ of M pre-trained architectures (e.g., different versions of Vision Transformers). Since the models are fixed and we only choose the one with the minimum empirical risk $\hat{R}_n(h)$, our hypothesis class is finite. By Massart's Lemma, the complexity of this selection process scales with $\sqrt{\log M}$, meaning we can safely choose from thousands of models without significantly increasing the generalization gap, provided our validation set n is sufficiently large. For instance, if there are* 100 *models, the overfitting risk from picking the best model grows only with $\sqrt{\log(100)}$.*

Proof. For simplicity, let us denote $W_f = \frac{1}{n}\sum_{i=1}^{n} \sigma_i f(Z_i)$. Recall that the empirical Rademacher complexity is defined as

$$\hat{R}_n(F) = \mathop{\mathbb{E}}_{\sigma_{1:n}} \left[\sup_{f \in \mathcal{H}} W_f \mid Z_{1:n} \right]$$

Let us write down the moment generating function of $\sup_{f \in \mathcal{H}} W_f$, and we'll use the convexity of the exponential function

$$\begin{aligned} \exp\left(t \cdot \mathbb{E}\left[\sup_{f \in \mathcal{H}} W_f \mid Z_{1:n} \right] \right) &\leq \mathbb{E}\left[\exp\left(t \cdot \sup_{f \in \mathcal{H}} W_f \right) \mid Z_{1:n} \right] \\ &= \mathbb{E}\left[\sup_{f \in \mathcal{H}} \exp\left(t \cdot W_f \right) \mid Z_{1:n} \right] \end{aligned}$$

The above can be further bounded by

$$|\mathcal{H}| \cdot \mathbb{E}\left[\exp\left(t \cdot W_f \right) \mid Z_{1:n} \right]$$

Finally, recall that W_f is the sum of n independent random variables. Hence, we invoke the product property of the individual MGFs, to get that $M_{W_f}(t)$ is the product of the individual random variables. Recall that we've conditioned on $Z_{1:n}$, which can be treated as constants. Thus, the randomness is now just on $\sigma_{1:n}$—each σ_i is either $+1$ or -1, scaled by $f(Z_i)$ (which is a constant conditioned on Z_i). We can show that σ_i is sub-Gaussian with parameter 1, since we have the fact that for any random variable X bounded between a and b (for some $a < b$), then X must be $(b-a)^2/4$ sub-Gaussian.[7]

As a result, W_f is sub-Gaussian with parameter at most $\frac{1}{n^2}\sum_{i=1}^n (f(Z_i))^2 \le M^2/n$. Therefore, we now get that

$$\mathbb{E}\left[\exp(t \cdot W_f) \mid Z_{1:n}\right] \le \exp\left(\frac{t \cdot M^2}{2n}\right)$$

We can now conclude that

$$\exp(t\hat{R}_n(f)) \le |\mathcal{H}| \exp\left(\frac{tM^2}{2n}\right)$$

Taking log on both sides and dividing by t, we get:

$$\hat{R}_n(\mathcal{H}) \le \frac{\log(|\mathcal{H}|)}{t} + \frac{tM^2}{2n}$$

By setting t to minimize the right-hand side above, we conclude that $\hat{R}_n(\mathcal{H}) \le \sqrt{\frac{2\log(|\mathcal{H}|)M^2}{n}}$.

We can apply the above result along with (22) to derive the result for learning from a bounded set of 0-1 valued functions.

2.5.2 Norm-constrained linear models

The next two examples involve learning a linear model from a norm-bounded hypothesis space. The first example applies to linear regression with ℓ_2-norm constraints, for instance, training models with weight decay. The second example applies to linear regression with ℓ_1-norm constraints, i.e., training with ℓ_1-regularization. Next, we'll derive the Rademacher complexity of these two examples.

Proof of Example 2.11. The key idea is to exploit the linear algebraic structure of the Rademacher complexity

$$\begin{aligned} R_n(\mathcal{H}) = \mathbb{E}\left[\sup_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^n \sigma_i h(z_i)\right] &= \mathbb{E}\left[\sup_{w\in\mathcal{W}} \frac{1}{n}\sum_{i=1}^n \sigma_i \langle w, z_i\rangle\right] \\ &\le \mathbb{E}\left[\sup_{w\in\mathcal{W}} \left\|\frac{w}{n}\right\| \cdot \left\|\sum_{i=1}^n \sigma_i z_i\right\|\right] \\ &\le \frac{B}{n} \cdot \mathbb{E}\left[\left\|\sum_{i=1}^n \sigma_i z_i\right\|\right] \end{aligned}$$

by the condition placed on $\mathcal{W}$. Finally, by Jensen's inequality, the above is at most

$$\frac{B}{n} \cdot \sqrt{\mathbb{E}\left[\left\|\sum_{i=1}^n \sigma_i z_i\right\|^2\right]} = \frac{B}{n} \cdot \sqrt{\sum_{i=1}^n \mathbb{E}\left[\|z_i\|^2\right]} \le \frac{B \cdot C}{\sqrt{n}}$$

[7]For a proof see this blog post: https://statisticaloddsandends.wordpress.com/2018/10/05/bounded-random-variables-are-sub-gaussian/

Example 2.16 (Rademacher complexity of ℓ_1-norm constrained balls)**.** *In some applications, we have a finite but large set of features, and we believe that only a small subset of them are relevant to our task. For such applications, ℓ_1-regularization has proven to be a working strategy (e.g., pruning, compression). Assume that the entries of the input vector are bounded from above by some constant $C > 0$: $\max_{j=1}^d z_{i,j} \le C$, for $i = 1, 2, \ldots, n$. Let $\mathcal{H} = \{z \to \langle w, z\rangle \mid \|w\|_1 \le B\}$ be a set of linear functions. Then, the Rademacher complexity of $\mathcal{H}$ is bounded as follows*

$$R_n(\mathcal{H}) \le \frac{B \cdot C \cdot \sqrt{2\log(2d)}}{\sqrt{n}}$$

Proof. The key idea is that the ℓ_1 ball is the convex hull of $2d$ weight vectors aligned with the basis

$$W = \{Be_1, -Be_1, Be_2, -Be_2, \ldots, Be_d, -Be_d\}$$

Since the Rademacher complexity of this class is equal to the Rademacher complexity of its convex hull, we just look at the finite class W since

$$R_n(\mathcal{H}) = \mathbb{E}\left[\sup_{w\in W} \frac{1}{n}\sum_{i=1}^n \sigma_i \langle w, z_i\rangle\right]$$

Since W is a finite set, we can apply Massart's finite lemma (see equation (27)). In order to do so, we shall first check the moment condition:

$$\frac{1}{n}\sum_{i=1}^n (\sigma_i\langle w, z_i\rangle)^2 \le \|w\|_1^2 \cdot \|z\|_\infty^2 = B^2 \cdot C^2.$$

Thus, by equation (27), we now get that

$$\hat{R}_n(\mathcal{H}) \le \sqrt{\frac{2B^2C^2\log(2d)}{n}}.$$

2.5.3 Learning two-layer neural networks

Proof of Example 2.12. For this result, we will use composition property of Rademacher complexity and result for ℓ_2-norm bounded linear predictors. Let us first write the Rademacher complexity as

$$R_n(\mathcal{H}) = \mathop{\mathbb{E}}_{\sigma_{1:n}}\left[\sup_{\Phi\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^n \sigma_i w^\top \phi(Ux)\right]$$

We first deal with the supremum over the second layer weight w by setting its direction to be the same as the other vector

$$R_n(\mathcal{H}) \le B' \cdot \mathop{\mathbb{E}}_{\sigma_{1:n}}\left[\sup_{\Phi\in\mathcal{H}} \left\|\frac{1}{n}\sum_{i=1}^n \sigma_i\phi(Ux_i)\right\|_2\right]$$

Next, because there are m neurons. Hence, by Cauchy-Schwartz inequality, we reduce the above to less than

$$B' \cdot \sqrt{m} \cdot \mathbb{E}\left[\sup_{\|u\|\le B}\left|\frac{1}{n}\sum_{i=1}^n \sigma_i \phi(u^\top x_i)\right|\right]$$

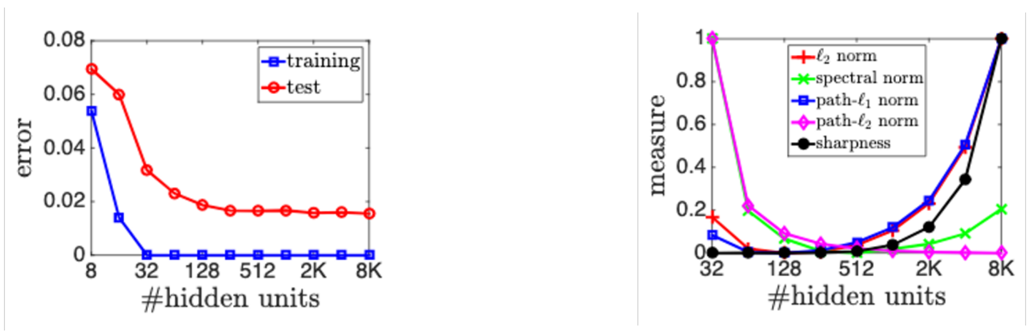


Figure 2: Vary the number of neurons m in a two-layer neural net. Path norm scales nicely with the test error. Results taken from Neyshabur et al. (2017).

The idea here is that since the operator norm of the first layer weight U is bounded by B, the norm of every row of U is also bounded by B. Furthermore, the absolute value can be bounded by adding a factor of two.

Next, we can use the composition property: ϕ is 1-Lipschitz continuous. Hence, the above is at most

$$2B' \cdot \sqrt{m} \cdot \mathbb{E}\left[\sup_{\|u\| \le B} \frac{1}{n} \sum_{i=1}^{n} \sigma_i u^\top x_i\right].$$

To finish the proof, we invoke the Rademacher complexity bound for ℓ_2-norm bounded linear predictors, to get that the above is at most $2B'B\sqrt{m/n}$.

Provided with the Rademacher complexity of two-layer neural nets, we can get a generalization bound for neural networks, with an additional error term of $\sqrt{\log(\delta^{-1})/n}$.

The above result scales with $\sqrt{m}$ – number of hidden units. Hence, the result gets worse when we over-parametrize with more hidden units. Empirically, researchers have observed that generalization error can improve as number of hidden units increases. Here's the result from training a two-layer neural net on MNIST (Neyshabur et al., 2017).

We'll show a generalization bound that does not depend on the number of hidden units m using the path norm $P(w, U) = \sum_{i=1}^{m} |w_i| \cdot \|u_i\|$, where u_i is the i-th row of U. Let $\mathcal{H} = \{f_\Phi \mid P(w, U) \le B\}$. Then, the Rademacher complexity of $\mathcal{H}$ is bounded as follows

$$R_n(\mathcal{H}) \le \frac{2B}{\sqrt{n}}.$$

Proof. Let's first write the Rademacher complexity. Recall that u_i is the i-th row of U. We expand out the Rademacher complexity using

$$w^\top \phi(Ux_i) = \sum_{j=1}^{m} w_j \phi(u_j^\top x_i)$$

Let $\bar{u}_j = u_j / \|u_j\|$. Because ReLU activation is 1-homogeneous, meaning that $\phi(cx) = c \cdot \phi(x)$. Hence,

$$w^\top \phi(Ux_i) = \sum_{j=1}^{m} w_j \|u_j\| \, \phi(\bar{u}_j^\top x_i)$$

We apply this expansion to the Rademacher complexity

$$R_n(\mathcal{H}) = \mathbb{E}\left[\sup_{\Phi} \frac{1}{n}\sum_{i=1}^{n}\sigma_i\left(\sum_{j=1}^{m} w_j \|u_j\| \phi(\bar{u}_j^\top x_i)\right)\right].$$

Taking the supremum over Φ means that we'll maximize the path norm above! Hence, after switching the order of the sum, we can get

$$R_n(\mathcal{H}) \le B \cdot \mathbb{E}\left[\max_{1\le j\le m}\left|\frac{1}{n}\sum_{i=1}^{n}\sigma_i \phi(\bar{u}_j^\top x_i)\right|\right] \le \frac{2B}{\sqrt{n}}.$$

Remark 2.17. *The second result implies the first result. To see this, we use the Cauchy-Schwartz inequality:*

$$P(w,U) = \sum_{i=1}^{m}|w_i|\cdot\|u_i\| \le \sqrt{\sum_{i=1}^{m} w_i^2}\cdot\sqrt{\sum_{i=1}^{m}\|u_i\|^2} \le B'\cdot B\cdot\sqrt{m}$$

Technically speaking, the path norm still scales with the number of hidden units. However, as we saw from the previous slide, some neurons turn out to be quite small in the experiments, resulting in a small path norm even as m increases.

2.6 Matrix completion (Lecture 9)

Suppose you are working on a recommendation system. The goal of this system is to predict the review ratings of a movie or a music database on YouTube or on Netflix. This problem can be formulated as a matrix completion problem. Suppose there is an unknown matrix $M \in \mathbb{R}^{d_1\times d_2}$, corresponding to the true ratings in the database. From the database, you see a subset of the entire matrix, denoted by $\widehat{M}$, such that $\widehat{M}_{i,j} = M_{i,j}$ for $(i,j)\in\Omega$ for some observed set of indices Ω, and $\widehat{M}_{i,j} = 0$ otherwise. Given the partially-observed data matrix $\widehat{M}$, how can we recover the unobserved entries outside Ω and impute the full matrix M?

A natural, first solution is to use rank-constrained minimization. Given an input array $\widehat{M}$, we seek to find X as close to $\widehat{M}$ as possible on Ω, subject to a rank constraint:

$$\begin{aligned}\min_{X} \quad & \sum_{(i,j)\in\Omega}(X_{i,j} - M_{i,j})^2 \\ \text{s.t.} \quad & \operatorname{rank}(X) = r\end{aligned}$$

However, the rank constraint leads to a non-convex problem, resulting in a search space whose complexity increases exponentially. Instead, consider relaxing the rank constraint into a nuclear norm constraint. Given a matrix $X \subset \mathbb{R}^{d_1\times d_2}$, let $\|X\|_\star = \sum_{j=1}^{\min(d_1,d_2)}\lambda_j(X)$, which is the sum of all the singular values of X. One can show that $\|X\|_\star$ is a convex operator in X. This is called the nuclear norm of X. Instead, consider the following nuclear norm constrained optimization:

$$\begin{aligned}\min_{X} \quad & \sum_{(i,j)\in\Omega}(X_{i,j} - M_{i,j})^2 \\ \text{s.t.} \quad & \|X\|_\star \le \alpha,\end{aligned} \tag{28}$$

for some hyperparameter $\alpha > 0$.

How well does nuclear norm minimization work? Imagine we want to recover an unknown low-rank matrix $M \in \mathbb{R}^{d_1 \times d_2}$. We observe m entries from M uniformly at random. Then, given another matrix X, we may define the mean squared recovery error as

$$\frac{1}{d_1} \frac{1}{d_2} \sum_{i=1}^{d_1} \sum_{j=1}^{d_2} (X_{i,j} - M_{i,j})^2. \tag{29}$$

Without further assumptions, we would need to see the entire matrix: Imagine the unknown matrix only has an extremely large entry but remains nonzero elsewhere. Without sampling this entry, we are only observing zeros. Therefore, we shall assume that M is bounded in all entries—this is a reasonable assumption for modeling many applications (e.g., movie ratings). In general, one may assume that M is incoherent (meaning that the row norms of the low-rank factors are bounded at the order of $O(n^{-1})$).

Remark 2.18 (Rank-constrained matrix factorization)**.** *An alternative way to formulate the above is by writing it as a convex optimization problem:*

$$\min_{X \in \mathbb{R}^{d_1 \times d_2}} \sum_{(i,j) \in \Omega} (X_{i,j} - M_{i,j})^2 + \lambda \left\| X \right\|_\star$$

This is a semi-definite program (SDP), which is convex in X. Instead, we may consider the re-parametrized optimization problem:

$$\min_{U \in \mathbb{R}^{d_1 \times r}, V \in \mathbb{R}^{d_2 \times r}} \sum_{(i,j) \in \Omega} (U_i^\top V_j - M_{i,j})^2,$$

where U_i, V_j are both r-dimensional vectors, for any i, j. Put together, $U = [U_1, U_2, \ldots, U_{d_1}]$, $V = [V_1, V_2, \ldots, V_{d_2}]$ are both rank-r matrices.

2.6.1 Recovery guarantees

Let $\alpha = dr$ and denote by $M^\star$ the output of program (28). We will show that the recovery error of $M^\star$ given by equation (29) is at most:

$$2r \sqrt{\frac{\log(d)}{md}},$$

where $d = \max(d_1, d_2)$. For example, if we want the test error less than ϵ, then the number of samples needs to be at least

$$m \geq \frac{dr^2}{\epsilon^2 \log(d)}.$$

We will use Rademacher complexity to establish this result. Define the hypothesis class to be the set of nuclear norm bounded and entrywise bounded matrices as follows:

$$\mathcal{H} = \left\{ X \in \mathbb{R}^{d_1 \times d_2} \middle| \left\| X \right\|_\star \leq \alpha, \left\| X \right\|_\infty \leq 1 \right\}$$

Then, define the Rademacher complexity of $\mathcal{H}$ as

$$R_n(\mathcal{H}) = \mathop{\mathbb{E}}_{\Omega, \sigma_{1:m}} \left[\sup_{X \in \mathcal{H}} \frac{1}{m} \sum_{(i,j) \in \Omega} \sigma_{i,j} \cdot |X_{i,j} - M_{i,j}| \right].$$

Here we use the following result. Suppose $\mathcal{H}$ is a set of matrices whose nuclear norm is bounded by C:

$$\mathcal{H} = \left\{ X \in \mathbb{R}^{d_1 \times d_2} \middle| \, \|X\|_\star \le C \right\}$$

Then, the Rademacher complexity of nuclear norm bounded matrices is at most

$$R_n(\mathcal{H}) = \mathop{\mathbb{E}}_{\Omega}\left[\sup_{X \in \mathcal{H}} \frac{1}{m} \sum_{(i,j)\in\Omega} \sigma_{i,j} |X_{i,j} - M_{i,j}| \right] \le 2C\sqrt{\frac{\log d}{md}} \tag{30}$$

Recall our result from equation (22), if $m \ge dr^2 \log(d\delta^{-1})/\epsilon^2$, then the above Rademacher complexity is at most ϵ for $\alpha = dr$. Based on equation (22), with probability at least $1-\delta$, the recovery error of the minimizer must be at most $O(\epsilon)$.

It remains to show that equation (30) is true. By symmetry, the absolute function inside $R_n(\mathcal{H})$ can be removed without changing the value of $R_n(\mathcal{H})$:

$$R_n(\mathcal{H}) = \mathbb{E}\left[\sup_{X \in \mathcal{H}} \frac{1}{m} \sum_{(i,j)\in\Omega} \sigma_{i,j} (X_{i,j} - M_{i,j}) \right]$$

Let us introduce a zero-one matrix Σ such that $\Sigma_{i,j} = \sigma_{i,j}$ if $(i,j) \in \Omega$, and $\Sigma_{i,j} = 0$ otherwise. Hence, the above is equal to

$$\begin{aligned} \mathop{\mathbb{E}}_{\Omega,\sigma_{1:n}}\left[\sup_{X\in\mathcal{H}} \frac{1}{m} \langle \Sigma, X - M \rangle \right] &\le \mathbb{E}\left[\sup_{X\in\mathcal{H}} \frac{1}{m} \|\Sigma\|_2 \cdot \|X - M\|_\star \right] \\ &\le \frac{2C}{m} \mathbb{E}\left[\|\Sigma\|_2 \right] \end{aligned} \tag{31}$$

Above we use the fact that both X, M have nuclear norm bounded above by C. Hence, to finish the proof, we are left to deal with the expectation of $\|\Sigma\|_2$. We use the claim that

$$\mathbb{E}\left[\|\Sigma\|_2\right] \le \sqrt{\frac{m \log d}{d}}. \tag{32}$$

Plugging this claim back into equation (31), we have thus shown that

$$R_n(\mathcal{H}) \le 2C\sqrt{\frac{\log(d)}{md}}.$$

Showing equation (32) requires using some facts related to the spectrum of random matrices. We will consider covering this result at a later point in class.

3 Optimization and generalization in neural networks

In the previous section, we saw the concept of uniform convergence, and showed that with this concept, we can get pretty strong results on a variety of problems, including ℓ_2/ℓ_1-regularized linear functions, matrix completion, and two-layer neural networks. However, it turns out that moving beyond these settings with the techniques we've developed is quite challenging. A couple of key challenges are:

- In terms of the sample/learning complexity, how could we go beyond two layers? How does this complexity depend on the data distribution?
- How could we incorporate the inductive bias induced by the choice of specific optimization algorithms into the learning complexity?

The goal of this section is to tackle the above questions. In particular, we shall begin by establishing the folklore intuition that the learning complexity of an ML model scales with its number of parameters. This kind of folklore is especially intriguing in the context of deep networks, as these models have millions (and now billions) of parameters. However, this high complexity clearly does not explain/corroborate with the empirical results we typically see with supervised fine-tuning and in-context learning.

3.1 Neural tangent kernels (Lecture 10)

The performance of a machine learning model breaks down into two parts:

$$L(\hat{f}) - \inf_f L(f) = \underbrace{L(\hat{f}) - \inf_{f\in F} L(f)}_{\text{Estimation error}} + \underbrace{\inf_{f\in F} L(f) - \inf_f L(f)}_{\text{Approximation error}}$$

Both optimization and generalization results aim to reduce estimation error. Approximation error results, on the other hand, relate to the expressivity of a function class.

Example 3.1 (Linear models and kernel features)**.** *For linear methods (i.e., $f_w(x) = w^\top x$ for some parameter $w \in \mathbb{R}^d$ and input feature vector $x \in \mathbb{R}^d$), the estimation error is small, but the approximation error is large.*

Kernel methods represent one way to move beyond linear methods. In kernel methods, we replace $\langle x, w\rangle$ with $\langle \phi(x), w\rangle$, where $\phi(x)$ is an arbitrary feature map of x. Then, we may use the kernel features in a regression model. Suppose we are minimizing the mean squared error using a kernel method:

$$\ell(w) = \frac{1}{2n}\sum_{i=1}^{n}(\langle \phi(x_i), w\rangle - y_i)^2$$

Definition 3.2 (Kernels)**.** *A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive semi-definite kernel if and only if for every subset of inputs $\{x_1, x_2, \ldots, x_n\} \subseteq \mathcal{X}$, the matrix $K \in \mathbb{R}^{n\times n}$ defined by using the kernel map*

$$K_{i,j} = k(x_i, x_j),\ \text{for every } 1 \le i, j \le n$$

is a positive semi-definite matrix.[8]

[8]Recall that a symmetric matrix K is positive semi-definite if for every x, $x^\top Kx \ge 0$.

Example 3.3. *Linear kernels: $k(x, x') = \langle x, x' \rangle$. This follows from the fact that $K = XX^\top$ is positive semi-definite.*

Gaussian kernels: $k(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$. This is followed by the fact that either the sum or the product of two kernels (the element-wise product (Hadamard product) of two PSD matrices is still PSD) is still a kernel.

Polynomial kernels: $k(x, x') = (1 + \langle x, x' \rangle)^p$.[9]

An alternative definition of kernels is through feature maps. Recall that a feature map ϕ maps an input x to a feature embedding $\phi(x)$. Consider the kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ defined by

$$k(x, x') = \langle \phi(x), \phi(x') \rangle$$

This must be a kernel. To show this, we will follow the definition of a kernel by taking a vector α and let

$$\alpha^\top K \alpha = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \langle \phi(x_i), \phi(x_j) \rangle = A^\top A,$$

where $A = \sum_{i=1}^{n} \alpha_i \phi(x_i)$.

3.1.1 Motivation

It has been widely observed that gradient-based optimization algorithms often converge to small training errors on a complex neural network model. A widely believed explanation for this surprising phenomenon is that the neural net is over-parametrized. The neural tangent kernel represents one of the earliest attempts to mathematically formulate the theory of over-parameterized neural networks. The neural tangent kernel arises from the dynamics of the predictions applied to the training data.

First, we state the problem setup. Given a training dataset $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, let $w \in \mathbb{R}^N$ represent the parameters of a neural network. Let $f_w(x)$ denote the output of the network. We shall restrict our attention to the mean squared loss (for solving regression problems)

$$\ell(w) = \frac{1}{2} \sum_{i=1}^{n} (f_w(x_i) - y_i)^2$$

For the purpose of illustration, let us consider the gradient flow update. Let $w(t)$ denote the iterate configuration at time $t \geq 0$. Then, $w(t)$ evolves according to

$$\frac{dw(t)}{dt} = -\nabla \ell(w(t))$$

We now show that the learning dynamics are characterized by a positive semi-definite matrix $H(t) \in \mathbb{R}^{n \times n}$, defined as

$$H_{i,j}(t) = \left\langle \frac{\partial f_{w(t)}(x_i)}{\partial w_i}, \frac{\partial f_{w(t)}(x_j)}{\partial w_j} \right\rangle \tag{33}$$

Then, let the network outputs at time t be defined as

$$u(t) = [f_{w(t)}(x_1), f_{w(t)}(x_2), \ldots, f_{w(t)}(x_n)],$$

[9] Exercise: verify that both the polynomial kernel and the Gaussian kernel are indeed kernels

for any $t \geq 0$. Finally, let $y = [y_1, y_2, \ldots, y_n]$. Then, we will show that the network outputs will follow the dynamics as

$$\frac{du(t)}{dt} = -H(t)(u(t) - y)$$

Proof. Recall that the evolution of the network parameters follows an update as

$$\frac{dw(t)}{dt} = -\nabla \ell(w(t)) = -\sum_{i=1}^{n} (f_{w(t)}(x_i) - y_i) \frac{df_{w(t)}(x_i)}{dw}$$

Using chain rule and multi-variate calculus, the evolution of the network output satisfies

$$\frac{df_{w(t)}(x_i)}{dt} = \left\langle \frac{df_{w(t)}(x_i)}{dw}, \frac{dw(t)}{dt} \right\rangle$$

By applying the first step to the second step, we get

$$\begin{aligned}
\frac{df_{w(t)}(x_i)}{dt} &= \left\langle \frac{df_{w(t)}(x_i)}{dw}, -\sum_{j=1}^{n} (f_{w(t)}(x_j) - y_j) \frac{df_{w(t)}(x_j)}{dw} \right\rangle \\
&= -\sum_{j=1}^{n} (f_{w(t)}(x_j) - y_j) \underbrace{\left\langle \frac{df_{w(t)}(x_i)}{dw}, \frac{df_{w(t)}(x_j)}{dw} \right\rangle}_{i,j\text{-th entry of the } H(t) \text{ matrix}} \\
&= -H(t)(u(t) - y)
\end{aligned}$$

In summary, the learning dynamics are captured by a symmetric matrix $H(t)$. Now, how does the NTK arise? We will define an ultra-wide neural net whose width goes to infinity. In the limit, it can be shown that the matrix $H(0)$ remains constant during training, i.e., equal to $H(0)$. Moreover, under a random initialization of parameters, $H(0)$ converges to a deterministic kernel matrix $H^\star$ — the Neural Tangent Kernel (NTK).

3.1.2 Definition

Consider the mapping of a two-layer neural network

$$x \to \sum_{i=1}^{m} a_i \sigma(w_i^\top x), \tag{34}$$

where σ represents a nonlinear activation function (such as ReLU or sigmoid). Now consider the mapping rescaled by $\frac{1}{\sqrt{m}}$:

$$x \to \frac{1}{\sqrt{m}} \sum_{i=1}^{m} a_i \sigma(w_i^\top x)$$

We shall linearize the right-hand side around the initialization $w_i(0)$, for all $i = 1, 2, \ldots, m$; Essentially, performing Taylor's expansion to derive the following

$$\begin{aligned}
&x \to \frac{1}{\sqrt{m}} \sum_{i=1}^{m} a_i \left(\sigma((w_i(0))^\top x) + (w_i - w_i(0))^\top x \sigma'(w_i(0)^\top x) \right) \\
&= \frac{1}{\sqrt{m}} \sum_{i=1}^{m} a_i \left(\sigma((w_i(0))^\top x) - (w_i(0))^\top x \sigma'((w_i(0))^\top x) \right) + \frac{1}{\sqrt{m}} x^\top \left(\sum_{i=1}^{m} a_i w_i \sigma'((w_i(0))^\top x) \right)
\end{aligned}$$

Next, recall that the dynamics of predictions is governed by $H(t) \in \mathbb{R}^{n\times n}$ (see (33)). Recall that f is given in (34). Therefore, the i, j-th entry of the kernel matrix $H(t)$ is

$$H_{i,j}(0) = x_i^\top x_j \sum_{r=1}^{m} \left\langle \frac{a_r \sigma'((w_r(0)^\top x_i))}{\sqrt{m}}, \frac{a_r \sigma'((w_r(0))^\top x_j)}{\sqrt{m}} \right\rangle$$

Suppose we sample every $w_r(0)$ from a standard Gaussian distribution. In addition, suppose we sample a_r uniformly between $\{+1, -1\}$. Then, one can view the above as the average of m independent random variables. When m becomes very large, then by the law of large numbers, the average is close to its expectation. This gives rise to the Neural Tangent Kernel evaluated at x,x_j as follows:

$$H^\star_{i,j} = x_i^\top x_j \mathop{\mathbb{E}}_{w\sim\mathcal{N}(0,\mathrm{Id})} \left[\sigma'(w^\top x_i)\sigma'(w^\top x_j)\right]$$

3.1.3 Convergence analysis

Based on the NTK defined above, we now show that the neural network stays close to initialization long enough to get a small loss value. There are essentially two steps here:

- Step 1: For a sufficiently wide network, the randomly initialized neural net is close to the expectation – the NTK.
- Step 2: For a sufficiently wide network, the kernel matrix at time t remains close to the initialization.

For reference, much of the proof in this section is taken from the monograph by Arora et al., 2020.

$H(0)$ is close to $H^\star$ for sufficiently large m: Suppose the activation function (e.g., ReLU) is 1-Lipschitz continuous. Suppose for every input x_i, the Euclidean norm of x_i is less than 1, for any $i = 1, 2, \ldots, n$. For any $\epsilon > 0$, if $m \geq O(n^2 \log(\delta^{-1})/\epsilon^2)$ (recall n is the number of samples at the input), then with probability at least $1 - \delta$, we have that

$$\|H(0) - H^\star\|_F \leq \epsilon$$

To show that this is true, we will first show that every entry of $H(0)$ is close to $H^\star$ with probability $1 - O(\delta/n^2)$. First, we can see that every individual entry of $H(0)$ is bounded from above by 1:

$$\left| x_i^\top x_j \cdot \sigma'((w_r(0))^\top x_i)\sigma'((w_r(0))^\top x_j) \right| \leq 1$$

Now we shall apply Hoeffding's inequality (see (13)), and set the deviation as ϵ/n. Then, the failure probability becomes

$$2\exp\left(-\frac{2\epsilon^2 m}{n^2}\right)$$

When $m \geq O(\frac{n^2 \log(n\delta^{-1})}{\epsilon^2})$, the above probability can be reduced below $O(\delta/n^2)$. By taking a union bound over all the entries of $H(0)$, we have that the above will hold with probability $1 - \delta$.

Finally, we convert the entry-wise error bound to the operator norm bound:

$$\|H(0) - H^\star\|_F \leq \sqrt{n^2 \cdot \frac{\epsilon^2}{n^2}} = \epsilon$$

$H(t)$ **remains close to $H^\star$:** Suppose that $y_i = O(1)$ for all $i = 1, 2, \ldots, n$. Let $t > 0$, suppose that for all $0 < s < t$, $u_i(s) = O(1)$ for all $i = 1, 2, \ldots, n$. Then if $m \geq O(\frac{n^6 t^2 \log(n\delta^{-1})}{\epsilon^2})$, with probability $1 - \delta$,

$$\|H(t) - H^\star\|_F \leq \epsilon$$

There are two steps in the proof of this result:

- We first show that every weight vector remains close to the initialization if the width is large
- We then show that this implies the kernel matrix remains close to the NTK

Consider the movement of a single weight vector w_r:

$$\begin{aligned}
\|w_r(t) - w_r(0)\|_2 &= \left\| \int_0^t \frac{dw_r(\tau)}{d\tau} d\tau \right\|_2 \\
&= \left\| \int_0^t \frac{1}{\sqrt{m}} \sum_{i=0}^n (u_i(\tau) - y_i) a_r x_i \sigma'(w_r(\tau)^\top x_i) d\tau \right\|_2 \\
&\leq \int_0^t \left\| \frac{1}{\sqrt{m}} \sum_{i=0}^n (u_i(\tau) - y_i) a_r x_i \sigma'(w_r(\tau)^\top x_i) d\tau \right\|_2 \\
&\leq \frac{1}{\sqrt{m}} \sum_{i=0}^n \int_0^t \left\| (u_i(\tau) - y_i) a_r x_i \sigma'(w_r(\tau)^\top x_i) \right\|_2 d\tau \leq O\left(\frac{tn}{\sqrt{m}} \right),
\end{aligned}$$

where we use the fact that the inputs x_i, y_i and $a_r, u_i(\tau)$ are all bounded by some constants, and the derivative of σ is also bounded by a constant (e.g., sigmoid).

Consider the movement of a single entry of the kernel matrix

$$\begin{aligned}
H_{i,j}(t) - H_{i,j}(0) =& \frac{1}{m} \left| \sum_{r=1}^m \left(\sigma'(w_r(t)^\top x_i) \sigma'(w_r(t)^\top x_j) - \sigma'(w_r(0)^\top x_i) \sigma'(w_r(0)^\top x_j) \right) \right| \\
\leq& \frac{1}{m} \left| \sum_{r=1}^m \sigma'(w_r(t)^\top x_i) (\sigma'(w_r(t)^\top x_j) - \sigma'(w_r(0)^\top x_j)) \right| \\
&+ \frac{1}{m} \left| \sum_{r=1}^m \sigma'(w_r(0)^\top x_j) (\sigma'(w_r(t)^\top x_i - \sigma'(w_r(0)^\top x_i)) \right| \\
\lesssim& \frac{1}{m} \sum_{r=1}^m \max_{i=1}^n \left\| \sigma'(w_r(t)^\top x_i) - \sigma'(w_r(0)^\top x_i) \right\|_2 \\
\leq& O\left(\frac{tn}{\sqrt{m}} \right)
\end{aligned}$$

Taken together, we have shown that if m is large enough, then $H(t)$ will stay close to $H^\star$ throughout the entire gradient descent dynamic.

3.1.4 Implications

The dynamics of network outputs are governed by the following approximation using the NTK matrix

$$\frac{du(t)}{dt} \approx -H^\star(u(t) - y)$$

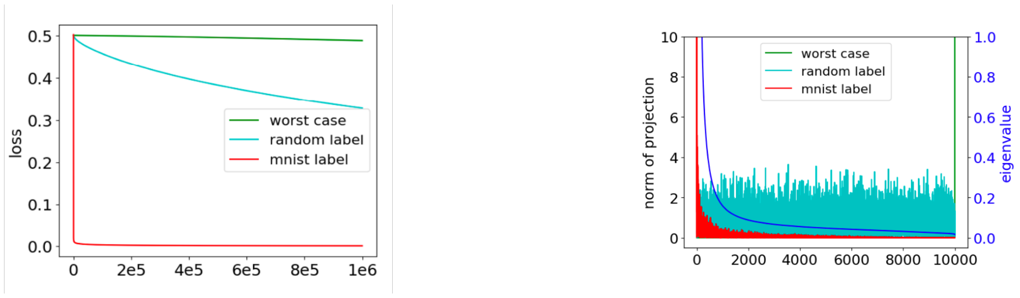


Figure 3: Illustration of the loss and the projected values onto the eigenspace.

We'll use this to describe two implications: i) Convergence to global minima; ii) Explaining why NTK converges faster using original labels than using random labels.

Notice that this is a linear dynamical system. Denote the eigen-decomposition of $H^\star$ as $\sum_{i=1}^n \lambda_i v_i v_i^\top$. Now we consider the dynamics of $u(t)$ on each eigenvector separately. Consider one eigenvector v_i for instance. We have

$$\frac{dv_i^\top u(t)}{dt} \approx -v_i^\top H^\star(u(t)-y) = -\lambda_i v_i^\top u(t) + \lambda_i v_i^\top y,$$

which is equivalent to

$$\frac{dv_i^\top (u(t)-y)}{dt} = -\lambda_i v_i^\top (u(t)-y)$$

One function that satisfies this differential equation is when

$$v_i^\top (u(t)-y) = \exp(-\lambda_i t)$$

Notice that $u(t)-y$ is the difference between predictions and training labels at time t.

We would like to show that the difference converges to zero when t is large. The above implies that each component of $u(t)-y$ projected to eigenvector v_i converges to zero exponentially fast at a rate of $\exp(-\lambda_i t)$.

Also notice that the eigenvectors $\{v_i\}_i$ form an orthonormal basis. Since we know that each component converges to zero, together they imply that $u(t)-y$ must also converge to zero. This shows that the outputs will eventually converge to the training labels, meaning the training loss will go to zero.

Next, we can see that each component goes to zero at different rates. For larger eigenvalues, the convergence is faster. Hence in order to have fast convergence, we would like the projections of the labels onto the top eigenvectors to be large.

For a set of labels, if they align with top eigenvectors, then gradient descent converges quickly. On the other hand, if the projections are uniform, or aligns with eigenvectors with small eigenvectors, then gradient descent converges with a slow rate. Below, we compare convergence rates of gradient descent between using original labels, random labels, and worst-case labels. Data: two classes of MNIST. Model: $f(a,w) = \frac{1}{\sqrt{m}} \sum_{i=1}^m a_i \sigma'(w_i^\top x)$.

P.S.

- Q&A form: https://forms.gle/SCjXUW6dkM8cDi4o9.
- Latex source code: https://github.com/hongyangsg/cs7140-advanced-ml.

References

Arora, R., Arora, S., Bruna, J., Cohen, N., Du, S., Ge, R., Gunasekar, S., Jin, C., Lee, J., Ma, T., Neyshabur, B., and Song, Z. (2020). *Theory of deep learning* (page 35).

Bach, F. (2024). *Learning theory from first principles*. MIT press (page 12).

Dhifallah, O. and Lu, Y. M. (2021). "Phase transitions in transfer learning for high-dimensional perceptrons". In: *Entropy* 23.4, p. 400 (page 8).

Duchi, J. (2019). "Lecture notes for statistics 311/electrical engineering 377". In: *URL: http://web.stanford.edu/class/stats311/lecture-notes.pdf* (page 9).

Garg, S., Tsipras, D., Liang, P. S., and Valiant, G. (2022). "What can transformers learn in-context? a case study of simple function classes". In: *Advances in neural information processing systems* 35, pp. 30583–30598 (page 7).

Kouw, W. M. and Loog, M. (2018). "An introduction to domain adaptation and transfer learning". In: *arXiv preprint arXiv:1812.11806* (page 7).

Ledoux, M. and Talagrand, M. (2013). *Probability in Banach Spaces: isoperimetry and processes*. Springer Science & Business Media (page 21).

Mitzenmacher, M. and Upfal, E. (2017). *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press (pages 14, 23).

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. (2017). "Exploring generalization in deep learning". In: *Advances in neural information processing systems* 30 (page 28).

Ruder, S. (2017). "An Overview of Multi-Task Learning in Deep Neural Networks". In: *arXiv preprint arXiv:1706.05098* (page 8).

Valiant, L. G. (1984). "A theory of the learnable". In: *Communications of the ACM* 27.11, pp. 1134–1142 (page 10).

Wainwright, M. J. (2019). *High-dimensional statistics: A non-asymptotic viewpoint*. Vol. 48. Cambridge university press (pages 4, 12).

Zhang, T. (2023). *Mathematical analysis of machine learning algorithms*. Cambridge University Press (page 12).