Mathematical Machine Learning

Yahya Saleh, Tizian Wenzel, Kamal Sharma

May 3, 2024

Contents

Introduction

Underlying the success of artificial intelligence are learning algorithms, i.e., algorithms that learn from data to perform a certain task. We start by two concrete examples of supervised learning algorithms. In the first example we consider the problem of approximating functions from pointwise evaluations using linear regression. In the second example we look at the task of classifying hand-written digits. In these two examples we identify and familiarize ourselves with the main components of learning algorithms; *datasets*, *a hypothesis class*, and *optimization algorithms*. We further identify important aspects of supervised learning algorithms, such as overfitting, and underfitting. Finally, we motivate in these examples, two problems at the forefront of research in mathematical machine learning, namely *the curse of dimensionality (CoD)* and *double/multiple descent phenomenon*.

I.1 Supervised Learning: Motivating Examples

In supervised learning tasks the dataset D is made up of two components, input variables $D_x = \{x_i\}_{i=1}^N$ and targets $D_y = \{y_i\}_{i=1}^N$. The dataset is assumed to be generated by an unknown function $f : \text{input} \to \text{target}$. The goal in a supervised learning task is to approximate the unknown function f pointwise, i.e., to find a function h such that $h(x) \approx f(x)$ for any x, whether it belongs to D_x or not. The target value can take finitely many values, e.g., target $\in \{0, 1, \dots, M\}$. In such a case, the supervised learning task is called a *classification task*. If the target can take infinitely many values, the learning task is called a *regression task*. Supervised learning problems are approached by first choosing a *hypothesis space* $\mathfrak{H}$, in which one looks for an approximation to the unknown function f. For example, if the data x is one-dimensional and the target takes values in $\mathbb{R}$ one can define the hypothesis class to be the set of all affine mappings, i.e.,

$$\mathfrak{H} = \big\{ f \mid f(x) = ax + b,\ a, b, \in \mathbb{R} \big\}. \tag{I.1}$$

Then, one can define a loss function l that measures how well a hypothesis function h approximates an unknown function f at a point x. Using the dataset D, the supervised learning problem can then be formulated as an optimization problem

$$\min_{h \in \mathfrak{H}} \frac{1}{N} \sum_{i=1}^{N} l(h(x_i), y_i). \tag{I.2}$$

While there are many alternatives to solve this optimization problem, by-far the most used algorithms are variants of the gradient-descent algorithm.

Let's look at some concrete examples.

Example I.1. [Regression] ☛ I.1 Let x be a random variable that takes values in the interval $[-1, 1]$. And assume we have access to a dataset $D = \{(x_i, y_i)_{i=1}^{200}\}$ generated by the unknown function

$$f(x) = x^2 \cos(5x) \exp(-x).$$

Assume that the dataset is corrupted by Gaussian noise. To learn a function h that approximates f, let your hypothesis class be the class of affine functions (I.1). Let the loss function be the absolute error, i.e.,

$$\begin{aligned} l(h(x_i), y_i) &= |h(x_i) - y_i| \\ &= |ax_i + b - y_i|. \end{aligned}$$

Use a gradient-descent-like algorithm to choose the best hypothesis h, i.e., the best scalars a and b.

Change your hypothesis class to the class of all polynomials up to order 20 and repeat the optimization process. Which class is better for optimization? Figure I.1 shows the outcome of such an experiment.

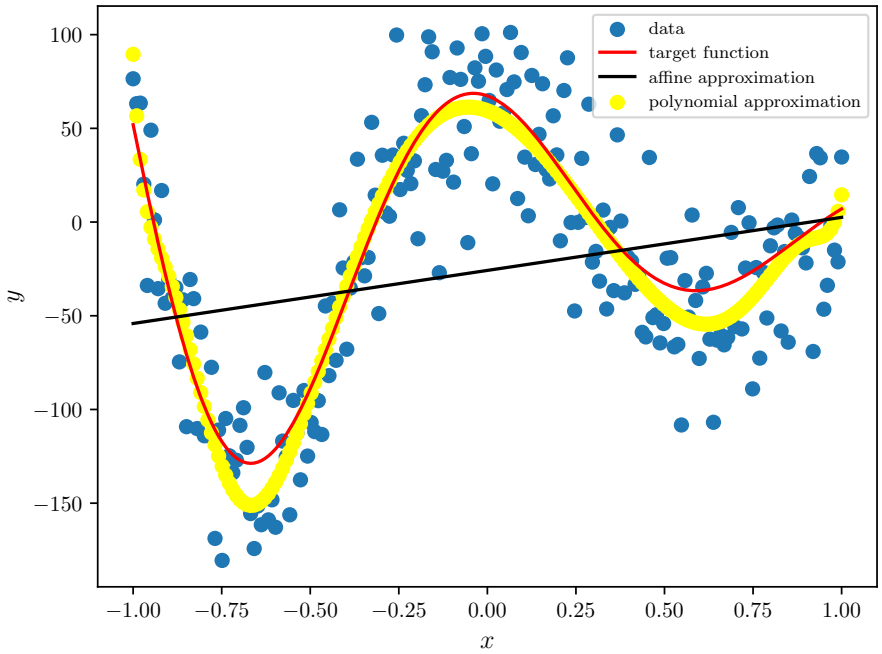

Figure I.1: A regression task; the goal is to fit noisy data (blue dots) assumed to be generated from a true function (solid red line). The data is fitted using an affine mapping (solid black line) and a polynomial mapping (solid yellow line).

Example I.2. [Classification] ☛ I.2 We consider a classification problem of hand-written digits. The input to the problem is an 8×8 image of a hand-written digit, and the output should be the predicted value of the digit. Formally, we consider x to be a random variable taking values in $[0, 16]^{8\times 8} \subset \mathbf{N}^{8\times 8}$, i.e., x is a random variables in a matrix representation, where each matrix element takes an integer value between 0 and 16. Here, the value of a certain matrix element represents its color, where 0 denotes black, and 16 denotes white. Let the target value y be a random variable taking values in the discrete set $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. To solve this supervised learning problem we consider as a hypothesis class the following multilayer perceptron:

$$\mathfrak{H} = \left\{\text{softmax } w_2 \left(\sigma(w_1 \cdot x + b_1)\right) + b_2; w_1 \in \mathbb{R}^{\text{dh},8}, b_1 \in \mathbb{R}^{\text{dh}}, w_2 \in \mathbb{R}^{10,\text{dh}}, b_2 \in \mathbb{R}^{10}\right\},$$

where dh is called the number of hidden units. In this class, the linear parameters are the weight matrices w_1, w_2 and the biases b_1, b_2. σ is a nonlinear non-learnable function, often referred to by the *activation function*. A common choice is the ReLU (Rectified Linear Unit) function

$$\text{ReLU}(x) = \max(0, x)$$

The softmax function (or layer) takes a set of real-valued input and transforms it into a probability distribution over multiple classes. In our example we have ten classes and the softmax is given by

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{10} e^{z_j}}.$$

Therefore, the output of the hypothesis function is a probability distribution over the 10 classes. Concretely, the output is 10-dimensional, where each entry denotes the probability of the input image to represent a certain digit.

For facilitating the implementation we represent the target value as a one-hot vector. For example, given a target value 4, we represent it as the vector $y = (0, 0, 0, 1, 0, 0, 0, 0, 0, 0)$. A suitable loss function for such problems is the categorical cross entropy function given by

$$l(h(x_i), y_i) = \sum_{c=0}^{9} y_i^c \log(h(x_i)^c),$$

where (x_i, y_i) is a specific training example. y_i^c refers to the c-th entry of the one-hot vector representation of the target.

Compute the training and test errors and study how they change when changing the number of hidden units or the number of layers.

Figure I.1 shows two hypotheses, one linear and one nonlinear, that we learned to fit the data in Example I.1. The figure depicts an interesting phenomenon; a certain hypothesis h can fit the data too accurately; notice for example in Figure I.1 that the polynomial-regression model fits badly local minima of the target function. In these regions, it is optimized to fit the noise. The outcome of such a result is that the polynomial-regression model will fail to generalize well in these regions, i.e., it will have large error on unseen data in these regions. This is called *overfitting*. On the other hand, the linear-regression model produces largely deviated results from the data everywhere, and would, hence, also generalizes badly to unseen data. This is called an *underfitting* phenomenon.

Think about the influence of the following factors on the underfitting and overfitting:

- Complexity of the model. For a polynomial-regression model this can be the degree of the polynomial.
- Size of the dataset. For example, would adding more data decrease or increase underfitting?

I.2 This Course

In this section we introduce and motivate some questions that guide the structure of this course.

I.2.1 Generalization Error and Minimization Principles

In Example I.2 and Example I.1 we saw that a model trained on a certain dataset D_{train} is expected to generalize well on *unseen* data. This is a crucial difference from standard interpolation paradigms. Formally, the training data is assumed to follow an unknown probability distribution P, i.e., $D \sim P$. It is desirable that the learned model not only performs well on D, but also on any other dataset D_{test} that also follows the distribution P. The problem is somehow ill-posed; *how can one fit a model to a training data D and expect it to perform well on unseen data D_{test}?*

One strategy to tackle this question is *via* induction principle. Formally, obtaining a hypothesis that minimizes the error over the whole distribution, also known as the *true risk*, is impossible. Instead, one can do the next best thing. This is formally done by deriving an upper bound of the true risk that includes, among other terms, the empirical risk, i.e., the loss on the training data. Other terms include the so-called *Rademacher complexity*, a term which describes how complex the hypothesis class is. The original task of minimizing the error over the whole probability distribution is then replaced by the task of minimizing its upperbound. Such induction strategies are called *Empirical Risk Minimization Principles*. These principles show that minimizing only the loss function of the training dataset is not enough to obtain good generalization. One needs to regularize such loss functions, i.e., to add some terms, whose minimization reduces the complexity of the hypothesis class. This is directly linked to our previous discussion on overfitting.

The topic will be discussed in more detail in chapter 2.

I.2.2 Hypothesis Classes and Approximation Capabilities

In Example I.1 and Example I.2, we have seen some examples of hypothesis classes, such as the class of all affine mappings, the class of polynomials up to a predefined degree, and the class of single-layer neural networks. Another major hypothesis class is Kernel methods. Some important questions here are as follows: given a certain learning problem, what class do we use? Are we guaranteed to find an optimal solution in the chosen class? Can we increase accuracy simply by increasing the complexity of our class?

In chapter 3 we will study two important classes of models, neural networks and kernel methods. We will survey some approximation properties of these classes and perform error analysis for approximating important functions, such as continuous functions, L^p- functions and Sobolev functions.

Moreover, we look with some detail at specific important application domains, such as image recognition, and natural language processing.

I.2.3 Curse of Dimensionality

Assume that we want to optimize a potentially multi-variate unknown function $f : \mathbb{R}^d \to \mathbb{R}$ using a dataset of points sampled from it. Let $d = 1$, i.e., assume for now that f is uni-variate and let our hypothesis class be a linear class. Let us denote by T the computational costs needed to achieve a certain accuracy ϵ. It turns out that T grows exponentially with respect to d. In other words, the computational costs required to achieve accuracy ϵ grow exponentially with the dimension of the problem. This is known as the *Curse of Dimensionality* phenomenon.

Formally, when approximating an unknown function f by a linear approximator $\tilde{f}$, *appriori* error estimates are often given by

$$\|f - \tilde{f}\| \leq c(d)\|f\|$$

where $\|.\|$ denotes some Sobolev norm of interest, and $c(d)$ is a constant that scales exponentially with the dimension of the problem. See, for example, error bounds for approximating Schwartz functions in the linear span of Hermite functions [1].

Example I.3. [CoD:Fitting] ☛ I.3 Consider fitting a dataset generated from the 1-dimensional target function

$$f(x) = \cos(2x)\exp(-x),$$

where you use the root-mean-squared error as a loss function and a polynomial-regression model as a hypothesis class. What is the degree of the polynomial necessary to achieve a training set error less than 10^{-5}. Similarly, study the same issue for fitting a dataset generated from the 2-dimensional target function

$$f(x, y) = \cos(2xy)\exp(-x).$$

In fact, this phenomenon is a major bottleneck for numerical methods to solve differential equations, such as finite differences, finite volumes or spectral methods.

Example I.4. [CoD:Solving Schrödinger Equation] I.4 Consider the following differential operator over $\mathbb{R}^d$

$$H = -\frac{1}{2}\big(\Delta + |x|^2 + \frac{1}{2}|x|^4\big),$$

where Δ denotes the Laplacian operator, and $|x| = \sqrt{\sum_{i=1}^{d} |x_i|^2}$. Its eigenvalue problem reads as follows: find all eigenpairs (E_n, ψ_n) that satisfy

$$H\psi_n = E_n\psi_n.$$

Assume we are interested only in the smallest eigenvalue E_0 and its corresponding eigenfunction ψ_0. Consider approximating ψ_0 in the linear span of truncated Hermite functions $(\gamma_n)_{n=0}^{\infty}$, i.e.,

$$\begin{aligned}\psi_0 &\approx \tilde{\psi}_0 \\ &= \sum_{n=0}^{N-1} c_n\gamma_n.\end{aligned}$$

Set $d = 1$. How many Hermite functions N are necessary to approximate E_0 to a relative accuracy of 10^{-1}? Repeat the same task for $d = 2$ and 3. What do you conclude? Answers are shown in Table I.1. For details on calculations refer to [2].

There is evidence, however, that neural networks are less prone to the CoD phenomenon. In other words, the computational scaling for using them to achieve a certain accuracy on a given task do not scale dramatically with the dimension of the problem. Characterizing such cases and providing

Table I.1: The size of a truncated Hermite basis N that is required to compute the smallest eigenvalue of the differential operator in Example I.4 in 1,2 and 3 dimensions to a relative absolute error $< 10^{-1}$.

d	1	2	3
N	3	45	286

rigorous understanding of this is a crucial point in modern mathematical machine learning. We will touch on this topic in chapter 4. Moreover, this topic will be of major interest to us when considering physics-informed neural networks.

I.2.4 Approximating Highly-Oscillatory Functions

The computational costs of approximation models, whether linear or nonlinear, seem to increase exponentially with an increase in the oscillation of a target function. This is a major bottleneck in some applications such as quantum dynamics. An important example here is approximating solutions to time-independent Schrödinger equations. Similar to Example I.4, the task here is to diagonalize a differential operator that describes a certain quantum system. We look here at a specific example, where H represents an operator, that describes vibrational motions inside a molecule. Given a linear numerical method to compute the eigenvalues, we look in Figure I.2 at the relative accuracy of the first 100 eigenvalues as a function of the truncation parameter N. It is clear that larger eigenvalues are harder to approximate. Morever, increasing the truncation parameter N does a worse job for improving the accuracy of higher eigenvalues than lower eigenvalues. This is partially because, larger eigenvalues correspond to highly-oscillatory functions. For details on the calculations see [3].

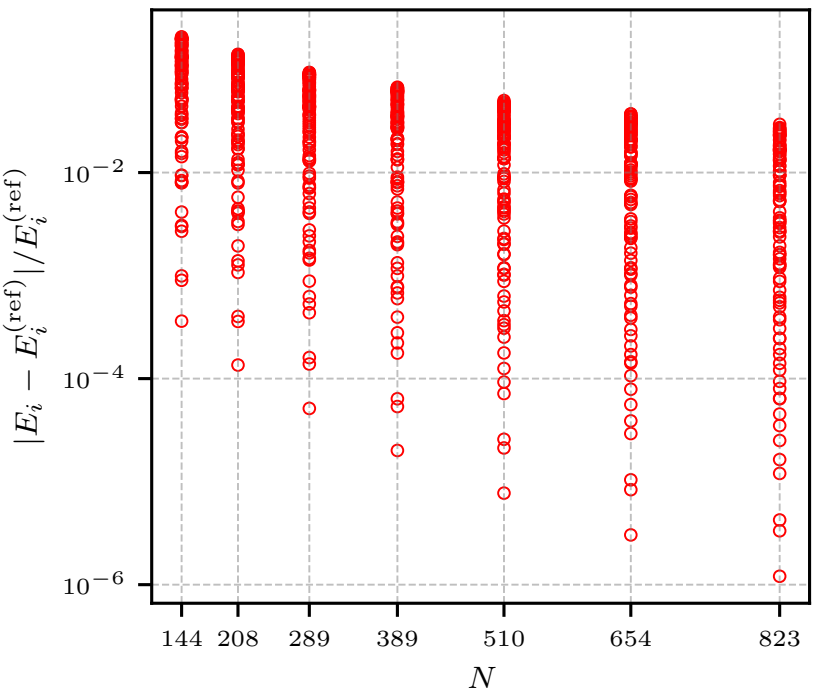

Figure I.2: The relative accuracy of the approximate first 100 eigenvalues of the vibrational Schrödinger equation for H_2S as a function of the truncation parameter N.

We will see in chapter 4 that neural-network approximation methods can significantly improve approximation capabilities for highly-oscillatory functions.

I.2.5 Validity of Occam's Razor and the Overparametrized Regime

Theoretical and empirical results have demonstrated in the past 40 years that Occam's Razor principle is valid when dealing with machine-learning problems: *Simple solutions are favored over unnecessarily complex ones.* Indeed, one can see in, e.g., Example I.1 that using a very high-order polynomial can produce very good results on the training data, but fail badly to produce sensible predictions on unseen data, resulting in an overfitting phenomenon. However, new evidence implies that very complex neural networks seem to generalize well on unseen data. In particular, many of the very successul machine-learning models that we use in our everyday life are heavily overparamterized, i.e., they have way more optimizable parameters than training data. Neverthess they generalize well on unseen data. Understanding the behavior of machine-learning models in this overparameterized regime is an important research direction in modern machine learning.

What is not covered in this course

The field of mathematical machine learning is huge and spans many standard mathematical areas, ranging from (geometric) measure theory to statistical learning theory, optimization theory and functional analysis. It is hence very challenging to cover all topics in a master course. Here is a list of topics that we do not cover in detail in this course and some references for self-study.

- Optimizers and their convergence are largely ignored in this course. We use standard first order optimizers in all our numerical examples and do not comment on their convergence properties.
- Bayesian learning models are largely marginalized.
- Unsupervised learning algorihtms such as clustering, and dimensionality reduction.

Wait! What is what?

Here is a list of questions that can help you check your understanding of key concepts in this chapter.

1. What are some examples of hypothesis classes? Which of them are linear or nonlinear approximation methods?
2. What loss functions do you know for regression and classification tasks? Can you think of other examples than the ones mentioned in this chapter?
3. For a fitting problem, can we get more accurate results by increasing the complexity of the hypothesis class?
4. How are the computational costs related to the dimension of the problem?

Statistical Learning Theory

In the previous chapter we have observed the phenomenon of overfitting; a model trained to minimize the empirical risk on a training dataset can still fail to generalize well over unseen dataset. A fundamental question in statistical learning theory is how to design hypothesis classes that do not overfit.

We will see in this chapter that restricting the complexity of the hypothesis classes can help reduce the overfitting. First, we start by looking at finite hypothesis classes and show that they do not overfit. This result will also motivate a notion of statistical learning, that of *probably approximately correct (PAC)* learning. However, finiteness of the hypothesis class is, indeed, a very restricting condition. We will discuss other measures of the complexity of a hypothesis class, such as the *Vapnik-Chervonenkis (VC) dimension* and the *Rademacher complexity*. At the end of this chapter, we will discuss what these theoretical results imply for the design of machine learning algorithms and link to the concept of *overparameterization*, which is a hot topic in the field of deep learning.

The reader is referred to [4] and [5] for more details.

We start this section by recalling some basic definitions of probability measure theory.

II.1 Probability Measure Theory

As we have seen in Introduction, training datasets are treated as random variables. This makes the following tools from probability theory essential.

Let $(\Omega, \mathcal{A}, \mathbb{P})$ be a probability measure space, where $\Omega \subseteq \mathbb{R}$. It is common to refer to any $A \in \mathcal{A}$ by an *event*. An event A s.t. $\mathbb{P}(A) = 1$ is said to happen *almost surley*.

Important probability measures are those induced by measurable transformations in the following way.

Definition II.1 (Push-forward Measure)**.** Given two measurable spaces $(\Omega_1, \mathcal{A}_1)$, $(\Omega_2, \mathcal{A}_2)$ and a measurable mapping $h : \Omega_1 \to \Omega_2$ the *push-forward measure* of a measure $\mathbb{P}$ on $(\Omega_2, \mathcal{A}_2)$ is

$$h_{\#}\mathbb{P}(A) := \mathbb{P}(h^{-1}(A)) \quad \forall A \in \mathcal{A}_2.$$

The push-forward measure is sometimes denoted by $\mathbb{P}h^{-1}$.

We look now at a special kind of measurable mappings and the measures they induce.

Definition II.2 (Real Random Variables and Distributions)**.** Let $(\Omega, \mathcal{A}, \mathbb{P})$ be a probability measure space.

(i) A measurable mapping $V : \Omega \to \mathbb{R}$ is called a *real random variable*.

(ii) The push-forward measure $\mathcal{P}_V := V_{\#}\mathbb{P}$ induced by a real random variable V is called the *(probability) distribution of* V.

A mapping $V : \Omega \to \mathbb{R}^n$ for $n > 1$ is usually referred to as a *random vector*. In our treatment we will refer to V by a random variable for any $n \geq 1$.

Definition II.3 (Expected Value)**.**

- The expected value of a random variable $X : \Omega \to \mathbb{R}$, denoted by $\mathbb{E}[X]$ is defined as:

$$\begin{aligned}\mathbb{E}[X] &:= \int_{\Omega} X(\omega)\, d\mathbb{P}(\omega) \\ &= \int_{\mathbb{R}} x\, d\mathcal{P}_X(x),\end{aligned}$$

where $\mathcal{P}_X$ is the pushforward-measure of X.

- Similarly, the expected value of a measurable mapping $g : \mathbb{R} \to \mathbb{R}$ as a function of the random variable X is given by

$$\begin{aligned}\mathbb{E}[g(X)] &:= \int_{\Omega} g(X)(\omega)\ d\mathbb{P}(\omega) \\ &= \int_{\mathbb{R}} g(x)\ d\mathcal{P}_X(x).\end{aligned}$$

  Sometimes, one writes $\mathbb{E}[g] = \mathbb{E}_{x\sim\mathcal{P}_x}[g]$ to highlight the measure on $\mathbb{R}$ against which one integrates.

Definition II.4 (Independent Events)**.** Two events A, B are said to be *independent* if

$$\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B).$$

Given an index set I, consider the family $A_i \in \mathcal{A}$ for $i \in I$. The family $(A_i)_{i\in I}$ of events is said to be *independent* if

$$\mathbb{P}(\cap_{j\in J} A_j) = \prod_{j\in J} \mathbb{P}(A_j) \quad \forall J \subset I.$$

The independence of events (i.e., sets) can be generalized to independence of families of sets.

Definition II.5 (Independence of Families of Sets)**.** Let I be an index set and consider $\mathcal{E}_i \subseteq \mathcal{A}$ for all $i \in I$. The family $(\mathcal{E}_i)_{i\in I}$ is called *independent* if, for any finite subset $J \in I$ and any choice of $E_j \in \mathcal{E}_j, j \in J$, one has

$$\mathbb{P}(\cap_{j\in J} E_j) = \prod_{j\in J} \mathbb{P}(E_j).$$

The following is an important family of random variables that one often encounters in machine learning and statistics.

Definition II.6 (Independent and Identically Distributed Random Variables)**.** Let I be an index set and $(V_i)_{i\in I}$ be a family of real random variables. Endow $\mathbb{R}$ with the Borel σ-algebra $\mathcal{B}$.

(i) The family $(V_i)_{i\in I}$ is said to be *identically distributed* if

$$\mathcal{P}_{V_i} = \mathcal{P}_{V_j} \quad \text{for all } i, j \in I.$$

(ii) The family $(V_i)_{i\in I}$ is said to be *independent* if the family of generated σ-algebras $\big(\sigma(V_i)\big)_{i\in I}$, where $\sigma(V_i) = V_i^{-1}(\mathcal{B})$ is independent.

A family of real random variables satisfying both conditions is said to be *independent and identically distributed (i.i.d.)*. In such a case set $\mathcal{P} = \mathcal{P}_{V_i}$.

II.2 A Formal Setting of Learning

We start by defining the standard setting of supervised learning. In our treatment, we will not use this very general setting. We will be imposing restrictions, for example, by considering only binary classification problems, or by using special loss functions. Nevertheless, it is useful to see the general setting first.

Let $z = (x, y)$ be a random variable where $x : \Omega \to \mathbb{X} \subseteq \mathbb{R}^n$ and $y : \Omega \to \mathbb{Y} \subset \mathbb{R}$. Denote by $\mathcal{P}$ the probability distribution of z and by $\mathcal{P}_x$ the marginal probability distribution corresponding to the random variable x. Further, we set $\mathbb{Z} := \mathbb{X} \times \mathbb{Y}$ [1]. We denote by $p(z)$ and $p(x)$ the probability density functions of z and x, respectively. These two densities are related by the formula $p(z) = p(x)p(y|x)$.

Given a hypothesis class $\mathfrak{H}$ of functions $h : \mathbb{X} \to \mathbb{Y}$ and a loss function $l : \mathbb{Y}^2 \to \mathbb{R}_{>0}$, the goal of a supervised-learning algorithm is to solve

$$\underbrace{R_{\mathcal{P}}(h) := \int_{\mathbb{Z}} l(y, h(x))\ d\mathcal{P}}_{\text{True risk}} \longrightarrow \min_{h\in\mathfrak{H}} \implies h^\star,$$

[1] Do not confuse the notation with the set of integers.

that is, to minimize the *true risk*. Note that the true risk is also called the *generalization error*. However, one only has access to a finite realiazation of the random variable z, i.e., to a training set $D = \{(x_i, y_i)_{i=1}^m\}$. A reasonable thing to do is, hence, to minimize a finite/empirical representation of the true risk, i.e., to solve

$$\underbrace{\hat{R}_{\mathcal{P}}(h) := \sum_{(x,y)\in D} \frac{1}{|D|} l(y, h(x))}_{\text{Empirical risk}} \longrightarrow \min_{h\in\mathfrak{H}} \implies \hat{h}^\star.$$

A learning algorithm that replaces the original task of minimizing the true risk by the task of minimizing the empirical risk is called an *empirical risk minimization (ERM) learner* or is said to be using the *ERM learning rule*. To highlight the dependence of the empirical risk on the training data, we sometimes write $\hat{R}_{\mathcal{P}}(h; D)$.

One of the main problems of statistical learning theories is to study the validity of approximating h^* by $\hat{h}^*$. In particular, under what conditions on the hypothesis class does $\hat{h}^*$ have a small generalization error? In the next section, we will see that the finiteness of the hypothesis class is a sufficient condition to this end. Is it necessary though?

II.3 Probably Approximately Correct Learning

We start this section by looking closely at the problem of overfitting and show that finite classes do not overfit. Our results to this end motivate a notion of statistical learning that we discuss.

For this section, we will adopt the setting defined in section II.2 and further assume that $\mathbb{Y} = \{0, 1\}$, i.e., we restrict ourselves to the binary classification case. Moreover, we assume that y is generated from x by the deterministic functional relation $y = f(x)$, where $f : \mathbb{X} \to \{0, 1\}$. Unless otherwise stated, we will set the loss function to be the 0-1 loss which we define as follows.

$$l(y, h(x)) := \begin{cases} 1: & h(x) \neq y \\ 0: & \text{otherwise.} \end{cases}$$

Note that in this case, the true and empirical risks simplify to

$$R_{\mathcal{P}}(h) = \mathcal{P}_x\big(\{x : h(x) \neq y\}\big)$$

$$\hat{R}_{\mathcal{P}}(h) = \frac{1}{m}|\{x : h(x) \neq y\}|$$

Moreover, we restrict ourselves to working under the *realizability assumption*, i.e., that there exists $h^* \in \mathfrak{H}$ s.t. $R_{\mathcal{P}}(h^*) = 0$. Note that this assumption implies that the empirical risk of the hypothesis obtained by the ERM rule is zero, i.e., $\hat{R}(\hat{h}^*) = 0$ with probability 1 over the choice of the training data. In other words, the realizability assumption implies that the ERM rule provides a hypothesis that is *consistent* on the training data. To see this, note that $R_{\mathcal{P}}(h^*) = 0$ implies that $\hat{R}_{\mathcal{P}}(h^*; D) = 0$ with probability 1 over the choice of a dataset D that is i.i.d. generated by $\mathcal{P}$. In turn, this implies that $\hat{R}_{\mathcal{P}}(\hat{h}^*; D) = 0$ with probability 1 over the choice of D since $\hat{R}_{\mathcal{P}}(\hat{h}^*; D) \leq \hat{R}_{\mathcal{P}}(h^*; D)$ by the definition of the ERM rule.

II.3.1 Finite Hypothesis Classes do not Overfit

The goal in this section is to show that we won't encounter an overfitting problem if the hypothesis class has finitely many elements. We will deal with the overfitting problem in an approximate manner, i.e., we will say that a hypothesis h does not overfit if $R_{\mathcal{P}}(h) \leq \epsilon$ for some small $\epsilon > 0$.

In what follows, given a dataset D, we denote by $D|_x$ the input of the dataset, i.e., $D|_x = \{x_i : i = 1, \ldots, m\}$. Note that since each $x \in D|_x$ is a random variable with a distribution $\mathcal{P}_x$, the dataset $D|_x$ has distribution $\mathcal{P}_x^m$ over $\mathbb{X}^m$.

Lemma II.1. *Under the realizability assumption and for accuracy $\epsilon > 0$ it holds that*

$$\mathcal{P}_x^m\big(\{D|_x : R_{\mathcal{P}}(\hat{h}^*) > \epsilon\}\big) \leq |\mathfrak{H}|e^{-\epsilon m}.$$

In words; the probability of sampling m training data points and obtaining a learner $\hat{h}^*$ by the ERM rule that does not generalize well is upperbounded. Note that this upperbound is finite if the cardinality of $\mathfrak{H}$ is finite. Also note that

$$\mathcal{P}_x^m\big(\{D|_x : R_{\mathcal{P}}(\hat{h}^*) > \epsilon\} = \mathbb{P}_{D|_x \sim \mathcal{P}_x^m}\big(\{R_{\mathcal{P}}(\hat{h}^*) > \epsilon\}\big).$$

Proof. We start by defining the set $\mathfrak{H}_b$ of bad hypotheses, i.e., the set of all hypotheses that lead to a generalization error $> \epsilon$,

$$\mathfrak{H}_b := \{h \in \mathfrak{H} \text{ s.t. } \mathbb{R}_{\mathcal{P}}(h) > \epsilon\}.$$

Next, we define the set of misleading training data, i.e., the set of all training datasets of cardinality m, on which there is at least one hypothesis that produces zero training error and a generalization error $> \epsilon$,

$$M := \{D|_x : \ |D|_x| = m, \text{ and s.t. there exists } h \in \mathfrak{H}_b : \ \hat{R}_{\mathcal{P}}(h; D) = 0\}.$$

Note that the realizability assumption implies that $\hat{R}_{\mathcal{P}}(\hat{h}^*) = 0$ as discussed before. This in turn implies that $\{D|_x : R_{\mathcal{P}}(\hat{h}^*) > \epsilon\} \subseteq M$. It thus follows that

$$\begin{aligned}
\mathcal{P}_x^m\big(\{D|_x : R_{\mathcal{P}}(\hat{h}^*) > \epsilon\}\big) &\leq \mathcal{P}_x^m(M) \\
&\leq \sum_{h \in \mathfrak{H}_b} \mathcal{P}_x^m\big(\{D|_x : \hat{R}_{\mathcal{P}}(h; D) = 0\}\big) \\
&= \sum_{h \in \mathfrak{H}_b} \prod_{i=1}^{m} \mathcal{P}_x\big(\{x : h(x) = y\}\big) \\
&= \sum_{h \in \mathfrak{H}_b} \prod_{i=1}^{m} (1 - R_{\mathcal{P}}(h)) \quad \text{(definition of true risk)} \\
&\leq \sum_{h \in \mathfrak{H}_b} (1 - \epsilon)^m \quad \text{(since } h \in \mathfrak{H}_b\text{)} \\
&\leq |\mathfrak{H}_b|(1 - \epsilon)^m \\
&\leq |\mathfrak{H}|(1 - \epsilon)^m \\
&\leq |\mathfrak{H}|e^{-\epsilon m}.
\end{aligned}$$

□

The above lemma shows that the probability of overfitting is exponentially small with the size of the training data. It also provides us with a way to find the minimal amount of training data that guarantees a small generalization error.

Corollary II.1. *Let $\mathfrak{H}$ be a finite hypothesis class. Let $\delta \in (0, 1)$ be a confidence parameter and $\epsilon \in (0, 1)$ be an accuracy parameter. Let m be an integer that satisfies*

$$m \geq \frac{1}{\epsilon} \log(|\mathfrak{H}|/\delta).$$

Under the realizability assumption it holds that

$$\mathbb{P}_{D|_x \sim \mathcal{P}_x^m}\big(\{R_{\mathcal{P}}(\hat{h}^*) > \epsilon\}\big) \leq \delta.$$

In other words,

$$R_{\mathcal{P}}(\hat{h}^*) \leq \epsilon$$

holds with probability of at least $1 - \delta$ over the choice of the training data D.

Note that this result holds for any labeling function f and any distribution $\mathcal{P}_x$.

Proof. Follows straight-forwardly from the previous lemma. □

The results we proved so far show that finite hypothesis classes do not overfit. Here, overfitting is defined in an *approximate sense* controlled by parameter ϵ. The results guarantee that the ERM rule provides a hypothesis that generalizes well in a *probabilistic sense*. The probability here is controlled by a parameter δ. This motivates the following definition.

Definition II.7. A hypothesis class $\mathfrak{H}$ is PAC learnable if there exist a function $m_{\mathfrak{H}} : (0, 1)^2 \to \mathbb{N}$ and a learning algorithm with the following properties

- for every $(\epsilon, \delta) \in (0, 1)^2$
- for every distribution $\mathcal{P}_x$ over $\mathbb{X}$
- for every labeling function $f : \mathbb{X} \to \{0, 1\}$

s.t. if the realizability assumption holds, when running the learning algorithm on $m \geq m_{\mathfrak{H}}(\delta, \epsilon)$ i.i.d. samples generated by $\mathcal{P}_x$ and labeled by f, the algorithm returns a hypothesis h such that

$$R_{\mathcal{P}}(h) \leq \epsilon$$

with probability of at least $1 - \delta$ over the choice of the samples.

Definition II.8 (samples complexity)**.** The sample complexity of leaning a hypothesis class $\mathfrak{H}$ is the minimal integer that satisfies the requirement of PAC learnability with accuracy ϵ and confidence δ.

With these definitions we can restate our corollary in the following manner.

Corollary II.2. *Every finite hypothesis calss is PAC learnable with sample complexity*

$$m \leq \lceil \frac{1}{\epsilon} \log(|\mathfrak{H}|/\delta) \rceil$$

One may wonder whether the finiteness of the hypothesis class is a necessary condition for PAC learnability. We will see now that this is indeed not the case. For example, we consider here the class of threshold functions.

Definition II.9 (Class of Threshold Functions)**.** $\mathfrak{H} := \{h_a : \mathbb{R} \to \{0, 1\}, h_a(x) = \mathbf{1}_{x<a},\ a \in \mathbb{R}\}$

In words, this class consists of all functions that assign 1 to all inputs that are smaller than a threshold a and 0 otherwise. The class of threshold functions has infinite cardinality. However, we will see that this class is PAC learnable.

Lemma II.2. *The class of threshold functions $\mathfrak{H}$ is PAC-learnable using the ERM rule with sample complexity of $m_{\mathfrak{H}} \leq \lceil \frac{1}{\epsilon} \log(2/\delta) \rceil$.*

Proof. It suffices to show that

$$\mathbb{P}_{D|_x \sim \mathcal{P}_x^m}\big(\{R_{\mathcal{P}}(\hat{h}^*) > \epsilon\}\big) \leq 2\exp(-\epsilon m).$$

To prove this note that we work under the realizability assumption. Hence, it is possible to find a hypothesis $\hat{h}^* = h_{\hat{a}^*}$ that is consistent with the training data.

Let a^* be the threshold of the labeling function f. Let $a_r^* > a^*$ be such that $\mathbb{P}_{x \sim \mathcal{P}_x}\big(x \in (a^*, \hat{a}_r^*)\big) = \epsilon$. Similarly, let $a_l^* < a^*$ be such that $\mathbb{P}_{x \sim \mathcal{P}_x}\big(x \in (\hat{a}_l^*, a^*)\big) = \epsilon$. Notice that picking datapoints that are outside the interval (a_l^*, a^*) or outside (a^*, a_r^*) is a sufficient condition for picking training datasets that lead to a bad hypothesis. Formally

$$\begin{aligned}\mathbb{P}_{D|_x \sim \mathcal{P}_x^m}\big(\{R_{\mathcal{P}}(\hat{h}^*) > \epsilon\}\big) &\leq \mathbb{P}_{D|_x \sim \mathcal{P}_x^m}\big(\{\forall x \in D|_x, x \notin (a_l^*, a^*) \vee x \notin (a^*, a_r^*)\}\big) \\ &\leq \underbrace{\mathbb{P}_{D|_x \sim \mathcal{P}_x^m}\big(\{\forall x \in D|_x, x \notin (a_l^*, a^*)\}\big)}_{\text{term1}} + \mathbb{P}_{D|_x \sim \mathcal{P}_x^m}\big(\{\forall x \in D|_x, x \notin (a^*, a_r^*)\}\big).\end{aligned}$$

Note that

$$\begin{aligned}\text{term 1} &\leq \mathbb{P}_{D|_x \sim \mathcal{P}_x^m}\big(\{x_1 \notin (a_l^*, a^*) \wedge x_2 \notin (a_l^*, a^*) \wedge \cdots \wedge x_m \notin (a_l^*, a^*)\}\big) \\ &= \prod_{i=1}^{m} \mathbb{P}_{x \sim \mathcal{P}_x}\big(x \notin (a_l^*, a^*)\big) \\ &= (1 - \epsilon)^m.\end{aligned}$$

The claim follows by doing the same inequality for the second term and using $(1-\epsilon)^m \leq \exp(-\epsilon m)$. □

In summary, our results show that finiteness of the hypothesis class is not a good condition for characterizing PAC learnability, since infinite classes can be PAC learnable. We thus move in the next section to introduce a better measure of the complexity of a hypothesis class.

II.3.2 Vapnik-Chervonenkis Dimension

The Vapnik-Chervonenkis (VC) dimension is a measure of the complexity of a hypothesis class. The intuition behind this measure is to define the richness of a class with respect to a given dataset. For example, a class of threshold functions is infinite dimensional. However, given a dataset D of only one point $x \in \mathbb{R}$, the whole class of threshold functions can give this datapoint only two different values, either 0 if the hypothesis has a threshold lying to the left of the datapoint, or 1 otherwise. Now consider a dataset of two points. The class of threshold functions can label these two points in three different ways, (0,0), (1,0) or (1,1). However, the hypothesis class of threshold functions cannot represent the labeling (0,1). Hence, the class of threshold functions cannot exhaust all the possible labeling of a dataset of two points. This is indeed the motivation behind the following definitions.

Definition II.10 (Restriction of $\mathfrak{H}$ to C)**.** Let $\mathfrak{H}$ be a class of functions from $\mathbb{X}$ to $\{0,1\}$ and let $C = \{x_1, \dots, x_m\} \subset \mathbb{X}$. The restriction of $\mathfrak{H}$ to C is the set of functions from C to $\{0,1\}$ that can be derived from $\mathfrak{H}$. That is,

$$\mathfrak{H}_C := \{(h(x_1), \dots, h(x_m)) : h \in \mathfrak{H}\}.$$

If $\mathfrak{H}_C$ is the set of all functions from C to $\{0,1\}$ we say that $\mathfrak{H}$ shatters C. Formally we write as follows.

Definition II.11 (Shattering)**.** A hypothesis class $\mathfrak{H}$ shatters a finite set $C \subset \mathbb{X}$ if $|\mathfrak{H}_C| = 2^{|C|}$.

According to this definition we observe that the class of threshold functions shatters a set of one point but does not shatter a set of two points, because it does not exhaust all of its possible labelings.

The following definition quantifies the number of different labelings that a hypothesis class can provide for a dataset of fixed size.

Definition II.12 (Growth Function)**.** Let $\mathfrak{H}$ be a hypothesis class. The growth function of $\mathfrak{H}$, denoted by $\tau_{\mathfrak{H}} : \mathbb{N} \to \mathbb{N}$ is defined as

$$\tau_{\mathfrak{H}}(m) := \max_{C \subset \mathbb{X} : |C| = m} |\mathfrak{H}_c|$$

In words; $\tau_{\mathfrak{H}}(m)$ is the maximum number of different functions from a set C of size m to $\{0,1\}$ that can be obtained by restricting $\mathfrak{H}$ to C.

It turns out that the growth function is a good quantity to measure the complexity of a hypothesis class, since a bounded growth function implies PAC learnability.

Theorem II.1. *Let $\delta \in (0,1)$ be a confidence parameter and $\epsilon \in (0,1)$ be an accuracy parameter. Let m be an integer that satisfies*

$$\epsilon \geq \frac{1}{m} \log(|\tau_{\mathfrak{H}}(2m)|/\delta).$$

Under the realizability assumption it holds that

$$\mathbb{P}_{D|_x \sim \mathcal{P}_x^m}\big(\{R_{\mathcal{P}}(\hat{h}^*) > \epsilon\}\big) \leq \delta.$$

Note that this result is very similar to Lemma II.1. The main difference is that the growth function is used instead of the cardinality of the hypothesis class.

Proof. To prove the result it suffices to show that

$$\mathcal{P}_x^m\big(\{D|_x : R_{\mathcal{P}}(\hat{h}^*) > \epsilon\}\big) \leq 2\tau_{\mathfrak{H}}(2m)2^{-\epsilon m/2}.$$

Given a hypothesis function h we define the quantity $M(h; D)$ as the number of errors that h makes on the dataset D. Since we are using the 0-1 loss we note that $\hat{R}_{\mathcal{P}}(h; D) = M(h; D)/m$. We define the event B as the event that a hypothesis is consistent with the training data but does not generalize well, i.e.,

$$B : \exists h \in \mathfrak{H} : M(h; D) = 0 \wedge R_{\mathcal{P}}(h) > \epsilon.$$

To have an empirical estimate of the true risk we assume that we have access to a test set D' of size m that is i.i.d. generated by $\mathcal{P}_x$. We now define the event B'

$$B' : \exists h \in \mathfrak{H} : M(h; D) = 0 \wedge M(h; D') > \epsilon m/2.$$

Now consider the following experiment. For each element $x \in D|_x$ and $x' \in D'|_x$ we flip a fair coin. If the coin lands heads we swap x with x'. Otherwise we do not swap. We denote the resulting datasets by T and T'. We now define the event B'' as follows

$$B'' : \exists h \in \mathfrak{H} : M(h;T) = 0 \wedge M(h;T') > \epsilon m/2.$$

We now claim that

$$\mathbb{P}(B) \underbrace{\leq}_{*_1} 2\mathbb{P}(B') \underbrace{=}_{*_2} 2\mathbb{P}(B'') \underbrace{\leq}_{*_3} 2\ \tau_{\mathfrak{H}}(2m)2^{-\epsilon m/2}.$$

Proving this claim would complete the proof. We now prove the inequality1 $*_1$,

Note that

$$\begin{aligned}\mathbb{P}(B') &\geq \mathbb{P}(B' \wedge B) \\ &= \mathbb{P}(B'|B)\mathbb{P}(B)\end{aligned}$$

The claim now follows from the fact that $\mathbb{P}(B'|B) \geq 1/2$ if $m \geq 8/\epsilon$. While we will not formally prove this result, here is the intuition behind it. Assume that the event B has occurred. This means that there is a hypothesis h that is consistent with the training data but does not generalize well. Now consider $M(h;D')$ as a random variable and note that $\mathbb{E}_{D'|_x \sim \mathcal{P}_x^m} M(h;D') = R_{\mathcal{P}}(h)m$. Since $R_{\mathcal{P}}(h) > \epsilon$ we have that $\mathbb{E}_{D'|_x \sim \mathcal{P}_x^m} M(h;D') > \epsilon m/2$. Hence, the probability of B' is equivalent to the probability of the random variable $M(h;D')$ being larger than a lower bound to its expected value. The desired result is based on the use of a measure-concentration inequality, i.e., an inequality that study the convergence of an empirical average of a random variable to its true expected value.

To see the inequality $*_2$ note that the samples D, D' are i.i.d. generated by $\mathcal{P}_x$. Since T, T' were generated by equally likely permutations, we have that T, T' are also i.i.d. generated by $\mathcal{P}_x$. Hence, the probability of B'' is the same as the probability of B'.

The inequality $*_3$ follows from the application of the law of total expectation which states that for two random variables X, Y it holds that $\mathbb{E}(X) = \mathbb{E}(\mathbb{E}(X|Y))$. Note that the probability of some event A can be written as the expected value of the indicator function of A. It hence follows that

$$\mathbb{P}_{D,D' \sim \mathcal{P}_x^{2m}}(B'') = \mathbb{E}_{D,D' \sim \mathcal{P}_x^{2m}} \mathbb{P}(B''|D,D').$$

Now let $\mathfrak{H}'$ be a hypothesis class containing one representative for each different labeling of $\mathfrak{H}$ of D, D'. We have that

$$\begin{aligned}\mathbb{P}(B''|D,D') &= \mathbb{P}(\exists h \in \mathfrak{H} : \underbrace{M(h;T) = 0 \wedge M(h;T') > \epsilon m/2}_{:=b(h)} |D,D') \\ &= \mathbb{P}(\exists h \in \mathfrak{H}' : b(h)|D,D') \\ &\leq \sum_{h \in \mathfrak{H}'} \mathbb{P}(b(h)|D,D') \\ &\leq |\mathfrak{H}'|2^{-\epsilon m/2} \quad \text{assume } \mathbb{P}(b(h)|D,D') \leq 2^{-\epsilon m/2} \\ &\leq \tau_{\mathfrak{H}}(2m)2^{-\epsilon m/2}.\end{aligned}$$

We prove the assumption $\mathbb{P}_{D,D' \sim \mathcal{P}_x^{2m}}(b(h)|D,D') \leq 2^{-\epsilon m/2}$ by considering three different cases. For a fixed h, we note that D, D' already occurred. Hence, the only randomness is coming from the random permutation process to construct T, T'.

Assume for the first case that there exists an index i s.t. h makes a mistake on both $x_i \in D|_x$ and $x_i' \in D'|_x$. In this case, it is not possible to have $M(h;T) = 0$. Hence, the event $b(h)$ occurs with probability 0.

For the next two cases define r to be the number of pairs $(x_i, x_i'), x_i \in D|_x, x_i' \in D'|_x$ where h is wrong on exactly one between the two, i.e., either on x_i, or on x_i'.

Consider for the second case $r < m\epsilon/2$. Even if we get lucky enough and all elements on which h is correct end up in T, we will not have enough datapoints to satisfy the condition $M(h;T') > \epsilon m/2$. Hence, the event $b(h)$ occurs with probability 0.

For the third case consider $r \geq m\epsilon/2$. In this case, we need that all the coin flips end up in the correct way to have zero mistakes in T and all mistakes in T'. Since the coin flips are independent and fair, we have that

$$\mathbb{P}(b(h)|D,D') = 2^{-r} \leq 2^{-m\epsilon/2}.$$

□

Due to the combinatorial nature of the growth function, its asymptotic behavior is not obvious. Getting hold of it would provide a better bound in Theorem II.1. We will now introduce the more intuitive quantity of the VC-dimension, which will enable us to calculate how many datapoints we need to get PAC learnability with a hypothesis class of infinite cardinality.

Definition II.13 (VC (Vapnik-Chervonenkis) dimension)**.** The VC dimension of a hypothesis class $\mathfrak{H}$, denoted VC-dim($\mathfrak{H}$) is the cardinality of the largest set C that can be shattered by $\mathfrak{H}$.

Example II.5. To show that a finite hypothesis class has a VC-dimension d, it suffices to show that there exists a set of size d that can be shattered by the hypothesis class and that there exists no set of size $d+1$ that can be shattered by the hypothesis class.

- The VC dimension of threshold functions is 1.
- Consider the following hypothesis class

$$\mathfrak{H} = \{h : h(x) = \mathbf{1}_{x\in(a,b)}, a < b, a, b \in \mathbb{R}\}.$$

  The VC dimension of this class is 2.

The previous examples may convey the idea that the VC dimension of a certain class equals the number of free parameters that the class has. This is not always the case. For example, the VC dimension of the class

$$\mathfrak{H} = \{h(x) = \sin(\theta x) : \theta \in \mathbb{R}\}$$

is infinite.

The following lemma allows us to replace the growth function by the VC-dimension in Theorem II.1.

Lemma II.3 (Sauer's lemma)**.** *Let $\mathfrak{H}$ be a hypothesis class. If*

- *$VC-dim(\mathfrak{H}) < \infty$ then $\tau_{\mathfrak{H}} = O(m^d)$ for all $m \in \mathbb{N}$.*
- *$VC-dim(\mathfrak{H}) = \infty$ then $\tau_{\mathfrak{H}} = 2^m$ for all $m \in \mathbb{N}$.*

Corollary II.3. *Let $\delta \in (0,1)$ be a confidence parameter and $\epsilon \in (0,1)$ be an accuracy parameter. Assume VC-dim($\mathfrak{H}$) $= d < \infty$. Let m be an integer that satisfies*

$$\epsilon \geq C\frac{d\log m + \log(1/\delta)}{m}.$$

for some constant C. Under the realizability assumption it holds that

$$\mathbb{P}_{D|_x\sim\mathcal{P}_x^m}\left(\{R_{\mathcal{P}}(\hat{h}^*) > \epsilon\}\right) \leq \delta.$$

Proof. The proof follows from applying Sauer's lemma to Theorem II.1. □

So far we have seen that the VC-dimension is a good measure of the complexity of a hypothesis class in the sense that having a finite VC-dimension implies PAC learnability. Now we will see that an infinite VC-dimension implies that the hypothesis class is not PAC learnable. To see this, we use the following theorem that states that a VC-dimension that is larger than twice the size of the training data implies that we might end up with a hypothesis that does not generalize well. Intuitively, the theorem states that having a high VC-dimension implies that the hypothesis class is too rich that it can fit random labels, so it is not capable of learning.

Lemma II.4. *Let $\mathfrak{H}$ be a hypothesis class of functions from $\mathbb{X}$ to $\{0,1\}$. Let m be a training set size. Assume that VC-dim($\mathfrak{H}$) $\geq 2m$. Then, for any learning algorithm A there exist a distribution $\mathcal{P}$ over $\mathbb{X}\times\{0,1\}$ s.t. $\hat{R}_{\mathcal{P}}(h^A) = 0$ but with probability of at least 1/7 over the choise of $D|_x \sim \mathcal{P}_x^m$ we have that*

$$R_{\mathcal{P}}(h^A) \geq 1/8.$$

Corollary II.4. *Let $\mathfrak{H}$ be a class of inifinite VC-dimension. Then, $\mathfrak{H}$ is not PAC learnable.*

Proof. Since VC-dim($\mathfrak{H}$) $= \infty$, for any training set of size m, there exists a shattered set of size $2m$. Hence, the result follows from the previous lemma. □

While we managed to show that the VC-dimension is a better measure of complexity than the finiteness of the hypothesis class, it is still not optimal. One of the major limitations of the VC-dimension is its restriction to binary classification. In the next section we will see a more general measure of the complexity of a hypothesis class, that is the Rademacher complexity.

II.4 Agnostic PAC Learning

So far we worked under the realizability assumption, i.e., we assumed that the labeling function f is a member of the hypothesis class $\mathfrak{H}$. We showed that this implies the existence of functions in $\mathfrak{H}$ that have zero empirical risk on the training data. The realizability assumption is, indeed, a very strong assumption and in many practical scenarios it does not hold. In this section we will relax this assumption and consider the more general case where the labeling function f is not necessarily a member of the hypothesis class $\mathfrak{H}$. This is known as the agnostic setting. We will generalize the PAC-learning framework to the agnostic setting.

In the previous section we always assumed that the target value y is generated form x by the functional relation f. From now on, we relax this assumption and assume that there is some randomness in the generation of y from x that is described by the conditional distribution $\mathcal{P}(y|x)$.

Generalizing PAC-learning to the agnostic setting is straightforward.

Definition II.14. A hypothesis class $\mathfrak{H}$ is agnostic PAC learnable if there exist a function $m_{\mathfrak{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm with the following properties

- for every $(\epsilon, \delta) \in (0,1)$
- for every distribution $\mathcal{P}$ over $\mathbb{X} \times \mathbb{Y}$

when running the learning algorithm on $m \geq m_{\mathfrak{H}}(\delta, \epsilon)$ i.i.d. samples generated by $\mathcal{P}$, the algorithm returns a hypothesis $\hat{h}^*$ such that

$$R_{\mathcal{P}}(\hat{h}^*) \leq \min_{h \in \mathfrak{H}} R_{\mathcal{P}}(h) + \epsilon$$

with probability of at least $1 - \delta$ over the choice of the samples.

We note that under the realizability assumption, agnostic PAC-learning reduces to PAC-learning. We now define a tool that will allow us to prove agnostic PAC learnability of a hypothesis class.

Definition II.15 (ϵ-representative sample)**.** A training set D is said to be $\epsilon-$representative (w.r.t. a domain $\mathbb{Z}$, a hypothesis class $\mathfrak{H}$, a loss l, and a distribution $\mathcal{P}$) if for every $h \in \mathfrak{H}$ it holds that

$$|R_{\mathcal{P}}(h) - \hat{R}_{\mathcal{P}}(h)| \leq \epsilon$$

We emphasize that this definition is uniform for $h \in \mathfrak{H}$.

Whenever a dataset D is $\epsilon/2$-representative, the ERM learner is guaranteed to return a good hypothesis (in the agnostic PAC sense).

Theorem II.2. *Assume that a training set D is $\epsilon/2$-representative. Then, for any*

$$\hat{h}^* \in argmin_{h \in \mathfrak{H}} \hat{R}_{\mathcal{P}}(h; D)$$

it holds that

$$R_{\mathcal{P}}(\hat{h}^*) \leq \min_{h \in \mathfrak{H}} R_{\mathcal{P}}(h) + \epsilon$$

Proof. For any h in $\mathfrak{H}$ we have that

$$\begin{aligned} R_{\mathcal{P}}(\hat{h}^*) &\leq \hat{R}_{\mathcal{P}}(\hat{h}^*) + \epsilon/2 \quad D \text{ is } \epsilon/2 - \text{representative} \\ &\leq \hat{R}_{\mathcal{P}}(h) + \epsilon/2 \quad \hat{h}^* \in \text{argmin}_{h \in \mathfrak{H}} \hat{R}_{\mathcal{P}}(h) \\ &\leq R_{\mathcal{P}}(h) + \epsilon \quad D \text{ is } \epsilon/2 - \text{representative}. \end{aligned}$$

The result holds in particular for the hypothesis $\hat{h}^*$. □

The previous theorem suggests that it is enough to prove uniform convergence of the empirical risk to the true risk to show agnostic PAC learnability, i.e., it is enough to show

$$\hat{R}_{\mathcal{P}}(h) \leq R_{\mathcal{P}}(h) + \epsilon/2$$

for all $h \in \mathfrak{H}$ with high probability. We apply this new knowledge to show that a finite hypothesis class is agnostic PAC-learnable.

II.4.1 Finite Hypothesis Classes Revisited

Previously, we showed that a finite hypothesis class is PAC learnable. It turns out that we can extend this to the agnostic setting. For this we rely on the following concentration-of-measure inequality.

Theorem II.3 (Hoeffding's inequality)**.** *Let $\theta_1, \dots, \theta_n$ be a sequence of i.i.d. random variables and assume that for all i $\mathbb{E}[\theta_i] = \mu$ and $\mathbb{P}[a \leq \theta_i \leq b] = 1$. Then, for any $\epsilon > 0$*

$$\mathbb{P}[|\frac{1}{m}\sum_{i=1}^{m}\theta_i - \mu| > \epsilon] \leq 2\exp(-2m\epsilon^2/(b-a)^2).$$

We are now ready to state our first result in agnostic PAC learning setting.

Theorem II.4. *Let $|\mathfrak{H}| < \infty$ then $\mathfrak{H}$ is agnostically PAC learnable with sample complexity*

$$m_{\mathfrak{H}}(\epsilon, \delta) \leq \lceil \frac{2\log(2|\mathfrak{H}|/\delta)}{\epsilon^2} \rceil$$

Proof. As previously discussed, it is sufficient here to show the uniform convergence property, i.e., to show that

$$\mathbb{P}_{D\sim\mathcal{P}^m}\big(\{D:\ \exists h \in \mathfrak{H}, |R_{\mathcal{P}}(h) - \hat{R}_{\mathcal{P}}(h)| > \epsilon\}\big) \leq \delta.$$

To this end we write

$$\begin{aligned}\mathbb{P}_{D\sim\mathcal{P}^m}\big(\{D:\ \exists h \in \mathfrak{H}, |R_{\mathcal{P}}(h) - \hat{R}_{\mathcal{P}}(h)| > \epsilon\}\big) &\leq \sum_{h\in\mathfrak{H}} \mathbb{P}_{D\sim\mathcal{P}^m}\big(\{D:\ |R_{\mathcal{P}}(h) - \hat{R}_{\mathcal{P}}(h)| > \epsilon\}\big) \\ &\leq |\mathfrak{H}| \cdot 2\exp(-2m\epsilon^2) \\ &\leq \delta,\end{aligned}$$

where we used Hoeffding's inequality after having noted that $R_{\mathcal{P}}(h) = \mathbb{E}_{D\sim\mathcal{P}^m}\hat{R}_{\mathcal{P}}(h)$. □

We now talk about a new measure of complexity of a hypothesis class and show agnostic PAC learnability in the new setting.

II.4.2 Rademacher Complexity

We found out in the last section that the VC-dimension is a good measure of the complexity of a hypothesis class. However, the VC-dimension is limited to binary classification problems. While there are ways to rectify this limitation, we will look into a more general measure of the complexity of a hypothesis class, mainly that of the Rademacher complexity.

We start by motivating the Rademacher complexity with a simple example.

Consider the binary classification task with the zero-one loss. Further, assume that the target values $y \in \{-1, 1\}$. Note that

$$\begin{aligned}\hat{R}_{\mathcal{P}}(h) &= \frac{1}{m}\sum_{i=1}^{m} l(y_i, h(x_i)) \\ &= \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}_{\{h(x_i)\neq y_i\}} \quad \text{since we are using the 0-1 loss} \\ &= \frac{1}{m}\sum_{i=1}^{m} \frac{1 - h(x_i)y_i}{2} \quad \text{since } y_i \in \{-1, 1\} \text{ for any } i \\ &= \frac{1}{2} - \frac{1}{2m}\sum_{i=1}^{m} h(x_i)y_i.\end{aligned}$$

Thus,

$$\frac{1}{2m}\sum_{i=0}^{m} h(x_i)y_i = 1 - 2\hat{R}_{\mathcal{P}}(h).$$

The quantity $\frac{1}{2m}\sum_{i=0}^{m} h(x_i)y_i$ is a good measure of how well the hypothesis h fits the data. Note also that minimizing the empirical risk is equivalent to maximizing this quantity. The Rademacher complexity generalizes this quantity to be a measure of how well h fits any dataset, not only

D. Formally, one replaces y_i by a random variable σ_i that takes values in $\{-1, 1\}$ with equal probability. The Rademacher complexity is defined to be

$$\mathrm{Rad}(\mathfrak{H}) := \mathbb{E}_\sigma\big[\sup_{h\in\mathfrak{H}} \frac{1}{m}\sum_{i=1}^{m} \sigma_i h(x_i)\big].$$

Intuitively, the Rademacher complexity measures how well a hypothesis class can fit random labels. Let us now compute the Rademacher complexity for two simple hypothesis classes. First, let the hypothesis class contain one element, i.e., $\mathfrak{H} = \{h\}$. We then compute

$$\begin{aligned}
\mathrm{Rad}(\mathfrak{H}) &= \mathbb{E}_\sigma\big[\frac{1}{m}\sum_{i=1}^{m} \sigma_i h(x_i)\big] \\
&= \frac{1}{m}\sum_{i=1}^{m} \mathbb{E}_\sigma[\sigma_i] h(x_i) \\
&= \frac{1}{m}\sum_{i=1}^{m} 0 h(x_i) \\
&= 0.
\end{aligned}$$

Second, we consider the case, where $\mathfrak{H}$ shatters the dataset D. In this case, we have that the empirical risk of the ERM hypothesis is zero. Hence, the Rademacher complexity is one.

We will now show that our definition of the Rademacher complexity extends in a meaningful way to regression problems.

Consider now a generic supervised learning problem. Given a certain dataset D we define its representativeness to be

$$\mathrm{Rep}(D) := \sup_{h\in\mathfrak{H}} (\hat{R}_\mathcal{P}(h; D) - R_\mathcal{P}(h)).$$

Clearly, one cannot compute this quantity since one cannot compute the true risk. However, one can approximate this quantity by considering a test dataset D' generated from the true distribution $\mathcal{P}$. We define the empirical representativeness of D to be

$$\hat{\mathrm{Rep}}(D) := \sup_{h\in\mathfrak{H}} (\hat{R}_\mathcal{P}(h; D) - \hat{R}_\mathcal{P}(h; D')).$$

Note that

$$\hat{\mathrm{Rep}}(D) = \sup_{h\in\mathfrak{H}} (\frac{1}{|D|}\sum_{(x_i,y_i)\in D} l(h(x_i), y_i) - \frac{1}{|D'|}\sum_{(x_i,y_i)\in D'} l(h(x_i), y_i'))$$

Now assume that $|D| = |D'| = m/2$ and let set

$$\sigma_i = \begin{cases} 1 & \text{if } (x_i, y_i) \in D \\ -1 & \text{if } (x_i, y_i) \in D' \end{cases}.$$

It follows that

$$\hat{\mathrm{Rep}}(D) = \sup_{h\in\mathfrak{H}} \frac{2}{m}\sum_{i=1}^{m} \sigma_i l(h(x_i), y_i).$$

The Rademacher complexity generalizes this idea by taking the expectation over all possible choices of σ_i.

We note that the Rademacher complexity of a hypothesis class depends on the choice of a loss function and training data set D. Hence, we introduce the following notation to make this dependence explicit.

$$l \circ \mathfrak{H} \circ D := \{\big(l(h(x_1), y_1), \ldots, l(h(x_m), y_m)\big) : h \in \mathfrak{H}\}$$

$$\mathfrak{F} := l \circ \mathfrak{H} := \{z \to l(h, z) : h \in \mathfrak{H}\}$$

We are now ready to state the formal definition of the Rademacher complexity.

Definition II.16 (Rademacher Complexity)**.** Let $\sigma = (\sigma_1, \ldots, \sigma_m) \in \{-1, 1\}^m$ be a vector of random variables such that $\mathbb{P}[\sigma_i = 1] = \mathbb{P}[\sigma_i = -1] = 1/2$. The Rademacher complexity of $l \circ \mathfrak{H} \circ D$ is defined as

$$\begin{aligned}\mathrm{Rad}(l \circ \mathfrak{H} \circ D) :&= \frac{1}{2}\mathbb{E}_\sigma\big[\hat{\mathrm{Rep}}(D)\big] \\ &= \mathbb{E}_\sigma\big[\sup_{f \in \mathfrak{F}} \frac{1}{m} \sum_{i=1}^{m} \sigma_i f(x_i)\big] \\ &= \mathbb{E}_\sigma\big[\sup_{h \in \mathfrak{H}} \frac{1}{m} \sum_{i=1}^{m} \sigma_i l(h(x_i), y_i)\big]\end{aligned}$$

The following result shows that a finite Rademacher complexity allows for agnostic PAC learning.

Theorem II.5. *Assume that for all $z = (x, y)$ and $h \in \mathfrak{H}$, $|l(h(x), y)| \leq c$. Then, for any $\delta > 0$ with probability of at least $1 - \delta$ over the choice of the sample D, for any $h \in \mathfrak{H}$ it holds that*

$$R_{\mathcal{P}}(h) \leq \hat{R}_{\mathcal{P}}(h) + 2Rad(l \circ \mathfrak{H} \circ D) + c\sqrt{\frac{2\log(4/\delta)}{m}} \tag{II.3}$$

and

$$\hat{R}_{\mathcal{P}}(h) \leq R_{\mathcal{P}}(h) + 2\mathbb{E}_{D' \sim \mathcal{P}^m}\big[Rad(l \circ \mathfrak{H} \circ D')\big] + c\sqrt{\frac{2\log(2/\delta)}{m}}$$

These results hold in particular for $h = \hat{h}^*$.

We accept the previous theorem without proof. However, we note that the proof is similar to the proof of the agnostic PAC learnability of finite hypothesis classes. The main difference here lies in the utilization of another concentration-of-measure inequality, namely the McDiarmid's inequality.

Remark II.1 (Empirical Risk Minimization Principle)**.** *Let us now spend some time to discuss the interpretation of the last theorem. (II.3) suggests that the true risk of any hypothesis h is bounded by the empirical risk of h plus terms that depend on the complexity of the hypothesis class and the size of the training data set. This result offers us a way to escape the dilemma of supervised learning, i.e., the fact that we want to minimize the true risk, although we do not have access to the true distribution of data. Since we cannot directly minimize the true risk, we can do the next best thing available to us, i.e., minimize an upperbound to it. Such result is called the ERM principle. We emphasize that the main message here is that we can solve the supervised learning problem by minimizing the empirical risk while controlling the complexity of the hypothesis class.*

We will now derive a meaningful bound to the Rademacher complexity of the linear hypothesis class given by

$$\mathfrak{H} = \{x \to \langle x, w\rangle : \quad w \in \mathbb{R}^n\},$$

where $\langle ., . \rangle$ denotes the inner product on $\mathbb{R}^n$. To this end, we need the following result.

Lemma II.5 (Letting out of the Loss Function)**.** *For any training example, let $\phi_i : \mathbb{R} \to \mathbb{R}$ be a $\rho-$Lipschitz function. For $\mathbf{a} = (a_1, \ldots, a_m) \in \mathbb{R}^m$ let $\phi(a) = (\phi(a_1), \ldots, \phi(a_m))$. Let $\phi \circ A = \{\phi(a) : \quad a \in A\}$. Then, it holds that*

$$Rad(\phi \circ A) \leq \rho Rad(A)$$

The previous lemma allows us to consider the Rademacher complexity of $\mathfrak{H} \circ D$ instead of $l \circ \mathfrak{H} \circ D$ under a Lipschitz constraint on the loss function l.

Theorem II.6 (Rademacher complexity of Linear Classes)**.** *It holds that*

$$Rad(\mathfrak{H} \circ D) \leq \frac{\|w\|_2}{m} \max_i \|x_i\|_2$$

Proof.

$$
\begin{aligned}
m\mathrm{Rad}(l \circ \mathfrak{H} \circ D) &= \mathbb{E}_\sigma\big[\sup_{a \in \mathfrak{H} \circ D} \sum_{i=1}^{m} \sigma_i a_i\big] \\
&= \mathbb{E}_\sigma\big[\sup_{w \in \mathbb{R}^n} \sum_{i=1}^{m} \sigma_i \langle x_i, w\rangle\big] \\
&= \mathbb{E}_\sigma\big[\sup_{w \in \mathbb{R}^n} \langle \sum_{i=1}^{m} \sigma_i x_i, w\rangle\big] \\
&\leq \|w\|_2 \mathbb{E}_\sigma\big[\|\sum_{i=1}^{m} \sigma_i x_i\|_2\big] \quad \text{by the Cauchy-Schwarz inequality} \\
&\leq \|w\|_2 \mathbb{E}_\sigma\big[\|\sum_{i=1}^{m} \sigma_i x_i\|_2^2\big]^{1/2} \quad \text{using Jensen's inequality} \\
&= \|w\|_2 \mathbb{E}_\sigma\big[\sum_{i,j} \sigma_i \sigma_j \langle x_i, x_j\rangle\big]^{1/2} \\
&= \|w\|_2 \mathbb{E}_\sigma\big[\sum_{i \neq j} \sigma_i \sigma_j \langle x_i, x_j\rangle + \sum_i \sigma_i^2 \langle x_i, x_i\rangle\big]^{1/2} \\
&= \|w\|_2 \big(\sum_{i \neq j} \mathbb{E}_\sigma[\sigma_i \sigma_j] \langle x_i, x_j\rangle + \sum_i \mathbb{E}_\sigma[\sigma_i^2] \langle x_i, x_i\rangle\big)^{1/2} \\
&= \|w\|_2 \big(\sum_{i \neq j} 0 \ \langle x_i, x_j\rangle + \sum_i 1 \langle x_i, x_i\rangle\big)^{1/2} \quad \text{since } \sigma_i \text{ are independent} \\
&= \|w\|_2 \big(\sum_{i=1}^{m} \|x_i\|_2\big)^{1/2} \\
&\leq \|w\|_2 \ m^{1/2} \ \max_i \|x_i\|_2^{1/2}.
\end{aligned}
$$

□

Remark II.2 (Link to Regularization)**.** *The previous result shows that the Rademacher complexity of a linear hypothesis is bounded by the product of the norm of the weight vector and the maximum norm of the data points. This result implies that a smaller 2-norm of the weight vector implies a smaller complexity of the hypothesis class. This, indeed, aligns with a common practice in applications, where one adds the 2-norm of the weight vector to the loss function to prevent overfitting, see also Example I.2 and Example I.1.*

II.5 Occam's Razor and the Overparameterization Regime

We have seen in this chapter that the unachievable task of minimizing the true risk can be replaced by the next best thing, i.e., minimizing an upperbound to it. In particular, this was suggested by upperbounds of the form

$$\text{true risk} \leq \text{empirical risk} + \text{complexity term}.$$

But how do the empirical risk and the complexity term behave? Assume that we have two hypothesis classes that result in roughly similar empirical risk minimizers. In other words, for two different hypothesis classes, running the ERM minimization algorithm returns hypothesis with similar empirical errors. Which of the two hypothesis classes is better? The guiding upperbound suggests that the hypothesis class with the smaller complexity term is better. So, in a sense, the Occam's razor principle is at play here; the simpler answer is better.

It turns out, however, that the relationship between the empirical risk and the complexity term is not straightforward. In particular, note that the empirical risk can be made arbitrarily small by increasing the number of learnable parameters of the hypothesis class. However, a hypothesis class with more learnable parameters is also more complex. For example, consider a certain classification task and let the hypothesis class be that of neural networks. Moreover, assume that we measure the complexity of the hypothesis class by the VC-dimension. To better fit the data, one can simply increase the number of learnable parameters n_p in the neural network. However, one can also show that the VC-dimension of the hypothesis class is bounded by $\mathcal{O}(n_p \log n_p)$. This suggests

that increasing the number of parameters in the neural network increases the complexity term. In summary, increasing the expressivity of the hypothesis class allows us to better fit the training data, but it also increases the complexity term, and hence the generalization error. This is known as bias-variance tradeoff and is a central concept in machine learning[2].

Indeed, the bias-variance tradeoff is an established concept in machine learning and statistics. It has been observed for multiple learning settings and a wide variety of hypothesis classes. However, lately, there has been a surge of counter examples. In particular, it has been observed that increasing the number of parameters in a neural network does not necessarily increase the generalization error. In fact, in many applications, one uses neural networks with a number of parameters that is much larger than the number of training data points and still obtains a small generalization error. This phenomenon is known as overparameterization.

Figure section II.5 summarizes the clash between common wisdom and new observations.

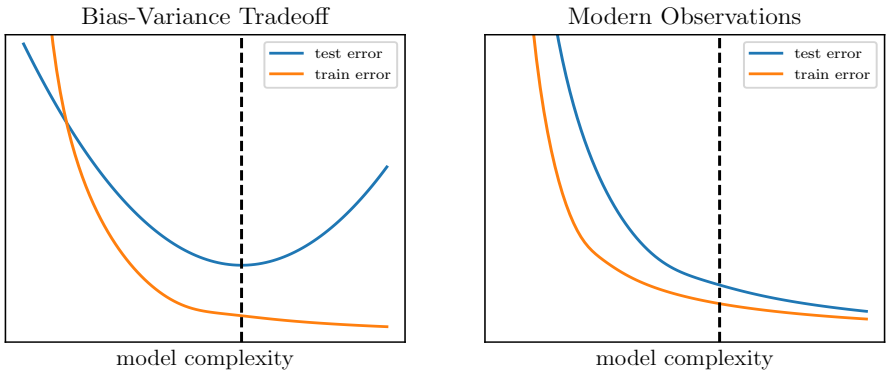

Figure II.3: Common wisdom and many empirical observations suggest that the training error decreases with the complexity of the hypothesis class. The test error decreases as well, only to increase again after a certain threshold as shown in the figure on the left. Here, the threshold usually corresponds to the point where the number of learnable parameters equals the size of the dataset. However, modern observations suggest that deep neural networks can have many parameters and still generalize well as shown in the figure on the right.

Recently, there has been a lot of research to understand the phenomenon of overparameterization. While we will not delve into the details of this research, we look at a one specific example.

Example II.6. Consider as a hypothesis class the space of 2-layer neural networks with ReLU activation functions. Assume that the learnable weights corresponding to the first layer do not deviate much in the 2-norm from their initial values. Further, assume that the weights of the second layer remain bounded in the 2-norm throughout the training. Formally, this class can be written as

$$\mathfrak{H} := \{h(x) = V \cdot [U \cdot x]_+ : U, V \in W\},$$

where

$$\begin{aligned} W := \{(U,V) \quad & U \in \mathbb{R}^{n \times n_{\text{hu}}}, V \in \mathbb{R}^{n_{\text{hu}} \times n_{\text{class}}} \quad \text{and} \\ & \|u_i - u_i^0\|_2 \leq \beta, \|v_i\|_2 \leq \alpha \quad \text{for all units}\}, \end{aligned}$$

and n_{hu} is the number of hidden units, n_{class} is the number of classes, $[.]_+$ denotes the ReLU activation function, u_i, v_i denote the i-th column of U and V, respectively, u_i^0, v_i^0 are the i-th column of the initial weight matrix U, and V, respectively, and β and α are constants.

Train such a neural network on the publicly available CIFAR-10 dataset. Study empirically both the training error and the test error as a function of the number of hidden units. What do you observe? Experiment with n_{hu} ranging from 8 to 30000.

[2]Note that in practice, the test error is a proxy of the generalization error

Now, empirically study the quantities $\|U\|_2$, $\|V\|_2$, and $\|U - U^0\|_2$ as a function of the number of hidden units. What do you observe?

In [6] the authors did the empirical study explained in Example II.6 and observed the following:

- Both the training error and test error decrease as a function of n_{hu}. This indicates that we are in the overparameterization regime.
- The quantity $\|U\|_2$ initially decreases as a function of n_{hu} and then increases. The quantities $\|V\|_2$ and $\|U - U^0\|_2$ decrease as a function of n_{hu}.

The second observation, and specifically the decrease of $\|U - U^0\|_2$ as a function of n_{hu}, is particularly interesting. It suggests that the weights of the neural network do not deviate much from their initial values during training. It appears here that training the neural network is then more focused on optimizing the weights of the second layer. When interpreting the hidden units as features, this result suggests that when one has many features, one has consequently many relevant features that are useful for the task at hand. Hence, there is no need to learn new features.

In a next step, the authors bounded the Rademacher complexity of the specific hypothesis class in Example II.6 by the quantities $\|U - U^0\|_2$ and $\|V\|_2$. In particular, they showed that

$$\text{Rad}(l \circ \mathfrak{H} \circ D) \leq \frac{C}{\sqrt{m}} \max_i \|x_i\|_2 \; \alpha \; (\beta + \|U^0\|_2)$$

This suggests that the Rademacher complexity of the hypothesis class decreases as a function of n_{hu}, which is in line with the empirical observations.

Wait! What is what?

Here is a list of questions that help you check your understanding of key concepts inside this chapter.

1. What is the ultimate goal of a supervised learning problem?
2. Where does the randomness in probably-approximately-correct (PAC) learning come from?
3. What is the realizability assumption in PAC learning?
4. What is the empirical risk minimization principle?
5. How do we judge whether a certain type of a complexity measure of a hypothesis class is good?
6. What is the bias-variance tradeoff? How does it relate to the complexity of the hypothesis class?
7. What are the limitations of the VC-dimension as a complexity measure of a hypothesis class?
8. What is the overparameterization regime?

Machine Learning Models

III.1 Neural Networks

III.2 Kernel Methods

Modern Machine Learning

Modern Mathematical Machine Learning

Bibliography

[1] C. Lubich, *From quantum to classical molecular dynamics: reduced models and numerical analysis*, European Mathematical Society, 2008.

[2] Y. Saleh, "Spectral and active learning for enhanced and computationally scalable quantum molecular dynamics", Dissertation, Hamburg, Germany: Universität Hamburg, 2023.

[3] Y. Saleh, Á. F. Corral, A. Iske, J. Küpper, and A. Yachmenev, "Computing excited states of molecules using normalizing flows", *arXiv preprint arXiv:2308.16468* (2023).

[4] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*, 2014.

[5] R. Schapire, *COS 511: Theoretical Machine Learning*, Lecture Notes, Theoretical Machine Learning, Princeton University Computer Science Department, 2019.

[6] B. Neyshabur, Z. Li, S. Bhojanapalli, Y. LeCun, and N. Srebro, "Towards understanding the role of over-parametrization in generalization of neural networks", *arXiv preprint arXiv:1805.12076* (2018).