

Mathematical Machine Learning

Yahya SALEH, Tizian WENZEL, Kamal SHARMA

March 26, 2024

Contents

I Introduction	v
I.1 Supervised Learning	v
I.1.1 Supervised Learning Informally	v
I.1.2 Supervised Learning Formally	vi
I.2 Curse of Dimensionality	vi
I.3 Approximating Highly-Oscillatory Functions	vii
I.4 Validity of Ocaam Razor and Double-Descent Phenomenon	vii
II Empirical Risk Minimization Principles	ix
III Machine Learning Models	xi
III.1 Neural Networks	xi
III.2 Kernel Methods	xi
IV Modern Machine Learning	xiii
V Modern Mathematical Machine Learning	xv

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 two examples 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

I.1.1 Supervised Learning Informally

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 $f : \text{input} \rightarrow \text{target}$. 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., $\text{target} \in \{0, 1, \dots, M\}$. 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.,


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

Then, one defines 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 be then formulated as an optimization problem

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

While there are many alternatives to solve these optimization problems, the by-far 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.

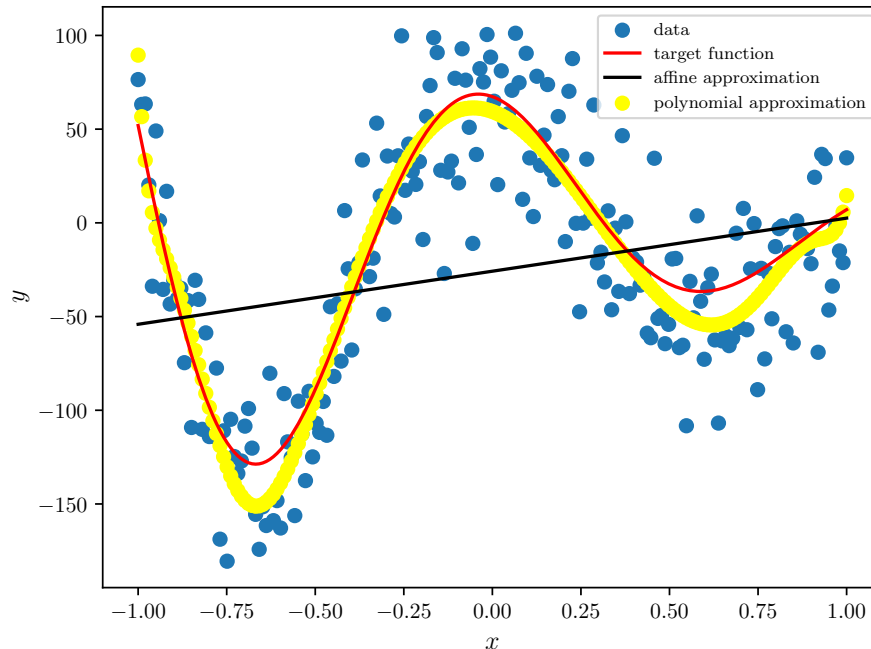


Figure I.1: A regression task; the goal is to fit noisy data (blue dots) assumed to be generated from a true function (solid red line). The data is fitted using an affine mapping (solid black line) and a polynomial mapping (solid yellow line).

Example I.2. [Classification] 📌 I.2

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 ?? 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 from the data everywhere, and would, hence, also generalize badly to unseen data. This is called an *underfitting* phenomenon.

Think about the influence of the following factor 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.1.2 Supervised Learning Formally

A formal definition of learning.

I.2 Curse of Dimensionality

Programming exercise and theorem about approximating smooth functions in higher dimensions.

Example I.3. [Classification] 📌 I.3 tba.

I.3 Approximating Highly-Oscillatory Functions

Example I.4. [Classification] 📌 I.4 tba.

I.4 Validity of Ocaam Razor and Double-Descent Phenomenon

Example I.5. [Double descent] 📌 I.5 tba.

Wait! What is what?

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

Empirical Risk Minimization Principles

Machine Learning Models

III.1 Neural Networks

III.2 Kernel Methods

Modern Machine Learning

Modern Mathematical Machine Learning