Mathematical Introduction to Machine Learning

Lecture 1: Machine Learning: A Mathematical Perspective

September 20, 2025

Lecturer: Lei Wu *Scribe: Lei Wu*

This lecture offers an intuitive introduction to supervised learning from a mathematical perspective. We highlight the key idea of generalization, the ability of a model to go beyond the training data, and discuss why high-dimensional problems make generalization both more challenging and more interesting.

1 Supervised learning

In *supervised learning*, we are given n labeled samples $(x_1, y_1), \cdots, (x_n, y_n)$ with

$$y_i = f^*(x_i) + \varepsilon_i,$$

where:

- x_i's are the inputs and let $x_i \in \mathcal{X}$, where $\mathcal{X}$ represents the input domain.
- y_i's are the labels and let $y_i \in \mathcal{Y}$, where $\mathcal{Y}$ represents the output/label domain.
- ε_i's represent the noise.
- $f^* : \mathcal{X} \to \mathcal{Y}$ denotes the target function, also known as the ground truth, the label function, or the regression function.

We often refer to $S = \{(x_i, y_i)\}_{i=1}^n$ as the training set, since these data are used to train machine learning models.

Remark 1.1. In theoretical analysis, it is often assumed that $x_1, \ldots, x_n$ are independent and identically distributed (i.i.d.) samples drawn from an input distribution ρ. In practice, however, the i.i.d. assumption is often only an approximation and may not hold exactly.

The aim of supervised learning is to approximate/recover/learn the unknown target function f^* by using the training samples S along with some potential prior knowledge about f^* (e.g., f^* might be known to be rotationally or permutation invariant). Roughly speaking, supervised learning problems can be categorized as follows:

- **Regression.** The labels take real continuous values; for example, $\mathcal{Y} = \mathbb{R}$. Regression problems are commonly found in applications, such as financial forecasting, risk assessment, and scientific computing. Figure 1 illustrates an example of learning a univariate function in a regression setting.
- **Classification.** The labels are discrete, for example $\mathcal{Y} = \{1, 2, 3, \ldots, 10\}$. Note that the numerical ordering of the labels is not inherent but imposed for representation. Classification problems include applications, such as image recognition and semantic analysis.

It is worth highlighting that the intrinsic distinction between regression and classification is not simply whether labels are continuous or discrete. Two aspects are essential:

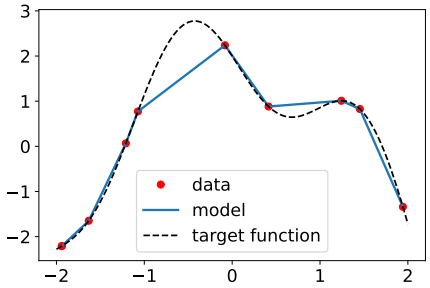


Figure 1: Learning a univariate function using only finite samples. Here, the model is a piecewise linear function.

- **Modeling choice.** Sometimes the same task can be phrased in both ways. For example, predicting human height could be formulated as a regression problem with real-valued outputs in centimeters, or as a classification problem by discretizing heights into $\{1, 2, \ldots, 200\}$ classes (each class representing an integer number of centimeters).
- **Nature of the data distribution.** Not every regression problem can be meaningfully converted into a classification problem. When the label is inherently real-valued with potentially unbounded or very fine resolution (e.g., predicting temperature, stock returns, or energy levels in physics), discretizing into classes either loses too much information or leads to an impractically large number of classes. In such cases, regression is the natural formulation.

Thus, while some problems admit both formulations, in general regression and classification reflect different underlying structures of the prediction task.

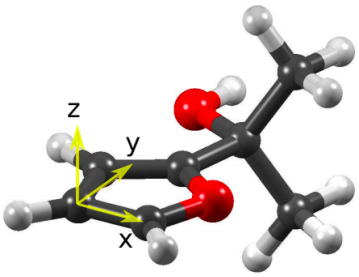
1.1 High-dimensional complex tasks

The regression task in Figure 1 is one-dimensional and can be handled effectively by classical methods such as polynomial or spline interpolation. In contrast, a major strength of modern machine learning lies in its ability to cope with *high-dimensional* problems. Below are several representative examples where ML has become indispensable.

- **Image recognition.** Shown below is the task of ImageNet. This dataset has around 1 million $224 \times 224 \times 3$ color images as well as the labels. The labels are image categories, e.g. bird, cat, and the number of categories is 1000. The aim is to learn a classifier: $[0, 1]^{224\times224\times3} \to \{1, 2, \ldots, 1000\}$. Here, the input dimension is $d = 150528 \gg 1$.

Figure 2: The ImageNet website: https://www.image-net.org/update-mar-11-2021.php

- **Molecule modeling.** Learn the potential energy function f^*, which maps the atom coordinates to the potential energy of that molecule. Assume that there are N atoms. Then, we know that $f^* : \mathbb{R}^{3\times N} \to \mathbb{R}$ must satisfy the following symmetries/invariances:
  - Translation and rotation,
  - Permutation among identical atoms.



- **Image translation.** Learn a map $\mathcal{T} : [0,1]^{214\times214\times3} \to [0,1]^{214\times214\times3}$. In such a case, the output space $\mathcal{Y}$ is high-dimensional too.

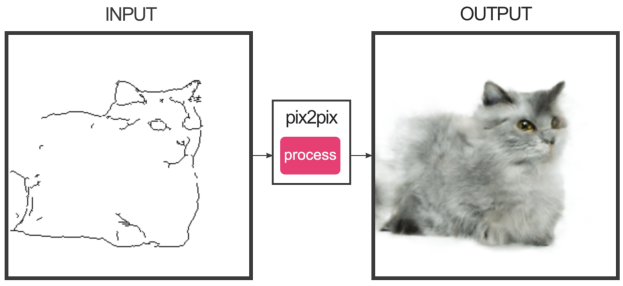


Figure 3: https://affinelayer.com/pixsrv/

- **Learning PDE operators.** Consider a PDE of the form $\mathcal{L}(u, \nabla u, \nabla^2 u) = f$, and define the solution operator $S : f \mapsto u$, which maps the source term f to the corresponding solution u. By learning this operator, one can directly obtain the solution from the source term without resorting to traditional numerical solvers. This operator-learning perspective has become a central theme in modern scientific machine learning. Note that the input f and the output u are functions, so both the input and output spaces are infinite-dimensional.

In all the examples above, the target functions are both high-dimensional and complex. As we will see later, traditional methods often struggle in such settings due to their limited approximation power (i.e., expressivity). To address these challenges, we require models with sufficient capacity to approximate high-dimensional nonlinear functions effectively. This is why modern machine learning is built upon neural network models, which excel at capturing complex high-dimensional structures.

2 Learning paradigm: Empirical risk minimization (ERM)

2.1 Learning Procedure

A standard approach to modern machine learning can be decomposed into three steps:

Step 1: Choose a parametric model. Select a family of functions $f(\cdot;\theta)$ parameterized by $\theta \in \mathbb{R}^m$. Here, θ denotes the parameters to be learned from data, and m denotes the number of parameters. For

notational brevity, we often write f_θ instead of $f(\cdot;\theta)$. This gives rise to the *hypothesis space* (or *model class*)

$$\mathcal{F}_m = \{f_\theta \,:\, \theta \in \mathbb{R}^m\}.$$

Common choices include:

- Linear functions: $f(x;\theta) = \theta^\top x + \theta_0$,
- Basis expansions: $f(x;\theta) = \sum_{j=1}^m \theta_j \varphi_j(x)$, where $\{\varphi_j\}$ are basis functions (e.g., polynomials, trigonometric functions, wavelets, random features, splines),
- (Deep) neural networks.

Classical machine learning is typically based on the first two types of models, while modern machine learning predominantly relies on deep neural networks, due to its strong approximation power.

Step 2: Formulate an optimization problem. The learning task is to choose $f \in \mathcal{F}_m$ that approximates the unknown target function f^*, using the training data S. In modern machine learning, this is typically achieved by minimizing the *empirical risk*:

$$\min_\theta \ \widehat{\mathcal{R}}(\theta) := \frac{1}{n}\sum_{i=1}^n \ell\big(f(x_i;\theta), y_i\big),$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to [0,\infty)$ is a loss function measuring prediction error.

If prior knowledge about f^* (e.g., smoothness) is available, it can be incorporated through a *regularization function* $r(\cdot)$. This corresponding optimization problem is

$$\min_\theta \ \widehat{\mathcal{R}}(\theta) + \lambda\, r(\theta), \tag{1}$$

where $\lambda > 0$ is a *hyperparameter* that controls the trade-off: a larger λ enforces stronger regularization (favoring low-complexity models), while a smaller λ emphasizes data fitting (risking overfitting). Unlike model parameters θ, hyperparameters are usually not learned from the training data directly and often need to be tuned manually. We denote the learned solution by

$$\hat\theta = \operatorname*{argmin}_\theta \widehat{\mathcal{R}}(\theta) + \lambda\, r(\theta).$$

Step 3: Choose an optimization method. Finally, we must solve the optimization problem (1). Depending on the setting, one may use optimization algorithms such as gradient descent, stochastic gradient descent, or second-order methods like Newton's method. This step is crucial: the choice of optimization algorithm affects not only the computational efficiency of training, but also the feasibility of handling large-scale models in modern machine learning.

Remark 2.1 (The advantage of the three-step paradigm). A key feature of modern machine learning is that almost all learning problems can be cast into the three-step paradigm above:

modeling → formulating an optimization problem → solving it with a generic optimization algorithm.

This stands in contrast to classical machine learning, where different problems often required problem-specific iterative algorithms. The benefit of the modern approach is twofold: it provides a simple and unified methodology, and it shifts the main intellectual effort to the modeling step, while the subsequent optimization can often be handled by standard, well-tested algorithms. This universality has been one of the driving forces behind the rapid development of modern machine learning.

2.2 Evaluating a model

Let $f_{\hat{\theta}}$ be the model obtained from the learning procedure. Then, the next key question is: *How good is this learned model?* In particular, we care about its **generalization ability** – how well it performs on unseen data. A natural way to measure this is the *generalization error*:

$$\mathcal{E}_{\mathrm{ge}}(\theta) = \mathbb{E}_{x\sim\rho}\left[\ell(f_\theta(x), f^*(x))\right], \tag{2}$$

which quantifies how the model f_θ performs on new samples drawn from the underlying distribution ρ.

Since $\mathcal{E}_{\mathrm{ge}}(\theta)$ cannot be computed exactly, a standard remedy is to use a *test set*, consisting of *fresh data* not used during training:

$$S_{\mathrm{te}} = \{(\tilde{x}_j, f^*(\tilde{x}_j))\}_{j=1}^{n_{\mathrm{te}}}.$$

We then approximate (2) by the *test error*:

$$\mathcal{E}_{\mathrm{te}}(\theta) = \frac{1}{n_{\mathrm{te}}} \sum_{j=1}^{n_{\mathrm{test}}} \ell(f_\theta(\tilde{x}_j), f^*(\tilde{x}_j)) \,.$$

This naturally leads to the question: how many test samples are needed for the test error to be a reliable estimate of the generalization error?

Gap for a fixed model. Suppose θ is fixed in advance (hence independent of the test set). When $\{\tilde{x}_j\}$ are i.i.d. samples from the underlying distribution ρ, the test error is simply a Monte Carlo estimate of the generalization error.

Lemma 2.2. *Under the above assumptions, we have for any fixed θ that*

$$\mathbb{E}\,\big|\,\mathcal{E}_{\mathrm{te}}(\theta) - \mathcal{E}_{\mathrm{ge}}(\theta)\,\big|^2 = O\big(n_{\mathrm{te}}^{-1}\big)\,. \tag{3}$$

Proof. For any fixed θ, define $Z_j = \ell(f_\theta(\tilde{x}_j), f^*(\tilde{x}_j))$ for $j = 1, \ldots, n_{\mathrm{te}}$. Then $Z_1, \ldots, Z_{n_{\mathrm{te}}}$ are i.i.d. random variables with mean

$$\mu = \mathbb{E}[Z_j] = \mathbb{E}_{x\sim\rho}\left[\ell(f_\theta(\tilde{x}_j), f^*(x))\right] = \mathcal{E}_{\mathrm{ge}}(\theta),$$

and $\mathcal{E}_{\mathrm{te}}(\theta) = \frac{1}{n_{\mathrm{te}}} \sum_{j=1}^{n_{\mathrm{te}}} Z_j$ is their empirical mean. Then, we have

$$\begin{aligned}
\mathbb{E}\Big[\big(\mathcal{E}_{\mathrm{te}}(\theta) - \mathcal{E}_{\mathrm{ge}}(\theta)\big)^2\Big] &= \mathbb{E}\left[\left(\frac{1}{n_{\mathrm{te}}}\sum_{j=1}^{n_{\mathrm{te}}}(Z_j - \mu)\right)^2\right] = \frac{1}{n_{\mathrm{te}}^2}\sum_{i,j=1}^{n_{\mathrm{te}}} \mathbb{E}\big[(Z_i-\mu)(Z_j-\mu)\big] \\
&= \frac{1}{n_{\mathrm{te}}^2}\sum_{i\neq j} \mathbb{E}\big[(Z_i-\mu)(Z_j-\mu)\big] + \frac{1}{n_{\mathrm{te}}^2}\sum_{i} \mathbb{E}\big[(Z_i-\mu)^2\big].
\end{aligned}$$

By independence, the cross terms vanish when $i \neq j$, leaving

$$\mathbb{E}\Big[\big(\mathcal{E}_{\mathrm{te}}(\theta) - \mathcal{E}_{\mathrm{ge}}(\theta)\big)^2\Big] = \frac{1}{n_{\mathrm{te}}^2}\sum_{i=1}^{n_{\mathrm{te}}} \mathbb{E}\big[(Z_i-\mu)^2\big] = \frac{\mathbf{Var}(Z)}{n_{\mathrm{te}}}.$$

Hence the standard deviation of the gap is of order $O(n_{\mathrm{te}}^{-1/2})$, which leads to (3). □

Thus, for any fixed model, a reasonably large test set ensures that $\mathcal{E}_{\text{te}}(\theta)$ reliably estimates $\mathcal{E}_{\text{ge}}(\theta)$. The key ingredient here is *independence* between the model and the test data. In practice, we are interested in $\hat{\theta}$ produced by the training procedure. Although $\hat{\theta}$ depends on the training data, it is still independent of the test set, so the bound (3) also applies to $f_{\hat{\theta}}$. Hence, if $\mathcal{E}_{\text{te}}(\hat{\theta})$ is small and the test set is large enough, we can conclude that the learned model has small generalization error.

Remark 2.3. The training error also has the form of an empirical mean:

$$\mathcal{E}_{\text{tr}}(\hat{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \ell\big(f_{\hat{\theta}}(x_i), f^*(x_i)\big)\,.$$

Why, then, does a small training error not guarantee small generalization error? The key difference is independence: test data are independent of the model parameters, while training data are not.

3 Understanding Generalization

In this section, we aim to develop high-level intuition about how the generalization error depends on two key factors: the sample size n and the capacity of the model. A natural question to ask is:

Does increasing model capacity always lead to smaller generalization error?

In general, model capacity is influenced not only by the number of parameters but also by the model structure. For simplicity, however, we will use the number of parameters m as a proxy for model capacity throughout the discussion.

3.1 Approximation vs. Estimation

To help understanding, we introduce a reference model

$$\theta^* = \operatorname*{argmin}_{\theta\in\mathbb{R}^m} \mathcal{E}_{\text{ge}}(\theta). \tag{4}$$

Obviously, f_{θ^*} represent the best approximation to f^* within the hypothesis space $\mathcal{F}_m$. Consider the following decomposition of the generalization error:

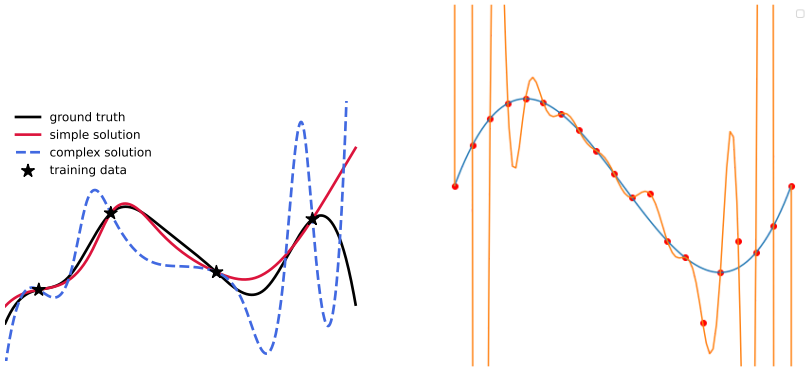
$$\underbrace{f^* - f_{\hat{\theta}}}_{\text{generalization error}} = \underbrace{f^* - f_{\theta^*}}_{\text{approximation error}} + \underbrace{f_{\theta^*} - f_{\hat{\theta}}}_{\text{estimation error}}\,. \tag{5}$$

Let us discuss the two terms of the RHS separately.

Approximation error. This error is independent of the sample size and only depends on the model choice. In general, the higher capacity is the model, the smaller is this approximation error. Particularly, if $f^* \in \mathcal{F}_m$, then this error becomes exactly zero.

Estimation error. This error reflects the uncertainty introduced by having only a finite training set: with limited data, the learning algorithm cannot perfectly identify the ideal solution θ^*.

- **Selection uncertainty (noiseless case).** When the data are clean, i.e. $y_i = f^*(x_i)$, many functions in the hypothesis space may fit the training set equally well, yet some of them can generalize poorly. Because the training set is finite, the algorithm may end up selecting such a poorly generalizing solution. We refer to Figure 4 for an illustration. Moreover, as the model class becomes larger, the number of these "bad" solutions increases, making this risk more severe.
  - Regularization can reduce (but not eliminate) this risk.
  - Collecting more training data reduces the uncertainty, thereby lowering the estimation error.
- **Overfitting noise (noisy case).** When the labels contain noise, a complex model can fit not only the underlying pattern but also the noise itself. This fitting of noise introduces additional error. This is the classical "overfitting" picture emphasized in textbooks. However, it is only part of the story: estimation error also exists without noise, as explained above.



(a) clean data (b) noisy data

Figure 4: Illustration of estimation error. **(a)** Black dots are training samples, the black curve is the ground truth, and the blue and red curves are two candidate hypotheses. With larger model capacity, the hypothesis space includes more poorly generalizing solutions (e.g., the blue curve). **(b)** When labels contain noise, the model may also fit the noise, leading to poor generalization.

3.2 Generalization error v.s. model capacity

Understanding how the generalization error depends on model size (a proxy for model capacity) is of central importance. It provides guidance for choosing the complexity of the model, such as the number of parameters or the network width, and for tuning regularization strength. In practice, striking the right balance between model capacity and available data is crucial for achieving good generalization.

Let $g_n(m), e_n(m), a(m)$ denote the generalization error, estimation error, and approximation error, respectively. By the decomposition (5), Roughly, we have

$$g_n(m) = a(m) + e_n(m).$$

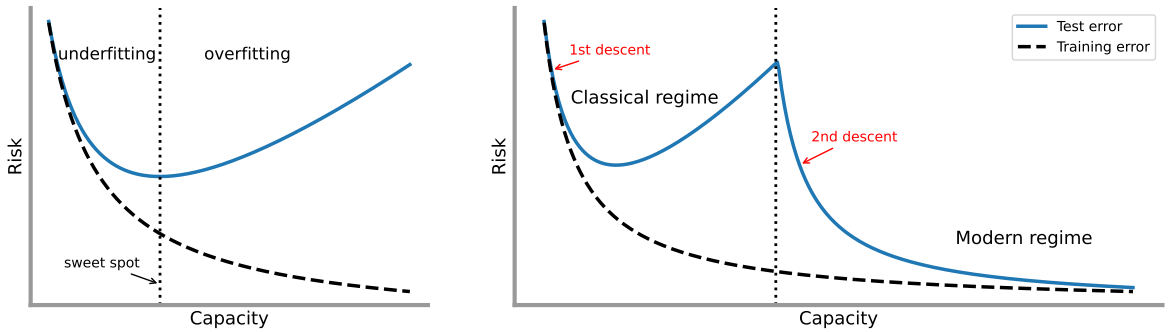


Figure 5: Generalization error versus model capacity. **(Left: Classical U-shaped curve)** As model capacity increases, the test error (blue) first decreases due to reduced approximation error, reaches a minimum at the "sweet spot," and then rises again as estimation error dominates. The training error (black dashed) decreases monotonically with capacity. **(Right: Double descent curve)** In modern machine learning, a richer behavior is often observed. After the classical U-shaped phase ("first descent" and subsequent rise in the *classical regime*), the test error decreases again once the model becomes over-parameterized beyond certain threshold ("second descent"), entering the *modern regime*. This double descent phenomenon highlights that large models, when trained with sufficient data and appropriate optimization, can generalize well despite having far more parameters than training samples.

From the discussion above, two qualitative properties are clear: 1) $a(m)$ decreases as m increases, and 2) $e_n(m)$ increases as m increases. Hence, there must exist a trade-off between the approximation error and estimation error for obtaining the optimal generalization error. We ask the question:

For fixed n, how $g_n(m)$ behaves when increasing m?

The U-shaped generalization curve. A classical view of how generalization depends the model capacity is the U-shaped curve shown in the left panel of Figure 5. As model capacity increases, the generalization error first decreases, reaches a minimum (the "sweet spot"), and then rises again.

This behavior reflects the approximation–estimation trade-off: with small models, approximation error dominates because the hypothesis space is too limited to approximate f^*. As capacity grows, approximation error decreases, but estimation error increases. Once, the approximation error becomes negligible compared to the estimation error; the total generalization error is very like to increase. Together, these two effects yield the U-shape. A simple illustration is given by

$$g_n(m) = a(m) + e_n(m) = \frac{1}{m^2} + \frac{m}{n},$$

which indeed has a U-shaped profile with the optimal choice at $m = O(n^{1/3})$.

The U-shaped curve implies that good generalization requires balancing approximation and estimation: the optimal model size grows with the amount of data, and this balance underlies the important role of regularization

Is the U-shaped picture really true? Although the U-shaped curve has long been a cornerstone of classical machine learning, it does not fully capture modern practice. Empirical evidence shows more complex behaviors, most notably the *double descent* phenomenon illustrated in the right panel of Figure 5.

Formally, since

$$g_n(m) = a(m) + e_n(m) \quad \Rightarrow \quad g_n'(m) = a'(m) + e_n'(m),$$

the shape of $g_n(m)$ depends on the relative rates of change of approximation and estimation errors. We generally know only that $a'(m) \le 0$ and $e_n'(m) \ge 0$, which by themselves do not guarantee a U-shape. For example, estimation error often saturates when m is large, so that $e_n'(m) \approx 0$ in this regime. Meanwhile, for complex tasks the approximation error decreases only slowly, and may continue to dominate even for very large m. In such cases, $g_n(m)$ can keep decreasing with m, giving rise to double descent as observed in random feature models and neural networks [Belkin et al., 2019].

In short, the generalization curve is determined by the balance between approximation and estimation errors. For simple problems such as low-dimensional linear regression, approximation error vanishes quickly and the classical U-shape emerges. For high-dimensional, complex tasks, approximation error remains significant, motivating the use of models with very large capacity to achieve good generalization.

4 Understanding the Curse of Dimensionality

We have emphasized many times that learning high-dimensional nonlinear functions is extremely hard. In this section, we provide both intuitive explanations and concrete quantification of this hardness.

4.1 A motivating example

We start with piecewise constant models. For a set $\Omega \subset \mathbb{R}^d$, denote the indicator function by

$$\mathbf{1}_\Omega(x) = \begin{cases} 1 & \text{if } x \in \Omega, \\ 0 & \text{if } x \notin \Omega. \end{cases}$$

One dimension. Consider a one-dimensional target function $f^* : [0,1] \to \mathbb{R}$. Suppose the input data $\{x_j\}_{j=1}^n$ lie on a uniform grid with spacing $h = 1/n$, i.e., $x_j = jh$ for $j = 1, \dots, n$. Let $S_j = [x_{j-1}, x_j]$. Consider the solution produced by the piecewise constant interpolation

$$\hat{f}_n(x) = \sum_{j=1}^n f^*(x_{j-1}) \mathbf{1}_{S_j}(x). \tag{6}$$

Lemma 4.1. *For any $f \in C^1([0,1])$, define $\mathrm{Lip}(f) = \sup_{x \in [0,1]} |f'(x)|$. Then*

$$\|\hat{f}_n - f^*\|_{L^2} \le \frac{\mathrm{Lip}(f^*)}{\sqrt{3}n}.$$

Proof. By the Lipschitz property of f^*,

$$\begin{aligned} \|\hat{f}_n - f^*\|_{L^2}^2 &= \sum_{j=1}^n \int_{x_{j-1}}^{x_j} |\hat{f}_n(x) - f^*(x)|^2 \, \mathrm{d}x = \sum_{j=1}^n \int_{x_{j-1}}^{x_j} |f^*(x_{j-1}) - f^*(x)|^2 \, \mathrm{d}x \\ &\le \sum_{j=1}^n \int_{x_{j-1}}^{x_j} \mathrm{Lip}^2(f^*) |x - x_{j-1}|^2 \, \mathrm{d}x = \mathrm{Lip}^2(f^*) n \frac{h^3}{3} = \frac{\mathrm{Lip}^2(f^*)}{3n^2}. \end{aligned}$$

□

Question 4.2. According to the proof, which property of the target function enables it to generalize beyond the training points?

High dimensions. Now consider the domain $\mathcal{X} = [0,1]^d$, partitioned into cubes of side length h. Each cube has volume h^d, so the total number of cubes is $n = h^{-d}$. Thus we can write

$$[0,1]^d = \bigcup_{j=1}^{n} S_j,$$

where S_j denotes the j-th cube. We again use the piecewise constant model introduced in (6), with x_{j-1} denoting the reference point (e.g., a corner) of cube S_j.

Theorem 4.3. *For any $f \in C^1([0,1]^d)$, define $\mathrm{Lip}(f) = \sup_{x\in[0,1]^d} \|\nabla f(x)\|_2$. Then*

$$\|\hat{f}_n - f^*\|_{L^2} \le \frac{\sqrt{d/3}\,\mathrm{Lip}(f^*)}{n^{1/d}}.$$

Proof. By a change of variables,

$$\begin{aligned}
\|\hat{f}_n - f^*\|_{L^2}^2 &= \sum_{j=1}^{n} \int_{S_j} |f^*(x_{j-1}) - f^*(x)|^2 \,\mathrm{d}x \\
&= \sum_{j=1}^{n} \int_{[0,h]^d} |f^*(x_{j-1}) - f^*(x_{j-1}+t)|^2 \,\mathrm{d}t \\
&\le \mathrm{Lip}^2(f^*) \sum_{j=1}^{n} \int_{[0,h]^d} \|t\|_2^2 \,\mathrm{d}t \\
&= \mathrm{Lip}^2(f^*) n \cdot d \cdot h^{d-1} \int_0^h z^2 \,\mathrm{d}z \\
&= \frac{d\,\mathrm{Lip}^2(f^*)}{3} n h^{d+2} = \frac{d\,\mathrm{Lip}^2(f^*)}{3n^{2/d}}.
\end{aligned}$$

□

The curse of dimensionality (CoD). The error rate is $O(n^{-1/d})$. To achieve accuracy ε, one needs

$$n_\varepsilon = \Omega\left(\frac{1}{\varepsilon^d}\right)$$

grid points – an exponential dependence on the input dimension d. For instance, to reach accuracy $\varepsilon = 0.1$, one requires $\Omega(10^d)$ grid points, which is infeasible for large d. This phenomenon is known as the *curse of dimensionality*.

What causes the CoD?

- *Where does generalization come from?* In this setting, generalization to unseen data relies on the smoothness of f^*, i.e., the fact that $\mathrm{Lip}(f^*) < \infty$. Indeed, the error behaves like $O(h)$ in any dimension, where h is the average distance from a test point to the nearest training sample.

- *Why does smoothness-based generalization lead to CoD?* In d dimensions, to achieve an average spacing of h, we need $n = h^{-d}$ samples. Hence the sample complexity grows exponentially with d, which is the essence of the curse of dimensionality.

4.2 A probabilistic viewpoint of the CoD

In this section, we attempt to explain the following (informal) phenomenon more quantitatively.

In a high dimensional space, the average distance between points is $O(1)$ unless there are exponentially many data points.

In the proceeding derivation, the above phenomenon is illustrated with the uniform grid points. Here, we prove it for randomly sampled points.

Theorem 4.4 (Hoeffding's inequality)**.** *Let $X_1, \dots, X_n$ be i.i.d. random variables. If $a \le X_i \le b$, then,*

$$\mathbb{P}\left\{\left|\frac{1}{n}\sum_{i=1}^{n} X_i - \mu\right| \ge t\right\} \le 2e^{-\frac{2nt^2}{(b-a)^2}}.$$

We do not prove this inequality here and interesting readers can find the proof in Lecture 13.

Theorem 4.5. *Suppose $x_i \overset{\text{i.i.d.}}{\sim} \mathrm{Unif}([0,1]^d)$ for $i = 1, \dots, n$ and $t < 1/3$. Then*

$$\mathbb{P}\left(\min_{i\in[n]} \|x_i\|_2 \le \sqrt{dt}\right) \;\le\; 2ne^{-2d(1/3-t^2)^2}.$$

This theorem shows that the probability of even a single point lying close to the origin decays exponentially with the dimension d. Equivalently, for any $\delta \in (0,1)$, with probability at least $1-\delta$ over the sampling of n points, we have

$$\min_{i\in[n]} \|x_i\|_2 \;\ge\; \sqrt{d}\left(\frac{1}{3} - \sqrt{\frac{\log(2n/\delta)}{2d}}\right). \tag{7}$$

As a concrete example, take $n = e^{4d/9}$ and $\delta = 2^{-d}$. Then, with probability at least $1 - 2^{-d}$,

$$\min_{i\in[n]} \|x_i\|_2 \;\ge\; \tfrac{1}{2}.$$

In other words, even with exponentially many samples, it remains overwhelmingly likely that none of the sampled points lies near the origin.

Proof. By definition,

$$\mathbb{E}[\|x\|_2^2] = \mathbb{E}\left[\sum_{i=1}^{d} x_i^2\right] = d\int_0^1 t^2\,dt = \frac{d}{3} =: d\mu_0.$$

Next, we estimate the probability that $\|x\|_2^2$ deviates from its mean. Applying Hoeffding's inequality to the i.i.d. variables $x_i^2 \in [0,1]$ gives

$$\mathbb{P}\left(\left|\frac{1}{d}\sum_{i=1}^{d} x_i^2 - \mu_0\right| \ge \lambda\right) \;\le\; 2e^{-2d\lambda^2}.$$

Equivalently,

$$\mathbb{P}\big(|\|x\|_2^2 - d\mu_0| \geq d\lambda\big) \;\leq\; 2e^{-2d\lambda^2}.$$

Now, for any $t < \sqrt{\mu_0}$,

$$\begin{aligned}
\mathbb{P}\Big(\min_i \|x_i\|_2 \leq \sqrt{d}\, t\Big) &\leq \sum_{i=1}^{n} \mathbb{P}\Big(\|x_i\|_2 \leq \sqrt{d}\, t\Big) \\
&= n\, \mathbb{P}\big(\|x\|_2^2 \leq dt^2\big) \\
&= n\, \mathbb{P}\big(|\|x\|_2^2 - d\mu_0| \geq d(\mu_0 - t^2)\big) \\
&\leq 2n \exp\big(-2d(\mu_0 - t^2)^2\big).
\end{aligned}$$

This completes the proof. □

Summary of high-dimensional phenomena. The theorem above illustrates a broader principle: in high dimensions, random samples tend to concentrate in surprising ways. This fact is often described in several equivalent heuristics:

- **Concentration of distances:** The distances between random points in high-dimensional space are all nearly the same.
- **Near-orthogonality:** Random high-dimensional vectors are almost orthogonal to each other.
- **Concentration on the shell:** In a high-dimensional ball, most of the volume is concentrated near the boundary rather than in the interior.

These perspectives are different manifestations of the same underlying concentration phenomenon, and they explain why intuition from low-dimensional geometry often fails in high-dimensional learning problems.

4.3 Can we overcome the CoD?

To overcome the curse of dimensionality, one must exploit special structural properties of the target function f^* beyond mere smoothness. Equally important is choosing a hypothesis class f_θ that is *adaptive* to these structures. Otherwise, even simple target functions may still suffer from CoD when approximated with a poorly chosen model.

- If f^* is a general Lipschitz function, **any** learning method suffers from CoD.
- If f^* is linear and we use linear regression, then CoD is avoided, since the model class matches the structure of f^*.
- If f^* is linear but we approximate it with piecewise-constant functions, we still face CoD.
- If f^* is constant, then piecewise-constant models avoid CoD.

References

[Belkin et al., 2019] Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854.