Lecture 1: Introduction to Supervised Learning

Readings: ESL (Ch. 1–2), ISL (Ch. 1–2), Bach (Ch. 2); code

Soon Hoe Lim

August 26, 2025

Outline

What is Statistical Learning?

- Statistical learning is a field at the intersection of statistics and machine learning.
- It is concerned with building mathematical models that can learn from data.
- The goal is to develop algorithms that detect patterns in data to predict future data or make decisions.
- This course: **"Principles of Statistical Machine Learning"**.

⚠ This is an intensive course, as a significant amount of material will be covered in a short amount of time.

Machine Learning in 2025

Automated learning from large, complex data drives breakthroughs in autonomous systems and generative AI tools such as ChatGPT.

- **Hype Cycles:**

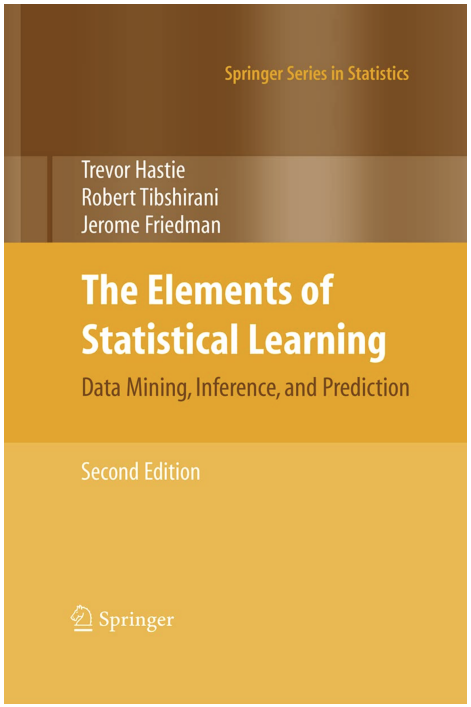
  Big Data $\rightarrow$ Data Science $\rightarrow$ Machine Learning $\rightarrow$ Deep Learning $\rightarrow$ Artificial Intelligence $\rightarrow$ Large Language Models

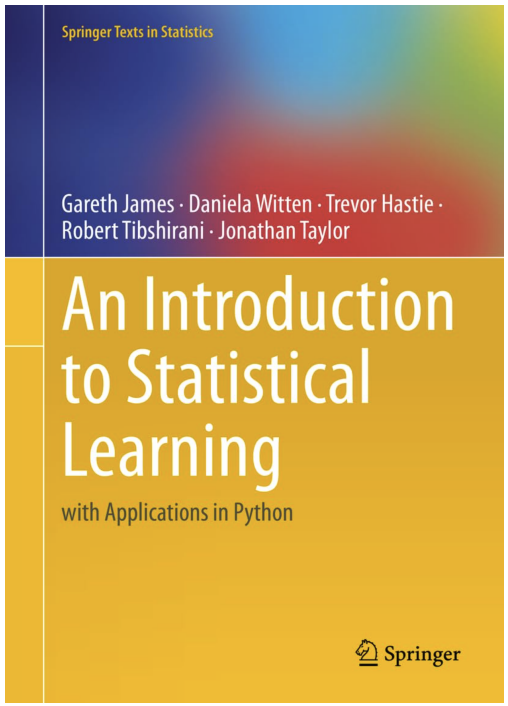
  Hype tells us what works, but not *why it works (or not)*.

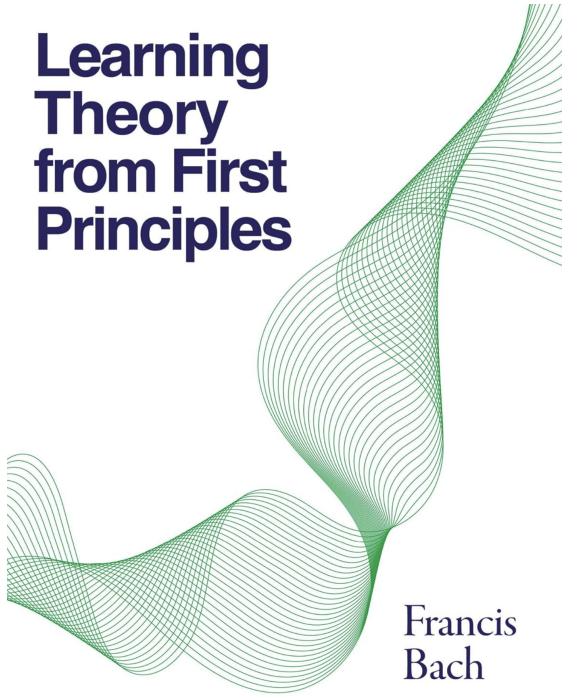
- **Why Learning Theory Matters:**
  - Helps us understand what can be learned from data and why, and in practice, its insights are complemented by empirical validation.
  - Connects statistics and computation in a principled way.
  - Keeps us grounded — avoids blindly chasing trends amid information overload.

Course Structure & Goals

- **Objective:** To study core machine learning methods and their statistical principles, reinforced through empirical experiments with data.
- **Syllabus Overview:** Will cover topics from linear models to deep neural networks, using ESL as the main text. Bach serves as a theoretical supplement, while ISL* provides a practical one. Additional materials will also be covered. See Canvas for a tentative schedule.







*More resources at `https://www.statlearning.com/resources-python`.

Course Assessments

- A **course project** (50%): a project proposal due on **Sept 12** and a final report due on **Oct 17** (see Canvas for details)
- A **final exam** (50%): a closed book exam scheduled for **Oct 23**.

Tips: Be sure to attend lectures and tutorials, as important and additional materials will be covered there. Also, complete the exercises assigned, as they will help you prepare for the final exam.

The Supervised Learning Problem

Supervised learning uses input (predictors, independent variables, or features) to predict one or more outputs (responses or dependent variables).

- **Input/Feature Space ($\mathcal{X}$):** The space of input variables X. This is typically a vector space (e.g., $\mathcal{X} = \mathbb{R}^p$).
- **Output/Target Space ($\mathcal{Y}$):** The space of output variables Y we want to predict (outcomes).
- **Training Data:** A dataset $D_N = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ of N pairs.
- **Assumption:** The underlying ground truth function f_{true} that relates x_n to y_n is unknown and can only be accessed through the training set. Typical to assume each pair (x_n, y_n) is an i.i.d. sample from an unknown joint probability distribution P on $\mathcal{X} \times \mathcal{Y}$.

Goal: "Learn" a **model** $\hat{f} : \mathcal{X} \to \mathcal{Y}$ using the dataset, so that we can use the learned model to accurately predict outcomes for new, previously unseen inputs.

Types of Supervised Learning

The type of problem depends on whether the output variable Y is quantitative or qualitative.

1. **Regression:** The output Y is a quantitative, real-valued number.
   - $\mathcal{Y} \subseteq \mathbb{R}$.
   - **Example:** Predicting a house price Y based on its features X.
2. **Classification:** The output G is a qualitative, categorical/class label from a finite set $\mathcal{G}$.
   - $\mathcal{G} = \{c_1, \ldots, c_K\}$.
   - Dummy variables: G is commonly coded using a vector of K binary variables (bits): only one of these K bits is "on" (set to 1) at any given time
   - **Example:** Classifying an email as spam or not spam based on its text content (X). We can treat the target Y as a quantitative output (as a 0-1 coded version of G), with the predictions $\hat{Y} \in [0, 1]$, and assign to $\hat{G}$ the class label according to whether $\hat{y} > 0.5$.

Both can be viewed as a task in function approximation.

From Data to Predictions

We have access to a training set of paired data, and so far we can:

- Identify the input and output variables.
- Determine the type of regression problem.
- State key assumptions about the data.

❓ What are the challenges? Despite them, how can we turn the ingredients into useful predictions?

Many possible **models** f — each with its own pros and cons.
They also come with **design choices**, **loss functions** to evaluate performance and **learning algorithms** to optimize the model.

Together, these elements constitute a **prediction method**. Let's take a look at two methods that can be learned easily next.

Two Simple Prediction Methods

1. Linear Models (Least Squares)

- Assumption: There is a linear relationship between inputs and outputs.
- Parametric Model: Given an input column vector $X = (X_1, X_2, \ldots, X_p)$, predict the output Y via: $f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$.
- Parameter estimation: The "best" linear fit is found by minimizing the sum of squared residuals (RSS) from the training data:

$$RSS(\beta) = \sum_{n=1}^{N} (y_n - f(x_n))^2.$$

- They rely on strong structural assumptions, leading to stable but potentially inaccurate predictions if the true relationship is non-linear.

Two Simple Prediction Methods

2. k-Nearest-Neighbor (k-NN)

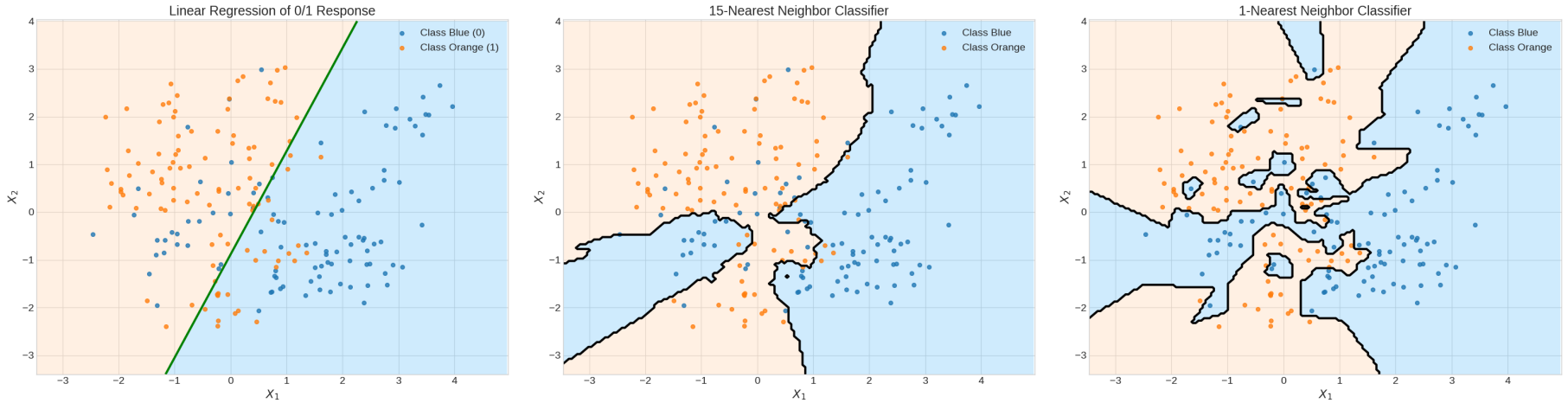
- Assumption: Makes very mild structural assumptions.
- Non-Parametric Model: For a new point x_0, identifies its k nearest training observations based on a chosen distance metric: $f(x) = \frac{1}{k} \sum_{x_n \in N_k(x)} x_n$, where $N_k(x)$ is the neighborhood of x defined by the k closest points in D_N.
  - Regression: The average of the outputs of the k nearest neighbors.
  - Classification: The majority vote among the classes of the k nearest neighbors.

- The predictions are often accurate, but can be unstable due to high variance, especially when k is small.
- Curse of Dimensionality: Large errors and increased variance in high dimensions (why?).

Example 1: Binary Classification in 2D

We are given a simulated dataset on input points $X = (X_1, X_2) \in \mathbb{R}^2$ and their corresponding color label (BLUE or ORANGE). We aim to classify new input points according to their color.

- The classes are coded as a binary variable (BLUE $= 0$, ORANGE $= 1$), and then fitted by different models.
- The line is the decision boundary defined by $X^T\beta = 0.5$.

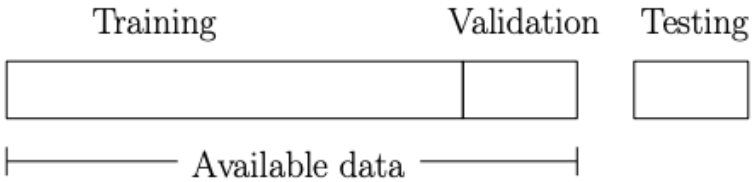


The orange shaded region denotes that part of input space classified as ORANGE, while the blue region is classified as BLUE.

Practical Performance Evaluation

In practice, we do not have access to the test distribution but only samples from it. In most cases, the data are split into three parts:

- The **training set**, on which learning models will be estimated.
- The **validation set**, to estimate hyperparameters (all methods have some) to optimize the performance measure (model selection).
- The **testing set**, to evaluate the performance of the final chosen model.

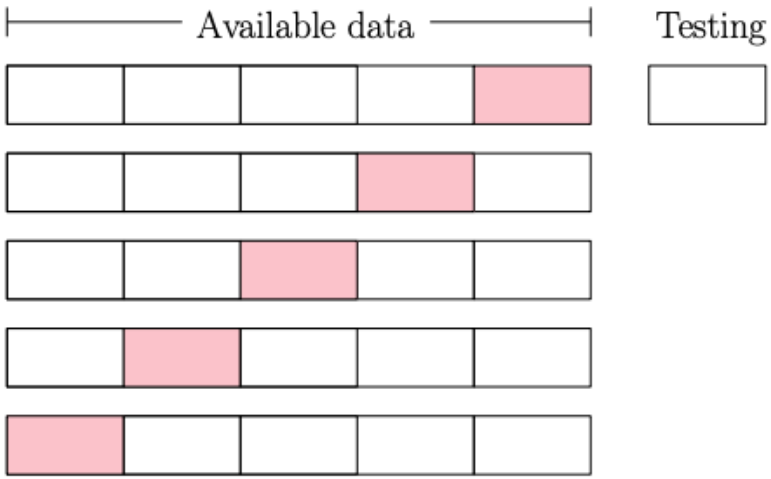


⚠ In theory, the test set can be used only once. In practice, this is unfortunately only sometimes the case. If the test data are seen multiple times, the estimation of the performance on unseen data is overestimated.

Moeel Selection with Cross-Validation

Use a maximal amount of training data and reduce the variability of the validation procedure.

- Divide the available data into k folds (typically $k = 5, 10$). Train/estimate all models k times, each time choosing a different fold as validation data (shaded below). Average the obtained k error measures.
- Can be applied to any learning method.



⚠ "Debugging" an implementation is often an art: on top of commonly found bugs, the learning method may not predict well enough with testing data.

Statistical Decision Theory

Provide a framework to analyze prediction models by viewing data as realizations of random variables from probability distributions.

- Let $X \in \mathcal{X} := \mathbb{R}^p$ and $Y \in \mathcal{Y} := \mathbb{R}$ denote the input vector and output variable respectively, and $P(X, Y)$ denote their joint distribution.
- Typical (but not always) to assume that the data arose from a statistical model of the form $Y = f(X) + \epsilon$ (additive error model), where the random error ϵ has mean zero and is independent of X.
- The goal is to find a (useful) approximator $\hat{f} : \mathcal{X} \to \mathcal{Y}$ that, for a new, unseen input X, can produce an acccurate* prediction $\hat{Y}$.

> What is the optimal prediction and performance given infinite data and infinite compute resource?

*In addition to **prediction**, we may also care about **inference**, where model interpretability is crucial to understand the relationship between X and Y.

Measuring Performance: Loss Functions

A **loss function** $L(y_{\text{true}}, y_{\text{predicted}})$, where $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, quantifies the penalty for errors in prediction.

Common Loss Functions:

- **Regression** $(\mathcal{Y} = \mathbb{R})$:
  - **Square Error:** $L(y, f(x)) = (y - f(x))^2$.
  - **Absolute Error:** $L(y, f(x)) = |y - f(x)|$.
  - Extension to $\mathcal{Y} = \mathbb{R}^q$: $\|y - f(x)\|_2^2$ (L_2 loss) and $\|y - f(x)\|_1$ (L_1 loss).
- **Classification** $(\mathcal{Y} = \{0, 1\})$:
  - **0-1 Loss:** $L(y, f(x)) = \mathbb{I}(y \neq f(x))$, where $\mathbb{I}(\cdot)$ is the indicator function.
  - **Cross-Entropy Loss:** For the two-class case, learn a probabilistic classifier that outputs $\hat{p}(x) = P(Y = 1|X = x)$:

$$L(y, \hat{p}(x)) = -[y \log(\hat{p}(x)) + (1 - y) \log(1 - \hat{p}(x))].$$

❓ How should we decide which loss function to use given a task?

The Ideal Goal: Minimizing True Risk

Since our data comes from a distribution P, we can define the **expected risk** of a function f as its expected loss over all possible data points from P.

Definition 1: Expected Risk

The expected risk of a function f is its average loss over the true distribution P:

$$R(f) = \mathbb{E}_{(X,Y)\sim P}[L(Y, f(X))] = \int_{\mathcal{X}\times\mathcal{Y}} L(y, f(x))\, dP(x, y).$$

The ultimate goal is to find the function f^* that minimizes this risk: $f^* = \arg\min_f R(f)$. This optimal function is called the **Bayes predictor***. The risk of Bayes predictors, $R(f^*)$, is called the **Bayes risk**, denoted as R^*.

*The Bayes predictor is not always unique, but all lead to the same Bayes risk (usually non-zero). The Bayes risk is the optimal performance.

Bayes Predictor for Regression

Theorem 1: Bayes Predictor for Squared Error Loss

For a regression problem with squared error loss, the optimal predictor is the conditional expectation of Y given X (also known as regression function):

$$f^*(x) = \mathbb{E}[Y|X = x],$$

and the Bayes risk is $R^* = \mathbb{E}_X[\text{Var}(Y|X)]$.

Proof.

See blackboard. □

Thus, the best prediction of Y at any point $X = x$ is the conditional mean, when best is measured by average squared error.

Bayes Predictor for Classification

Theorem 2: Bayes Predictor for 0-1 Loss

For a classification problem with 0-1 loss, the optimal classifier (the **Bayes classifier**) assigns the most probable class for a given x:

$$f^*(x) = \arg\max_{c_k \in \mathcal{G}} P(G = c_k | X = x).$$

Proof.

See Exercise 1. □

❓ What is the Bayes risk for the Bayes classifier? What if we use the cross-entropy loss instead?

See Proposition 2.1 in Bach for results for general loss functions.

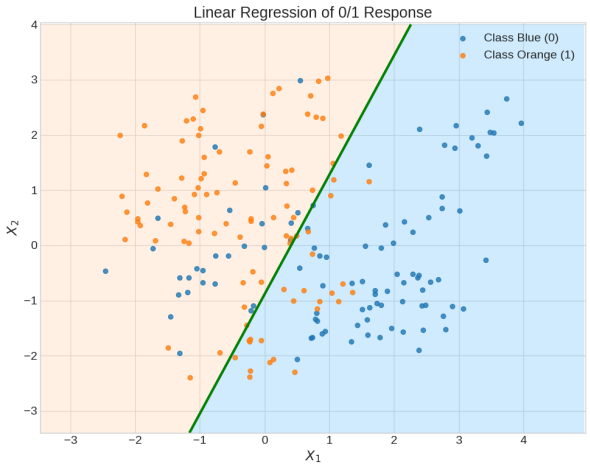
The Bias-Variance Tradeoff

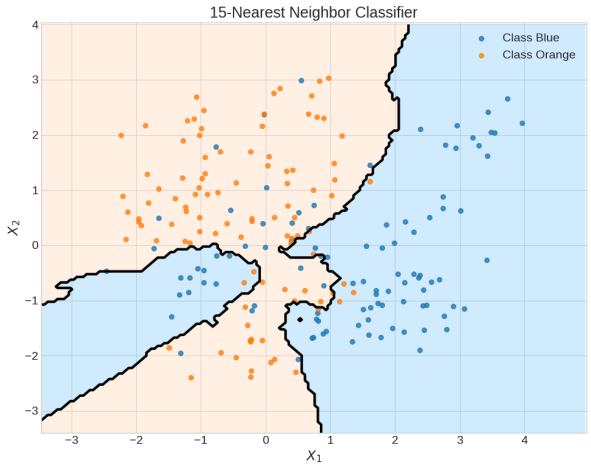
The ideal goal is to choose a function f that minimizes the average loss over the joint distribution of (X, Y). But would a learned model $\hat{f}$ generalize well to new input data?

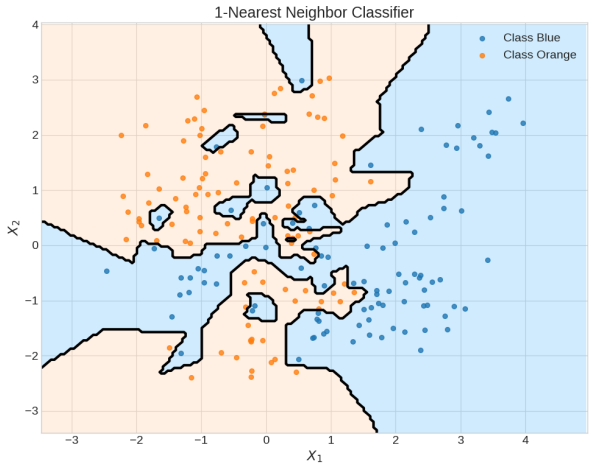
- Consider a regression model $Y = f(X) + \epsilon$, $\mathbb{E}[\epsilon|X] = 0$, $\text{Var}(\epsilon|X) = \sigma^2$.
- At a new test point x_0, the EPE (with squared error loss) for a predictor $\hat{f}$ can be decomposed into three components:

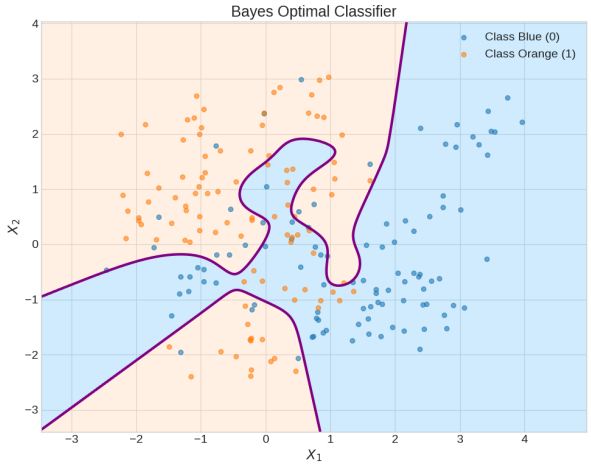
$$
\begin{aligned}
& EPE(\hat{f}(x_0)) \\
&= \mathbb{E}[(Y - \hat{f}(x_0))^2 | X = x_0] && (1) \\
&= \text{Var}(Y|X = x_0) + \text{Bias}^2(\hat{f}(x_0)) + \text{Var}(\hat{f}(x_0)) && (2) \\
&= \underbrace{\sigma^2}_{\textbf{Irreducible Error}} + \underbrace{[\mathbb{E}[\hat{f}(x_0)] - \hat{f}(x_0)]^2}_{\textbf{Bias}^2} + \underbrace{\mathbb{E}[\hat{f}(x_0) - \mathbb{E}[\hat{f}(x_0)]]^2}_{\textbf{Variance}}. && (3)
\end{aligned}
$$

Example: Binary Classification in 2D



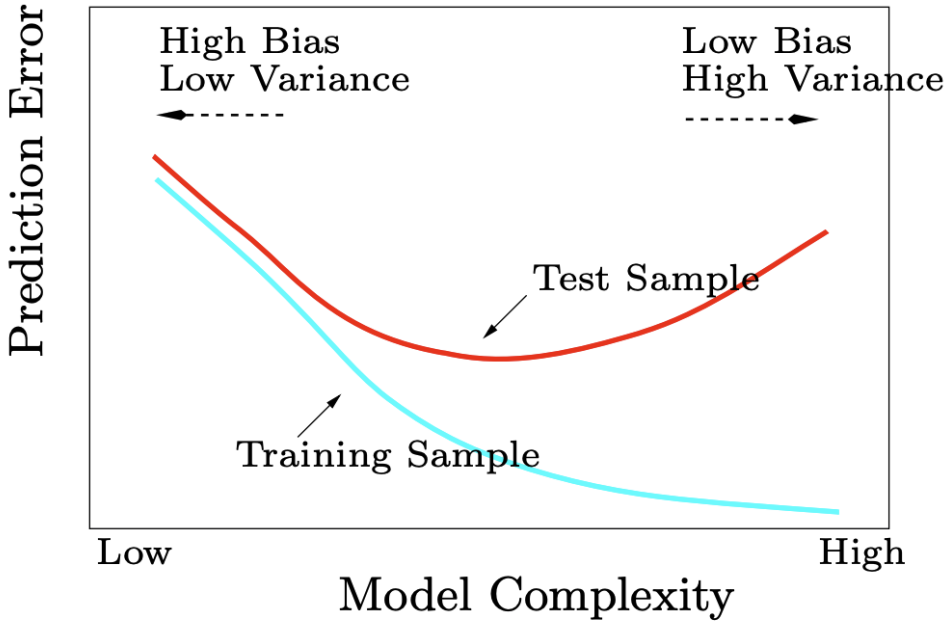






The Bias-Variance Tradeoff

- A simpler model (e.g., linear regression) typically has higher bias (may not capture the true relationship) but lower variance (predictions are stable across different training sets).
- A more complex model (e.g., 1-NN) typically has lower bias (can fit the training data very closely) but higher variance (predictions can vary significantly with different training sets).
- Effective **model selection** (e.g., cross-validation) involves finding an optimal balance in this trade-off to minimize the overall EPE.



A Practical Approach: Empirical Risk Minimization (ERM)

- In practice, we do not know the true distribution P, so we cannot compute the true risk $R(f)$.
- Instead, we use the training data as a proxy for P.

Definition 2: Empirical Risk

The empirical risk (or training error) is the average loss on the training data D_n:

$$\hat{R}_N(f) = \frac{1}{N} \sum_{n=1}^{N} L(y_n, f(x_n)).$$

The **Empirical Risk Minimization (ERM)** principle is to choose the function that minimizes this empirical risk.

⚠ Our method should aim to achieve a low test error but training error can be a poor estimate of test error.

Challenges with Function Approximation

We frame supervised learning as the task of finding a useful approximation $\hat{f}(x)$ to the true underlying regression function $f(x) = E[Y|X = x]$.

- **Challenge 1.** If we search for a minimizer of $\hat{R}_n(f)$ over the space of *all possible functions*, we run into **overfitting**. We could simply choose a function that memorizes the training data, for instance:

$$f(x) = \begin{cases} y_n & \text{if } x = x_n \text{ for some } n \in \{1, \dots, N\}, \\ \text{any value} & \text{otherwise.} \end{cases}$$

  This function achieves an empirical risk of 0 (for many loss functions), but it will likely generalize very poorly to new data.
- **Challenge 2.** The curse of dimensionality for local methods like k-nearest neighbors (see ESL Ch. 2.5). To maintain the same estimation accuracy as in low dimensions, the size of the training set must increase exponentially with the number of dimensions.

Constraining Complexity: Hypothesis Spaces

To combat overfitting, we restrict our search to a pre-defined class of functions $\mathcal{F}$, called a **hypothesis space**. The learning problem becomes finding the best function *within this class*:

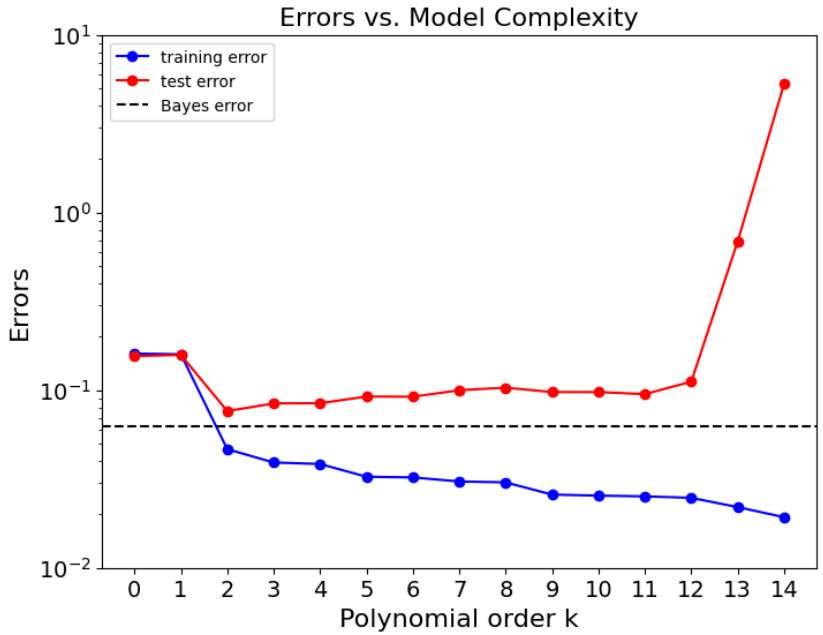
$$\hat{f}_N = \arg\min_{f \in \mathcal{F}} \hat{R}_N(f)$$

Bias-variance tradeoff: If $\mathcal{F}$ is too simple, the model may not be expressive enough to approximate the optimal f^* (high **bias**). If $\mathcal{F}$ is too complex, the model becomes too sensitive to training data and fits the noise in the data (high **variance**).

Example 2: Approximations by Polynomials in Regression

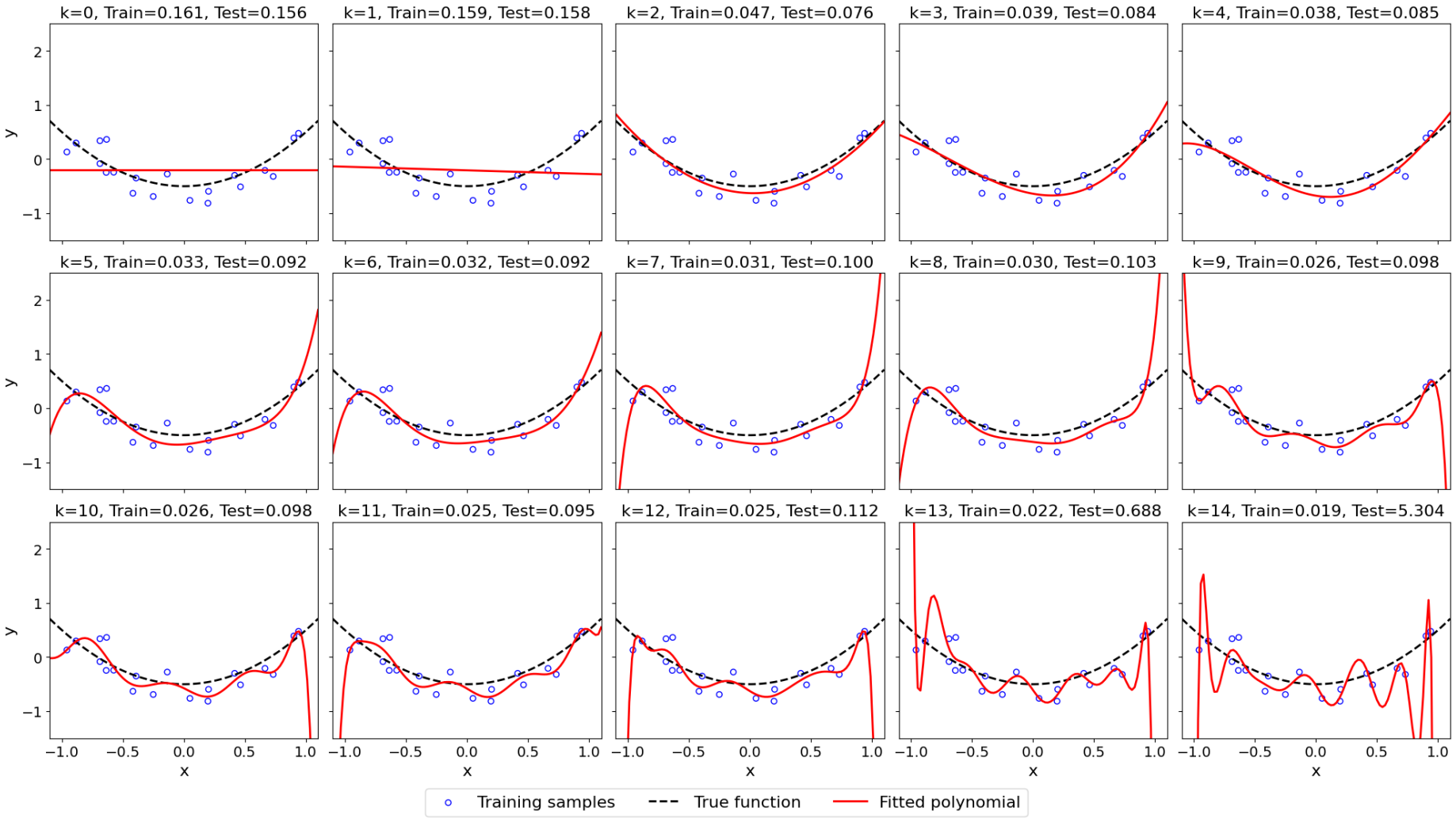
We consider $(x, y) \in \mathbb{R} \times \mathbb{R}$, with prediction functions that are polynomials of order k, from $k = 0$ (constant functions) to $k = 14$. This corresponds to linear regression with $f_\theta(x) = \theta^\top \phi(x)$, where $\phi(x) = (1, x, \ldots, x^k)^\top \in \mathbb{R}^{k+1}$.

The training error (square loss) is minimized with $N = 20$ samples.



- The data were generated with inputs uniformly distributed on $[-1, 1]$ and outputs given by the quadratic function $f(x) = x^2 - \frac{1}{2}$ plus independent additive Gaussian noise with standard deviation $1/4$.

Empirical Results



Approximation-Estimation Decomposition

Why doesn't our learned predictor $\hat{f}_N$ achieve the Bayes risk R^*? The gap, or **excess risk**, can be decomposed.

Proposition 1: Decomposition of Excess Risk

$$\underbrace{R(\hat{f}_N) - R^*}_{\text{Excess Risk}} = \underbrace{\left(\inf_{f \in \mathcal{F}} R(f) - R^* \right)}_{\textbf{Approximation Error}} + \underbrace{\left(R(\hat{f}_N) - \inf_{f \in \mathcal{F}} R(f) \right)}_{\textbf{Estimation Error}} \geq 0.$$

Approximation Error (Bias)

- Due to the choice of model $\mathcal{F}$.
- How well can the *best* function in our class approximate the true Bayes predictor?
- A more complex/flexible class $\mathcal{F}$ leads to a smaller approximation error.

Estimation Error (Variance)

- Due to having finite data.
- How much our estimate $\hat{f}_N$ suffers because we only have N samples, not the true distribution?
- A more complex class $\mathcal{F}$ is harder to estimate, leading to a larger estimation error.

Examples of Hypothesis Spaces

- **Linear models:** $\mathcal{F} = \{f_\theta(x) = x^T\theta \mid \theta \in \mathbb{R}^p\}$ (# parameters $= p$).
- **Basis functions expansion:** $\mathcal{F} = \{f_\theta(x) = \sum_{m=1}^{M} h_m(x)\theta_m\}$, where the $h_m(x)$ can be polynomial terms, trigonometric expansions, or nonlinear transformations like the sigmoid function used in neural networks.
- **Regularization:** Introduce penalties on functions to enforce smoothness and prevent overfitting. An example is smoothing spline, which minimizes the RSS plus a penalty on the function's roughness. For a 1D case:

$$\min_f \sum_{n=1}^{N} (y_n - f(x_n))^2 + \lambda \int [f''(t)]^2 dt.$$

- **Kernel Methods and Local Regression:** predict locally by taking weighted averages of nearby data points. The "local" neighborhood is defined by a kernel function (e.g., Nadaraya-Watson), or adaptively.

"No Free Lunch" Principle

 **Learning is impossible without making assumptions** (see Ch 2.5 in Bach).

 No single learning method dominates all others across all possible datasets. Different methods perform best on different datasets. Therefore, selecting the best approach for a given dataset is crucial and challenging.

This is where theory can be useful to understand when a method is supposed to work or not.

Summary of Notations

Notation	Description
N	Number of data points (samples).
p	Number of predictors (features).
$x_n \in \mathbb{R}^p$	Input vector for the n-th data point.
$y_n \in \mathbb{R}$	Response value for the n-th data point.
D_N	The training dataset $\{(x_n, y_n)\}_{n \in \{1,\ldots,N\}}$.
$\mathcal{X}$	Input space (e.g., $\mathbb{R}^p$).
$\mathcal{Y}$	Output space (e.g., $\mathbb{R}$ for regression, $\{0, 1\}$ for classification).
X, Y	Random variables for an input and its corresponding output.
$P(X, Y)$	The (unknown) joint probability distribution of the data.
$f(X)$	A model/prediction function that maps inputs from $\mathcal{X}$ to $\mathcal{Y}$.
$\hat{f}_N(X)$ / $\hat{f}(X)$	An estimated prediction function, learned from the training dataset.
$f^*(X)$	The Bayes predictor, the theoretically optimal function that minimizes risk.
$L(Y, f(X))$	A loss function measuring the error between a true output Y and a prediction $f(X)$.
$R(f)$	The risk (or expected loss): $R(f) = \mathbb{E}[L(Y, f(X))]$.
$\hat{R}_N(f)$ / $\hat{R}(f)$	The empirical risk: an average of the loss over the training data, $\frac{1}{N}\sum_{n=1}^{N} L(y_n, f(x_n))$.
$\mathbb{E}[\cdot]$	Expectation operator.
$\mathrm{Var}(\cdot)$	Variance operator.

Exercises

Exercise 1

1. Generalize the cross-entropy loss to the multi-class setting and use it to formulate an ERM framework for the multi-class classification task.
2. Show that minimizing the cross-entropy loss for a single data point is equivalent to maximizing the predicted probability of the correct class.
3. What is the Bayes predictor and the Bayes risk if we replace the L_2 loss with the L_1 loss $\mathbb{E}|Y - f(X)|$?
4. Prove Theorem 2 and answer the questions that follows it.
5. For the k-NN regression estimator, derive its bias–variance decomposition, and use your result to explain how increasing or decreasing k affects the estimator's bias and variance. Validate this by plotting test error vs. N/k (degrees of freedom) for the classification task (you can build on the provided code).