Robotics & Internet-of-Things

جامعة الأمير سلطان PRINCE SULTAN UNIVERSITY

CS316: INTRODUCTION TO AI AND DATA SCIENCE

CHAPTER 7
STATISTICAL LEARNING

LECTURE 1
FUNDAMENTAL STATISTICAL THEOREMS IN DATA SCIENCE

Prof. Anis Koubaa

February 2024

www.riotu-lab.org

Outline

- Introduction to Statistical Learning
- Fundamental Statistical Theorem in Data Science
  - Bayes Theorem
  - Law of Large Numbers
  - Central Limit Theorem
  - Confidence Intervals

CHAPTER 7
STATISTICAL LEARNING

Introduction

Anis Koubaa

CS316: INTRODUCTION

TO AI AND DATA SCIENCE

What is Statistical Learning?

- **Statistical Learning: A Framework for Data-Driven Understanding**
  - A subfield of machine learning
  - Utilizes statistical principles to **model relationships within data**.
  - It focuses on the **discovery of patterns** and the ability to **generate predictions**.
- **Key Goals**
  - **Inference:** Deriving **insights** and understanding the underlying structure of data.
  - **Prediction:** **Forecasting** or **estimating future** outcomes based on observed data.
- **Essential Tools**
  - Probability theory
  - Statistical models (e.g., linear regression, decision trees)
  - Algorithms for optimization and fitting models to data

Role in Machine Learning and AI

- **Core to ML Algorithms:** Statistical Learning is essential for algorithms like decision trees, neural networks, support vector machines.
- **Functions in ML:**
  - **Model Selection & Evaluation:** Statistical criteria guide the selection and assessment of models.
  - **Data Understanding:** Statistical analysis reveals data structures and relationships.
- **Framework for Validation:** Statistics offer a lens to understand and validate algorithm outcomes.
- **Boosted by Big Data:** Advanced statistical methods expand capabilities in handling big data and complex structures.
- **Foundational in Data Science:** Indispensable for deriving insights, a pillar for data engineers and scientists.

CHAPTER 7
STATISTICAL LEARNING

LECTURE 1
FUNDAMENTAL STATISTICAL THEOREMS IN DATA SCIENCE

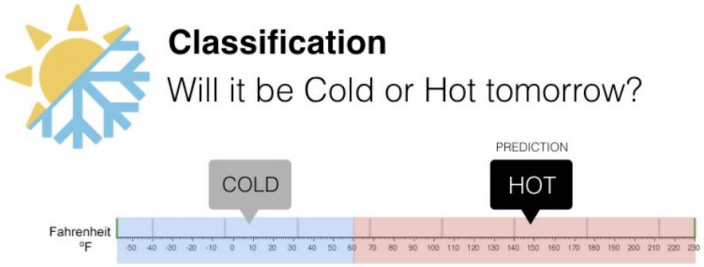
Data Categories

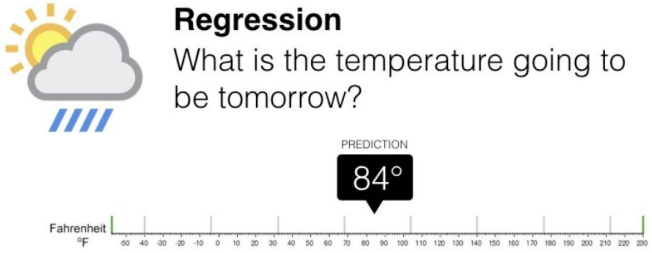
CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Distinguishing Types of Data

- **Qualitative Data - Categorical Data**
  - Describes characteristics or categories.
  - Non-numerical in nature.
  - Example: City, weather conditions (sunny, cloudy, rainy)
- **Quantitative Data - Numerical Data**
  - Numeric measurements or counts.
  - Represents quantities or magnitudes
  - Example: Temperature, rainfall, wind speed.





Discrete vs. Continuous Data

- **Discrete Data**:
  - **Definition**: Takes distinct, separate values (e.g., days of the month).
  - **Representation**: $D = \{d_1, d_2, ..., d_m\}$, specific integers only.
  - **Example**: 'Day' variable, limited to integer values like 1, 2, ..., 31.
- **Continuous Data**:
  - **Definition**: Can assume any value within a range, including fractions.
  - **Representation**: $C = \{c \in \mathbb{R} | a \leq c \leq b\}$, a continuous range.
  - **Example**: 'Temperature', varies within a range, e.g., 17.68°C to 33.01°C.

$$D = \{d_1, d_2, \ldots, d_m\}$$

$$C = \{c \in R \mid a \leq c \leq b\}$$

CHAPTER 7
STATISTICAL LEARNING

LECTURE 1
FUNDAMENTAL STATISTICAL THEOREMS IN DATA SCIENCE

Population vs. Sample

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Population

- **Population Example**: All temperature readings in Riyadh for the year 2018.
  - **Representation**: $P_{\text{temp}} = \{T_1, T_2, ..., T_N\}$, every temperature reading recorded every hour of every day in 2018.
- **Sample Example**: Temperature readings from a specific day or month.
  - **Representation**: $S_{\text{temp}} = \{T_{\text{Nov6}_1}, T_{\text{Nov6}_2}, ..., T_{\text{Nov6}_{24}}\}$, readings from November 6, 2018.

- **Mathematical Notation:**
  - $P = \{x_1, x_2, \ldots, x_N\}$
  - Where x_i is an individual member, and N is the total count.
- **Riyadh Weather Example:** The population is all temperature readings for Riyadh in 2018.

$$P = \{x_1, x_2, \ldots, x_N\}$$

Sample

- **Population Example**: All temperature readings in Riyadh for the year 2018.
  - **Representation**: $P_{\text{temp}} = \{T_1, T_2, ..., T_N\}$, every temperature reading recorded every hour of every day in 2018.
- **Sample Example**: Temperature readings from a specific day or month.
  - **Representation**: $S_{\text{temp}} = \{T_{\text{Nov6}_1}, T_{\text{Nov6}_2}, ..., T_{\text{Nov6}_{24}}\}$, readings from November 6, 2018.

- **Mathematical Notation:**
  - $S = \{x_1, x_2, \ldots, x_n\}$
  - Where $n<N$ indicating a smaller group selected from the population.
- **Riyadh Weather Example:** A sample could be temperature readings **for a specific month** or day in 2018.

$$S = \{x_1, x_2, \ldots, x_n\}$$

Why Samples Matter?

- **Practicality:** Analyzing entire populations is often costly, time-consuming, or impossible.
  - Training a computer vision neural network on a set of images (e.g. face recognition)
  - Estimate the price of a real-estate in a certain country
- **Representativeness:** A well-chosen sample can accurately reflect the characteristics of the **larger population**.
- **Inference:** Statistical methods allow us to draw conclusions about the whole population based on the sample we study.

CHAPTER 7
STATISTICAL LEARNING

LECTURE 1
FUNDAMENTAL STATISTICAL THEOREMS IN DATA SCIENCE

Fundamental Statistical Theorems in Data Science

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Fundamental Statistical Theorems in Data Science

- **Bayes' Theorem:**
  - **Purpose:** Enables updating of hypothesis probabilities with new evidence.
  - **Impact:** Fundamental for predictive modeling, inferential statistics and decision-making.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

- **Law of Large Numbers (LLN):**
  - **Principle:** Sample means converge to the true population mean as sample size increases.
  - This underpins the reliability of insights we draw from data.
  - **Application:** Ensures reliability of empirical averages in large datasets.

$$\lim_{n\to\infty} P\left(\left|\bar{X} - \mu\right| > \epsilon\right) = 0$$

- **Central Limit Theorem (CLT):**
  - **Key Insight:** Distribution of sample means tends towards a normal distribution as sample size grows, regardless of the population's original distribution.
  - This is fundamental for hypothesis testing and confidence intervals.
  - **Significance:** Crucial for inferential statistics, allowing for the application of normal distribution in various statistical models.

$$\lim_{n\to\infty} P\left(a \leq \frac{1}{\sqrt{n}} \frac{\bar{X} - \mu}{\sigma} \leq b\right) = \int_a^b \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}\, dx$$

These theorems enable data scientists to confidently make inferences about populations based on samples.

They form the basis for predictive modeling, decision-making under uncertainty, and robust statistical analysis.

CHAPTER 7
STATISTICAL LEARNING

LECTURE 1
FUNDAMENTAL STATISTICAL THEOREMS IN DATA SCIENCE

BAYES THEOREM

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Bayes' Theorem: Updating Beliefs with Data

- **Definition**: A fundamental result in probability theory that updates the likelihood of an event based on new evidence.
- **Formula**: $P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$
  - Where:
    - $P(A|B)$: Probability of event A given B is true.
    - $P(B|A)$: Probability of event B given A is true.
    - $P(A)$: Probability of event A.
    - $P(B)$: Probability of event B.
- **Significance in Data Science**:
  - **Predictive Modeling**: Enables probabilistic predictions incorporating new data.
  - **Hypothesis Testing**: Fundamental for updating model predictions based on evidence.
  - **Machine Learning**: Underpins many algorithms, especially in areas requiring continual learning from data.

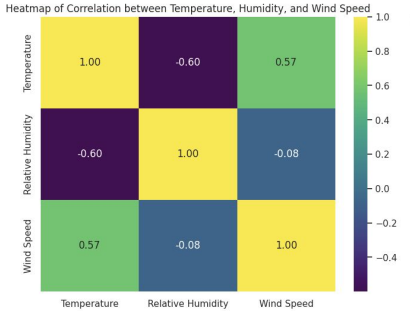
Bayes' Theorem in Action: Predicting Weather Patterns

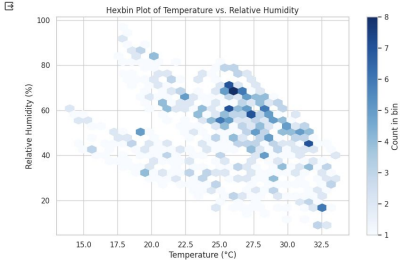
- **Objective**: Calculate $P(A|B)$ - Probability of temperature > 20°C given relative humidity > 40%.
- **Defining Events**:
  - **A**: Temperature is above 20°C.
  - **B**: Relative humidity is above 40%.

- **Bayes' Theorem**: $P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$
  - $P(A|B)$: Probability of A given B, the target calculation.
  - $P(A)$: Prior probability of A (temperature > 20°C).
  - $P(B|A)$: Likelihood of B given A (humidity > 40% when temperature > 20°C).
  - $P(B)$: Marginal probability of B (humidity > 40%).

- **Prior Probability:** The prior probability reflects our initial belief about the likelihood of an event occurring before considering any new evidence.
- **Marginal Probability:** The marginal probability represents the overall probability of an event occurring across all possible values of another related variable.

	A	B	C	D	E	F
1	Year	Month	Day	Hour	Temperature	Relative Humidity
2	2018	11	6	0	21.65	38
3	2018	11	6	1	20.94	40
4	2018	11	6	2	20.29	41
5	2018	11	6	3	19.67	43
6	2018	11	6	4	18.76	45
7	2018	11	6	5	18.25	47
8	2018	11	6	6	17.68	48
9	2018	11	6	7	19.63	42
10	2018	11	6	8	24.18	31





Bayes' Theorem in Action: Predicting Weather Patterns

- **Calculations**:
  - **Prior Probability** ($P(A)$): $\frac{n_A}{n_{\text{total}}}$
    - Where n_A = Number of days with temperature > 20°C, n_{total} = Total number of observed days.
- **Likelihood** ($P(B|A)$): $\frac{n_{AB}}{n_A}$
  - Where n_{AB} = Number of days with both conditions met (temp > 20°C and humidity > 40%).
- **Marginal Probability** ($P(B)$): $\frac{n_B}{n_{\text{total}}}$
  - Where n_B = Number of days with humidity > 40%.
- **Updated Probability**:
  $P(A|B) = \frac{n_{AB} \cdot n_{\text{total}}}{n_A \cdot n_B}$
  - Simplifies to a formula using observed frequencies to estimate $P(A|B)$.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Bayes' Theorem in Python

```python
import pandas as pd

# Load the dataset
df = pd.read_csv('https://www.riotu-lab.org/cs313/weather.csv')

# Calculate the prior probability P(A): P(Temp > 20°C)
prior_prob_A = len(df[df['Temperature'] > 20]) / len(df)
print(f"prior_prob_A: {prior_prob_A:.4f}")

# Calculate the marginal probability P(B): P(Humidity > 40%)
marginal_prob_B = len(df[df['Relative Humidity'] > 40]) / len(df)
print(f"marginal_prob_B: {marginal_prob_B:.4f}")

# Calculate the likelihood P(B|A): P(Humidity > 40% | Temp > 20°C)
# This is the number of times humidity is > 40% when temp is > 20°C divided by the number of times temp is > 20°C
likelihood_B_given_A = len(df[(df['Temperature'] > 20) & (df['Relative Humidity'] > 40)]) / len(df[df['Temperature'] > 20])
print(f"likelihood_B_given_A: {likelihood_B_given_A:.4f}")

# Apply Bayes' Theorem to find P(A|B): P(Temp > 20°C | Humidity > 40%)
# P(A|B) = (P(B|A) * P(A)) / P(B)
posterior_prob_A_given_B = (likelihood_B_given_A * prior_prob_A) / marginal_prob_B
print(f"posterior_prob_A_given_B: {posterior_prob_A_given_B:.4f}")

print(f"The updated probability that the temperature is above 20°C given that the relative humidity is above 40% is: {posterior_prob_A_given_B:.4f}")
```

```
prior_prob_A: 0.7583
marginal_prob_B: 0.5333
likelihood_B_given_A: 0.4469
posterior_prob_A_given_B: 0.6354
The updated probability that the temperature is above 20°C given that the relative humidity is above 40% is: 0.6354
```

CHAPTER 7
STATISTICAL LEARNING

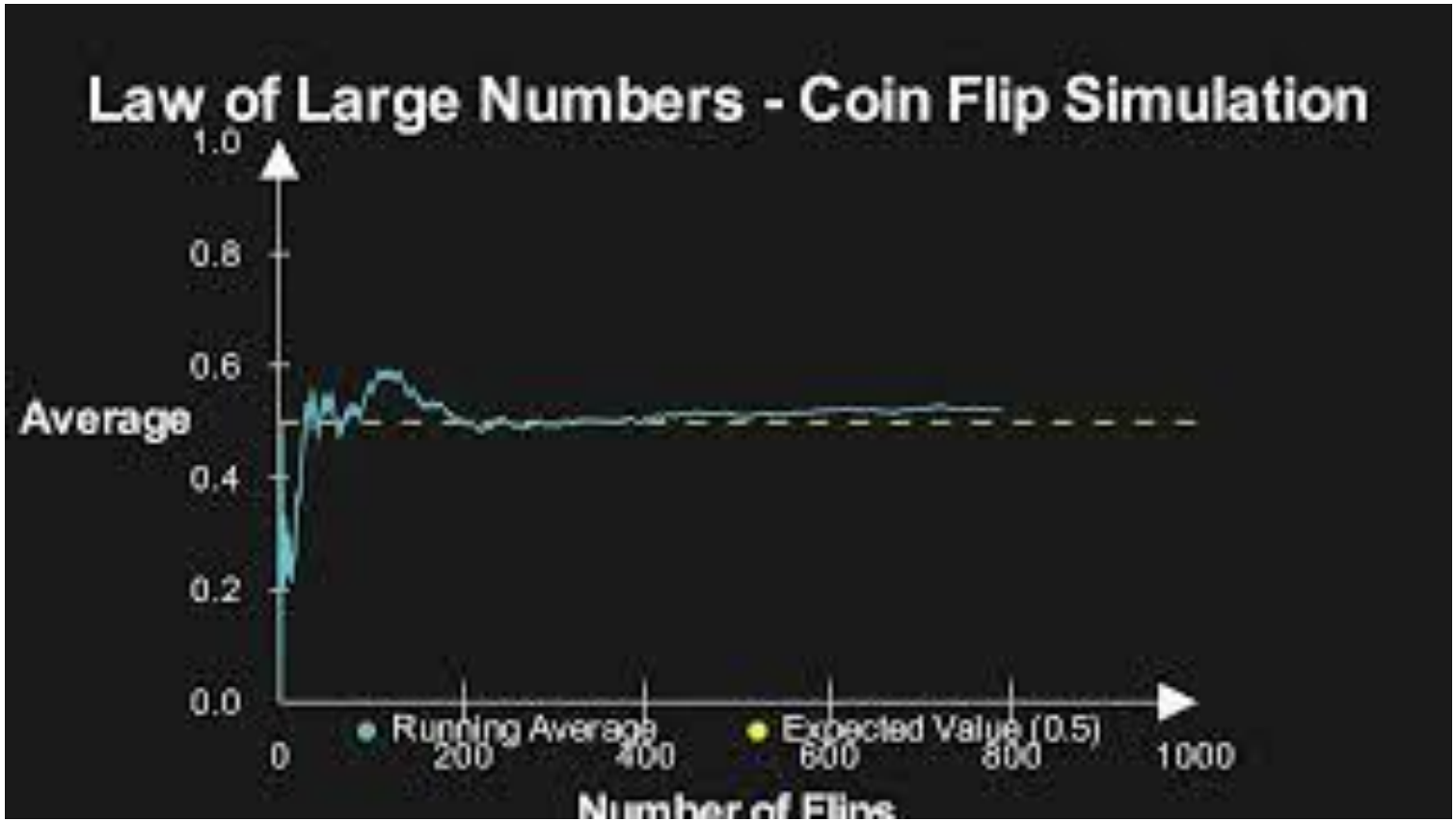
LECTURE 1
FUNDAMENTAL STATISTICAL THEOREMS IN DATA SCIENCE

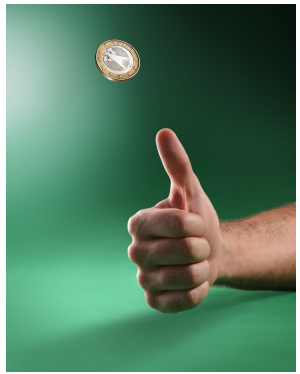
LAW OF LARGE NUMBERS

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Law of Large Numbers





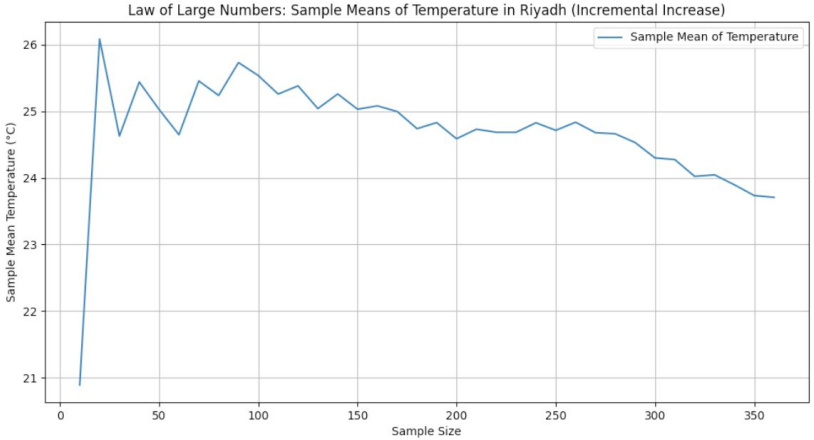
1: Head
0: Tail

Law of Large Numbers

- **Definition:** The Law of Large Numbers (LLN) ensures that the average of results from a large number of trials converges to the expected mean as trials increase.

$$\lim_{n \to \infty} P\left(\left|\bar{X} - \mu\right| > \epsilon\right) = 0$$

- **Key Points:**
  - **Convergence:** Sample averages approximate the population mean with a large number of trials.
  - **Foundation:** Essential for validating statistical models and results in data science.
  - **Applications:** Underlies inferential statistics, enabling population generalizations from sample data.
- **Two Forms:**
  - **Weak Law (WLLN):** Convergence in probability to the expected value as sample size increases.
  - **Strong Law (SLLN):** Almost sure convergence to the expected value, regardless of sample size growth.
- **Significance:** Validates that, given sufficient data, sample statistics can reliably estimate population parameters, critical for data-driven decision-making.



The Laws of Large Numbers in Data Science

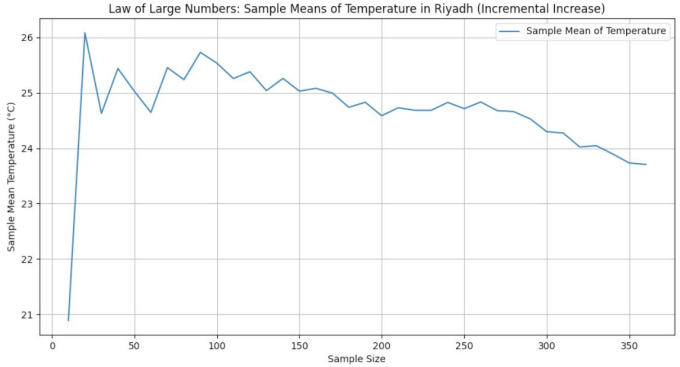
- **Weak Law of Large Numbers (WLLN)**:
  - **Mathematical Insight**: As sample size (n) increases, $P(|\bar{X}_n - \mu| > \epsilon) \to 0$, meaning the chance the sample mean significantly deviates from the population mean diminishes.
  - **Practical Implication**: Ensures the sample mean is a reliable estimator of the population mean for large n, but doesn't guarantee absolute convergence to μ for every sequence.
- **Strong Law of Large Numbers (SLLN)**:
  - **Guarantee**: Asserts $\bar{X}_n$ will almost surely (with probability 1) converge to μ as $n \to \infty$, a stronger form of convergence compared to the WLLN.
  - **Absolute Convergence**: Indicates that the sample mean will definitely approximate the population mean given an infinitely large sample size.
- **Comparison**:
  - **WLLN vs. SLLN**: WLLN suggests likelihood of close approximation to μ with large samples but doesn't ensure almost sure convergence, making it a "weaker" assurance than the SLLN.
  - **Application**: WLLN is often sufficient for practical statistical work, offering a balance between mathematical rigor and ease of proof.

The Laws of Large Numbers in Data Science

- **Strong Law of Large Numbers (SLLN)**:
  - **Definition**: For i.i.d. random variables $X_1, X_2, \ldots, X_n$ with mean μ, the SLLN states that $P(|\bar{X} - \mu| > \epsilon) \to 0$ as $n \to \infty$, ensuring the sample mean $\bar{X}$ almost surely converges to μ.
  - **Significance**: Guarantees that sample averages will almost surely match the population mean with large enough samples, crucial for empirical research and statistical modeling accuracy.
- **Weak Law of Large Numbers (WLLN)**:
  - **Definition**: Similarly, for i.i.d. random variables $X_1, X_2, \ldots, X_n$ with mean μ, the WLLN asserts that $P(|X_n - \mu| > \epsilon) \to 0$ as $n \to \infty$, indicating convergence in probability of the sample average to the expected value.
  - **Mathematical Expression**: $\lim_{n\to\infty} P(|\bar{X}_n - \mu| > \epsilon) = 0$.
- **Mathematical Significance**:
  - The SLLN and WLLN underpin the reliability of using sample statistics to estimate population parameters, affirming that larger sample sizes lead to more accurate mean estimations.
  - Both laws provide a foundation for statistical modeling, validating the use of sample means as accurate estimators of population means in data science.

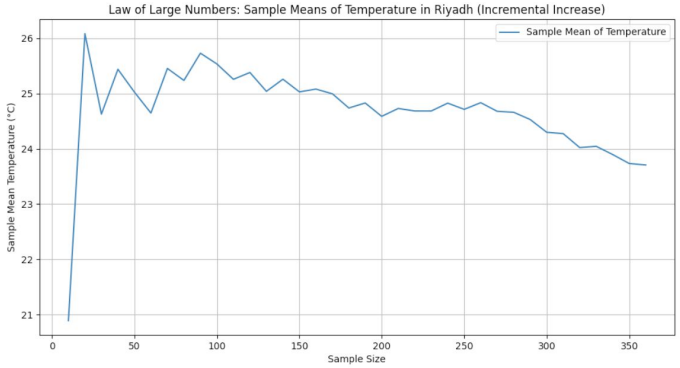
Law of Large Numbers & Temperature Data Analysis

- **Scenario**: Analyzing temperature readings in Riyadh over several years with a dataset $\{T_1, T_2, ..., T_n\}$.
- **Application of LLN**:
  - **Sample Expansion**: Increasing sample size (n) of temperature readings.
  - **Expected Outcome**: Average temperature $\bar{T} = \frac{1}{n}\sum_{i=1}^{n} T_i$ converges towards Riyadh's true average temperature.
- **Strong Law of Large Numbers (SLLN)**:
  - **Convergence**: With continuous sample size increase, $\bar{T}$ will almost surely converge to the city's true mean temperature.
  - **Implication**: Absolute certainty in convergence with an infinite sample size.
- **Weak Law of Large Numbers (WLLN)**:
  - **Convergence**: For a sufficiently large sample, $\bar{T}$ will be very close to the true mean with high probability.
  - **Distinction**: High probability of convergence, but not as absolute as SLLN guarantees.
- **Conclusion**:
  - **SLLN vs. WLLN**: SLLN assures almost sure convergence to true mean with infinite data, while WLLN suggests high likelihood of approximation with large, finite samples.
  - **Statistical Inference**: Both laws underpin the reliability of statistical estimates from large datasets, essential for accurate climate analysis and modeling.



Law of Large Numbers & Temperature Data Analysis

```python
import pandas as pd
import matplotlib.pyplot as plt

weather_data = pd.read_csv('https://www.riotu-lab.org/cs313/weather.csv')

# Function to calculate the sample mean of temperature for different sample sizes
def calculate_sample_means_incremental(data, start_sample_size, increment):
    sample_means = []
    for sample_size in range(start_sample_size, len(data) + 1, increment):
        sample_mean = data['Temperature'][:sample_size].mean()
        sample_means.append((sample_size, sample_mean))
    return sample_means

# Calculate sample means starting from 10 entries and increasing incrementally
start_sample_size = 10
increment = 10  # Incremental step
incremental_sample_means = calculate_sample_means_incremental(weather_data, start_sample_size, increment)

# Creating a DataFrame for the sample means
sample_means_df = pd.DataFrame(incremental_sample_means, columns=['Sample Size', 'Sample Mean Temperature'])

# Display the DataFrame
print(sample_means_df)

# Preparing data for plotting
sample_sizes = sample_means_df['Sample Size']
sample_means_values = sample_means_df['Sample Mean Temperature']

# Plotting the results
plt.figure(figsize=(12, 6))
plt.plot(sample_sizes, sample_means_values, label='Sample Mean of Temperature')
plt.xlabel('Sample Size')
plt.ylabel('Sample Mean Temperature (°C)')
plt.title('Law of Large Numbers: Sample Means of Temperature in Riyadh (Incremental Increase)')
plt.legend()
plt.grid(True)
plt.show()
```



	Sample Size	Sample Mean Temperature
0	10	20.891000
1	20	26.084000
2	30	24.627667
3	40	25.439000
4	50	25.026400
5	60	24.647500
6	70	25.454429
7	80	25.237500
8	90	25.730667
9	100	25.535400
10	110	25.257727
11	120	25.381333
12	130	25.040000
13	140	25.259786
14	150	25.030467
15	160	25.080688
16	170	24.995529
17	180	24.737444
18	190	24.829474
19	200	24.587250
20	210	24.730048
21	220	24.684909
22	230	24.684043
23	240	24.826750
24	250	24.713920
25	260	24.835154
26	270	24.678667
27	280	24.661000
28	290	24.530103
29	300	24.299133
30	310	24.274742
31	320	24.022531
32	330	24.045515
33	340	23.896235
34	350	23.734143
35	360	23.707583

CHAPTER 7
STATISTICAL LEARNING

LECTURE 1
FUNDAMENTAL STATISTICAL THEOREMS IN DATA SCIENCE

CENTRAL LIMIT THEOREM (CLT)

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

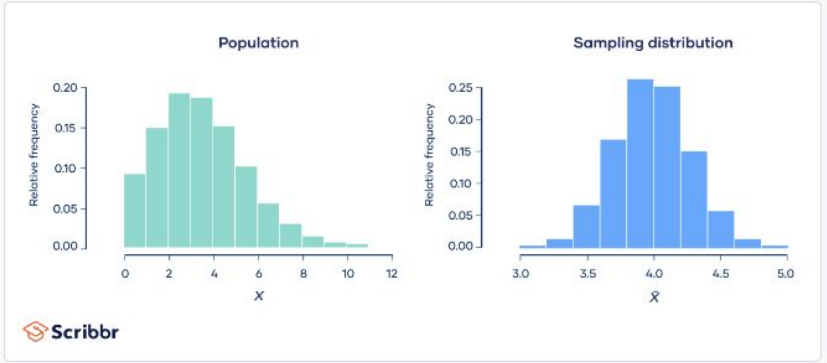
Central Limit Theorem (CLT)

- **Definition:**
  - The CLT states that if you take **sufficiently large, random samples** from a population with a finite mean (μ) and variance (σ^2), the **distribution of the sample means will be approximately normal**.
  - This holds true regardless of the shape of the original population distribution.
  - The larger the sample size, the closer the approximation to a normal distribution.
- **Key Components:**
  - **Large Number of Observations:** The theorem applies when dealing with a sufficiently large sample size.
  - **Finite Mean and Variance:** Each random variable must have a finite mean (μ) and variance (σ_2).
- **Significance in Data Science:**
  - **Justification for Normal Distribution:** Allows the use of normal distribution in statistical methods, even if the original data isn't normally distributed.
  - **Foundation for Statistical Methods:** Vital for hypothesis testing, constructing confidence intervals, and much of the inferential statistics toolkit.

$$\frac{X_1 + \cdots + X_n}{\sqrt{n}} \longrightarrow \mathcal{N}(0, \sigma^2). \qquad f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Example: Central limit theorem

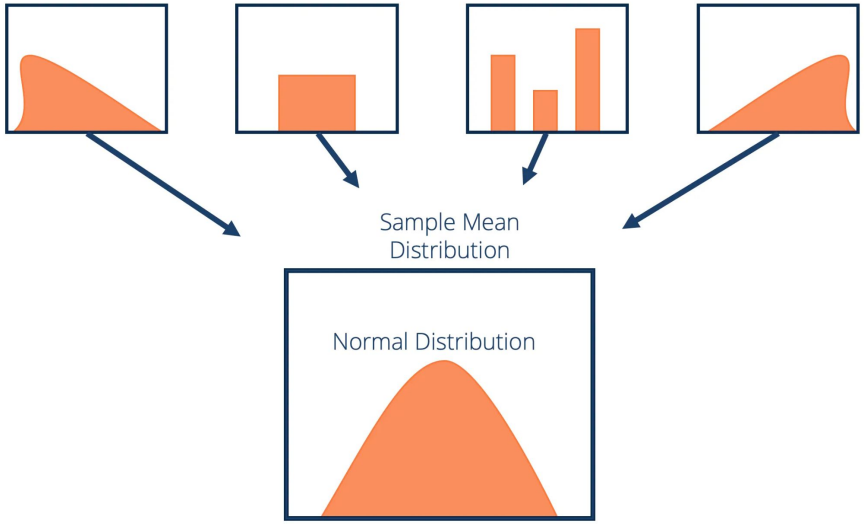
A **population** follows a **Poisson distribution** (left image). If we take 10,000 **samples** from the population, each with a sample size of 50, the sample means follow a normal distribution, as predicted by the **central limit theorem** (right image).



https://www.scribbr.com/statistics/central-limit-theorem/

The Power of Central Limit Theorem (CLT) in Diverse Data Analysis

- **Universal Insight:** Regardless of the original data's distribution, the mean of a sufficiently large sample will approximate a normal distribution.
- **Application Across Data Types:**
  - Temperature readings from various cities.
  - Survey responses across demographic groups.
  - Sales figures from multiple outlets.
- **Simplifying Data Complexity:**
  - Transforms diverse and variable data into a coherent, analyzable format.
  - Facilitates the application of well-understood statistical methods.
- **Foundation for Inference:**
  - Enables reliable inferences about a population based on sample data.
  - Critical for hypothesis testing and constructing confidence intervals.
- **Key to Statistical Methods:**
  - Makes the normal distribution a versatile tool for data analysis.
  - Opens up pathways for deeper understanding and accurate predictions.

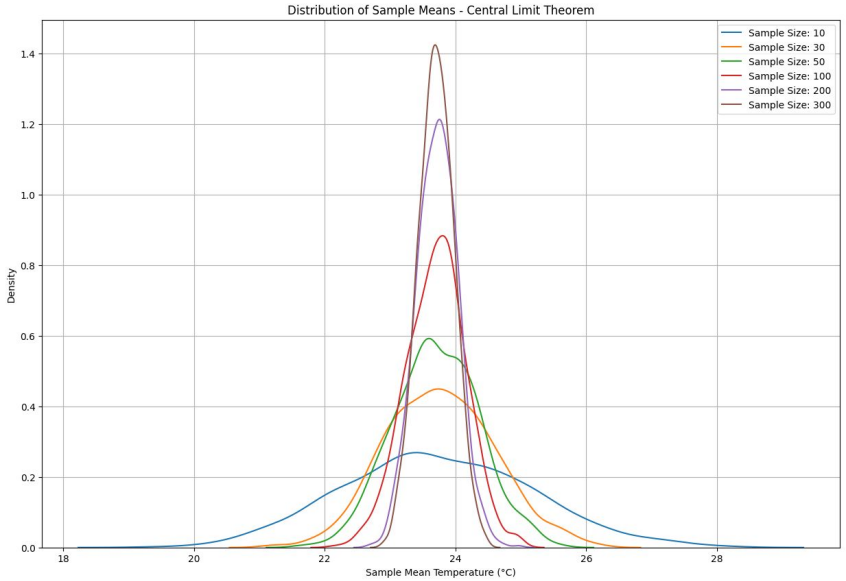


https://corporatefinanceinstitute.com/resources/data-science/central-limit-theorem/

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Central Limit Theorem: Real-World Application with Weather Data

```python
import pandas as pd
from numpy.random import choice
import seaborn as sns
import matplotlib.pyplot as plt


weather_data = pd.read_csv('https://www.riotu-lab.org/cs313/weather.csv')
# Function to compute sample means for different sample sizes
def compute_sample_means_for_CLT(data, sample_sizes, num_samples=1000):
    sample_means = {size: [] for size in sample_sizes}
    for size in sample_sizes:
        for _ in range(num_samples):
            sample = choice(data['Temperature'], size=size)
            sample_means[size].append(sample.mean())
    return sample_means

# Define sample sizes for the experiment
sample_sizes = [10, 30, 50, 100, 200]

# Compute sample means
sample_means_CLT = compute_sample_means_for_CLT(weather_data, sample_sizes)

# Plotting the distributions of the sample means
plt.figure(figsize=(15, 10))
for size in sample_sizes:
    sns.distplot(sample_means_CLT[size], label=f'Sample Size: {size}', kde=True, hist=False)

plt.xlabel('Sample Mean Temperature (°C)')
plt.ylabel('Density')
plt.title('Distribution of Sample Means - Central Limit Theorem')
plt.legend()
plt.grid(True)
plt.show()

```

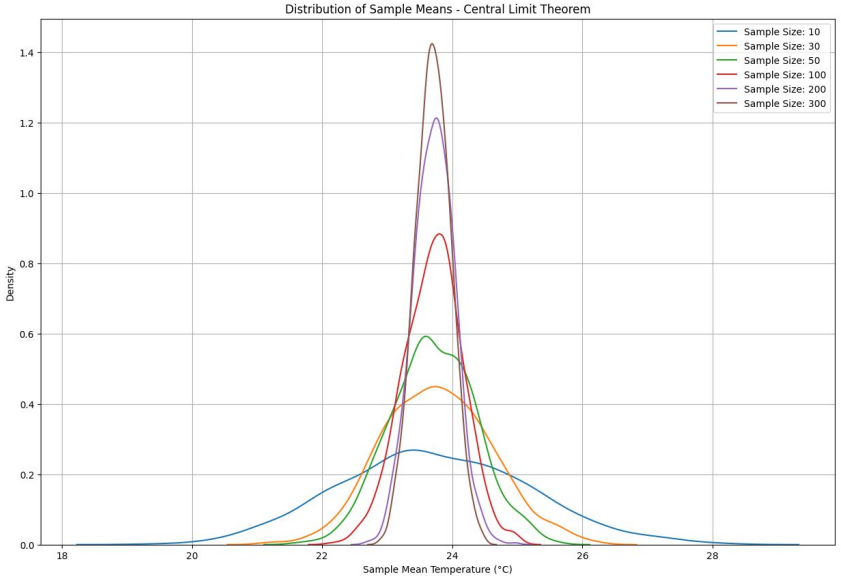


Central Limit Theorem: Real-World Application with Weather Data

- **Overview:** Multiple random samples (sizes: 10, 30, 50, 100, 200) analyzed to observe the Central Limit Theorem (CLT) effects on temperature data.

- **Key Findings:**
  - **Normal Distribution Emergence:** With increasing sample sizes, sample means' distribution shifts towards a more symmetric, bell-shaped curve, validating CLT predictions.
  - **Convergence to True Mean:** Larger samples show sample means converging towards the dataset's true mean temperature, enhancing the accuracy of population mean estimates.
  - **Variability Reduction:** As sample size grows, the spread of sample means narrows, indicating more reliable estimates of the population mean.
  - **Statistical Inference:** The observed trends enable applying normality-assuming statistical methods (hypothesis testing, confidence intervals) effectively.

- **Conclusion:** Analyzing Riyadh's temperature data with the CLT reveals that larger sample sizes cause sample means to closely follow a normal distribution, essential for precise population insights.



Formal Definition and Mathematical Representation

Let $X_1, X_2, ..., X_n$ be a sequence of independent and identically distributed random variables, each with an expected mean μ and a finite variance σ^2. The distribution of the sample means $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ will approximate a normal distribution (Gaussian distribution) as the sample size n becomes large, with the mean μ and variance $\frac{\sigma^2}{n}$.

This theorem is mathematically represented as:

$$\lim_{n\to\infty} P\left(a \leq \frac{1}{\sqrt{n}}\frac{\bar{X}-\mu}{\sigma} \leq b\right) = \int_a^b \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}\, dx$$

CLT Proof

Obtain Moments through Derivatives

- **Integral Function for $E[X]$**: The expectation of X, or the first moment, can be calculated using the integral of x times the probability density function $p_X(x)$ over all possible values of x:

$$E[X] = \int_{-\infty}^{\infty} x p_X(x) dx$$

- **Integral Function for $E[X^2]$**: Similarly, the expectation of X^2, or the second moment, is found by integrating x^2 times the probability density function $p_X(x)$ over all possible values of x:

$$E[X^2] = \int_{-\infty}^{\infty} x^2 p_X(x) dx$$

Moment of Generating Functions

- **Definition of MGF**: The moment-generating function (MGF) of a random variable X, denoted as $M_X(t)$, is defined as the expected value of e^{tX}, where t is a real number, and X is a random variable.

$$M_X(t) = E[e^{tX}]$$

- **Integral Form**: The MGF, when X has a continuous distribution with probability density function $p_X(x)$, is given by:

$$M_X(t) = \int_{-\infty}^{\infty} e^{tx} p_X(x) dx$$

- **Expansion by Taylor Series**: The MGF can be expanded as a Taylor series around $t = 0$:

$$E[e^{tX}] = E[1 + tX + \frac{(tX)^2}{2!} + \frac{(tX)^3}{3!} + \cdots]$$

- **Relation to Moments**: The coefficients of the Taylor series of $M_X(t)$ around $t = 0$ correspond to the moments of the random variable X:

$$M_X(t) = 1 + tE[X] + \frac{t^2}{2!}E[X^2] + \frac{t^3}{3!}E[X^3] + \cdots$$

- **Example for Gaussian Distribution**: For a Gaussian (normal) distribution with mean μ and variance σ^2, the MGF is:

$$M_X(t) = e^{\mu t + \frac{1}{2}\sigma^2 t^2}$$

Obtain Moments through Derivatives

- **Relation to MGF**: The moments obtained through these integrals are directly related to the derivatives of the Moment Generating Function (MGF) at $t = 0$.
  - **First Moment (Mean) via MGF**: By differentiating the MGF once with respect to t and evaluating at $t = 0$, we obtain the first moment, or the mean of the distribution:

$$\left.\frac{dM_X(t)}{dt}\right|_{t=0} = E[X]$$

  - **Second Moment via MGF**: By differentiating the MGF twice with respect to t and evaluating at $t = 0$, we obtain the second moment:

$$\left.\frac{d^2M_X(t)}{dt^2}\right|_{t=0} = E[X^2]$$

- **General Formula for nth Moment**: The nth moment of X can be obtained by taking the nth derivative of $M_X(t)$ with respect to t and then evaluating it at $t = 0$:

$$E[X^n] = \left.\frac{d^nM_X(t)}{dt^n}\right|_{t=0}$$

Moment of Generating Functions

- **Definition of MGF:** $M_X(t) = \mathbb{E}[e^{tX}]$
- **Integral Form:** $M_X(t) = \int_{-\infty}^{\infty} e^{tx} P_X(x) dx$
- **Expansion by Taylor Series:** $\mathbb{E}[e^{tX}] = \mathbb{E}\left[1 + tX + \frac{(tX)^2}{2!} + \frac{(tX)^3}{3!} + \cdots\right]$
- **Relation to Moments:** $M_X(t) = 1 + t\mathbb{E}[X] + \frac{t^2}{2!}\mathbb{E}[X^2] + \frac{t^3}{3!}\mathbb{E}[X^3] + \cdots$
- **Example for Gaussian Distribution:** $M_X(t) = e^{\mu t + \frac{1}{2}\sigma^2 t^2}$
- **Derivative to Obtain Moments:**
  - First Moment (Mean): $\frac{d}{dt} M_X(t)\Big|_{t=0} = \mathbb{E}[X]$
  - Second Moment: $\frac{d^2}{dt^2} M_X(t)\Big|_{t=0} = \mathbb{E}[X^2]$
- **Note:** The nth derivative of $M_X(t)$ evaluated at $t = 0$ gives the nth moment of the random variable X.

Example of Gaussian Distribution

To prove $E[X] = \mu$ for a Gaussian distribution using its Moment Generating Function (MGF):

Given the MGF of a Gaussian distribution:
$M_X(t) = e^{\mu t + \frac{1}{2}\sigma^2 t^2}$

The first derivative of $M_X(t)$ with respect to t is:
$\frac{dM_X(t)}{dt} = (\mu + \sigma^2 t)e^{\mu t + \frac{1}{2}\sigma^2 t^2}$

Evaluating this derivative at $t = 0$ gives:
$E[X] = \mu e^0 = \mu$

This concise proof demonstrates that for a Gaussian distribution, the expected value $E[X]$ is equal to its mean, μ.

Moment of Generating Functions - Python Example

```python
import numpy as np
import pandas as pd

# Load the dataset
df = pd.read_csv('https://www.riotu-lab.org/cs313/weather.csv')

# Calculate empirical mean and variance of Temperature
mean_temp = df['Temperature'].mean()
var_temp = df['Temperature'].var()

print(f"Mean Temperature: {mean_temp}")
print(f"Variance of Temperature: {var_temp}")

# Define a function to approximate MGF for Temperature at a given t
def approximate_mgf(t, temperatures):
    if t == 0:
        return 1  # Directly return 1 for the t=0 case
    return np.mean(np.exp(t * temperatures))

# Choose a t value to compute the MGF
t = 0.1

mgf_approx = approximate_mgf(t, df['Temperature'])

print(f"Approximate MGF of Temperature at t={t}: {mgf_approx}")

```

```
Mean Temperature: 23.707583333333332
Variance of Temperature: 21.03453704038997
Approximate MGF of Temperature at t=0.1: 11.894757989672568
```

$$M(t) = E(e^{tX})$$

$$= \sum_{x \in S} e^{tx} f(x)$$

The code calculates an Empirical MGF

CLT Proof Strategy using MGF

- **Proof Strategy**:
  1. **Define i.i.d. Variables**: $X_1, X_2, ..., X_n$ with common mean μ and variance σ^2.
  2. **Consider Sum**: $S_n = X_1 + X_2 + ... + X_n$.
  3. **Standardize** S_n: $Z_n = \frac{S_n - n\mu}{\sigma\sqrt{n}}$ to have mean 0 and variance 1.
  4. **MGF of** Z_n: Show $M_{Z_n}(t) \to e^{\frac{t^2}{2}}$ as $n \to \infty$, matching the MGF of a standard normal distribution.
  5. **Use Limits & Series**: Apply calculus principles to demonstrate convergence of $M_{Z_n}(t)$ to that of a normal distribution.
- **Conclusion**: By the uniqueness of MGFs, if the MGF of the standardized sum converges to that of a standard normal distribution, so does the distribution of the sum, proving the CLT.

CLT Proof using MGF - Problem Statement

Given Setup

1. Let $X_1, X_2, \ldots, X_n$ be i.i.d. random variables with mean μ and variance σ^2.
2. Define $S_n = \sum_{i=1}^{n} X_i$, the sum of the first n random variables.
3. The standardized version of S_n is $Z_n = \frac{S_n - n\mu}{\sigma\sqrt{n}}$.

Objective

Show that $M_{Z_n}(t) \to e^{\frac{t^2\sigma^2}{2}}$ as $n \to \infty$, where $M_{Z_n}(t)$ is the MGF of Z_n.

CLT Proof using MGF – Steps

MGF of X_i: By definition, the MGF of a single observation X_i is $M_{X_i}(t) = E[e^{tX_i}]$.

MGF of S_n: Since X_i are independent, $M_{S_n}(t) = (M_{X_1}(t))^n$ due to the property that the MGF of the sum of independent random variables is the product of their MGFs.

Taylor Expansion: Assuming X_i has a finite mean μ and variance σ^2, expand $M_{X_i}(t)$ around 0 to get $M_{X_i}(t) \approx 1 + \mu t + \frac{\sigma^2 t^2}{2}$ for small t.

MGF of Z_n: To find $M_{Z_n}(t)$, consider the transformation applied to S_n, yielding:

$$M_{Z_n}(t) = E\left[e^{t\frac{S_n - n\mu}{\sigma\sqrt{n}}}\right] = e^{-\frac{tn\mu}{\sigma\sqrt{n}}}\left(M_{X_1}\left(\frac{t}{\sigma\sqrt{n}}\right)\right)^n$$

Limit as $n \to \infty$:

$$\lim_{n\to\infty} M_{Z_n}(t) = \lim_{n\to\infty} e^{-\frac{tn\mu}{\sigma\sqrt{n}}}\left(1 + \frac{t\mu}{\sigma\sqrt{n}} + \frac{t^2\sigma^2}{2n}\right)^n$$

Applying Exponential Limit: Recognize that the expression inside the limit resembles the form of e^x as $x \to \infty$. Specifically, use the limit definition of e and properties of exponential functions to simplify:

$$\lim_{n\to\infty} M_{Z_n}(t) = e^{\frac{t^2\sigma^2}{2}}$$

CHAPTER 7
STATISTICAL LEARNING

LECTURE 1
FUNDAMENTAL STATISTICAL THEOREMS IN DATA SCIENCE

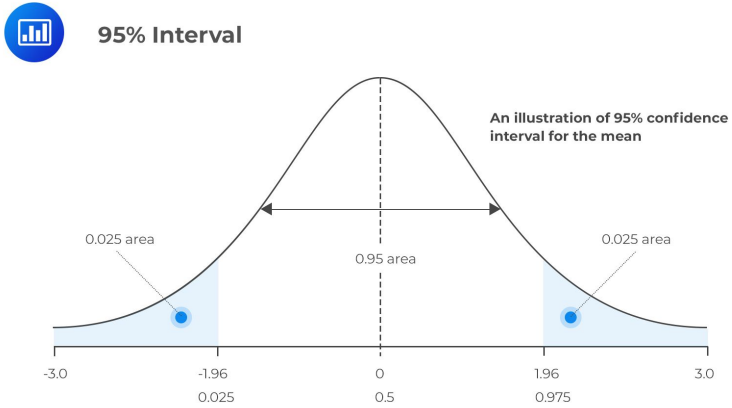
CONFIDENCE INTERVAL

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Fundamentals of Confidence Intervals

- **Definition**
  - CIs provide a **range** likely to contain the true population parameter (e.g., **population mean**).
  - They are associated with a level of confidence (e.g., 95%), indicating the probability that the true parameter falls within the interval.
- **Link to Central Limit Theorem (CLT):**
  - The CLT guarantees that the sampling distribution of the mean approximates a normal distribution for large sample sizes.
  - This normality allows us to create standardized confidence intervals using critical values from the standard normal distribution.



Z-Score

The Z-score of a value quantifies how many standard deviations it is from the mean. The formula for calculating a Z-score is:

$$Z = \frac{(X - \mu)}{\sigma}$$

- **Variables Explained:**
  - Z: The Z-score.
  - X: The value being standardized.
  - μ: The mean of the dataset.
  - σ: The standard deviation of the dataset.

Confidence Interval as a Function of Sample Mean

The confidence interval for the population mean, based on the sample mean, is expressed by the formula:

$$CI = \bar{x} \pm Z_{\frac{\alpha}{2}} \times \frac{s}{\sqrt{n}}$$

- **Formula Components:**
  - CI: Confidence Interval for the population mean.
  - $\bar{x}$: Sample mean, serving as the estimator for the population mean (μ).
  - $Z_{\frac{\alpha}{2}}$: Z-score corresponding to the desired confidence level (e.g., 1.96 for 95% confidence).
  - s: Sample standard deviation, used as an estimate of the population standard deviation (σ) in the absence of σ.
  - n: Sample size, or the number of observations in the sample.

Mathematical Foundation of Confidence Intervals

- **Confidence Interval Formula:**

$$\bar{x} \pm Z_{\frac{\alpha}{2}} \times \frac{\sigma}{\sqrt{n}}$$

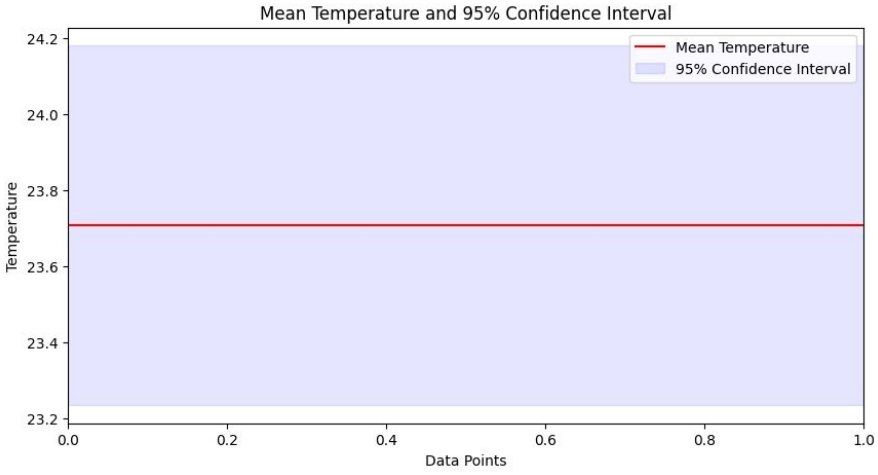
  - **Variables Explained:**
    - $\bar{x}$: Sample mean, an estimator of the population mean μ.
    - $Z_{\frac{\alpha}{2}}$: Z-score for the selected confidence level, defining the critical value that encapsulates the central area under the normal distribution curve.
    - σ: Population standard deviation, indicating the dispersion of population values.
    - n: Number of observations in the sample, influencing the margin of error.

CLI Effect on Confidence Intervals

- **Role of the Central Limit Theorem (CLT):**
  - **Foundation:** The CLT asserts that the distribution of the sample mean approximates a normal distribution as the sample size n becomes large, regardless of the population's distribution shape.
  - **Implication for Confidence Intervals:** This theorem validates the utilization of the sample mean ($\bar{x}$) as an effective estimator for the population mean (μ) in the formula, ensuring that $Z_{\frac{\alpha}{2}}$ can be applied to calculate the interval around $\bar{x}$ that likely contains μ.
  - **Significance:** By guaranteeing that the sampling distribution of the mean tends towards normality, the CLT enables the precise computation of confidence intervals even when the underlying population distribution is unknown or non-normal.

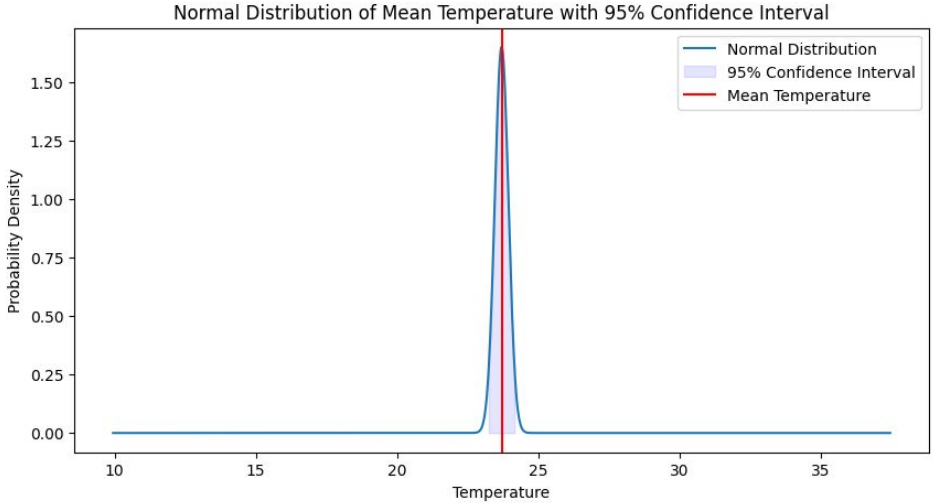
Confidence Levels and Interpretation

- **Confidence Levels**
  - Usually 95% or 99%.
  - Represents the percentage of confidence intervals that would contain the true population parameter if we repeated the sampling and calculation process many times.
- **Interpretation of a 95% Confidence Interval**
  - If we construct 100 confidence intervals from different samples, roughly 95 of them are expected to include the true population parameter.



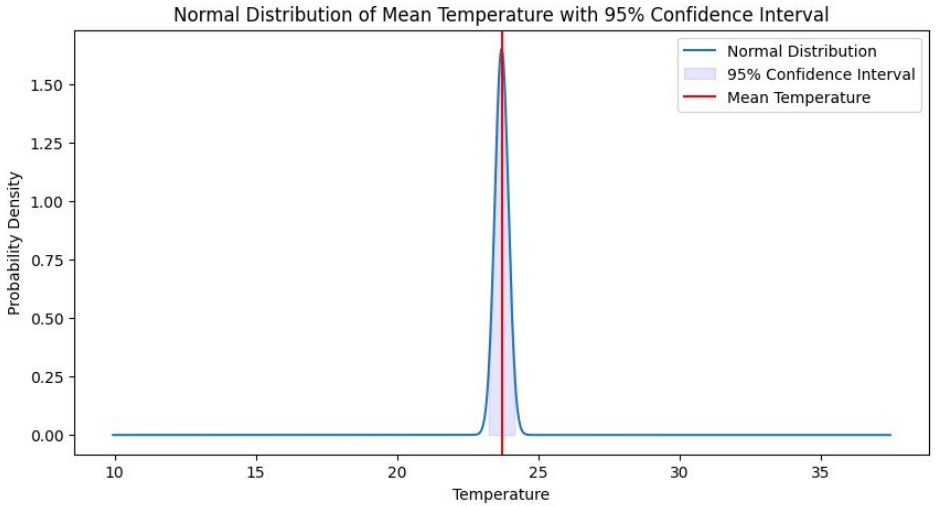
Confidence Levels Illustration

```python
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
url = "https://www.riotu-lab.org/cs313/weather.csv"
data = pd.read_csv(url)

# Calculate the mean temperature and the standard deviation
mean_temperature = data['Temperature'].mean()
std_temperature = data['Temperature'].std(ddof=1)  # ddof=1 for sample standard deviation
sample_size = len(data['Temperature'])

# Calculate the 95% Confidence Interval for the mean temperature
confidence_level = 0.95
z_score = stats.norm.ppf((1 + confidence_level) / 2)
margin_of_error = z_score * (std_temperature / np.sqrt(sample_size))

lower_bound = mean_temperature - margin_of_error
upper_bound = mean_temperature + margin_of_error

# Normal Distribution Plot with Confidence Interval
x = np.linspace(mean_temperature - 3*std_temperature, mean_temperature + 3*std_temperature, 1000)
y = stats.norm.pdf(x, mean_temperature, std_temperature/np.sqrt(sample_size))

plt.figure(figsize=(10, 5))
plt.plot(x, y, label='Normal Distribution')
plt.fill_between(x, y, where=(x >= lower_bound) & (x <= upper_bound), color='blue', alpha=0.1, label='95% Confidence Interval')
plt.axvline(x=mean_temperature, color='red', linestyle='-', label='Mean Temperature')
plt.title('Normal Distribution of Mean Temperature with 95% Confidence Interval')
plt.xlabel('Temperature')
plt.ylabel('Probability Density')
plt.legend()
plt.show()

```



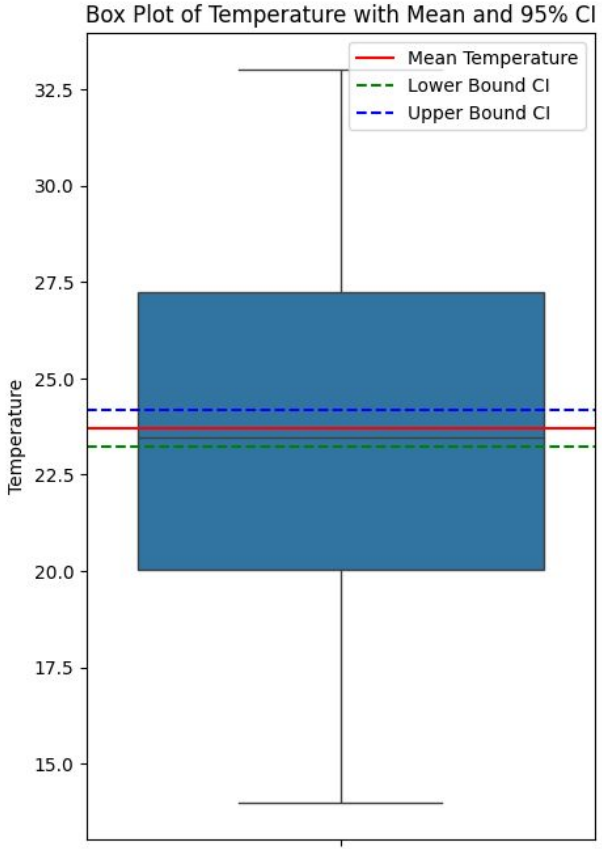
Confidence Levels Illustration

```python
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
url = "https://www.riotu-lab.org/cs313/weather.csv"
data = pd.read_csv(url)

# Calculate the mean temperature and the standard deviation
mean_temperature = data['Temperature'].mean()
std_temperature = data['Temperature'].std(ddof=1)  # ddof=1 for sample standard deviation
sample_size = len(data['Temperature'])

# Calculate the 95% Confidence Interval for the mean temperature
confidence_level = 0.95
z_score = stats.norm.ppf((1 + confidence_level) / 2)
margin_of_error = z_score * (std_temperature / np.sqrt(sample_size))

lower_bound = mean_temperature - margin_of_error
upper_bound = mean_temperature + margin_of_error

# Normal Distribution Plot with Confidence Interval
x = np.linspace(mean_temperature - 3*std_temperature, mean_temperature + 3*std_temperature, 1000)
y = stats.norm.pdf(x, mean_temperature, std_temperature/np.sqrt(sample_size))

plt.figure(figsize=(10, 5))
plt.plot(x, y, label='Normal Distribution')
plt.fill_between(x, y, where=(x >= lower_bound) & (x <= upper_bound), color='blue', alpha=0.1, label='95% Confidence Interval')
plt.axvline(x=mean_temperature, color='red', linestyle='-', label='Mean Temperature')
plt.title('Normal Distribution of Mean Temperature with 95% Confidence Interval')
plt.xlabel('Temperature')
plt.ylabel('Probability Density')
plt.legend()
plt.show()

```



Confidence Levels Illustration

```python
# Load the dataset
url = "https://www.riotu-lab.org/cs313/weather.csv"
data = pd.read_csv(url)

# Calculate the mean temperature and the standard deviation
mean_temperature = data['Temperature'].mean()
std_temperature = data['Temperature'].std(ddof=1)  # ddof=1 for sample standard dev
sample_size = len(data['Temperature'])

# Calculate the 95% Confidence Interval for the mean temperature
confidence_level = 0.95
z_score = stats.norm.ppf((1 + confidence_level) / 2)
margin_of_error = z_score * (std_temperature / np.sqrt(sample_size))

lower_bound = mean_temperature - margin_of_error
upper_bound = mean_temperature + margin_of_error

# Normal Distribution Plot with Confidence Interval
x = np.linspace(mean_temperature - 3*std_temperature, mean_temperature + 3*std_tempe
y = stats.norm.pdf(x, mean_temperature, std_temperature/np.sqrt(sample_size))

plt.figure(figsize=(10, 5))
plt.plot(x, y, label='Normal Distribution')
plt.fill_between(x, y, where=(x >= lower_bound) & (x <= upper_bound), color='blue', a
plt.axvline(x=mean_temperature, color='red', linestyle='-', label='Mean Temperature
plt.title('Normal Distribution of Mean Temperature with 95% Confidence Interval')
plt.xlabel('Temperature')
plt.ylabel('Probability Density')
plt.legend()
plt.show()

# Box Plot
plt.figure(figsize=(5, 8))
sns.boxplot(y=data['Temperature'])
plt.axhline(y=mean_temperature, color='r', linestyle='-', label='Mean Temperature')
plt.axhline(y=lower_bound, color='g', linestyle='--', label='Lower Bound CI')
plt.axhline(y=upper_bound, color='b', linestyle='--', label='Upper Bound CI')
plt.title('Box Plot of Temperature with Mean and 95% CI')
plt.ylabel('Temperature')
plt.legend()
plt.show()

```



Introduction to Statistical Learning

- **Statistical Learning vs. Machine Learning:**
  - **Statistical learning:** Model interpretation, quantifying uncertainty
  - **Machine learning:** Large-scale prediction, data mining.
- **Main challenge:** Mathematical analysis, not structuring/visualizing data
- **Core Goals of Data Modeling:**
  - Predict future quantities based on observed data.
  - Discover interesting patterns within the data.
- **Pillars of Statistical Learning:**
  - **Function approximation:** Understanding variable relationships, efficient representation
  - **Optimization:** Finding the best model fit, calibration to data
  - **Prob/Stats:** Assessing uncertainty, prediction accuracy, error sources

Robotics & Internet-of-Things

جامعة الأمير سلطان PRINCE SULTAN UNIVERSITY

CS316: INTRODUCTION TO AI AND DATA SCIENCE

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

Prof. Anis Koubaa

February 2024

www.riotu-lab.org

CS316: INTRODUCTION TO AI AND DATA SCIENCE

CHAPTER 7
STATISTICAL LEARNING

Lecture 1 Review

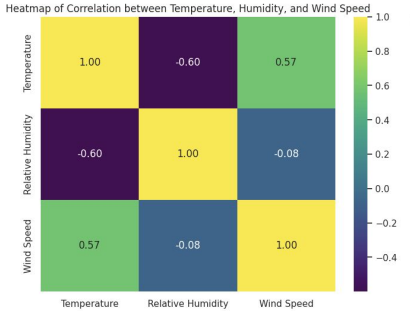
Bayes' Theorem

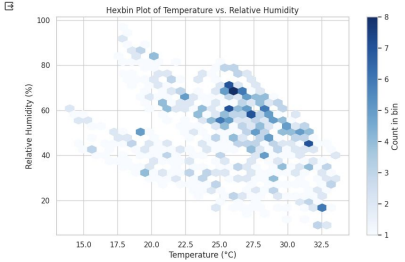
- **Objective**: Calculate $P(A|B)$ - Probability of temperature > 20°C given relative humidity > 40%.
- **Defining Events**:
  - **A**: Temperature is above 20°C.
  - **B**: Relative humidity is above 40%.

- **Bayes' Theorem**: $P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$
  - $P(A|B)$: Probability of A given B, the target calculation.
  - $P(A)$: Prior probability of A (temperature > 20°C).
  - $P(B|A)$: Likelihood of B given A (humidity > 40% when temperature > 20°C).
  - $P(B)$: Marginal probability of B (humidity > 40%).

  - **Prior Probability:** The prior probability reflects our initial belief about the likelihood of an event occurring before considering any new evidence.
  - **Marginal Probability:** The marginal probability represents the overall probability of an event occurring across all possible values of another related variable.

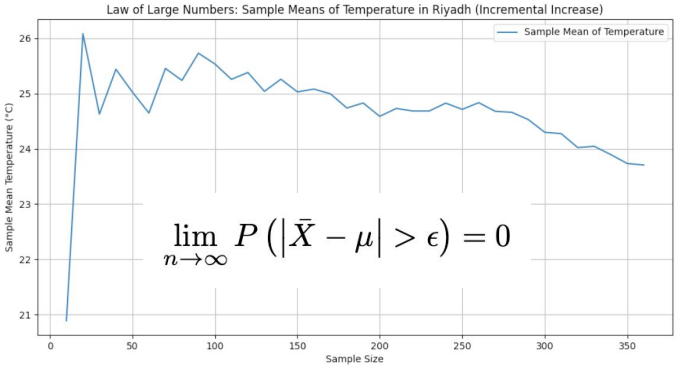
	A	B	C	D	E	F
1	Year	Month	Day	Hour	Temperature	Relative Humidity
2	2018	11	6	0	21.65	38
3	2018	11	6	1	20.94	40
4	2018	11	6	2	20.29	41
5	2018	11	6	3	19.67	43
6	2018	11	6	4	18.76	45
7	2018	11	6	5	18.25	47
8	2018	11	6	6	17.68	48
9	2018	11	6	7	19.63	42
10	2018	11	6	8	24.18	31





Law of Large Numbers & Temperature Data Analysis

```python
import pandas as pd
import matplotlib.pyplot as plt

weather_data = pd.read_csv('https://www.riotu-lab.org/cs313/weather.csv')

# Function to calculate the sample mean of temperature for different sample sizes
def calculate_sample_means_incremental(data, start_sample_size, increment):
    sample_means = []
    for sample_size in range(start_sample_size, len(data) + 1, increment):
        sample_mean = data['Temperature'][:sample_size].mean()
        sample_means.append((sample_size, sample_mean))
    return sample_means

# Calculate sample means starting from 10 entries and increasing incrementally
start_sample_size = 10
increment = 10  # Incremental step
incremental_sample_means = calculate_sample_means_incremental(weather_data, start_sample_size, increment)

# Creating a DataFrame for the sample means
sample_means_df = pd.DataFrame(incremental_sample_means, columns=['Sample Size', 'Sample Mean Temperature'])

# Display the DataFrame
print(sample_means_df)

# Preparing data for plotting
sample_sizes = sample_means_df['Sample Size']
sample_means_values = sample_means_df['Sample Mean Temperature']

# Plotting the results
plt.figure(figsize=(12, 6))
plt.plot(sample_sizes, sample_means_values, label='Sample Mean of Temperature')
plt.xlabel('Sample Size')
plt.ylabel('Sample Mean Temperature (°C)')
plt.title('Law of Large Numbers: Sample Means of Temperature in Riyadh (Incremental Increase)')
plt.legend()
plt.grid(True)
plt.show()
```



	Sample Size	Sample Mean Temperature
0	10	20.891000
1	20	26.084000
2	30	24.627667
3	40	25.439000
4	50	25.026400
5	60	24.647500
6	70	25.454429
7	80	25.237500
8	90	25.730667
9	100	25.535400
10	110	25.257727
11	120	25.381333
12	130	25.040000
13	140	25.259786
14	150	25.030467
15	160	25.080688
16	170	24.995529
17	180	24.737444
18	190	24.829474
19	200	24.587250
20	210	24.730048
21	220	24.684909
22	230	24.684043
23	240	24.826750
24	250	24.713920
25	260	24.835154
26	270	24.678667
27	280	24.661000
28	290	24.530103
29	300	24.299133
30	310	24.274742
31	320	24.022531
32	330	24.045515
33	340	23.896235
34	350	23.734143
35	360	23.707583

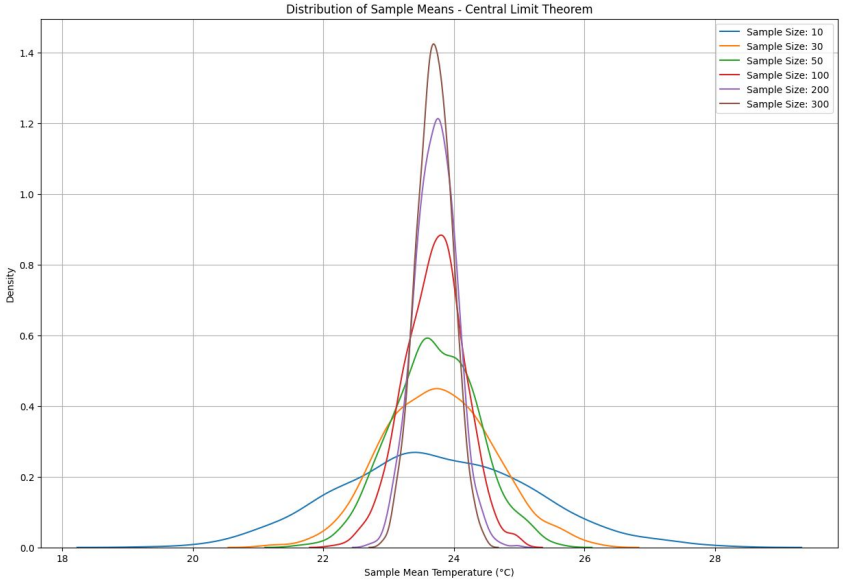
Central Limit Theorem: Real-World Application with Weather Data

```python
import pandas as pd
from numpy.random import choice
import seaborn as sns
import matplotlib.pyplot as plt


weather_data = pd.read_csv('https://www.riotu-lab.org/cs313/weather.csv')
# Function to compute sample means for different sample sizes
def compute_sample_means_for_CLT(data, sample_sizes, num_samples=1000):
    sample_means = {size: [] for size in sample_sizes}
    for size in sample_sizes:
        for _ in range(num_samples):
            sample = choice(data['Temperature'], size=size)
            sample_means[size].append(sample.mean())
    return sample_means

# Define sample sizes for the experiment
sample_sizes = [10, 30, 50, 100, 200]

# Compute sample means
sample_means_CLT = compute_sample_means_for_CLT(weather_data, sample_sizes)

# Plotting the distributions of the sample means
plt.figure(figsize=(15, 10))
for size in sample_sizes:
    sns.distplot(sample_means_CLT[size], label=f'Sample Size: {size}', kde=True, hist=False)

plt.xlabel('Sample Mean Temperature (°C)')
plt.ylabel('Density')
plt.title('Distribution of Sample Means - Central Limit Theorem')
plt.legend()
plt.grid(True)
plt.show()

```

Let $X_1, X_2, ..., X_n$ be a sequence of independent and identically distributed random variables, each with an expected mean μ and a finite variance σ^2. The distribution of the sample means $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ will approximate a normal distribution (Gaussian distribution) as the sample size n becomes large, with the mean μ and variance $\frac{\sigma^2}{n}$.

This theorem is mathematically represented as:

$$\lim_{n\to\infty} P\left(a \leq \frac{1}{\sqrt{n}}\frac{\bar{X}-\mu}{\sigma} \leq b\right) = \int_a^b \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}\, dx$$



Robotics & Internet-of-Things

جامعة الأمير سلطان PRINCE SULTAN UNIVERSITY

CS316: INTRODUCTION TO AI AND DATA SCIENCE

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

Prof. Anis Koubaa

February 2024

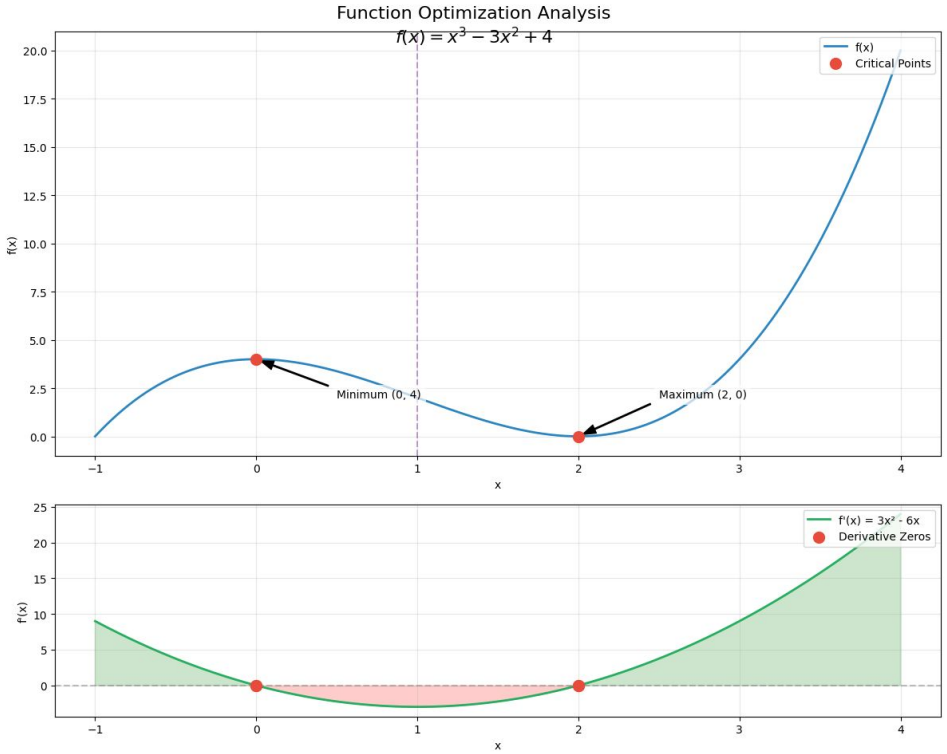
www.riotu-lab.org

Introduction to Statistical Learning

- **Statistical Learning vs. Machine Learning:**
  - **Statistical learning:** Model interpretation, quantifying uncertainty
  - **Machine learning:** Large-scale prediction, data mining.
- **Main challenge:** Mathematical analysis, not structuring/visualizing data
- **Core Goals of Data Modeling:**
  - Predict future quantities based on observed data.
  - Discover interesting patterns within the data.
- **Pillars of Statistical Learning:**
  - **Function approximation:** Understanding variable relationships, efficient representation
  - **Optimization:** Finding the best model fit, calibration to data
  - **Prob/Stats:** Assessing uncertainty, prediction accuracy, error sources

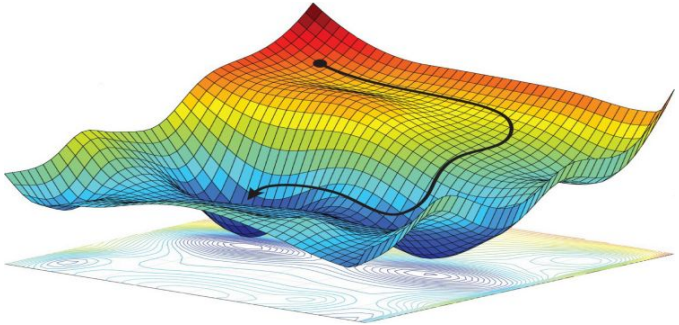
Function Optimization: Critical Points Where Derivative Equals Zero

- **Graph Description**:
  - Plot a cubic polynomial $f(x) = x^3 - 3x^2 + 4$.
  - This function has critical points where its derivative $f'(x) = 3x^2 - 6x$ equals zero.
- **Critical Points**:
  - **Highlight**: Identify the points on the graph where $f'(x) = 0$.
  - **Locations**: Calculate the derivative to find x values that satisfy $f'(x) = 0$ which are $x = 0$ and $x = 2$.
- **Annotations**:
  - Label the point $(0, 4)$ as "Potential Minimum".
  - Label the point $(2, 0)$ as "Potential Maximum".
  - Use arrows to point to these critical points, illustrating their importance in finding the maxima and minima of the function.

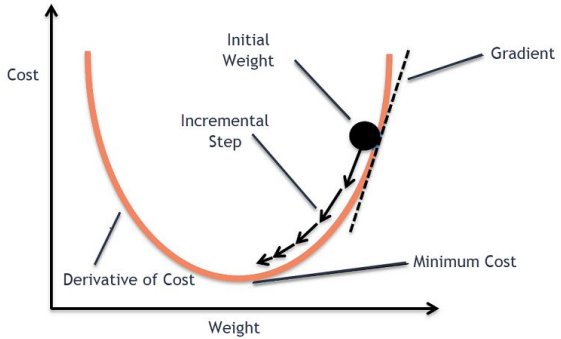


Optimization in Data Science

- **Objective:** Identify the optimal model within a predefined class of mathematical models.
  - **Minimize the error between predicted values and actual observations**
- **Methodology:** Employ efficient search or optimization techniques for model fitting.
  - Example: Gradient Descent
- **Application Example:**
  - Maximizing **recommendation system** effectiveness by tuning algorithms.
  - Enhancing **weather forecast** precision by calibrating prediction models.
  - Improving **email spam detection** accuracy by refining logistic regression parameters.



https://medium.com/analytics-vidhya



https://www.analyticsvidhya.com/

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

Math Foundations of Predictive Modeling in Data Science

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

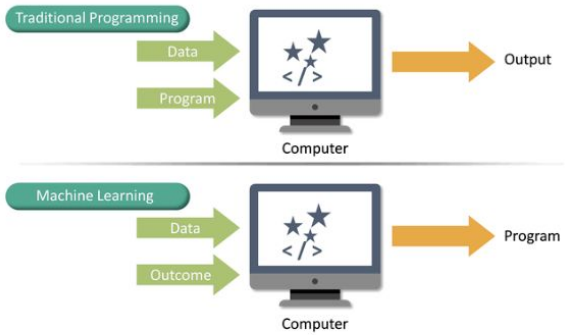
Understanding the Prediction Function

- **Learning from data:** The prediction function **(g)** is developed by **"learning" patterns** from existing **data**.
- **Underlying Relationships:** The function **(g)** attempts to **model** the true relationship between **input features (x)** and the **output variable (y)**.
- **Inherent Uncertainty:** Real-world data contains **randomness** and **unpredictable** variation, limiting the perfection of any prediction function.

Prediction Function Modeling

1. **Prediction Function Representation:**
   - $g(x) = \hat{y}$
     - Where $g(x)$ is the prediction function, x is the input feature vector, and $\hat{y}$ is the predicted output.
2. **Example Equations:**
   - For email spam detection:
     - $$g(x_{\text{email features}}) = \begin{cases} 1 & \text{if spam} \\ 0 & \text{if not spam} \end{cases}$$
   - For weather forecasting:
     - $g(x_{\text{temperature, humidity}}) = \hat{y}_{\text{weather condition}}$
       - $x_{\text{temperature, humidity}}$ represents input features like current temperature and humidity levels.
3. **Mathematical Formulation:**
   - Linear Regression Example: $g(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$
     - $x_1, x_2, \ldots, x_n$ are input features, $\beta_0, \beta_1, \ldots, \beta_n$ are model parameters.
4. **Binary Classification Example:**
   - Logistic Regression: $P(Y = 1|x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\ldots+\beta_n x_n)}}$
     - Predicts the probability P of $Y = 1$ (e.g., email being spam) given input features x.

```
def train(images, labels):
    … machine learning …
    return model

def predict(model, test_images):
    … use model to predict labels …
    return test_labels
```

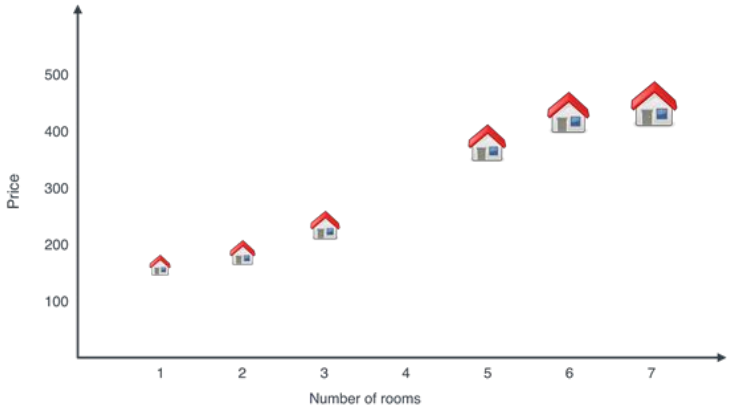


https://www.sketchbubble.com/en/presentation-machine-learning.html

Prediction Function Modeling

3. **Mathematical Formulation:**
   - Linear Regression Example: $g(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$
     - $x_1, x_2, \ldots, x_n$ are input features, $\beta_0, \beta_1, \ldots, \beta_n$ are model parameters.

```
def train(images, labels):
    … machine learning …
    return model

def predict(model, test_images):
    … use model to predict labels …
    return test_labels
```

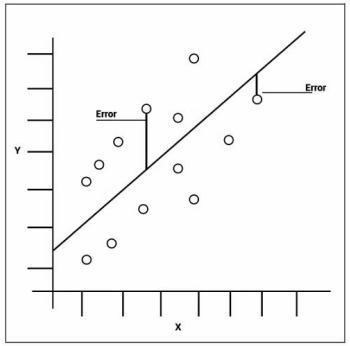


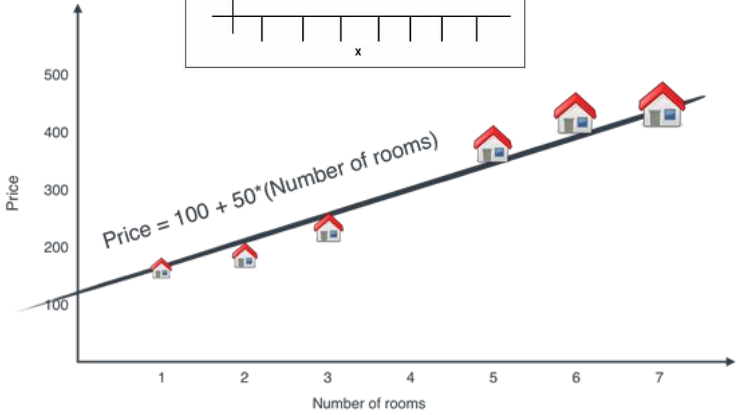
https://livebook.manning.com/book/grokking-machine-learning/3-2-the-solution-building-a-regression-model-for-housing-prices/v-4/44

Prediction Function Modeling

3. **Mathematical Formulation:**
   - Linear Regression Example: $g(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$
     - $x_1, x_2, \ldots, x_n$ are input features, $\beta_0, \beta_1, \ldots, \beta_n$ are model parameters.

```
def train(images, labels):
    … machine learning …
    return model

def predict(model, test_images):
    … use model to predict labels …
    return test_labels
```



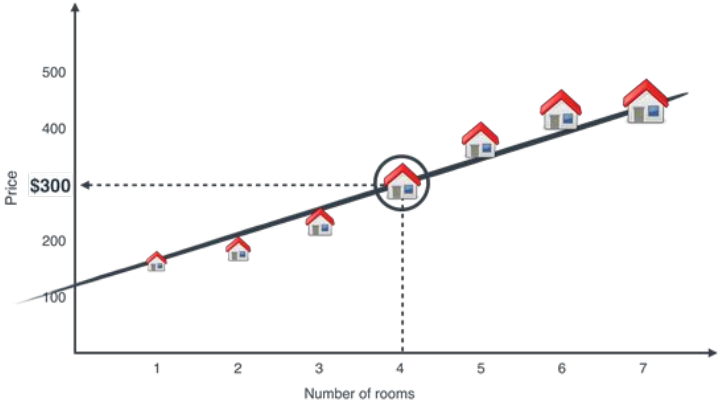


https://livebook.manning.com/book/grokking-machine-learning/3-2-the-solution-building-a-regression-model-for-housing-prices/v-4/44

Prediction Function Modeling

3. **Mathematical Formulation:**
   - Linear Regression Example: $g(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$
     - $x_1, x_2, \ldots, x_n$ are input features, $\beta_0, \beta_1, \ldots, \beta_n$ are model parameters.

```
def train(images, labels):
    … machine learning …
    return model

def predict(model, test_images):
    … use model to predict labels …
    return test_labels
```

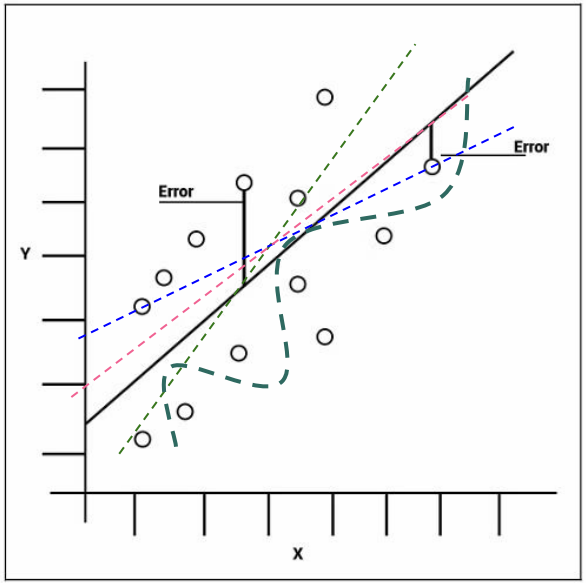


https://livebook.manning.com/book/grokking-machine-learning/3-2-the-solution-building-a-regression-model-for-housing-prices/v-4/44

Prediction Function Modeling

3. **Mathematical Formulation:**
   - Linear Regression Example: $g(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$
     - $x_1, x_2, \ldots, x_n$ are input features, $\beta_0, \beta_1, \ldots, \beta_n$ are model parameters.

```
def train(images, labels):
    … machine learning …
    return model

def predict(model, test_images):
    … use model to predict labels …
    return test_labels
```



https://livebook.manning.com/book/grokking-machine-learning/3-2-the-solution-building-a-regression-model-for-housing-prices/v-4/44

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

LOSS FUNCTION

CS316: INTRODUCTION TO AI AND DATA SCIENCE

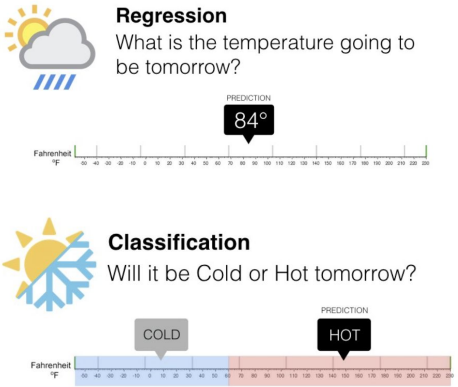
Prof. Anis Koubaa

Loss Functions

- **What are loss functions?**
  - **Loss functions** are used to measure the **difference (discrepancy)** between the **predicted** value of a model and the **true** value.
  - They are essential for **training** machine **learning** models, as they provide a way to evaluate how well a model is performing and guide the learning process.

Loss Functions in Prediction Models

- **Regression vs. Classification:**
  - **Regression:** y takes any real value.
  - **Classification:** y is in a finite set, making it a categorization task.
- **Loss Functions Overview:**
  - **Regression:** Commonly uses squared-error loss $(y - \hat{y})^2$.
  - **Classification:** Often employs zero–one loss $Loss(y, \hat{y}) = 1_{\{y \neq \hat{y}\}}$, incurring a loss of 1 for incorrect predictions.



- **Error Measurement:**
  - **Absolute Error:** $|y - \hat{y}|$ for real-valued y.
  - **Squared Error:** $(y - \hat{y})^2$, a measure of prediction accuracy.
  - **Vector Norms:** $||\mathbf{y} - \hat{\mathbf{y}}||$ and squared norm $||\mathbf{y} - \hat{\mathbf{y}}||^2$ for vector-valued responses.
- **Advanced Loss Functions:**
  - **Cross-Entropy:** For probabilistic classification.
  - **Hinge Loss:** Used in support vector machines (SVMs) for classification.

} **Classification Loss Functions**

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

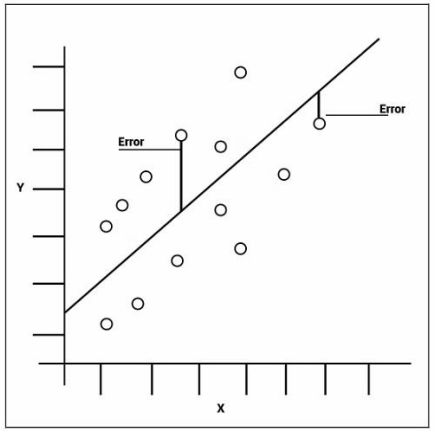
SQUARED ERROR LOSS

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Squared Error Loss in Data Science

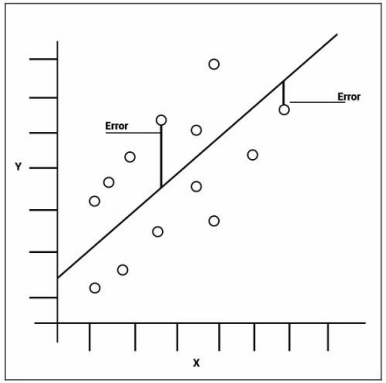
- **Definition of Squared Error Loss:**
  - **Mathematical Expression:** $(y - \hat{y})^2$
    - Where y is the actual value and $\hat{y}$ is the predicted value.
- **Core Concept:**
  - Measures the square of the difference between actual and predicted values, emphasizing larger errors.
- **Importance in Regression Analysis:**
  - **Widely Used:** The standard loss function for regression problems.
  - **Sensitivity to Outliers:** Heavily penalizes large deviations, ensuring model accuracy.



Squared Error Loss in Data Science

- **Mathematical Foundations:**
  - **Objective:** Minimize the sum of squared errors across all observations in the dataset.
  - **Equation:** $L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$
    - Aims to find model parameters that minimize L.
- **Applications in Data Science:**
  - **Model Evaluation:** Benchmark for assessing model performance.
  - **Gradient Descent:** Utilized in optimization algorithms to adjust model parameters.
- **Advantages:**
  - **Analytical Clarity:** Direct relationship with linear models, facilitating solution derivation.
  - **Predictive Performance:** Effective in capturing the variability of the data.

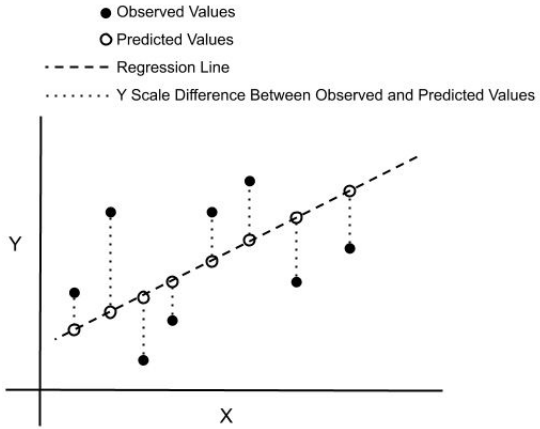
$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$



Squared Error Loss in Data Science

- **Mathematical Foundations:**
  - **Objective:** Minimize the sum of squared errors across all observations in the dataset.
  - **Equation:** $L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$
    - Aims to find model parameters that minimize L.
- **Applications in Data Science:**
  - **Model Evaluation:** Benchmark for assessing model performance.
  - **Gradient Descent:** Utilized in optimization algorithms to adjust model parameters.
- **Advantages:**
  - **Analytical Clarity:** Direct relationship with linear models, facilitating solution derivation.
  - **Predictive Performance:** Effective in capturing the variability of the data.

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$



Squared Error Loss in Data Science

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

```python
import pandas as pd
import numpy as np

# URL of the dataset
url = "https://www.riotu-lab.org/cs313/weather.csv"

# Load the dataset
data = pd.read_csv(url)

# Assuming we want to predict 'Temperature' and using the mean temperature as our prediction model
actual_temps = data['Temperature'].values
mean_temp = np.mean(actual_temps)  # Simple prediction model (mean temperature)

# Simulating predicted temperatures based on our simple model (mean temperature for all predictions)
predicted_temps = [mean_temp for _ in range(len(actual_temps))] # simple temperature estimator

# Calculating MSE manually
mse = sum((actual_temps[i] - predicted_temps[i]) ** 2 for i in range(len(actual_temps))) / len(actual_temps)

print(f"Mean Squared Error (using mean temperature as prediction): {mse}")

```

```
Mean Squared Error (using mean temperature as prediction): 20.976107770833337
```

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

ZERO-ONE LOSS

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Zero-One (Binary) Loss Function

- **Zero-One Loss Definition:**
  - **Equation:** $L(y, \hat{y}) = 1_{\{y \neq \hat{y}\}}$
  - A binary loss function where $L(y, \hat{y}) = 1$ if the predicted label $\hat{y}$ does not match the actual label y, and 0 otherwise.

- **Usage in Classification Problems:**
  - Primarily used in classification tasks to measure the accuracy of predictions.
  - Emphasizes the importance of exact matches between predicted and actual categories.
- **Mathematical Concept and Impact:**
  - Directly quantifies the number of misclassifications in a model's predictions.
  - **Objective:** Minimize the total number of incorrect predictions across the dataset.
- **Implications for Model Evaluation:**
  - Provides a clear, intuitive measure of model performance in classification tasks.
  - Influences model selection by prioritizing models with lower misclassification rates.

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

CROSS-ENTROPY LOSS

CS316: INTRODUCTION TO AI AND DATA SCIENCE

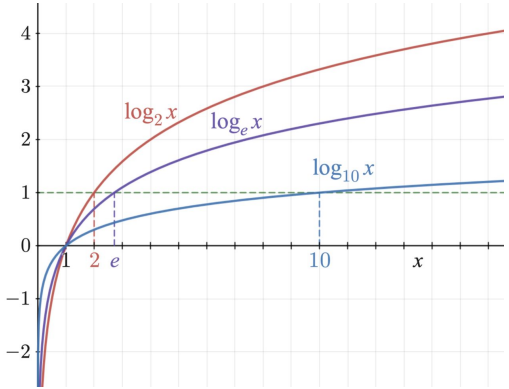
Prof. Anis Koubaa

Cross-Entropy Loss in Data Science

$$H(p, q) = -\sum_{x \in classes} p(x) \log q(x)$$

True probability distribution (one-shot)

Your model's predicted probability distribution



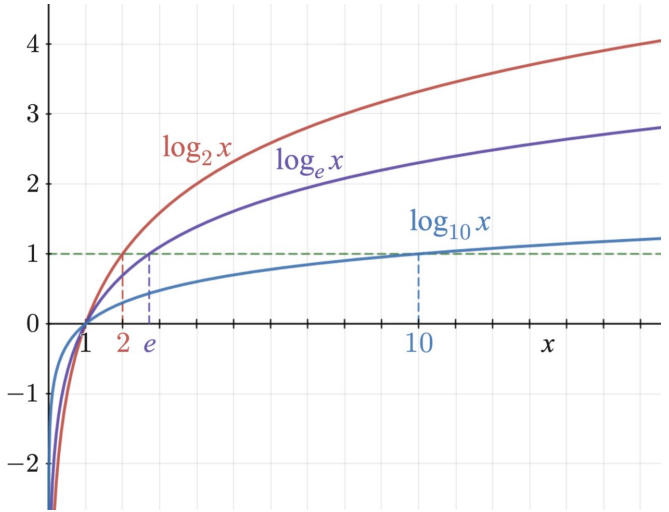
Intuition:

- Cross-entropy measures the "distance" between the true and predicted distributions.
- Minimizing cross-entropy pushes the predicted probabilities closer to the true probabilities.

- **Definition of Cross-Entropy Loss:**
  - **Mathematical Expression:** $-\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$
    - Where y_i is the actual label (0 or 1) and $\hat{y}_i$ is the predicted probability of belonging to class 1.
- **Core Concept:**
  - Measures the performance of a classification model whose output is a probability value between 0 and 1.
  - Emphasizes the distance between actual and predicted probability distributions.

Why Logarithms in Predictions?

- **Makes Big Mistakes Stand Out:** Using logs turns **small errors into big ones**, making it clear when predictions are way off.
- **Keeps Numbers Manageable:** Logs help avoid super tiny numbers by turning multiplication into addition, which is easier for computers to handle.
- **Measures Surprise:** The more unexpected a prediction, the higher the log value. It's like quantifying how surprised we are by a prediction.
- **Helps Computers Learn Better:** Logs adjust how much a prediction needs to change, making learning from mistakes more efficient.



Cross-Entropy Loss in Data Science

- **Importance in Classification Problems:**
  - **Widely Used:** Essential for binary and multi-class classification tasks.
  - **Sensitivity to Misclassification:** Heavily penalizes incorrect predictions with high confidence.
- **Mathematical Foundations:**
  - **Objective:** Minimize the cross-entropy to align the predicted probabilities closely with actual labels.
  - **Binary Classification:** Special focus on the binary case for simplicity.
- **Applications in Data Science:**
  - **Model Evaluation:** Benchmark for assessing model performance in classification.
  - **Deep Learning:** Particularly prevalent in training neural networks.
- **Advantages:**
  - **Efficiency:** Facilitates faster convergence in probabilistic frameworks.
  - **Interpretability:** Direct measure of model's predictive accuracy on the probabilistic scale.

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

PREDICTIVE MODELING RISK

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Risk in Predictive Modeling

Risk is a measure of the expected loss of a prediction function over all possible data points. It represents the average loss that the model would incur if it were applied to all possible data.

*The **loss function** is a function that measures the **difference** between the predicted value and the true value for a single data point.*

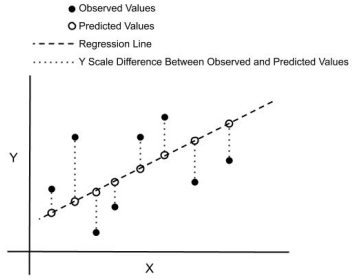
- **Definition**: Risk quantifies the expected loss of a prediction function across the data spectrum, symbolized as:

$$R(g) = \mathbb{E}[L(Y, g(X))]$$

where:

  - $R(g)$ represents the risk associated with prediction function g,
  - $\mathbb{E}$ denotes the expectation over the joint distribution of all possible data points (X, Y),
  - L is the loss function,
  - Y is the true value,
  - $g(X)$ is the predicted value by the function g for input X.

- **Interpretation**: Risk embodies the average loss a model incurs when applied universally, providing a metric for the model's overall performance.



Risk in Predictive Modeling - Math Foundation

- **Risk Definition:**
  - **Equation:** $l(g) = E[\text{Loss}(Y, g(X))]$
  - Explains the expected discrepancy between predicted outcomes $g(X)$ and actual outcomes Y.
- **Adopting Probabilistic Approach:**
  - Models are based on the joint probability density $f(x, y)$ of observing a pair (X, Y).
  - **Joint Probability Density:** $f(x, y)$ encapsulates the randomness and variability in data.

- **Statistical Inference and Risk:**
  - The accuracy of predictions and the inherent uncertainty are governed by the probability laws of the data.
  - **Key Concept:** Data is seen as a realization of a random process or vector, influencing model's predictive performance.

Empirical Risk in Statistical Learning

Empirical Risk: Estimating True Risk

- **Context**: Direct assessment of true risk, $R(g)$, is unfeasible due to the unknown comprehensive distribution of X and Y.
- **Definition**: Empirical Risk is the calculated average loss over a specific dataset, serving as a practical approximation of the true risk.
- **Formula**:

$$\hat{R}(g) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, g(x_i))$$

where:

- $\hat{R}(g)$ denotes the empirical risk for prediction function g,
- N is the number of samples in the dataset,
- L is the loss function,
- (x_i, y_i) represents the observed samples,
- y_i is the true value for the i^{th} sample,
- $g(x_i)$ is the predicted value by function g for the i^{th} input x_i.

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

RISK MINIMIZATION

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Risk Minimization

Risk minimization guides the selection of the model with the least expected loss, using loss functions tailored to specific problem types (classification or regression).

- **Primary Goal**: To identify and fine-tune a prediction function, g, that minimizes the expected loss (risk) across all possible data points in the distribution of X and Y.

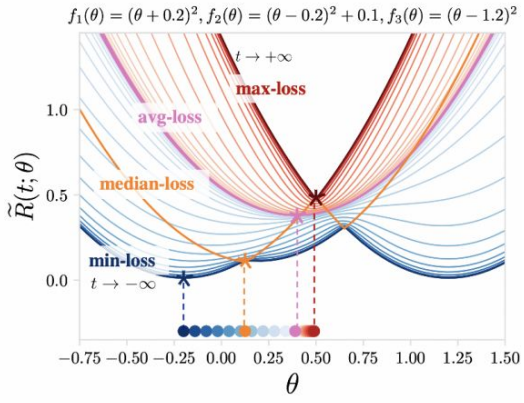
Formal Mathematical Framework

- **True Risk Minimization**:

$$g^* = \arg\min_g R(g)$$

where g^* is the optimal prediction function minimizing the true risk, $R(g)$, defined as:

$$R(g) = \mathbb{E}[L(Y, g(X))]$$



https://blog.ml.cmu.edu/2021/04/02/term/

g(X) is **a model** that we learn from data sample
g*(X) is **the best model** ever that we can learn from data

Empirical Risk Minimization

- **Primary Goal**: To identify and fine-tune a prediction function, g, that minimizes the expected loss (risk) across all possible data points in the distribution of X and Y.

Formal Mathematical Framework

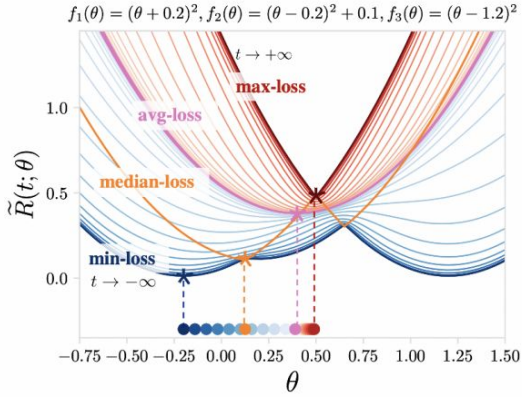
- **Empirical Risk Minimization (ERM)**:

$$\hat{g} = \arg\min_{g} \hat{R}(g)$$

with empirical risk, $\hat{R}(g)$, given by:

$$\hat{R}(g) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, g(x_i))$$

Here, $\hat{g}$ represents the best approximation of g^* based on the available data, aiming to minimize the average loss calculated over the dataset.



https://blog.ml.cmu.edu/2021/04/02/term/

g^(X) is an empirical model that we learn from data sample

Binary Loss Risk Minimization

Objective

Minimize binary loss in classification.

Binary Loss

$$L(Y, g(X)) = \begin{cases} 0, & \text{if } Y = g(X) \\ 1, & \text{otherwise} \end{cases}$$

Risk Minimization

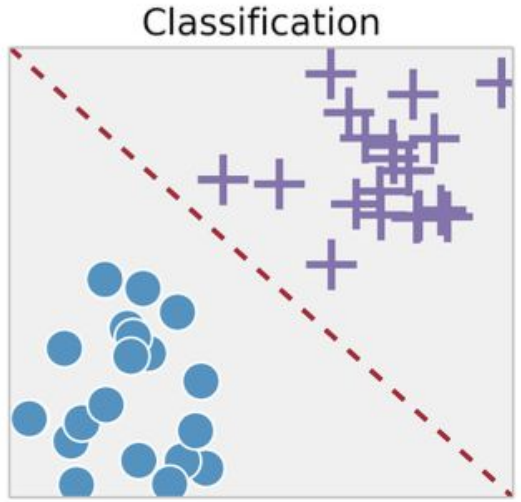
$$g^* = \arg\min_g R(g)$$

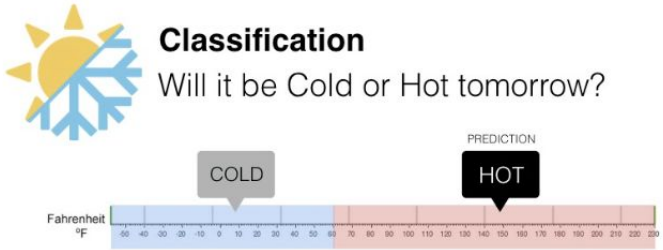
Risk with Binary Loss

$$R(g) = \mathbb{E}[L(Y, g(X))] = \mathbb{E}\left[\mathbf{1}_{\{Y \neq g(X)\}}\right]$$

Optimal Prediction

$$g^* = \arg\min_g \mathbb{E}\left[\mathbf{1}_{\{Y \neq g(X)\}}\right]$$





Regression Risk Minimization

Minimize Mean Squared Error (MSE) in regression.

Mean Squared Error (MSE)

$$L(Y, g(X)) = (Y - g(X))^2$$

Risk Minimization

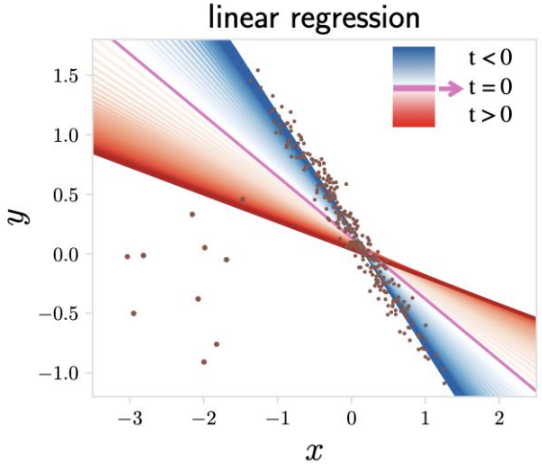
$$g^* = \arg\min_g R(g)$$

Risk with MSE

$$R(g) = \mathbb{E}[(Y - g(X))^2]$$

Optimal Prediction

$$g^* = \arg\min_g \mathbb{E}[(Y - g(X))^2]$$



$$L(x, y; A) = \tfrac{1}{2}(A^T x - y)^2$$

In regression, risk minimization involves finding the parameter vector A that reduces the average discrepancy between the predicted outputs $A^T x$ and the actual outputs y, which is quantified by a loss function L, such as the mean squared error shown in the graph.

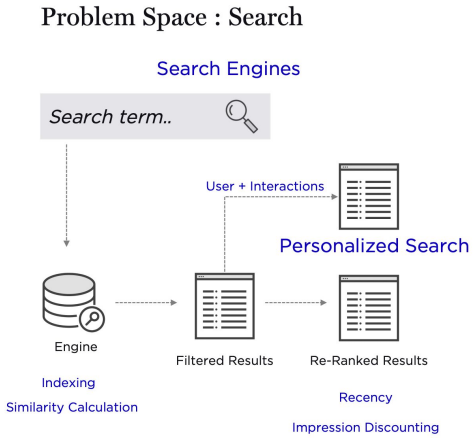
Practice Question

Formulating Loss, Risk, and Minimization in Ranking

Scenario: You are tasked with developing a model to rank a list of items (e.g., search engine results, product recommendations) based on relevance to a user query or preference. The goal is to minimize the discrepancy between the predicted ranking order ($g(X)$) and the true, desired ranking order (Y) provided by expert judgment or user feedback.

Task

1. **Formulate the Loss Function**: Define a loss function suitable for measuring the difference between the predicted ranking order and the actual ranking order.
2. **Express the Risk**: Using your loss function, express the risk associated with your ranking model over all possible ranking scenarios.
3. **Risk Minimization**: Determine the formula to find the optimal ranking function that minimizes this risk.



Ranking Web Pages in Search Engines

Objective: Develop a model to rank web pages based on relevance to user queries, minimizing the discrepancy between predicted and actual desired order.

Inputs (X): Factors Influencing Rankings

- **User Query**: Specific search terms (e.g., "best Italian restaurants in New York").
- **User Context**: Location, search history, time, device, personalization.
- **Page Content**: Keywords, metadata, semantic relevance to query.
- **User Engagement**: Click-through rates, time on page, bounce rates for similar queries.

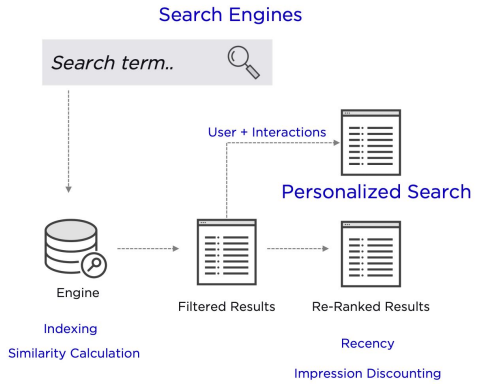
Desired Output (Y): True Ranking Order

- **Based On**:
  - Human relevance judgments.
  - User satisfaction signals (engagement, clicks, non-return rate).

Goal

- **Minimize Discrepancy**: Ensure model's ranking ($g(X)$) aligns closely with true, expert/user-determined order (Y), enhancing search relevance and user satisfaction.

Problem Space : Search



Example

- **Query**: "best Italian restaurants in New York".
- **Ideal Ranking (Y)**: Highly-rated Italian restaurants, based on reviews, ratings, and relevance.

Predicted vs. Actual Order in Ranking Models

Query:
Find the best Restaurant

Predicted Order $g(X)$

- **Definition**: Sequence determined by the ranking model from input features X.
- **Notation**: $g(X_i)$ for predicted rank of item i.

Actual Order Y

- **Definition**: True ranking order, based on expert judgment or user feedback.
- **Notation**: Y_i for actual rank of item i.

Predicted	**g(Restaurant 1)**	**g(Restaurant 2)**
Actual	y(Restaurant 1)	y(Restaurant 2)

Model Evaluation: Predicted vs. Actual Order

- **Pairwise Assessment**:
  - Check if $g(X_i) > g(X_j)$ aligns with $Y_i > Y_j$ for items i, j.
- **Goal**:
  - Minimize $|\{(i,j) : (g(X_i) > g(X_j)) \oplus (Y_i > Y_j), i < j\}|$, the count of misordered pairs.

Predicted vs. Actual Order in Ranking Models

Predicted Order $g(X)$

- **Definition**: Sequence determined by the ranking model from input features X.
- **Notation**: $g(X_i)$ for predicted rank of item i.

Actual Order Y

- **Definition**: True ranking order, based on expert judgment or user feedback.
- **Notation**: Y_i for actual rank of item i.

Query:
Find the best Restaurant

Predicted **g(Restaurant 1)** > **g(Restaurant 2)**

Actual y(Restaurant 1) > y(Restaurant 2)

Actual = Prediction
Loss = 0 (LOW LOSS)

Predicted vs. Actual Order in Ranking Models

Predicted Order $g(X)$

- **Definition**: Sequence determined by the ranking model from input features X.
- **Notation**: $g(X_i)$ for predicted rank of item i.

Actual Order Y

- **Definition**: True ranking order, based on expert judgment or user feedback.
- **Notation**: Y_i for actual rank of item i.

Query:
Find the best Restaurant

Predicted	**g(Restaurant 1)**	>	**g(Restaurant 2)**
Actual	y(Restaurant 1)	<	y(Restaurant 2)

Actual ≠ Prediction
Loss = 1 (HIGH LOSS)

Predicted vs. Actual Order in Ranking Models

Predicted Order $g(X)$

- **Definition**: Sequence determined by the ranking model from input features X.
- **Notation**: $g(X_i)$ for predicted rank of item i.

Actual Order Y

- **Definition**: True ranking order, based on expert judgment or user feedback.
- **Notation**: Y_i for actual rank of item i.

Query:
Find the best Restaurant

Predicted	**g(Restaurant 1)**	<	**g(Restaurant 2)**
Actual	y(Restaurant 1)	>	y(Restaurant 2)

Actual ≠ Prediction
Loss = 1 (HIGH LOSS)

Predicted vs. Actual Order in Ranking Models

Query:
Find the best Restaurant

Predicted Order $g(X)$

- **Definition**: Sequence determined by the ranking model from input features X.
- **Notation**: $g(X_i)$ for predicted rank of item i.

Predicted **g(Restaurant 1)** < **g(Restaurant 2)**

Actual Order Y

- **Definition**: True ranking order, based on expert judgment or user feedback.
- **Notation**: Y_i for actual rank of item i.

Actual y(Restaurant 1) < y(Restaurant 2)

Actual ≠ Prediction
Loss = 0 (LOW LOSS)

Input A	Input B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Loss

Predicted vs. Actual Order in Ranking Models

Predicted Order $g(X)$

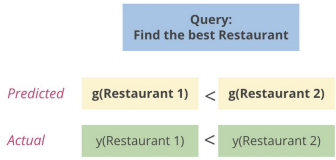
- **Definition**: Sequence determined by the ranking model from input features X.
- **Notation**: $g(X_i)$ for predicted rank of item i.

Actual Order Y

- **Definition**: True ranking order, based on expert judgment or user feedback.
- **Notation**: Y_i for actual rank of item i.

Model Evaluation: Predicted vs. Actual Order

- **Pairwise Assessment**:
  - Check if $g(X_i) > g(X_j)$ aligns with $Y_i > Y_j$ for items i, j.
- **Goal**:
  - Minimize $|\{(i,j) : (g(X_i) > g(X_j)) \oplus (Y_i > Y_j), i < j\}|$, the count of misordered pairs.



Actual ≠ Prediction
Loss = 0 (LOW LOSS)

Input A	Input B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Practice Question: Formulating Loss, Risk, and Minimization in Ranking

Input A	Input B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

HINTS

Guidance

- **Step 1**: Consider using a pairwise loss function, where the loss is calculated based on the number of incorrectly ordered pairs in the predicted ranking compared to the actual ranking.
- **Step 2**: The risk, in this context, is the expected value of the loss function across all possible ranking scenarios, reflecting the average performance of your model in preserving the correct order.
- **Step 3**: Formulate the objective to minimize this risk, identifying the best ranking function (g^*) that leads to the lowest average number of incorrectly ordered pairs.

Practice Question: Formulating Loss, Risk, and Minimization in Ranking

SOLUTIONS

1. **Loss Function** (Pairwise Loss):

$$L(Y, g(X)) = \sum_{i<j} \mathbf{1}_{\{(Y_i > Y_j) \oplus (g(X_i) > g(X_j))\}}$$

   Here, $\mathbf{1}$ is an indicator function that equals 1 if the predicted order of items i and j disagrees with their actual order, and 0 otherwise.

2. **Risk Expression**:

$$R(g) = \mathbb{E}\left[\sum_{i<j} \mathbf{1}_{\{(Y_i > Y_j) \oplus (g(X_i) > g(X_j))\}}\right]$$

3. **Risk Minimization**:

$$g^* = \arg\min_g R(g)$$

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

OPTIMAL PREDICTION FUNCTION OF LEAST SQUARE ERROR

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Optimal Prediction Function for Squared-Error Loss

Theorem 2.1: Optimal Prediction Function for Squared-Error Loss

For the squared-error loss $\text{Loss}(y, \widehat{y}) = (y - \widehat{y})^2$, the optimal prediction function g^* is equal to the conditional expectation of Y given $\boldsymbol{X} = \boldsymbol{x}$:

$$g^*(\boldsymbol{x}) = \mathbb{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}].$$

Proof: Let $g^*(\boldsymbol{x}) = \mathbb{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}]$. For any function g, the squared-error risk satisfies

$$\begin{aligned}
\mathbb{E}(Y - g(\boldsymbol{X}))^2 &= \mathbb{E}[(Y - g^*(\boldsymbol{X}) + g^*(\boldsymbol{X}) - g(\boldsymbol{X}))^2] \\
&= \mathbb{E}(Y - g^*(\boldsymbol{X}))^2 + 2\mathbb{E}[(Y - g^*(\boldsymbol{X}))(g^*(\boldsymbol{X}) - g(\boldsymbol{X}))] + \mathbb{E}(g^*(\boldsymbol{X}) - g(\boldsymbol{X}))^2 \\
&\geqslant \mathbb{E}(Y - g^*(\boldsymbol{X}))^2 + 2\mathbb{E}[(Y - g^*(\boldsymbol{X}))(g^*(\boldsymbol{X}) - g(\boldsymbol{X}))] \\
&= \mathbb{E}(Y - g^*(\boldsymbol{X}))^2 + 2\mathbb{E}\left\{(g^*(\boldsymbol{X}) - g(\boldsymbol{X}))\mathbb{E}[Y - g^*(\boldsymbol{X}) \mid \boldsymbol{X}]\right\}.
\end{aligned}$$

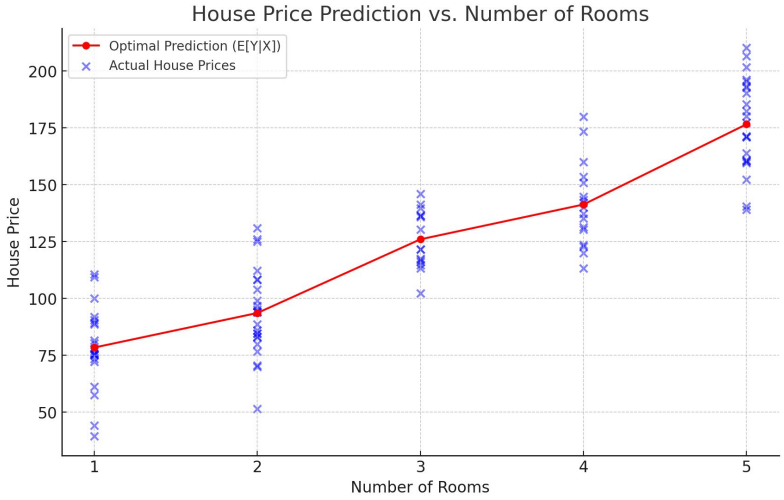
In the last equation we used the tower property. By the definition of the conditional expectation, we have $\mathbb{E}[Y - g^*(\boldsymbol{X}) \mid \boldsymbol{X}] = 0$. It follows that $\mathbb{E}(Y - g(\boldsymbol{X}))^2 \geqslant \mathbb{E}(Y - g^*(\boldsymbol{X}))^2$, showing that g^* yields the smallest squared-error risk. □

https://people.smp.uq.edu.au/DirkKroese/DSML/DSML.pdf

Optimal Prediction Function for Squared-Error Loss

- **Definition:** The optimal prediction function $f^*(X)$ minimizes expected squared error loss in regression, defined as $f^*(X) = \mathbb{E}[Y|X]$.
- **Intuition:** Predicting the conditional mean of the target Y given predictors X minimizes the squared differences between actual and predicted Y values.

- **Mathematical Rationale:** Squared error loss function penalizes deviations quadratically; conditional expectation is the value minimizing this penalty.



Imagine we have data on 3 houses of 1,000 square feet, priced at $100,000, $120,000, and $110,000.

- **Average Price for 1,000 square feet houses:** $\frac{100,000+120,000+110,000}{3} = 110,000$

Optimal Prediction Function for Squared-Error Loss

Theorem 2.1: Optimal Prediction with Squared-Error Loss

Objective: Understand the optimal prediction function g^* under squared-error loss.

Squared-Error Loss

- **Formula**: $\text{Loss}(y, \hat{y}) = (y - \hat{y})^2$
- **Goal**: Minimize the discrepancy between actual y and predicted $\hat{y}$.

Optimal Prediction Function

- **Definition**: $g^*(x) = E[Y|X = x]$
  - The optimal predictor g^* equals the conditional expectation of Y given $X = x$.

Theorem 2.1: Optimal Prediction Function for Squared-Error Loss

For the squared-error loss $\text{Loss}(y, \widehat{y}) = (y - \widehat{y})^2$, the optimal prediction function g^* is equal to the conditional expectation of Y given $\boldsymbol{X} = \boldsymbol{x}$:

$$g^*(\boldsymbol{x}) = \mathbb{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}].$$

PROOF. Optimal Prediction Function for SEL

Proof Overview

1. **Starting Point**: Consider any prediction function g and compare its risk to g^*.
2. **Expand Squared-Error Risk**:
   - $E(Y - g(X))^2 = E[(Y - g^*(X) + g^*(X) - g(X))^2]$
3. **Application of Expectation Properties**:
   - Decompose into $E(Y - g^*(X))^2 + 2E[(Y - g^*(X))(g^*(X) - g(X))] + E(g^*(X) - g(X))^2$
4. **Simplification**:
   - Utilizes the tower property and the fact that $E[Y - g^*(X)|X] = 0$ to show $E(Y - g(X))^2 > E(Y - g^*(X))^2$.
5. **Conclusion**: This demonstrates g^* minimizes the squared-error risk, confirming it as the optimal prediction function.

PROOF. Optimal Prediction Function for SEL

$$E[E[Y|X]] = E[Y]$$

The Proof Explained

1. **Start with the squared-error risk for any function g:**
   $E(Y - g(X))^2$
2. **Expand this expression by adding and subtracting $g^*(X)$:**
   $E[(Y - g^*(X) + g^*(X) - g(X))^2]$
3. **Apply the binomial expansion to the squared term:**
   $E(Y - g^*(X))^2 + 2E[(Y - g^*(X))(g^*(X) - g(X))] + E(g^*(X) - g(X))^2$
4. **Now, focus on the middle term, where the tower property is used:**
   The tower property, or law of total expectation, states that for any random variables Y, X, and a function $g(X)$:
   $E[E[Y|X]] = E[Y]$
   Applying this property, the term $2E[(Y - g^*(X))(g^*(X) - g(X))]$ can be seen as the expectation of the product of $g^*(X) - g(X)$ and the conditional expectation $E[Y - g^*(X)|X]$, since $g^*(X) - g(X)$ is a function of X.
5. **By the definition of $g^*(X)$ as the conditional expectation of Y given X, we have:**
   $E[Y - g^*(X)|X] = 0$
   This is because $g^*(X)$ is the expectation of Y given X, so their difference, on average, given X, is zero.
6. **Substitute back into the equation:**
   Since $E[Y - g^*(X)|X] = 0$, the term $2E[(Y - g^*(X))(g^*(X) - g(X))]$ collapses to 0, due to the multiplication by zero inside the expectation. This reduces our equation to:
   $E(Y - g(X))^2 = E(Y - g^*(X))^2 + E(g^*(X) - g(X))^2$
7. **The final inequality:**
   $E(Y - g(X))^2 > E(Y - g^*(X))^2$
   This follows because $E(g^*(X) - g(X))^2$ is always non-negative (it's a squared term), and it is strictly positive unless $g(X) = g^*(X)$ almost surely.

Understanding Conditional Expectation in Squared-Error Minimization

Key Concept:

- The optimal prediction function $g^*(X) = \mathbb{E}[Y|X]$ minimizes the squared error, $E(Y - g(X))^2$, for any prediction function $g(X)$.

Middle Term Analysis:

- Consider: $2E[(Y - g^*(X))(g^*(X) - g(X))]$
  - This term involves the product of two differences:
    1. $Y - g^*(X)$: The difference between actual outcomes and their optimal predictions.
    2. $g^*(X) - g(X)$: The difference between optimal predictions and any other prediction function.

Application of Conditional Expectation:

- **Critical Insight:** $E[Y - g^*(X)|X] = 0$ by the definition of $g^*(X)$ as the conditional expectation.
- **Result:** The middle term simplifies to zero because it involves the expectation of a product where one factor, conditioned on X, is zero.

Conclusion:

- Any deviation from the optimal prediction $g^*(X)$ increases the squared error, underscoring the efficiency of $g^*(X)$ as the best predictor under squared-error loss.

Appendix. Optimal Prediction Function for SEL

Understanding Conditional Expectation in Squared-Error Minimization

Key Concept:

- The optimal prediction function $g^*(X) = \mathbb{E}[Y|X]$ minimizes the squared error, $E(Y - g(X))^2$, for any prediction function $g(X)$.

Middle Term Analysis:

- Consider: $2E[(Y - g^*(X))(g^*(X) - g(X))]$
  - This term involves the product of two differences:
    1. $Y - g^*(X)$: The difference between actual outcomes and their optimal predictions.
    2. $g^*(X) - g(X)$: The difference between optimal predictions and any other prediction function.

Application of Conditional Expectation:

- **Critical Insight:** $E[Y - g^*(X)|X] = 0$ by the definition of $g^*(X)$ as the conditional expectation.
- **Result:** The middle term simplifies to zero because it involves the expectation of a product where one factor, conditioned on X, is zero.

Conclusion:

- Any deviation from the optimal prediction $g^*(X)$ increases the squared error, underscoring the efficiency of $g^*(X)$ as the best predictor under squared-error loss.

Appendix. Tower Property Definition

Objective: To grasp the essence of the Tower Property and its application in simplifying complex expectation calculations in probability and statistics.

Problem Statement:

- In statistical analysis, calculating the expectation of a random variable Y directly can be challenging.
- The Tower Property provides a systematic approach to decompose these calculations into more manageable parts, especially when dealing with conditional expectations.

Tower Property Definition:

- **Formula**: $E[E[Y|X]] = E[Y]$
- **Explanation**: The expectation of the conditional expectation of Y given X equals the overall expectation of Y.
- **Interpretation**: This property allows for the simplification of the expected value of Y by first conditioning on another variable X and then averaging these conditional expectations.

Key Concepts:

- **Conditional Expectation**: $E[Y|X]$ represents the expected value of Y given that X has occurred.
- **Unconditional Expectation**: $E[Y]$ is the expected value of Y without any condition.

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

SUPERVISED LEARNING
VS.
UNSUPERVISED LEARNING

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Empirical Prediction vs Optimal Prediction

- **Decomposition Formula:**

  $Y = g^*(x) + \epsilon(x)$

  - Where Y is the response given $X = x$.
- **Random Deviation:**
  - $\epsilon(x)$ represents deviation from the conditional mean.
  - $\mathbb{E}[\epsilon(x)] = 0$.
- **Conditional Variance:**

  $Var[\epsilon(x)] = v^2(x)$

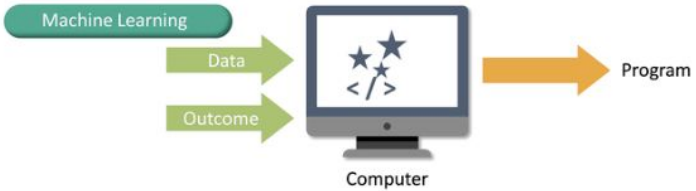
  - $v(x)$ is an unknown positive function.
- **Distribution:**
  - Distribution of $\epsilon(x)$ is unspecified.
- **Optimal Prediction Challenge:**
  - g^* is ideal but often unknown due to the joint distribution of X, Y being unknown.

Learning the Optimal Prediction Function

Problem Statement

- **Challenge of Optimal Prediction:**
  - Optimal function g^* depends on the unknown joint distribution of X, Y.
  - Practical approach uses a finite sample $T = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$, termed as the training set.
- **Training Set Notation:**
  - Random training set: T.
  - Deterministic outcome: τ or τ_n (to emphasize size).

$$\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$$

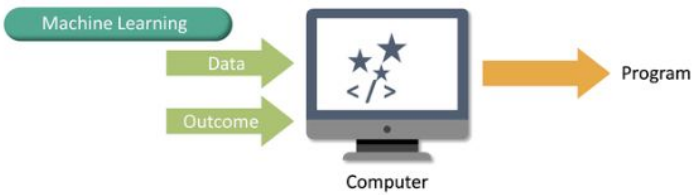


Dataset vs. Sample

number	student_id	gender	major01	major02	project	final	term	year
1	120060257	male	0.25	0	0	13	fall	2015
2	211110888	female	15.22	10.75	14.67	27	fall	2015
3	211210095	male	13.25	15.43	16.25	25	fall	2015
4	212110756	female	5.91	1.25	5	7	fall	2015
5	212110889	male		13	6.67	27	fall	2015
6	212110977	male	10.31	12.45		33	fall	2015
7	213110290	female	9.13	14.52	16.33	30	fall	2015
8	213110419	female	16	12.63	16.67	23	fall	2015
9	213110460	male		6.38	3		fall	2015
10	213110964	female	11.5	7.5	11.67	15	fall	2015
11	213111059	male	9.91	9.38	5.33	30	fall	2015
12	213111074	male	10.56	12	11.33	20	fall	2015
13	213111128	female	19	17.25	18	36	fall	2015
14	213210082	male	20	18.88	18.67	37	fall	2015
15	213210083	male	13.75	8	18	33	spring	2018
16	213210084	male	8.19	7.5	15.75	31	spring	2018
17	213210085	male	11		12	14	spring	2018
18	213210086	female	8.94	1	12	22	spring	2018
19	213210087	male	11.19	4.5	13	24	spring	2018
20	213210088	female	11	8.75	15	26	spring	2018
21	213210089	male	9.19		15	28	spring	2018
22	213210090	male	12.69	8	16	34	spring	2018
23	213210091	female	13.63	12	16	38	spring	2018
24	213210092	male	11.88	10.25	15.75	36	spring	2018
25	213210093	female		13	18	37	spring	2018
26	213210094	male	18.88	18.25	20	35	spring	2018
27	213210095	female	15.81	7.5	12	28	spring	2018
28	213210096	male	18.13	17	20	39	spring	2018
29	213210097	male	19.88	17.5	17.5	38	spring	2018
30	213210098	male	8.5	9.75		26	spring	2018
31	213210099	female	16.63	13	16	35	spring	2018
32	213210100	male	17.63	17.5	19	38	spring	2018
33	213210101	female	9.56		19	31	spring	2018
34	213210102	male	17.19	12.5	18.25	38	spring	2018
35	213210103	male	14.63	19	16	36	spring	2018
36	213210104	male	11.75	8.75	15	29	spring	2018
37	213210105	male	15.31		16	34	spring	2018

Learning the Optimal Prediction Function

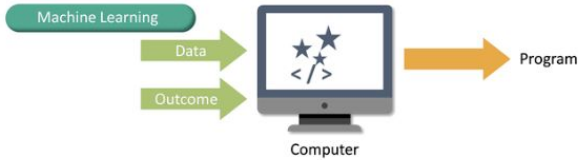
TEACHER-LEARNER PARADIGM



- **Educational Metaphor:**
  - Conceptualize g_T as a learner, trained via T to mimic $g^* : x \mapsto y$.
  - Training facilitated by a "teacher" providing n examples, guiding predictions on unseen data.
- **Supervised Learning Context:**
  - Engages in learning the function linking x (features) to y (response) under supervision.
  - Emphasizes on prediction, leveraging explanatory variables.

SUPERVISED LEARNING: CONCEPT AND APPLICATIONS

1. **Supervised Learning Defined:**
   - Learning the relationship between features (x) and response (y) with examples provided by a "teacher".
   - Focus on predicting y from x, using a set of explanatory variables.
2. **Learning Process:**
   - Objective: Learn unknown optimal function g^* from training data.
   - Approach: Use training set $T = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ to construct approximation g_T.
3. **Training Set Dynamics:**
   - Random set: T (training).
   - Deterministic outcome: τ (observed sample).
4. **Teacher-Learner Metaphor:**
   - g_T: Learner, trained to infer functional relationship $g^* : x \mapsto y$.
   - Teacher provides n examples, enabling g_T to predict on new inputs.

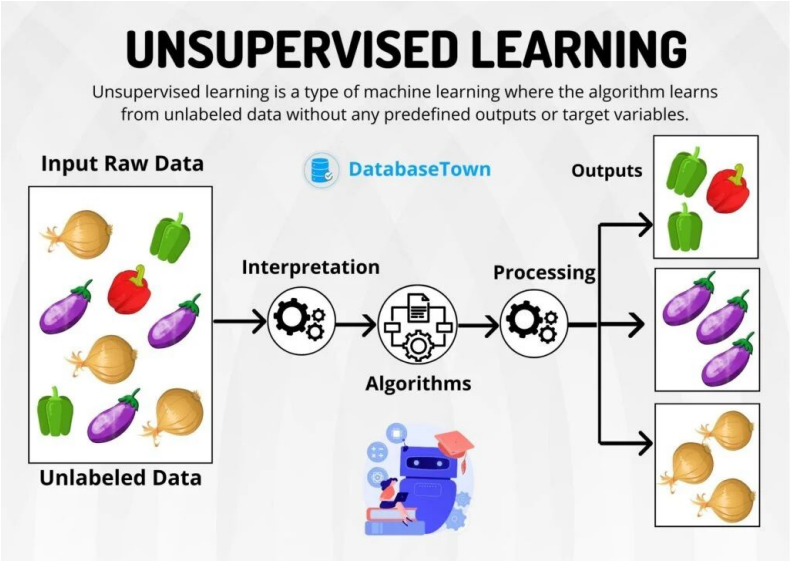


Significance of Conditional PDF:

- Understanding $f(y|x)$ allows for the determination of $g^*(x)$, facilitating prediction and analysis.

UNSUPERVISED LEARNING: EXPLORING DATA STRUCTURE

1. **Unsupervised Learning Overview:**
   - Focuses on learning the structure of an unknown distribution $f(x)$ without distinguishing between response and explanatory variables.
2. **Objective and Approach:**
   - Goal: Discover the intrinsic structure of data by approximating $f(x)$ with $g(x)$.
   - Risk Measurement: $l(g) = \mathbb{E}\,\text{Loss}(f(X), g(X))$.

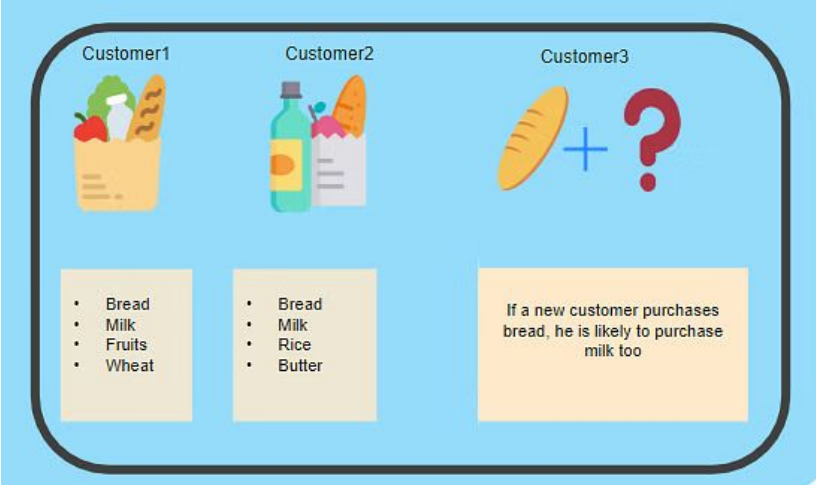


https://databasetown.com/unsupervised-learning-types-applications/

UNSUPERVISED LEARNING: EXPLORING DATA STRUCTURE

Example: Customer Purchase Patterns

- Analyze purchasing behaviors in a grocery shop with 100 items.
- Feature vector $x \in \{0,1\}^{100}$ represents items bought.
- Task: Identify patterns from binary vectors of purchases.



https://www.simplilearn.com/

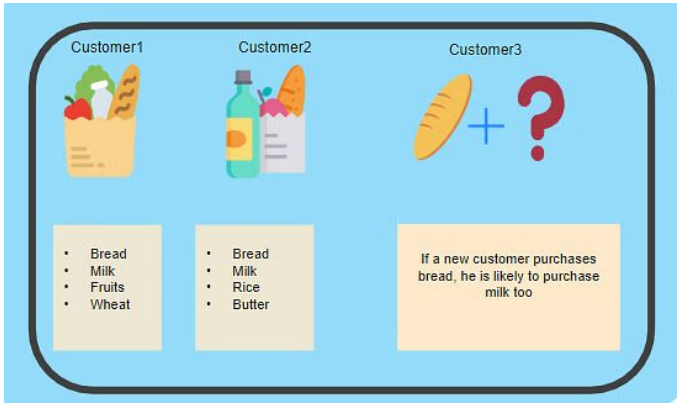
Data Representation:

- Feature vector $x \in \{0,1\}^{100}$: Indicates the presence (1) or absence (0) of 100 different grocery items in a customer's purchase.

UNSUPERVISED LEARNING: EXPLORING DATA STRUCTURE

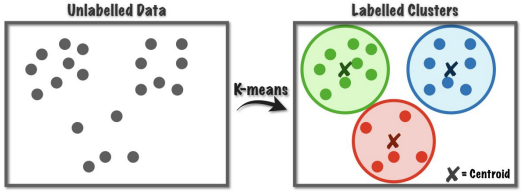
```python
binary_representation = [
    # Milk, Bread, Eggs
    [1, 1, 0],  # Transaction 1: Milk, Bread
    [0, 1, 1],  # Transaction 2: Bread, Eggs
    [1, 0, 1],  # Transaction 3: Milk, Eggs
    [1, 1, 1],  # Transaction 4: Milk, Bread, Eggs
    [0, 1, 0],  # Transaction 5: Bread
]

# Items for header
items_header = ['Milk', 'Bread', 'Eggs']

# Displaying as a simple text-based table
table_header = ' | '.join(items_header)
print(table_header)
print('-' * len(table_header))

for row in binary_representation:
    print(' | '.join(str(x) for x in row))
```

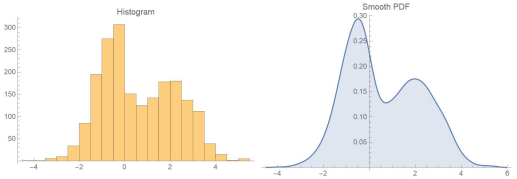
	Milk	Bread	Eggs	
0	1	1	0	Transaction 1
1	0	1	1	Transaction 2
2	1	0	1	Transaction 3
3	1	1	1	Transaction 4
4	0	1	0	Transaction 5



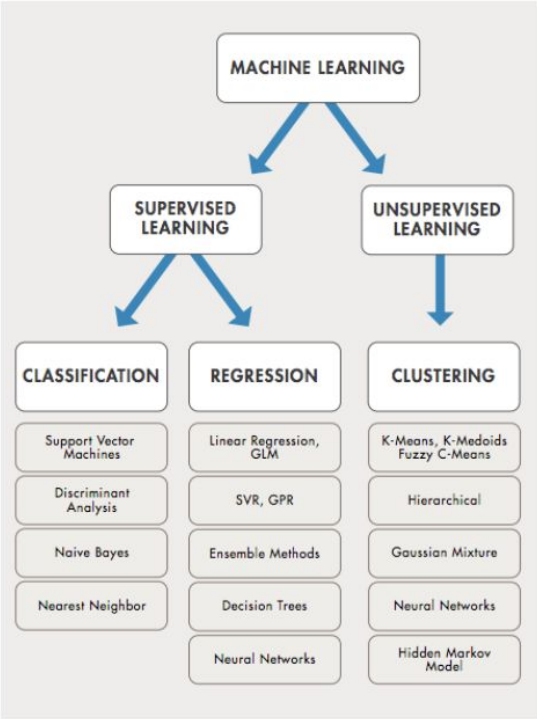
UNSUPERVISED LEARNING: EXPLORING DATA STRUCTURE

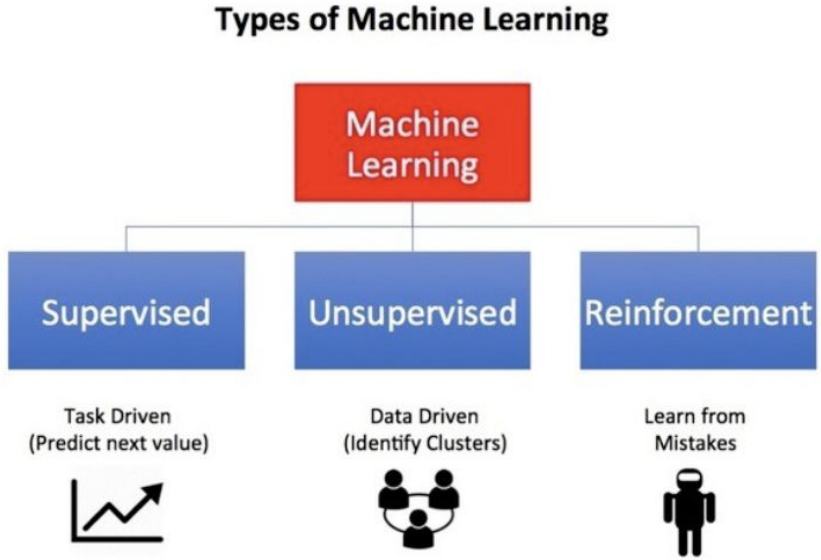
4. **Challenges in Unsupervised Learning:**
   - Lack of clear success metrics due to the absence of a "teacher" or labeled examples.
   - Difficulty in evaluating the performance of unsupervised learning algorithms.
5. **Key Methodologies:**
   - **Clustering:** Group similar data points.
   - **Principal Component Analysis (PCA):** Reduce dimensionality while preserving variance.
   - **Kernel Density Estimation:** Estimate the probability density function of a random variable.
6. **Significance:**
   - Enables discovery of hidden patterns and structures in data without predefined labels.
   - Essential for exploratory data analysis and understanding complex datasets.



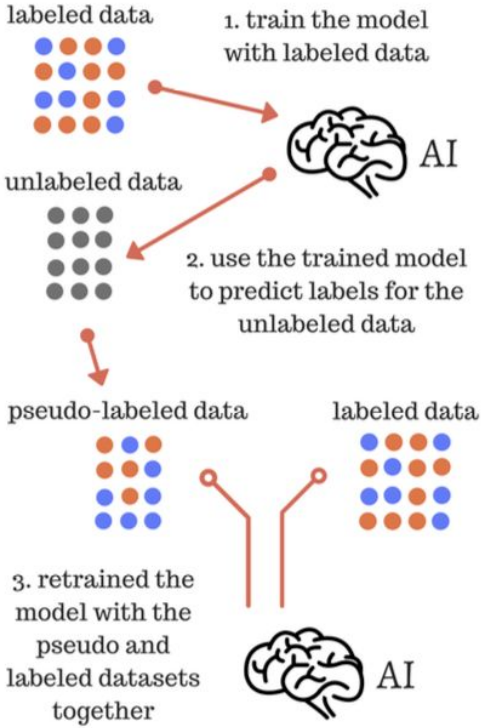


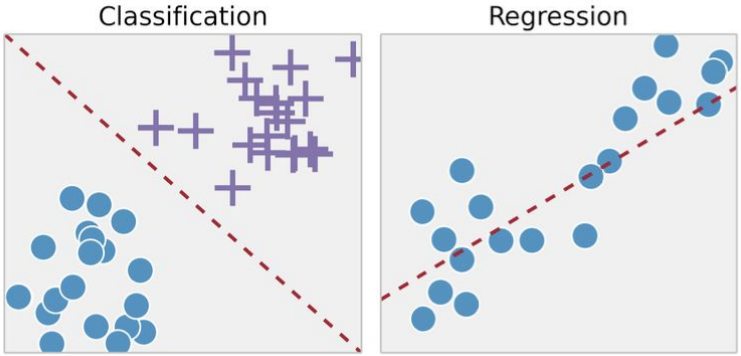
Supervised vs. Unsupervised Learning

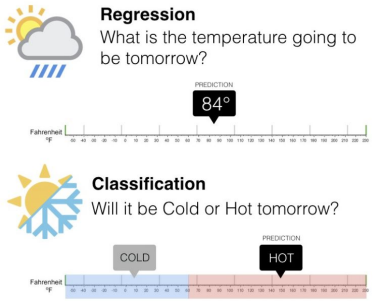




Supervised Learning





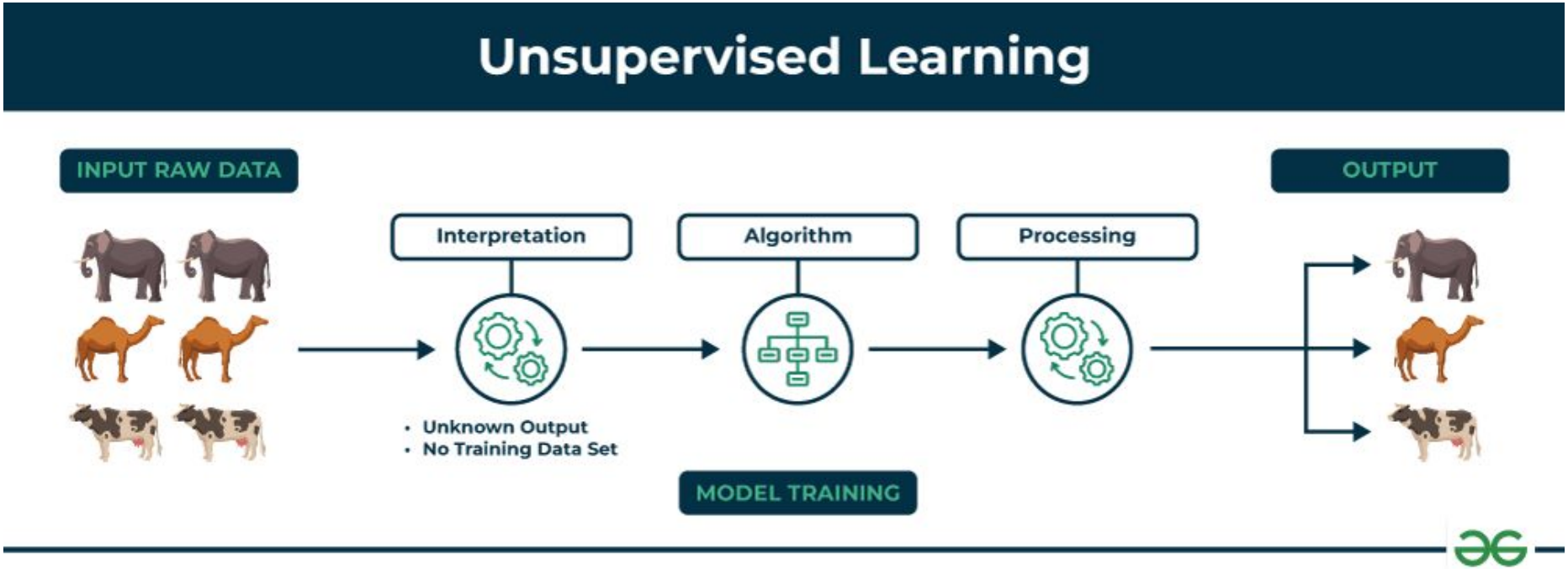


Regression and **classification** are categorized under the same umbrella of **supervised machine learning**.

The main difference the **output variable:**

- **regression** output is **numerical** (or continuous)
- **classification** output is **categorical** (or discrete)

Unsupervised Learning



CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

TRAIN vs. TEST LOSS

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Training Loss

Training Loss Definition:

$$l_T(g) = \frac{1}{n}\sum_{i=1}^{n} \text{Loss}(Y_i, g(X_i))$$

not possible to compute its real risk (Expected Loss) `R(g)`

$$R(g) = \mathbb{E}[L(Y, g(X))]$$

1. **Empirical Risk Approximation:**
   - **Training Loss ($l_T(g)$)**: Empirical risk calculated as the sample average loss:
     $l_T(g) = \frac{1}{n}\sum_{i=1}^{n} \text{Loss}(Y_i, g(X_i))$
   - Represents an unbiased estimate of the risk (expected loss) for g based on training data.

2. **Identifying the Optimal Prediction Function:**
   - **Selection Process**: Choose g_{GT} from a function set G that minimizes training loss:
     $g_{GT} = \arg\min_{g \in G} l_T(g)$
   - **Example of G**: The set of linear functions, where $g : x \mapsto \beta^\top x$ for a vector β.

Training Risk Minimization over set of Functions

- **Equation of Training Loss:**

  $l_T(g) = \sum_{i=1}^{n} \text{Loss}(Y_i, g(X_i))$

  Leverages loss function to quantify discrepancy between predictions and actual values.

- **Optimal Function Selection:**

  $g_{G_T} = \text{argmin}_g l_T(g)$

  Identifies function within class G that minimizes training loss.

- **Linear Function Class Example:**

  $g : x \mapsto \beta^\top x \quad (\text{where } \beta \text{ is a real-valued vector})$

  Illustrates a simple yet powerful model form, emphasizing linearity in parameters.

- **Minimizing Training Loss:**
  - Minimizing over $g \in G$ ensures meaningful optimization within a specified function class.
  - Minimizing over all possible functions g invites overfitting, sacrificing generalizability.

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

OVERFITTING

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Understanding Overfitting

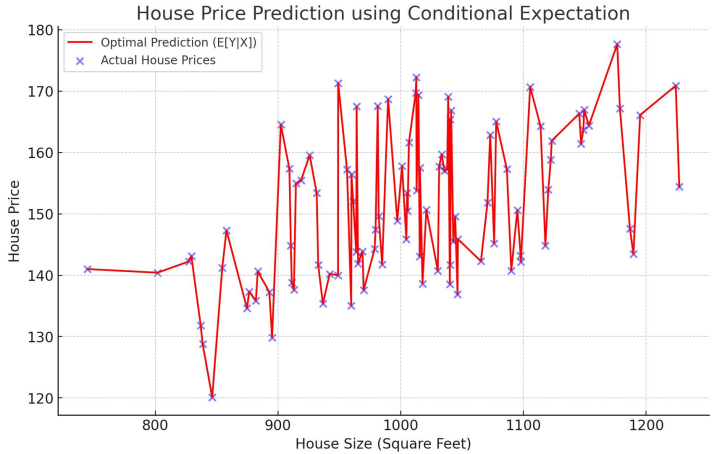
- **Overfitting Defined:**

  Overfitting occurs when a model learns the training data too closely, capturing noise and outliers, which results in poor performance on unseen data.

- **Training Loss Indicator:**

  $$\text{Training Loss} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

  Approaches zero in overfitting scenarios.



Overfitting Consequences

- **Generalization Failure:**

  Overfitting results in a substantial gap between training accuracy and the accuracy on novel, unseen data.

- **Quantifying Generalization Risk:**

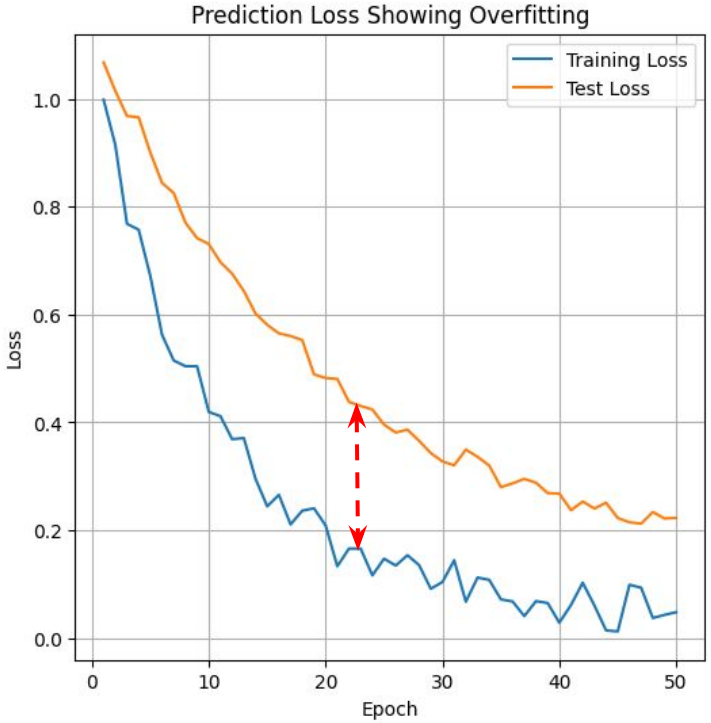
  $$\text{Generalization Risk} = \mathbb{E}[\text{Loss}(Y, g_{G_\mathcal{T}}(X))]$$

  $$l(g_{G_\mathcal{T}}) = \sum_{x,y} \text{Loss}(y, g_{G_\mathcal{T}}(x)) f(x, y)$$

  (In the continuous case, replace the sum with an integral.)

- **Impact of Overfitting on Generalization Risk:**

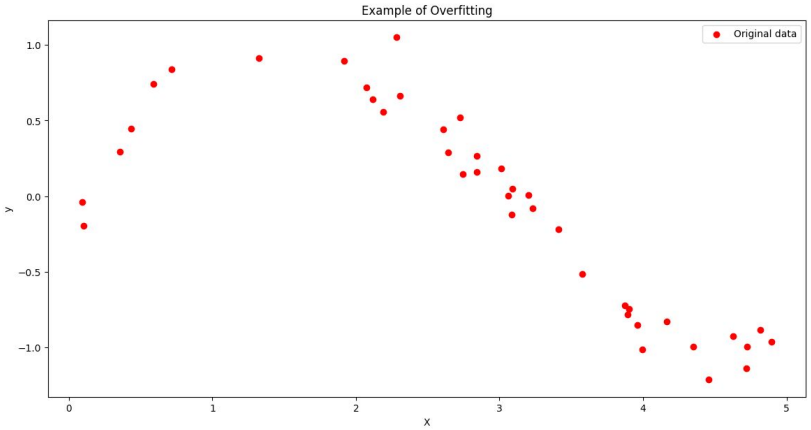
  Overfitting is marked by elevated generalization risk, signifying a model's poor generalization capability. This misalignment underscores the model's inability to accurately predict outcomes for unseen data, demonstrating the critical need for models that can generalize well beyond their training dataset.



Overfitting in Model Training

1. **Training Loss Minimization:**
   - Objective: Find g_{GT} within a function class G that minimizes training loss.
   - **Formula:**

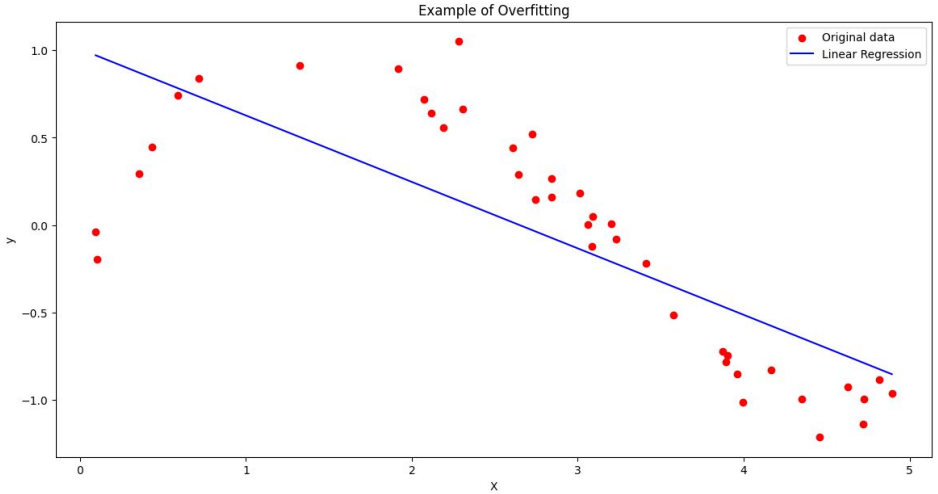
     $$g_{GT} = \arg\min_{g \in G} \frac{1}{n} \sum_{i=1}^{n} \text{Loss}(Y_i, g(X_i))$$
2. **Overfitting:**
   - Occurs when g_{GT} fits the training data too perfectly, leading to zero training loss.
   - **Implication:** Poor prediction on new, independent data due to lack of generalization.
3. **Example of Overfitting:**
   - A model where $g(X_i) = Y_i$ for all i in the training set, achieving zero squared-error training loss.



Overfitting in Model Training

1. **Training Loss Minimization:**
   - Objective: Find g_{GT} within a function class G that minimizes training loss.
   - **Formula:**

     $$g_{GT} = \arg\min_{g \in G} \frac{1}{n} \sum_{i=1}^{n} \text{Loss}(Y_i, g(X_i))$$
2. **Overfitting:**
   - Occurs when g_{GT} fits the training data too perfectly, leading to zero training loss.
   - **Implication:** Poor prediction on new, independent data due to lack of generalization.
3. **Example of Overfitting:**
   - A model where $g(X_i) = Y_i$ for all i in the training set, achieving zero squared-error training loss.

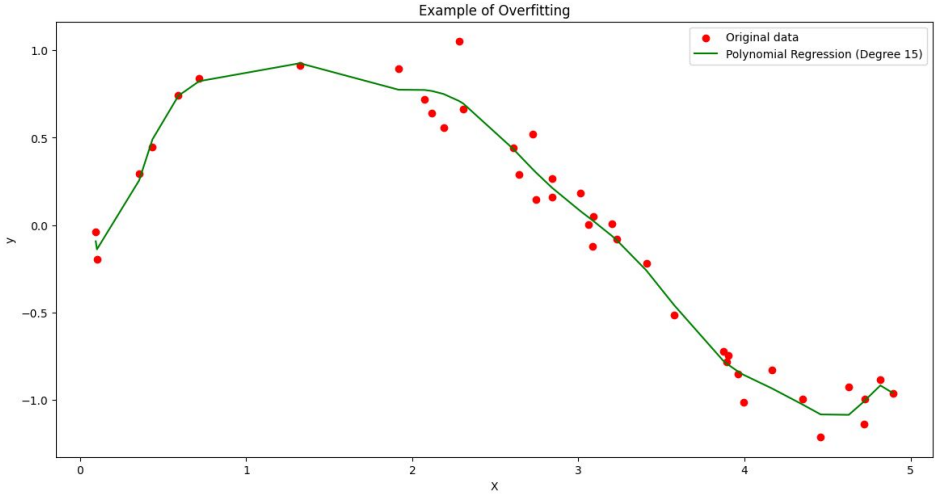


MSE = 0.1864

Overfitting in Model Training

1. **Training Loss Minimization:**
   - Objective: Find g_{GT} within a function class G that minimizes training loss.
   - **Formula:**

     $$g_{GT} = \arg\min_{g \in G} \frac{1}{n} \sum_{i=1}^{n} \text{Loss}(Y_i, g(X_i))$$
2. **Overfitting:**
   - Occurs when g_{GT} fits the training data too perfectly, leading to zero training loss.
   - **Implication:** Poor prediction on new, independent data due to lack of generalization.
3. **Example of Overfitting:**
   - A model where $g(X_i) = Y_i$ for all i in the training set, achieving zero squared-error training loss.

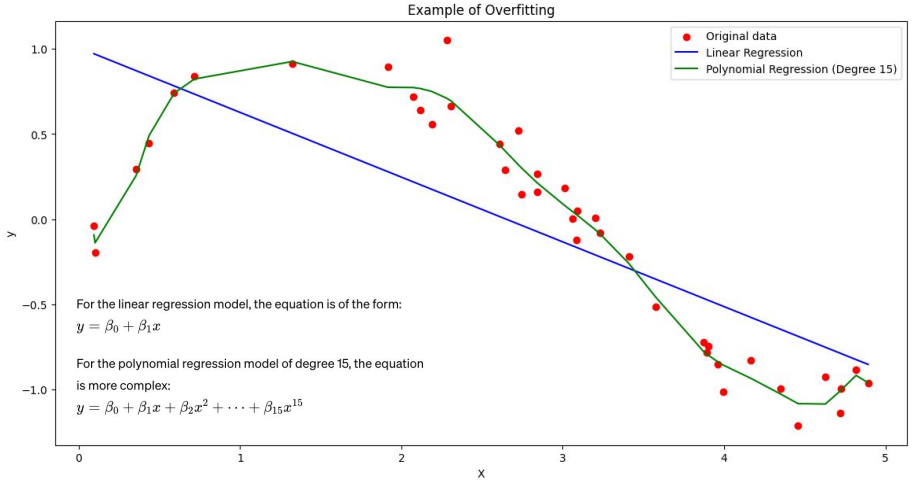


MSE = 0.010

Overfitting in Model Training

1. **Training Loss Minimization:**
   - Objective: Find g_{GT} within a function class G that minimizes training loss.
   - **Formula:**

     $$g_{GT} = \arg\min_{g \in G} \frac{1}{n} \sum_{i=1}^{n} \text{Loss}(Y_i, g(X_i))$$
2. **Overfitting:**
   - Occurs when g_{GT} fits the training data too perfectly, leading to zero training loss.
   - **Implication:** Poor prediction on new, independent data due to lack of generalization.
3. **Example of Overfitting:**
   - A model where $g(X_i) = Y_i$ for all i in the training set, achieving zero squared-error training loss.



Linear Regression Equation:

$$y = 1.01 - 0.38x$$

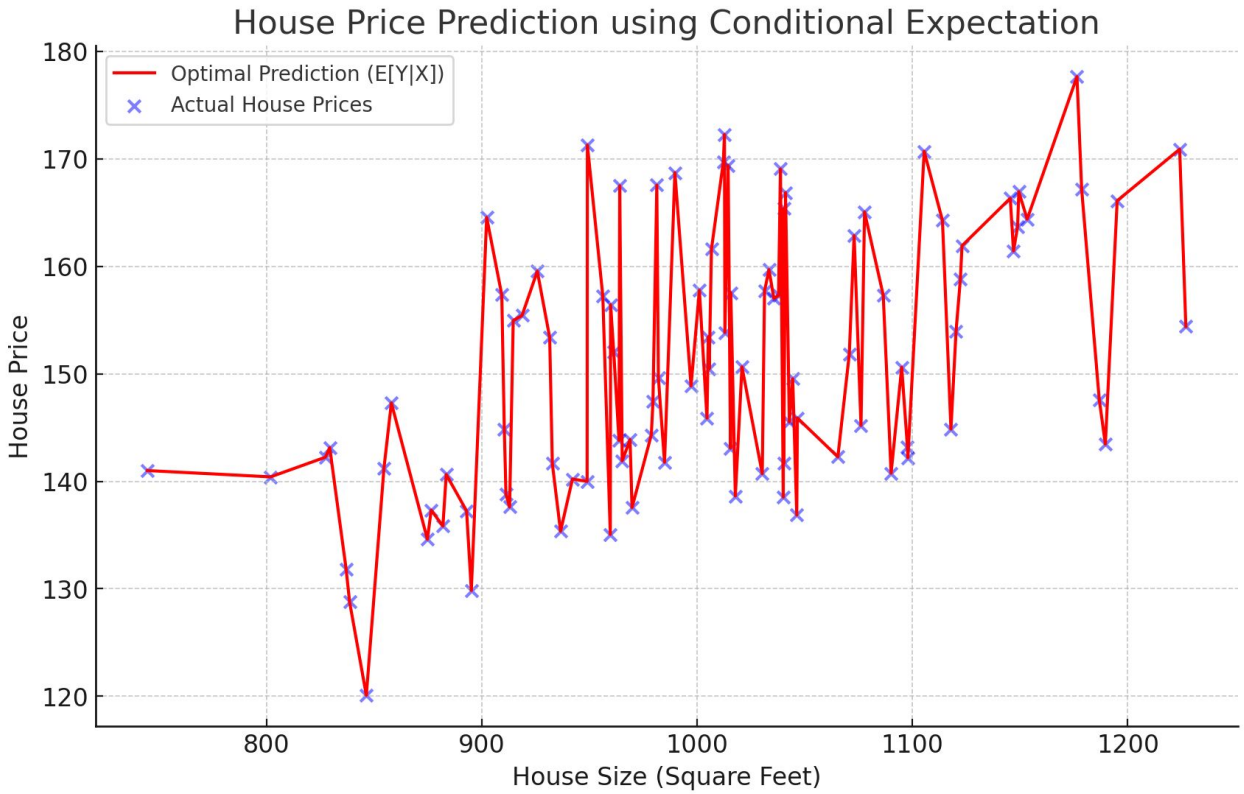
Polynomial Regression Equation (truncated):

$$y = 1.40e+00 - 2.75e+01x + 1.55e+02x^2 - 3.77e+02x^3 + 4.60e+02x^4 - 1.99e+02x^5 - \cdots$$

Mean Squared Error (MSE) Comparison:

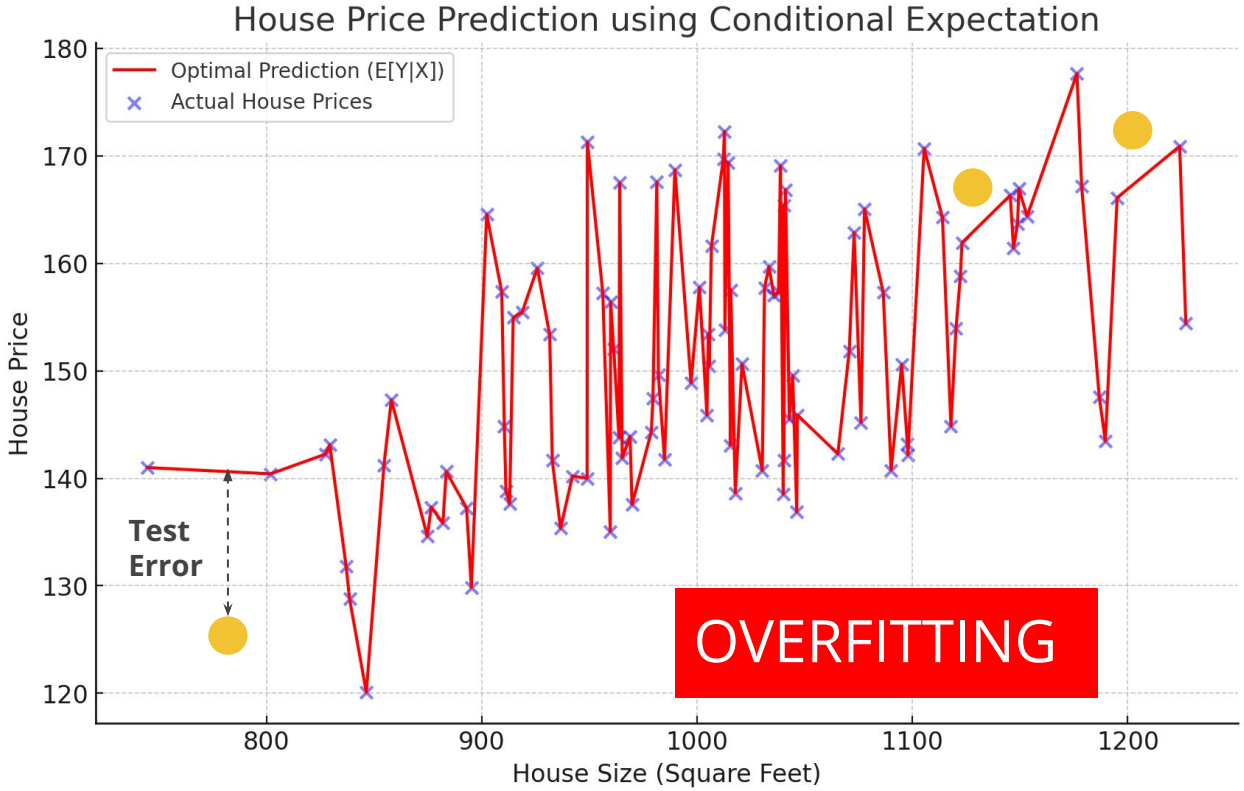
- **Linear Regression MSE:** 0.186
- **Polynomial Regression (Degree 15) MSE:** 0.011

What is the MSE Train Loss?



Train Loss = 0

What is the MSE Train/TEST Loss?



Train Loss = 0

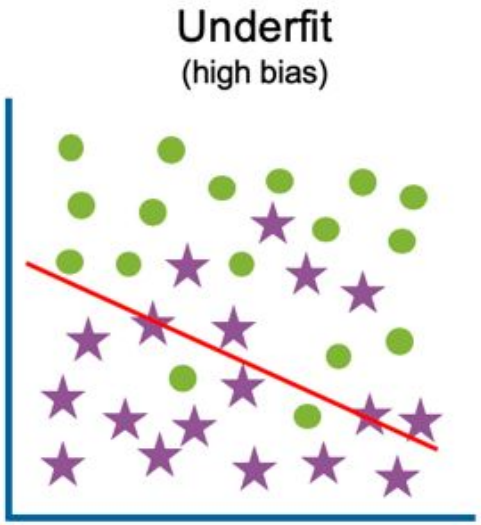
Unseen (Test) Data

Test Loss > 0

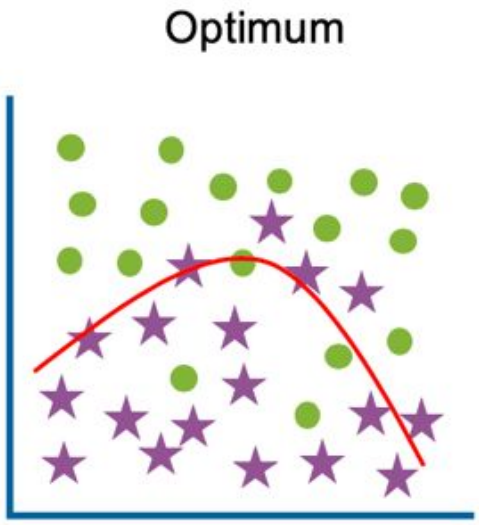
Training Goal

The ultimate goal is **not merely to minimize training loss** but to build models that **generalize well** to **unseen data**.

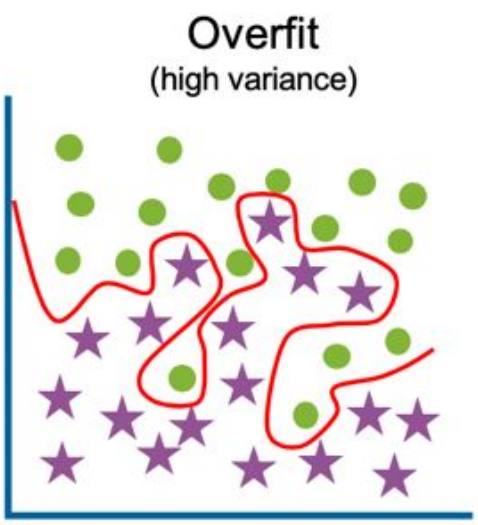
Overfitting | Optimal | Underfit



High training error
High test error



Low training error
Low test error



Low training error
High test error

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

Generalization Risk

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Generalization Risk and Overfitting

- **Generalization Risk Formula:**
  $l_\tau(g_\tau^G) = \mathbb{E}[\text{Loss}(Y, g_\tau^G(X))]$
  - *Clarification:* g_τ^G denotes a function within the class G, specifically trained on dataset τ.
- **Understanding the Context:**
  - This formula encapsulates the expected accuracy of model g_τ^G on new, unseen data, where (X, Y) are sampled according to distribution $f(x, y)$.
- **Addressing Discrete vs. Continuous Data:**
  - **Discrete Case:**
    $l_\tau(g_\tau^G) = \sum_{x,y} \text{Loss}(y, g_\tau^G(x)) f(x, y)$
  - **Continuous Case:**
    $l_\tau(g_\tau^G) = \int \text{Loss}(y, g_\tau^G(x)) f(x, y)\, dx\, dy$
- **Highlighting the Impact of Overfitting:**
  - Overfitting is indicated by an excessively low training loss for g_τ^G, leading to a high generalization risk, $l_\tau(g_\tau^G)$, for unseen data.
  - It showcases the challenge in achieving a balance where a model is sufficiently complex to learn from the training data τ without losing the capacity to generalize well to new data instances.

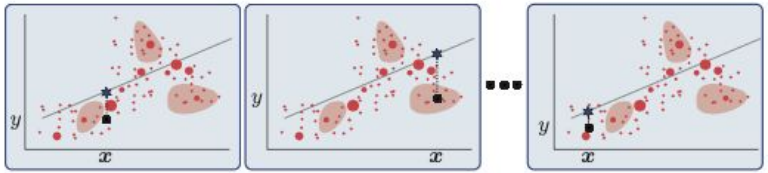


Figure 2.1: The generalization risk for a fixed training set is the weighted-average loss over all possible pairs (x, y).

FOR A FIXED TRAINING SET
In this scenario, generalization risk represents the **average expected loss** of a **prediction model (straight line)** on **new, unseen data points (black dots)**, taking into account the **underlying distribution of data (red dots).**

https://people.smp.uq.edu.au/DirkKroese/DSML/DSML.pdf

Expected Generalization Risk with Random Training Set T

- **Expected Generalization Risk for Random T:**

  $\mathbb{E}_T[l(g_T^G)] = \mathbb{E}[\text{Loss}(Y, g_T^G(X))]$

  - Averages generalization risk over all possible T.
  - g_T^G is the best predictor in G for training set T.
- **Expression in Discrete Case:**

  $\mathbb{E}_T[l(g_T^G)] = \sum_{x,y,x_1,y_1,\ldots,x_n,y_n} \text{Loss}(y, g_T^G(x)) f(x,y) \prod_{i=1}^n f(x_i, y_i)$

  - Considers all possible training sets and data points.
- **Key Insights:**
  - Generalization risk is a random variable due to T.
  - Expected generalization risk evaluates model's average performance across different training sets.
  - Aims for robustness in model prediction across varying training data instances.

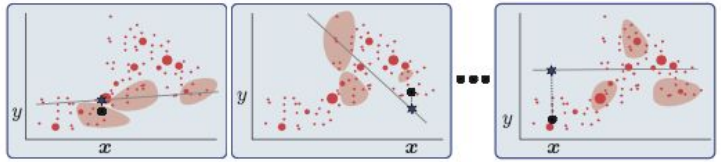


Figure 2.2: The expected generalization risk is the weighted-average loss over all possible pairs (x, y) and over all training sets.

FOR A FIXED TRAINING SET
In this scenario, the **Expected Generalization Risk** is the **average loss of a trained model (gGT)** over **all possible training sets (T)**, calculated using the loss function and the probability distributions of data and training instances.

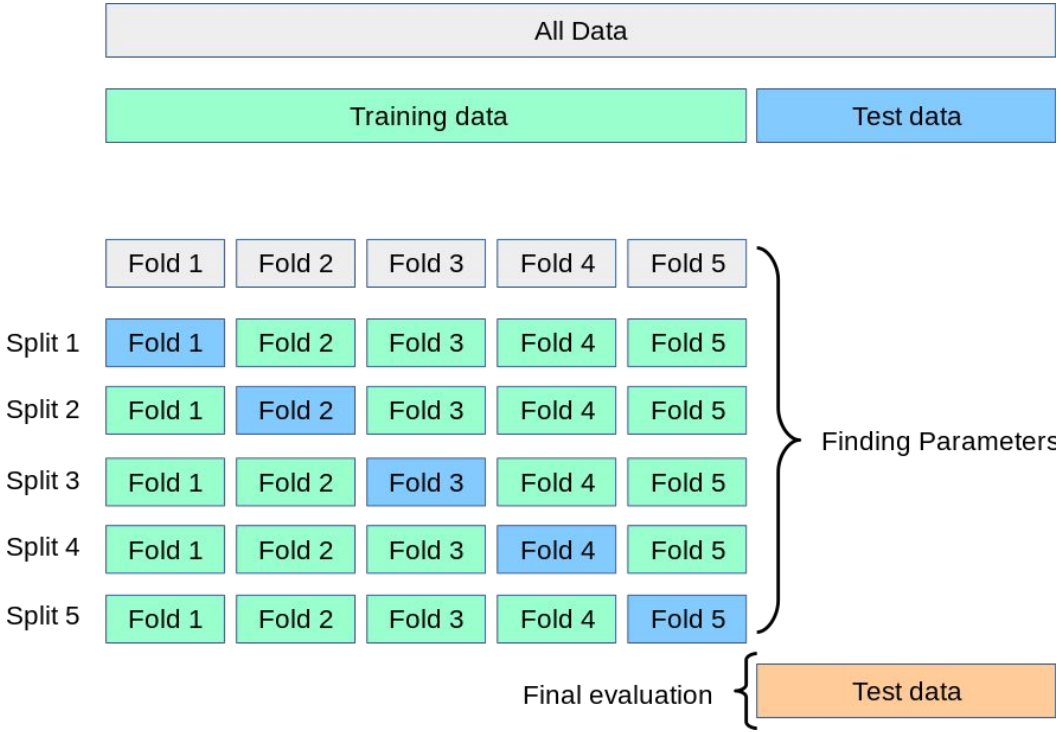
https://people.smp.uq.edu.au/DirkKroese/DSML/DSML.pdf

Note: Understanding τ vs. T in Training Datasets

- τ - **Specific Training Dataset:**
  - Represents a specific instance of a training dataset.
  - Used to discuss outcomes related to this particular dataset.
  - Focuses on the concrete realization of data points.
- T - **Random Training Set:**
  - Denotes the concept of a random training set, highlighting variability in dataset selection.
  - Generalizes findings across different instances of training data.
  - Used in theoretical analyses for expected performance over all possible training sets.
- **Key Differences:**
  - **Specificity:** τ is specific; T represents a random selection.
  - **Purpose in Analysis:** τ focuses on specific dataset properties; T aims for broader insights over randomness in data selection.
  - **Generalization:** T allows for evaluating model robustness across varying training datasets, unlike the fixed instance τ.

https://people.smp.uq.edu.au/DirkKroese/DSML/DSML.pdf

Application: Cross Validation



https://scikit-learn.org/stable/modules/cross_validation.html

- τ - **Specific Training Dataset:**
  - Represents a specific instance of a training dataset.
  - Used to discuss outcomes related to this particular dataset.
  - Focuses on the concrete realization of data points.
- T - **Random Training Set:**
  - Denotes the concept of a random training set, highlighting variability in dataset selection.
  - Generalizes findings across different instances of training data.
  - Used in theoretical analyses for expected performance over all possible training sets.

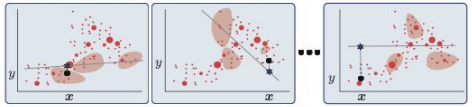


Figure 2.2: The expected generalization risk is the weighted-average loss over all possible pairs (x, y) and over all training sets.

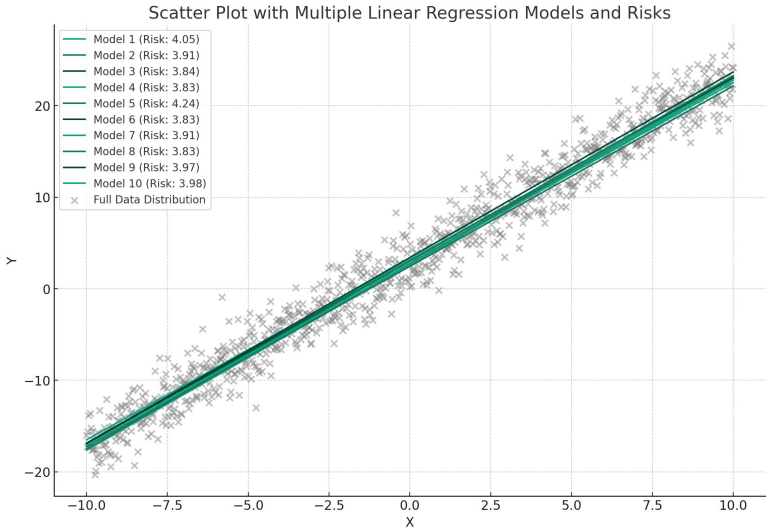
Example: Impact of Training Set Variability on Generalization Risk

- **Study Objective:**

  Examine the effect of training set selection on the generalization risk of linear regression models.

- **Methodology:**
  - Simulated 100 linear regression models using varied training sets.
  - True data model: $Y = 2X + 3$, with added noise.
  - Evaluated each model using Mean Squared Error (MSE) on a comprehensive dataset.
- **Results:**
  - **Average Generalization Risk:** Approximately 3.98.
  - **Risk Variability:** Standard deviation around 0.15.
- **Key Insight:**

  Variability in training sets significantly impacts generalization risk, highlighting the importance of evaluating models across a broad range of datasets to ensure robust performance assessment.

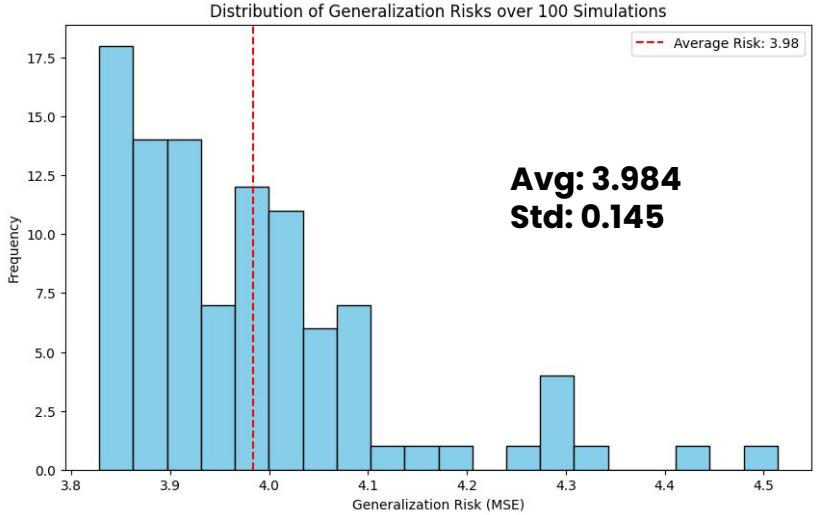


*The graph demonstrates the **variability in generalization risk** (measured by MSE) **across different training sets**, highlighting how **model performance can fluctuate** due to randomness in training data selection, with the average generalization risk providing a more reliable performance metric.*

https://scikit-learn.org/stable/modules/cross_validation.html

Example: Impact of Training Set Variability on Generalization Risk

- **Study Objective:**
  Examine the effect of training set selection on the generalization risk of linear regression models.
- **Methodology:**
  - Simulated 100 linear regression models using varied training sets.
  - True data model: $Y = 2X + 3$, with added noise.
  - Evaluated each model using Mean Squared Error (MSE) on a comprehensive dataset.
- **Results:**
  - **Average Generalization Risk:** Approximately 3.98.
  - **Risk Variability:** Standard deviation around 0.15.
- **Key Insight:**
  Variability in training sets significantly impacts generalization risk, highlighting the importance of evaluating models across a broad range of datasets to ensure robust performance assessment.



*The graph demonstrates the **variability in generalization risk** (measured by MSE) **across different training sets**, highlighting how **model performance can fluctuate** due to randomness in training data selection, with the average generalization risk providing a more reliable performance metric.*

https://scikit-learn.org/stable/modules/cross_validation.html

Example: Impact of Training Set Variability on Generalization Risk

```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np

# Define the true relationship between X and Y
a, b = 2, 3  # Coefficients for the linear relationship
np.random.seed(42)  # For reproducibility

# Generate a large pool of data points that represents the entire distribution of X, Y pairs
X_full = np.linspace(-10, 10, 1000).reshape(-1, 1)
Y_full = a * X_full + b + np.random.normal(0, 2, X_full.shape)  # Adding some noise

# Function to simulate training and evaluate generalization risk
def simulate_generalization_risk(n_simulations=100, n_samples=50):
    risks = []  # To store the generalization risks

    for _ in range(n_simulations):
        # Randomly sample a training set
        indices = np.random.choice(range(len(X_full)), size=n_samples, replace=False)
        X_train = X_full[indices]
        Y_train = Y_full[indices]

        # Fit a linear regression model
        model = LinearRegression().fit(X_train, Y_train)

        # Predict on the full dataset to evaluate generalization risk
        Y_pred = model.predict(X_full)

        # Calculate MSE as the generalization risk for this simulation
        risk = mean_squared_error(Y_full, Y_pred)
        risks.append(risk)

    # Return the average generalization risk
    return np.mean(risks), np.std(risks)

# Simulate the generalization risk over 100 instances of the training set
avg_risk, std_risk = simulate_generalization_risk()

avg_risk, std_risk

```

```python
import matplotlib.pyplot as plt

# Run the simulation to get risks and their standard deviation
risks, _ = np.array([simulate_generalization_risk(1, 50) for _ in range(100)]).T

# Plotting the distribution of generalization risks
plt.figure(figsize=(10, 6))
plt.hist(risks, bins=20, color='skyblue', edgecolor='black')
plt.title('Distribution of Generalization Risks over 100 Simulations')
plt.xlabel('Generalization Risk (MSE)')
plt.ylabel('Frequency')
plt.axvline(x=avg_risk, color='r', linestyle='--', label=f'Average Risk: {avg_risk:.2f}')
plt.legend()
plt.show()
```

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

Unbiased Generalization Risk Estimator using Test Data

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

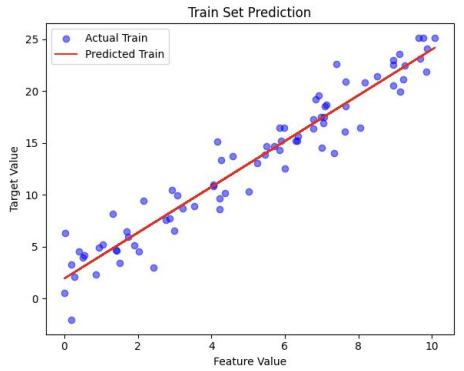
Unbiased Estimation of Generalization Risk

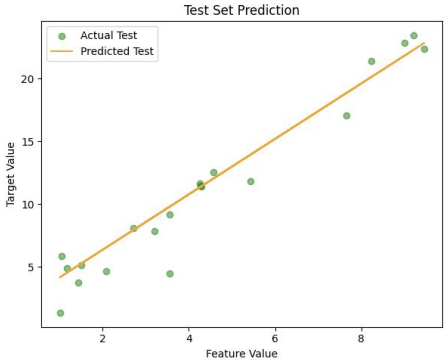
- **Objective**: Estimate the unbiased generalization risk of a predictive model.
- **Notation**:
  - **Training Data**, T: The dataset used to train the model.
  - **Test Data**, T': A separate dataset used to evaluate the model.
  - **Generalization Function**, $\hat{g}_\tau^G$: The predictive model trained on T and evaluated on T'.
- **Key Concept**:
  - The **Test Loss** is used to estimate the generalization risk of $\hat{g}_\tau^G$ without bias.
- **Formula**:

$$\frac{1}{n'}\sum_{i=1}^{n'} \text{Loss}(Y_i', \hat{g}_\tau^G(X_i')) =: l_{T'}(\hat{g}_\tau^G)$$

  where n' is the size of the test sample T', and $\text{Loss}(Y_i', \hat{g}_\tau^G(X_i'))$ calculates the discrepancy between the predicted and actual outcomes.
- **Assumption**: Independence between training set T and test set T' is crucial for unbiased estimation.
- **Application**: This methodology allows for the assessment of how well the model generalizes to new, unseen data, a fundamental aspect of supervised learning.

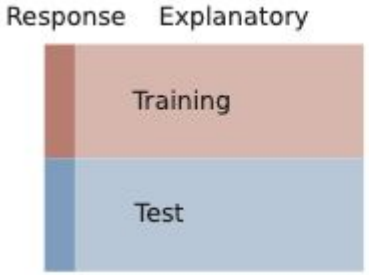
```
Train Loss (MSE): 3.301
Test Loss (MSE): 2.901
Train Accuracy (R^2 Score): 0.931
Test Accuracy (R^2 Score): 0.937
```



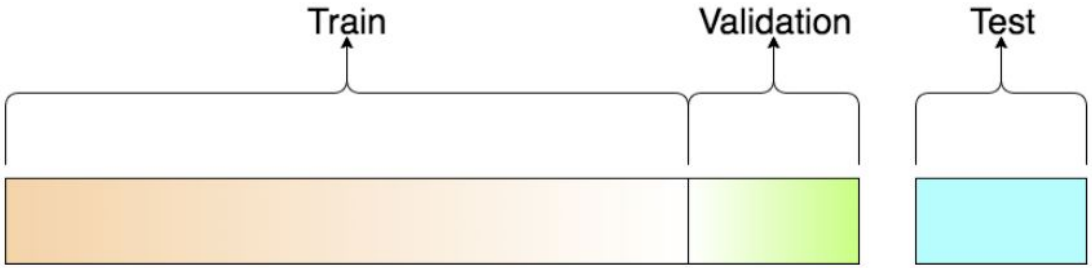


Comparing Predictive Performance

- **Objective**: Assess and compare the predictive performance of various learners within the function class G.
- **Notation**:
  - **Training Set**, τ: Used to train different learners ($g^{G_1}, g^{G_2}, \ldots$).
  - **Test Set**, τ': Initially used as a validation set to select the best learner based on the smallest test loss.
  - **Validation Set**: A subset of the dataset used to tune models' parameters and select the best performing model.
  - **Final Test Set**: An additional, separate dataset used to evaluate the predictive performance of the selected best learner.
- **Process**:
  - **Data Splitting**: The overall dataset is randomly divided into training and test/validation sets.
  - **Model Training**: Use the training data to construct and train various learners.
  - **Model Selection**: Use the validation set (initial test set) to select the learner with the best performance.
  - **Performance Evaluation**: Utilize a third set, a final test set, to assess the predictive performance of the chosen learner.



Train/Dev/Test Split



Typical 100-1000-10000	60%	20%	60%
Huge data size 1000000	98%	1%	1%

Mismatched Data Distribution

Train Dataset

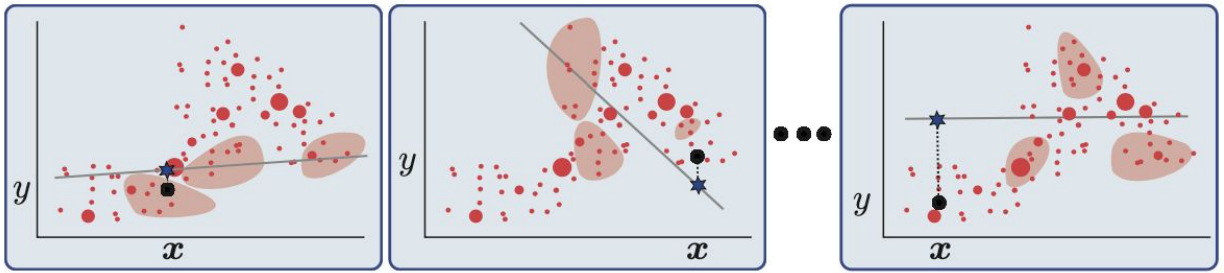
High Resolution

Different Distribution

Dev/Test Dataset

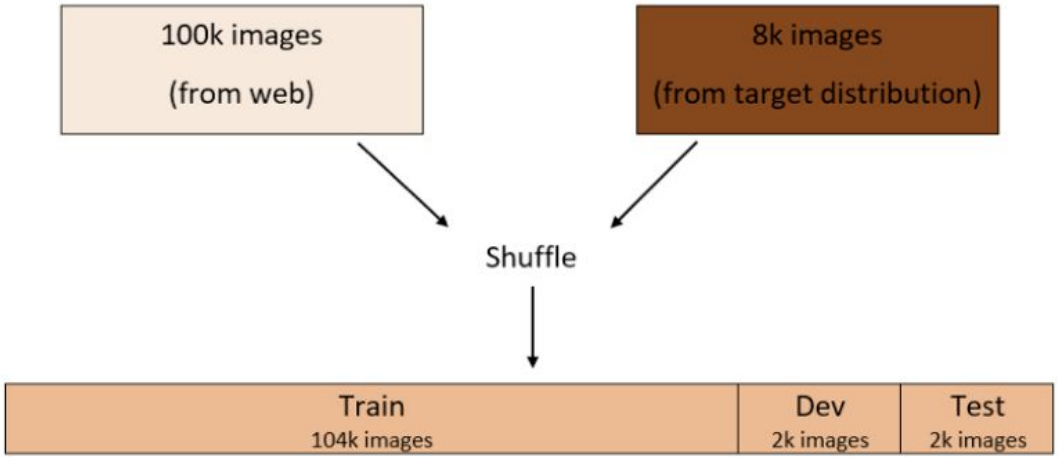
Low Resolution

Make sure that distributions of Train dataset and Dev/Test dataset are similar

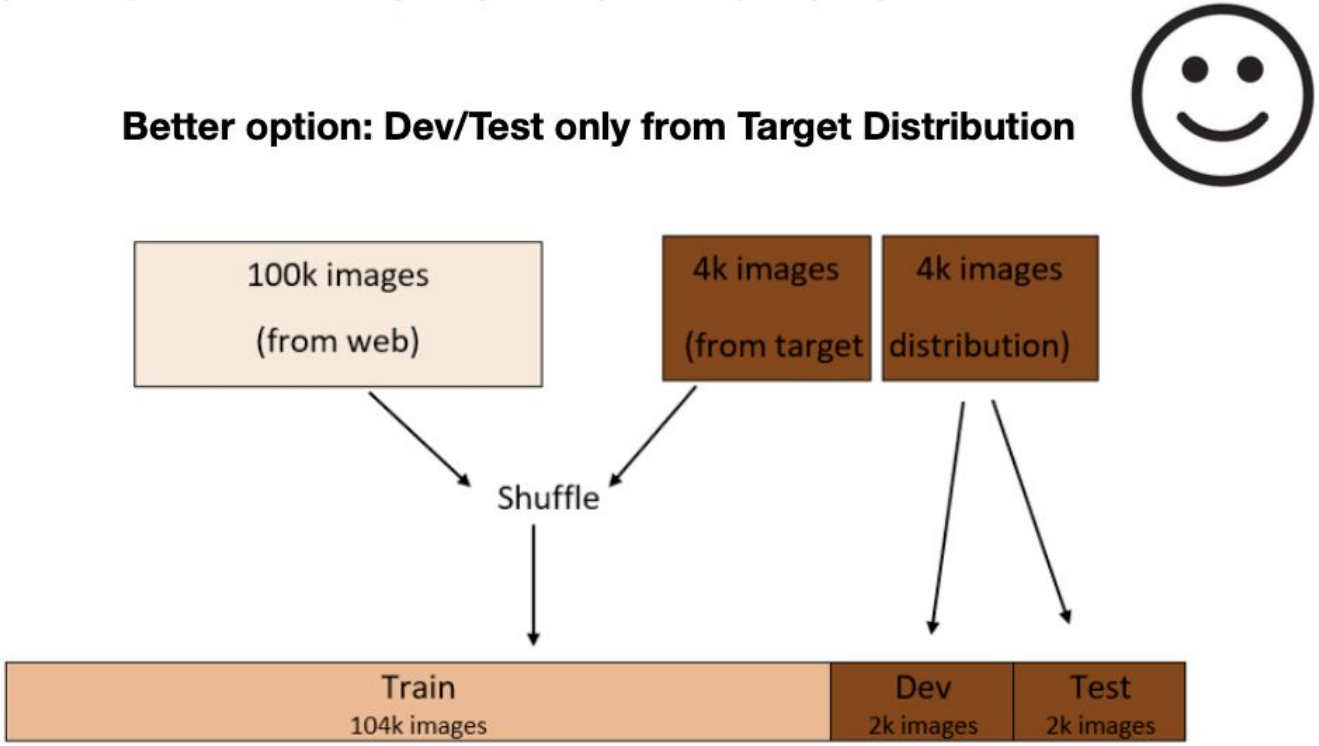


How to Split Mismatched Distribution Data

Option 1: **Shuffle images from different distributions**



How to Split Mismatched Distribution Data



CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

Bias vs. Variance

CS316: INTRODUCTION TO AI AND DATA SCIENCE

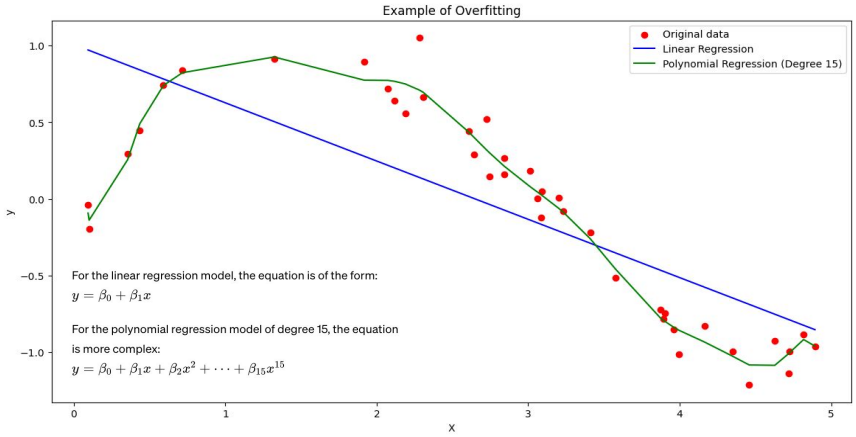
Prof. Anis Koubaa

Understanding Bias vs. Variance

الانحياز والتباين

- **Objective**: Dissect the trade-offs between bias and variance in statistical learning models.
- **Definitions**:
  - **Bias**: The error due to overly simplistic assumptions in the learning algorithm. High bias can cause the model to miss relevant relations between features and target outputs (underfitting).
  - **Variance**: The error from sensitivity to small fluctuations in the training set. High variance can cause model to capture random noise in the training data (overfitting).
- **Key Concept**:
  - **Bias-Variance Tradeoff**: Balancing the complexity of the model to minimize both errors and achieve good generalization.
- **Illustration**:
  - High Bias: Assumes a linear relationship in a non-linear problem.
  - High Variance: Fits noise in the data as if it were a real pattern.
- **Mathematical Perspective**:
  - Expected prediction error for a given point x can be decomposed as:

$$\text{Error}(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$



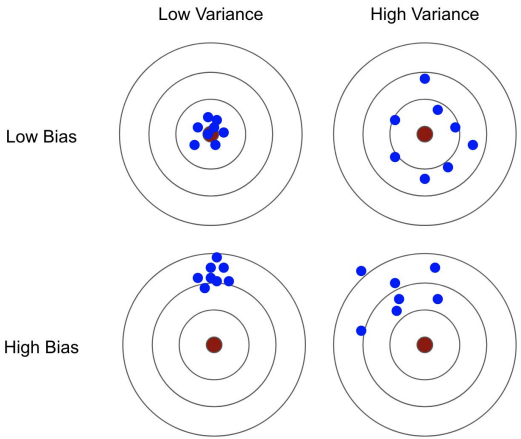
Simplifying the Bias-Variance Tradeoff

$$\text{Error}(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

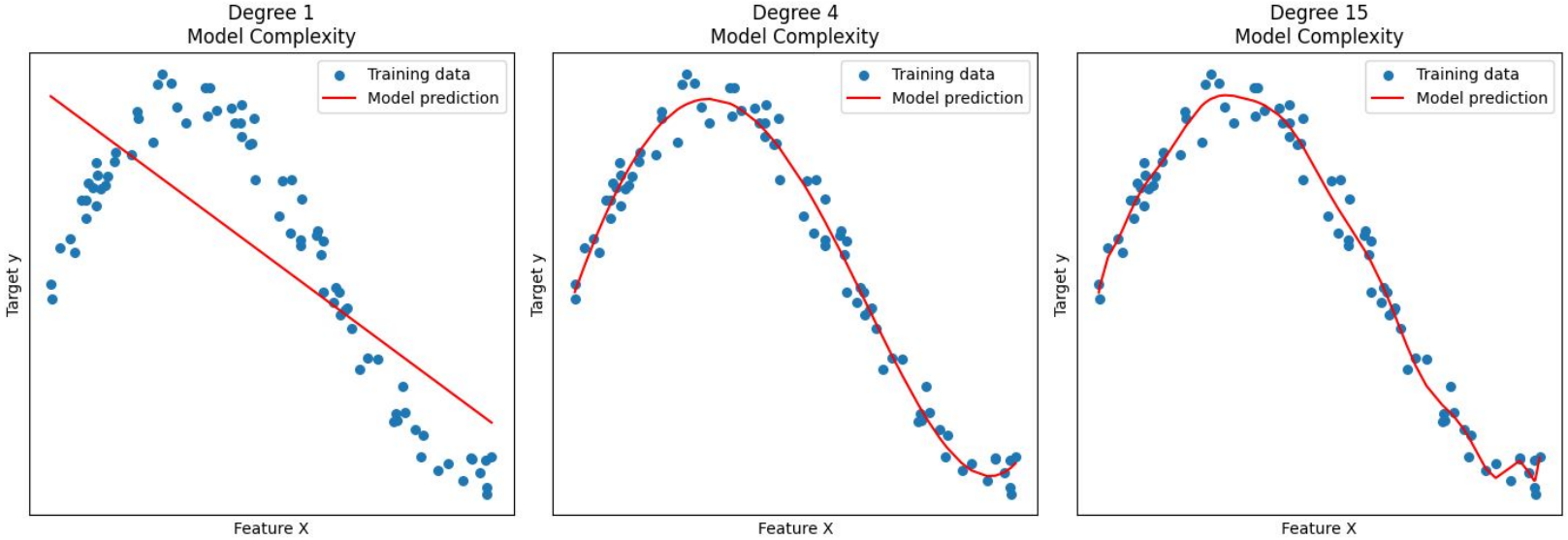
- **Understanding Prediction Error**:
  - Imagine you're trying to hit a target with arrows. Your goal is to be as close to the bullseye (the true value) as possible with each shot (prediction).
- **Components of Prediction Error**:
  - **Bias**: Suppose you consistently hit the target to the left of the bullseye. The average distance of your shots from the bullseye represents **Bias**. High bias means you're not even aiming correctly; your model is missing the mark because it's too simplistic.
  - **Variance**: Now, imagine your shots are spread out, sometimes to the left, sometimes to the right, above, and below the bullseye. The spread of your shots represents **Variance**. High variance means your model is sensitive to the training data and shoots all over the place.
  - **Irreducible Error**: There's always a bit of wind or some unseen bump in the ground that affects where your arrow lands, no matter how good your bow and your aim are. This is the **Irreducible Error**, which comes from randomness or unknown factors in the problem itself that we cannot predict or reduce.

- **Bias-Variance Tradeoff**: Balancing the complexity of the model to minimize both errors and achieve good generalization.
- **Illustration**:
  - High Bias: Assumes a linear relationship in a non-linear problem.
  - High Variance: Fits noise in the data as if it were a real pattern.
- **Mathematical Perspective**:
  - Expected prediction error for a given point x can be decomposed as:

$$\text{Error}(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$



Understanding Bias vs. Variance

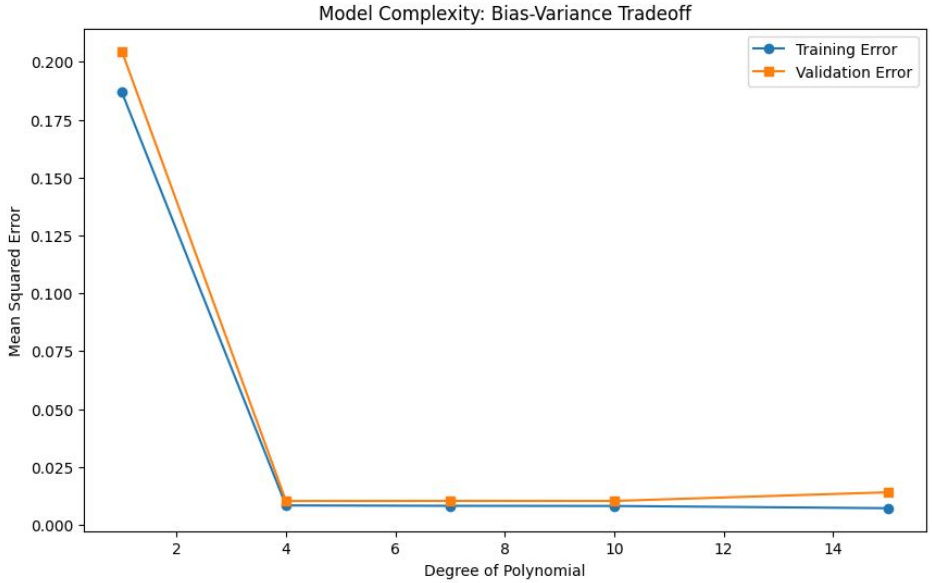


Understanding Bias vs. Variance

- **Training Error** decreases as the model complexity increases, which shows that the model is getting better at fitting the training data. However, this does not necessarily

mean it will perform well on unseen data.

- **Validation Error** initially decreases as the model becomes more capable of capturing the underlying pattern in the data but starts to increase at higher degrees of polynomial, indicating overfitting. This increase in validation error is due to the model's high variance, capturing noise in the training data as meaningful patterns, which do not generalize to new data.

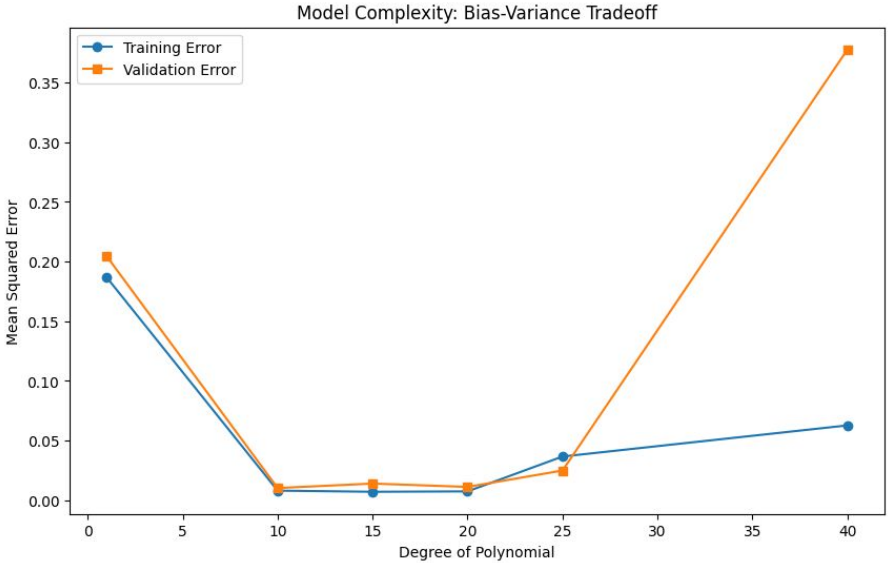


Understanding Bias vs. Variance

- **Training Error** decreases as the model complexity increases, which shows that the model is getting better at fitting the training data. However, this does not necessarily

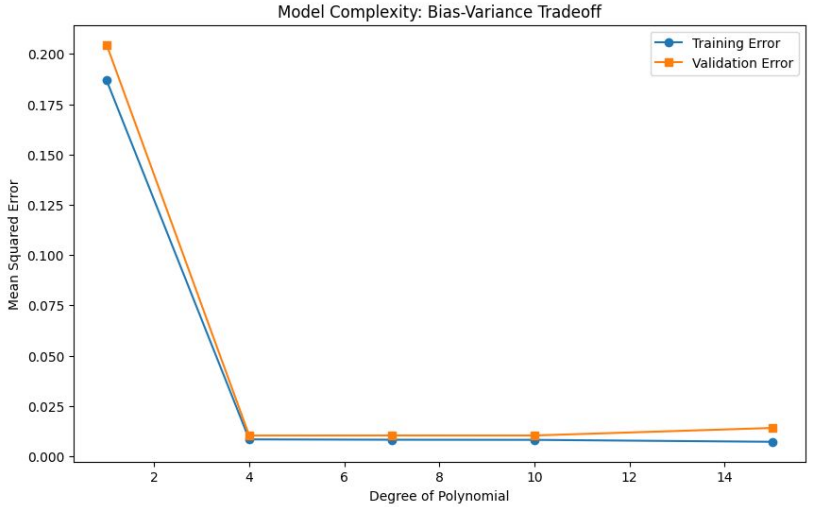
mean it will perform well on unseen data.

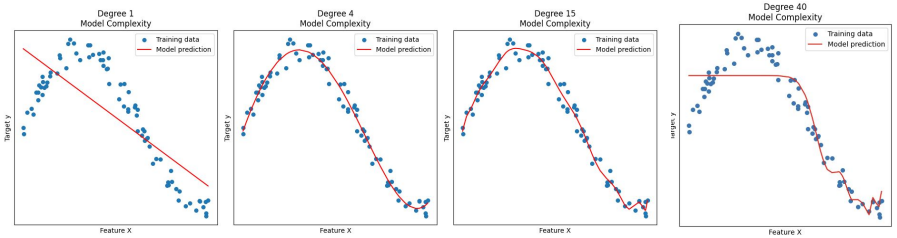
- **Validation Error** initially decreases as the model becomes more capable of capturing the underlying pattern in the data but starts to increase at higher degrees of polynomial, indicating overfitting. This increase in validation error is due to the model's high variance, capturing noise in the training data as meaningful patterns, which do not generalize to new data.



Understanding Bias vs. Variance

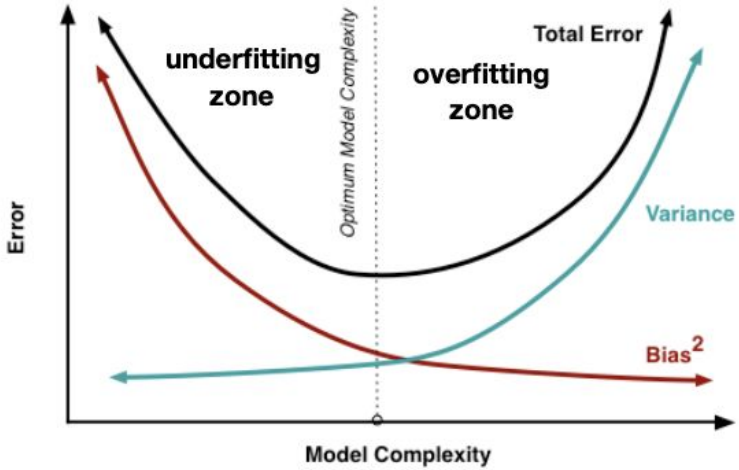
- **Bias**: At lower degrees of polynomial (e.g., 1), both training and validation errors are high because the model is too simple to capture the underlying pattern in the data (underfitting). This is indicative of high bias, where the model's assumptions are too far from the true relationship.
- **Variance**: At higher degrees of polynomial (e.g., 15), the training error is very low because the model fits the training data too closely, including noise. However, the validation error is high because the model fails to generalize to new data. This is indicative of high variance, where the model is too sensitive to small fluctuations in the training data.
- **Bias-Variance Tradeoff**: The optimal model complexity (in this case, around degree 4 to 7) achieves a balance between bias and variance. It is complex enough to capture the underlying pattern in the data (low bias) but not so complex that it fits the noise in the training data (controlled variance). This balance results in the lowest validation error, indicating the best generalization to unseen data.





How to Split Mismatched Distribution Data

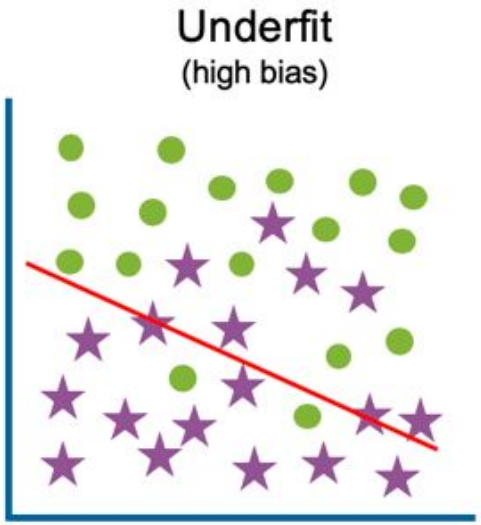
To build a good model, we need to find a good balance between bias and variance such that it minimizes the total error.



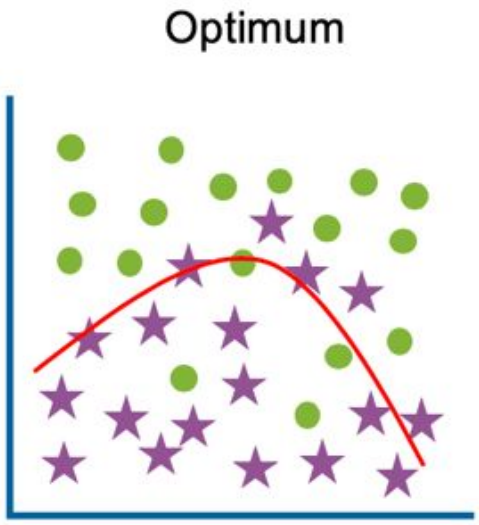
$$Err(x) = \left(E[\hat{f}\,(x)] - f(x)\right)^2 + E\left[\left(\hat{f}\,(x) - E[\hat{f}\,(x)]\right)^2\right] + \sigma_e^2$$

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

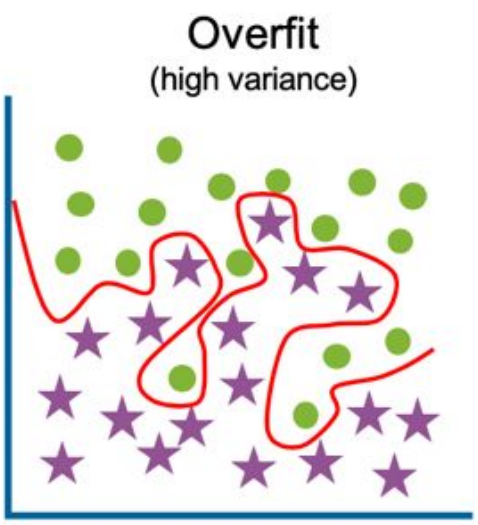
Overfitting | Optimal | Underfit



High training error
High test error



Low training error
Low test error



Low training error
High test error

Bias and Variance Cases

Train Error	1%	15%	15%	0.5%
Test Error	11%	16%	30%	1%
Result	High Variance	High Bias	High Bias High Variance	Low Bias Low Variance
Fitting Type	Overfitting	Underfitting	Overfitting and Underfitting	Good Fitting

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

FORMULATE A PREDICTIVE MODELING PROBLEM
Example of a Polynomial Regression

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Empirical Risk Minimization

- **Primary Goal**: To identify and fine-tune a prediction function, g, that minimizes the expected loss (risk) across all possible data points in the distribution of X and Y.

Formal Mathematical Framework

- **Empirical Risk Minimization (ERM)**:

$$\hat{g} = \arg\min_{g} \hat{R}(g)$$

with empirical risk, $\hat{R}(g)$, given by:

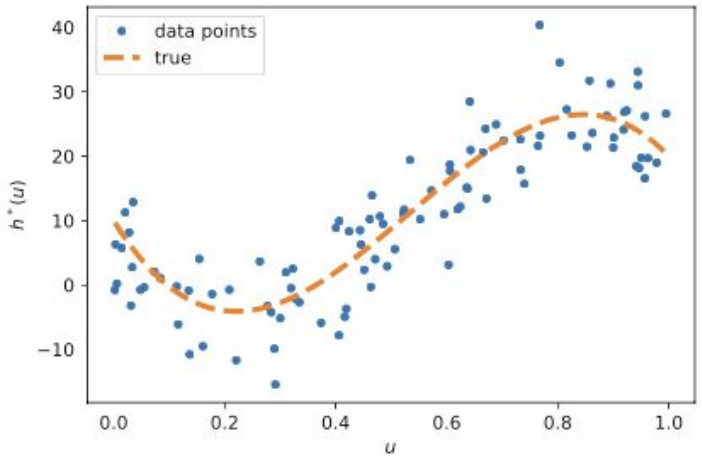
$$\hat{R}(g) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, g(x_i))$$

Here, $\hat{g}$ represents the best approximation of g^* based on the available data, aiming to minimize the average loss calculated over the dataset.

Introduction to Polynomial Regression

- **Objective**: Introduce Polynomial Regression as a tool for modeling nonlinear relationships.
- **Notation Change**: Transition from (x, g, G) to (u, h, H) to better align with this example.

- **Data Points**: (u_i, y_i), for $i = 1, \ldots, n$ with $n = 100$, represent iid samples.
- **Uniform Distribution**: U_is are uniformly spread over (0, 1).
- **Normal Distribution**: Given $U_i = u_i$, Y_i follows a normal distribution with mean $10 - 140u_i + 400u_i^2 - 250u_i^3$ and variance $\sigma^2 = 25$.



- **Model Definition**: A cubic polynomial regression model capturing the relationship between u and y.
- **Optimal Prediction Function**: $h^*(u) = 10 - 140u + 400u^2 - 250u^3$.
  - This function is derived using squared-error loss, representing the expected value of Y given $U = u$.

Parametric Optimization Problem

- **Objective**: Find the optimal polynomial function $h^*(u)$ that minimizes prediction error for the training dataset $\tau = \{(u_i, y_i), i = 1, \ldots, n\}$.

Key Equations and Concepts:

1. **Training Loss Minimization**:
   - Minimize the squared error loss to estimate the relationship between u and y.

     $\text{Minimize: } \frac{1}{n} \sum_{i=1}^{n} (y_i - h(u_i))^2$
2. **Polynomial Candidate Functions**:
   - Consider polynomial functions of order $p - 1$ as candidates for $h(u)$.

     $h(u) = \beta_1 + \beta_2 u + \beta_3 u^2 + \cdots + \beta_p u^{p-1}$

Parametric Optimization Problem

2. **Polynomial Candidate Functions:**
   - Consider polynomial functions of order $p-1$ as candidates for $h(u)$.

     $h(u) = \beta_1 + \beta_2 u + \beta_3 u^2 + \cdots + \beta_p u^{p-1}$
3. **Parametric Optimization:**
   - Optimization over H_p requires finding the best parameter vector β.
   - Linearization: Transforming the polynomial problem into a linear form using feature vector

     $x = [1, u, u^2, \ldots, u^{p-1}]^\top$.

     $g(x) = x^\top \beta$
4. **Optimal Solution:**
   - For $p > 4$, the optimal $h^*(u)$ corresponds to $g^*(x) = x^\top \beta^*$ within the linear function set G_p, where $\beta^* = [10, -140, 400, -250, 0, \ldots, 0]^\top$.

CONCEPT. Transforming Polynomial Regression to Linear Form

- **Core Concept**: Polynomial regression complexities can be simplified by transforming them into linear regression problems.

Key Insights:

1. **Polynomial to Linear Transformation**:
   - Polynomial regression models, though nonlinear in variables, can be linearized through feature transformation.
   - This transformation allows us to apply linear regression techniques to polynomial problems.
2. **Optimization in Linear Space**:
   - By mapping each polynomial term u^i to a corresponding feature x_i, we reformulate the polynomial regression as a linear model:
     $g(x) = x^\top \beta$
     where $x = [1, u, u^2, \ldots, u^{p-1}]^\top$ and β is the parameter vector.

Importance of Linearization in Statistical Learning

Expand the feature space to obtain a *linear* prediction function.

3. **Advantages of Linearization**:
   - Simplifies computation: Linear models are computationally less intensive and well-understood.
   - Enhances interpretability: Parameters in linear models directly correspond to the influence of each transformed feature, making the model easier to interpret.
4. **Optimal Solution for $p > 4$**:
   - The optimal polynomial function $h^*(u)$ in H_p is equivalent to the optimal linear function $g^*(x) = x^\top \beta^*$ in G_p, with $\beta^* = [10, -140, 400, -250, 0, \ldots, 0]^\top$.

Simplifying Polynomial Models via Feature Expansion

1. **Polynomial Model Representation**
   - $h(u) = \beta_0 + \beta_1 u + \beta_2 u^2 + \ldots + \beta_{p-1} u^{p-1}$
2. **Feature Space Expansion**
   - $u \mapsto x = [1, u, u^2, \ldots, u^{p-1}]^\top$
3. **Linear Model Formulation in Expanded Space**
   - $g(x) = x^\top \beta$
4. **Optimization Objective**
   - Find β^* to optimize $g(x)$
5. **Optimal Linear Prediction Example**
   - $\beta^* = [10, -140, 400, -250, 0, \ldots, 0]^\top$
6. **Key Concepts Emphasized**
   - Transition from Complex Polynomial (H_p) to Simplified Linear Form (G_p)
   - Enhancing Computational Efficiency & Simplification of Optimization

Linear Problem Formulation of Polynomial Regression

1. **Feature Variables and Matrix Formulation**
   - Feature vector for each observation: $x_i = [1, u_i, u_i^2, ..., u_i^{p-1}]^\top$
   - Feature matrix X:

$$X = \begin{bmatrix} 1 & u_1 & u_1^2 & \cdots & u_1^{p-1} \\ 1 & u_2 & u_2^2 & \cdots & u_2^{p-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & u_n & u_n^2 & \cdots & u_n^{p-1} \end{bmatrix}$$

Understanding the Feature Matrix X

1. **Batch of Observations**
   - Represents multiple observations or samples from the dataset.
2. **Structure of Features**
   - Columns correspond to feature variables (predictors).
   - Includes transformed features and an intercept term.
3. **Matrix Dimensions**
   - Shape: $n \times p$, where n is the number of observations, and p is the number of features.
4. **Purpose in Statistical Models**
   - Facilitates prediction of the response variable y through model training.

$$x_i = [1, u_i, u_i^2, ..., u_i^{p-1}]^\top$$

$$X = \begin{bmatrix} 1 & u_1 & u_1^2 & \cdots & u_1^{p-1} \\ 1 & u_2 & u_2^2 & \cdots & u_2^{p-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & u_n & u_n^2 & \cdots & u_n^{p-1} \end{bmatrix}$$

Batch (n×p)

Capturing Nonlinear Relationships with 2nd Order Polynomial Feature Matrix

1. **Polynomial Regression Equation**
   - $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon$
     - Where y is the response variable, x is the predictor variable, β_0, β_1, and β_2 are coefficients, and ϵ is the error term.

- y: House price
- x: Surface area of the house
- ϵ: Error term accounting for deviations

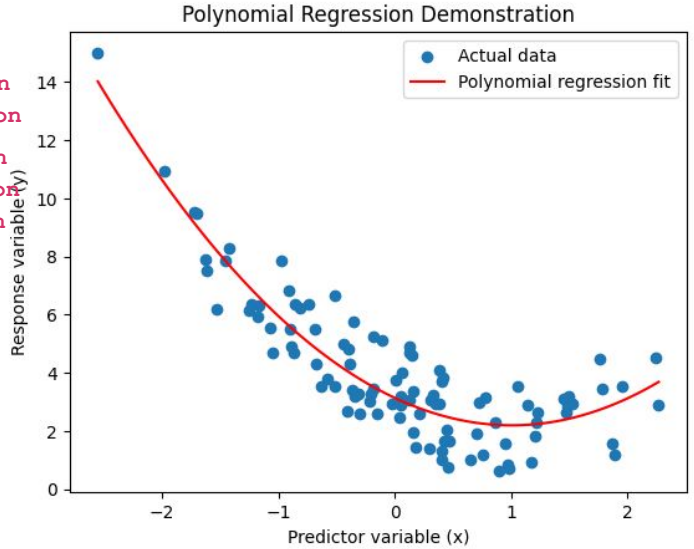
2. **Polynomial Feature Matrix Example**
   - For first 5 observations:

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix} \qquad X_{\text{poly}} = \begin{bmatrix} 1 & 1.76405235 & 3.11188068 \\ 1 & 0.40015721 & 0.16012579 \\ 1 & 0.97873798 & 0.95792804 \\ 1 & 2.2408932 & 5.02160233 \\ 1 & 1.86755799 & 3.48777285 \end{bmatrix}$$

x0: first observation
x1: second observation
x2: third observation
x3: fourth observation
x4: fifth observation

3. **Column Significance**
   - **First Column**: Intercept term, set to 1 for all observations.
   - **Second Column**: Original predictor variable x.
   - **Third Column**: x^2, capturing quadratic relationships.
4. **Importance of Polynomial Features**
   - Allows linear models to capture complex, nonlinear patterns.
   - Significantly enhances model's ability to fit and predict accurately.

```
Feature matrix (X_poly) for the first 5 observations:
 [[1.         1.76405235 3.11188068]
 [1.         0.40015721 0.16012579]
 [1.         0.97873798 0.95792804]
 [1.         2.2408932  5.02160233]
 [1.         1.86755799 3.48777285]]
```



Capturing Nonlinear Relationships with 2nd Order Polynomial Feature Matrix

1. **Polynomial Regression Equation**
   - $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon$
     - Where y is the response variable, x is the predictor variable, β_0, β_1, and β_2 are coefficients, and ϵ is the error term.

- y: House price
- x: Surface area of the house
- ϵ: Error term accounting for deviations

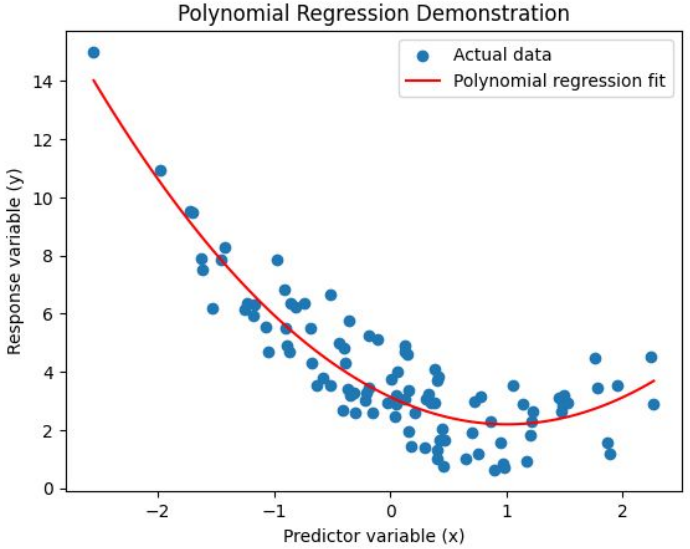
2. **Polynomial Feature Matrix Example**
   - For first 5 observations:

Feature $\beta 0$ Feature $\beta 1$ Feature $\beta 2$

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix} \qquad X_{\text{poly}} = \begin{bmatrix} 1 & 1.76405235 & 3.11188068 \\ 1 & 0.40015721 & 0.16012579 \\ 1 & 0.97873798 & 0.95792804 \\ 1 & 2.2408932 & 5.02160233 \\ 1 & 1.86755799 & 3.48777285 \end{bmatrix}$$

3. **Column Significance**
   - **First Column**: Intercept term, set to 1 for all observations.
   - **Second Column**: Original predictor variable x.
   - **Third Column**: x^2, capturing quadratic relationships.
4. **Importance of Polynomial Features**
   - Allows linear models to capture complex, nonlinear patterns.
   - Significantly enhances model's ability to fit and predict accurately.

```
Feature matrix (X_poly) for the first 5 observations:
 [[1.         1.76405235 3.11188068]
 [1.         0.40015721 0.16012579]
 [1.         0.97873798 0.95792804]
 [1.         2.2408932  5.02160233]
 [1.         1.86755799 3.48777285]]
```



- Intercept (β_0): 2.833085921825976
- Coefficient for x (β_1): -1.88039556
- Coefficient for x^2 (β_2): 0.97494202

Polynomial Regression Model of Degree 4

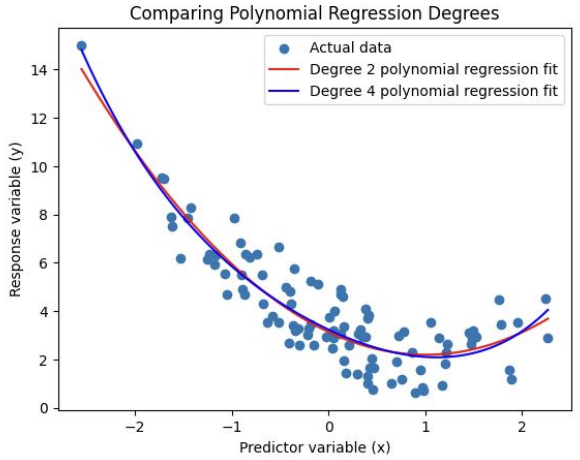
1. **Polynomial Regression Equation**
   - $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \epsilon$
     - **Variables:**
       - y: Response variable
       - x: Predictor variable
       - $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$: Model coefficients
       - ϵ: Error term

   - y: House price
   - x: Surface area of the house
   - ϵ: Error term accounting for deviations

2. **Feature Matrix Construction (X) for Degree 4**
   - Form:

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & x_1^4 \\ 1 & x_2 & x_2^2 & x_2^3 & x_2^4 \\ 1 & x_3 & x_3^2 & x_3^3 & x_3^4 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 & x_n^4 \end{bmatrix}$$

```
Feature matrix (X_poly_degree_4) for the first 5 observations:
 [[ 1.          1.76405235  3.11188068  5.48952041  9.68380136]
 [ 1.          0.40015721  0.16012579  0.06407549  0.02564027]
 [ 1.          0.97873798  0.95792804  0.93756056  0.91762613]
 [ 1.          2.2408932   5.02160233 11.25287451 25.21648996]
 [ 1.          1.86755799  3.48777285  6.51361805 12.16455943]]
```



- Intercept (β_0): 3.2398824750482764
- Coefficient for x (β_1): -1.87934064
- Coefficient for x^2 (β_2): 0.68842674
- Coefficient for x^3 (β_3): 0.00490558
- Coefficient for x^4 (β_4): 0.05556872

Linear Problem Formulation of Polynomial Regression

1. **Response Vector (y)**
   - Definition: Collects observed outcomes.
   - Notation: $y = [y_1, y_2, \ldots, y_n]^\top$.
2. **Training Loss Formula**
   - $$\text{Training Loss} = \frac{1}{n}\|y - X\beta\|^2$$
3. **Components**
   - y: Response vector with observed values.
   - X: Feature matrix.
   - β: Coefficients vector to be optimized.
4. **Objective**
   - Minimize training loss to optimize β, achieving the best fit between model predictions and observed data.

Linear Problem Formulation of Polynomial Regression

1. **Response Vector (y) and Training Loss**
   - Response Vector: $y = [y_1, y_2, ..., y_n]^\top$.
   - Training Loss Formula:

$$\text{Training Loss} = \frac{1}{n}\|y - X\beta\|^2$$

2. **Objective of Training Loss Minimization**
   - To accurately predict outcomes by optimizing the model coefficients (β).
3. **Minimization Process**
   - Minimizer formula to find optimal coefficients (β):

$$\hat{\beta} = \arg\min_{\beta} \|y - X\beta\|^2$$

   - $\hat{\beta}$: Optimal set of coefficients that minimize the training loss.
4. **Significance**
   - Identifying $\hat{\beta}$ is crucial for enhancing the model's predictive accuracy.
   - Directly influences the fit of the model to the observed data.

Polynomial Regression Equation

- $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \epsilon$

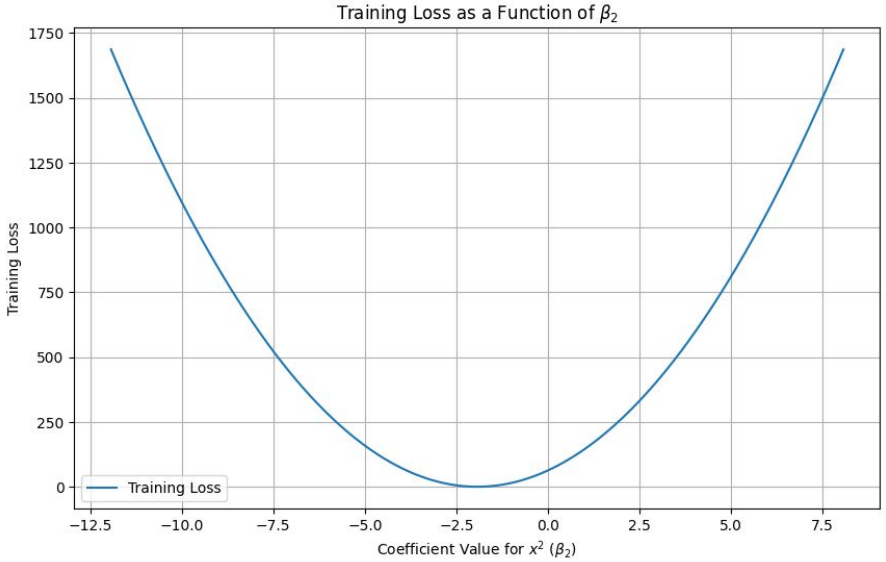
Components

- y: Response vector with observed values.
- X: Feature matrix.
- β: Coefficients vector to be optimized.

Learnable Parameters

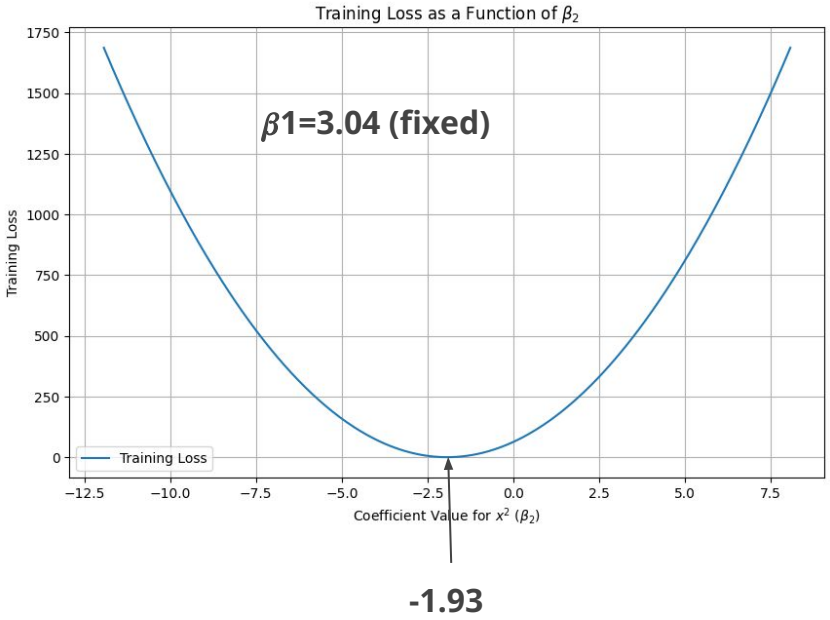
Linear Problem Formulation of Polynomial Regression

```python
import matplotlib.pyplot as plt

# Using the optimal coefficients as a starting point
beta_1_optimal = coefficients[1]
beta_2_optimal = coefficients[2]

# Varying beta_2 (coefficient for x^2) over a range
beta_2_values = np.linspace(beta_2_optimal - 10, beta_2_optimal + 10, 400)
losses = []

# Calculating the training loss for each beta_2 value
for beta_2 in beta_2_values:
    # Manually setting the coefficient for x^2 and keeping others at their optima
    y_pred_temp = intercept + beta_1_optimal * X[:, 1] + beta_2 * X[:, 2]
    loss = mean_squared_error(y, y_pred_temp)
    losses.append(loss)

# Plotting the loss as a function of beta_2
plt.figure(figsize=(10, 6))
plt.plot(beta_2_values, losses, label='Training Loss')
plt.xlabel('Coefficient Value for $x^2$ ($\\beta_2$)')
plt.ylabel('Training Loss')
plt.title('Training Loss as a Function of $\\beta_2$')
plt.legend()
plt.grid(True)
plt.show()
```



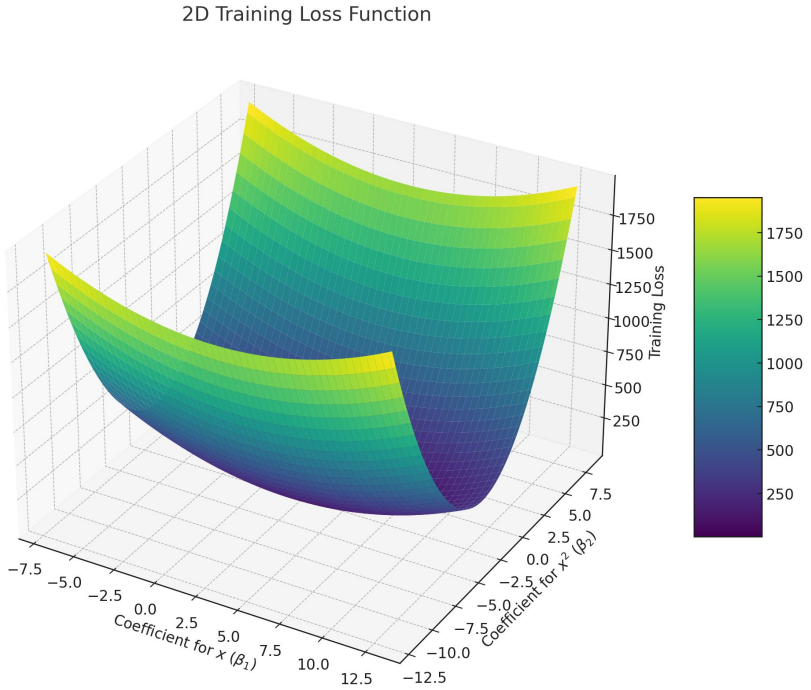
Linear Problem Formulation of Polynomial Regression

1. **Optimized Coefficients**
   - Coefficients: $\beta = \begin{bmatrix} 0 \\ 3.04860316 \\ -1.93172605 \end{bmatrix}$
   - Intercept: $3.7318080606045 55$
2. **Polynomial Equation**
   - $y = 3.7318 + 3.0486x - 1.9317x^2$
3. **Training Loss**
   - Loss Function: $L(\beta) = \frac{1}{n}\|y - X\beta\|^2$
   - Minimized Loss: 0.8641581808270844



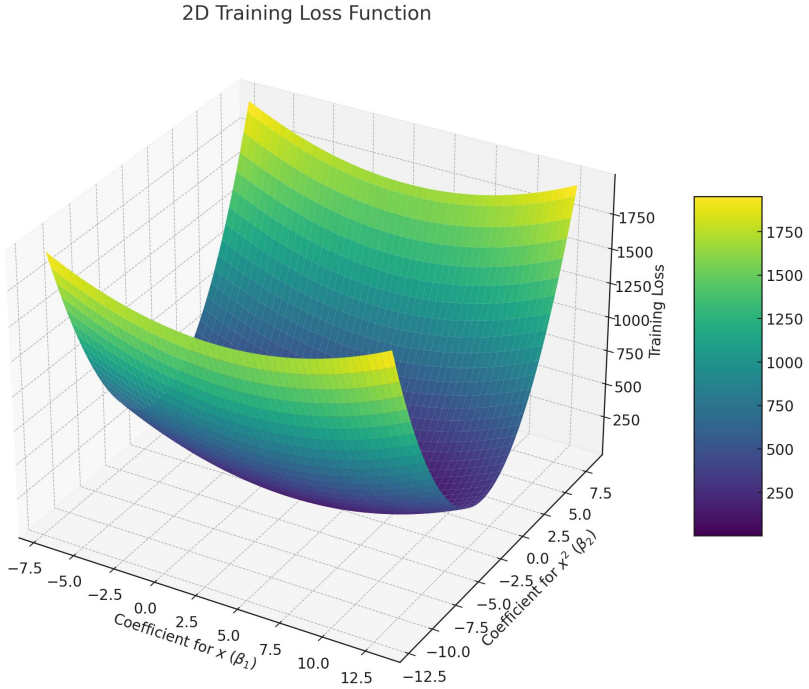
Linear Problem Formulation of Polynomial Regression

```python
from mpl_toolkits.mplot3d import Axes3D

# Defining ranges for beta_1 and beta_2
beta_1_range = np.linspace(beta_1_optimal - 10, beta_1_optimal + 10, 100)
beta_2_range = np.linspace(beta_2_optimal - 10, beta_2_optimal + 10, 100)

# Creating a meshgrid for beta_1 and beta_2
B1, B2 = np.meshgrid(beta_1_range, beta_2_range)
losses_2d = np.zeros(B1.shape)

# Calculating the training loss for each pair of (beta_1, beta_2)
for i in range(B1.shape[0]):
    for j in range(B1.shape[1]):
        y_pred_temp = intercept + B1[i, j] * X[:, 1] + B2[i, j] * X[:, 2]
        losses_2d[i, j] = mean_squared_error(y, y_pred_temp)

# Visualizing the 2D loss function
fig = plt.figure(figsize=(14, 10))
ax = fig.add_subplot(111, projection='3d')
surf = ax.plot_surface(B1, B2, losses_2d, cmap='viridis', edgecolor='none')
ax.set_xlabel('Coefficient for $x$ ($\\beta_1$)')
ax.set_ylabel('Coefficient for $x^2$ ($\\beta_2$)')
ax.set_zlabel('Training Loss')
ax.set_title('2D Training Loss Function')
fig.colorbar(surf, shrink=0.5, aspect=5)
plt.show()
```



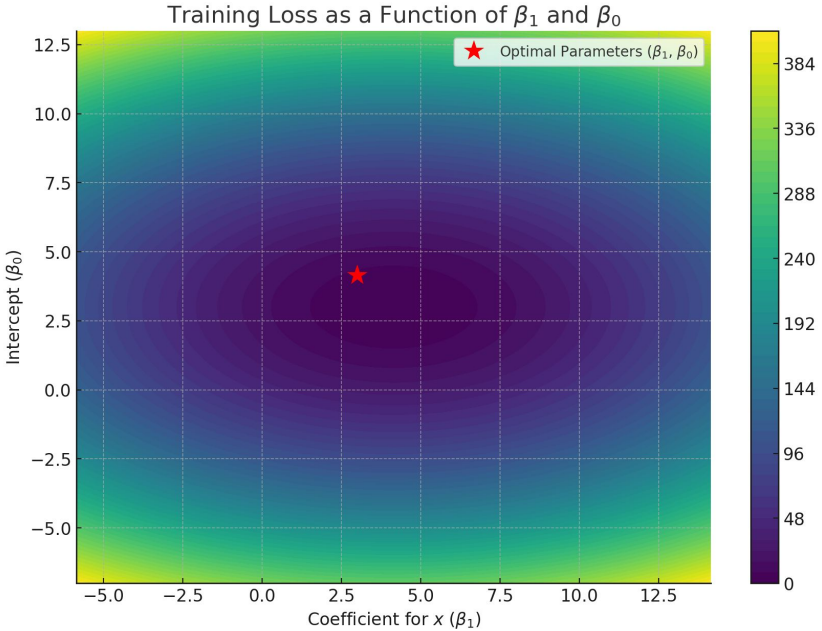
Linear Problem Formulation of Polynomial Regression

- **Meshgrid for β_1 and β_2**: Explores coefficient combinations.
- **Training Loss Evaluation**: Calculated for each (β_1, β_2) pair.
- **3D Surface Visualization**: Depicts how loss varies with β_1 and β_2.
  - **Key Insight**: Lowest point indicates optimal coefficients minimizing loss.
- **Implication**: Visual guide to model accuracy improvement via coefficient optimization.



Linear Problem Formulation of Polynomial Regression

- **Meshgrid for β_1 and β_2**: Explores coefficient combinations.
- **Training Loss Evaluation**: Calculated for each (β_1, β_2) pair.
- **3D Surface Visualization**: Depicts how loss varies with β_1 and β_2.
  - **Key Insight**: Lowest point indicates optimal coefficients minimizing loss.
- **Implication**: Visual guide to model accuracy improvement via coefficient optimization.



CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

Tradeoff in Statistical Learning

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Understanding Tradeoffs in Statistical Learning

Objective

- Aim: Minimize generalization risk or expected generalization risk, optimizing computational efficiency.

Factors Influencing the Choice of Prediction Function Class G

- **Complexity**: Ability of G to approximate or contain the optimal prediction function g^*.
- **Training Ease**: Simplifying the optimization process for learner training.
- **Loss Estimation Accuracy**: How well the training loss predicts actual risk within G.
- **Feature Types**: Handling of categorical, continuous, etc., features.

Tradeoffs

- **Simplicity vs. Accuracy**: Simple G classes train fast but may not closely approximate g^*. Rich G classes may encapsulate g^* but require significant resources.

Error Decomposition in Predictive Modeling

Equation Breakdown

- **Error Decomposition Formula**:

$$E(Y - \hat{Y})^2 = E[\hat{g}(X) + \epsilon - g(X)]^2$$
$$= [\hat{g}(X) - g(X)]^2 + \mathrm{Var}(\epsilon)$$

Components Explained

1. **Total Prediction Error**:
   - Represents the expected value of the squared difference between actual and predicted values of Y.
2. **Reducible Error**:
   - $[\hat{g}(X) - g(X)]^2$: Squared difference between the estimated model $\hat{g}$ and the true model g.
   - Reflects the part of the error that can be reduced by improving the model estimation.
3. **Irreducible Error**:
   - $\mathrm{Var}(\epsilon)$: Variance of the error term ϵ.
   - Inherent uncertainty in the prediction process, unaffected by the model.

Understanding Reducible and Irreducible Errors

Reducible Error

- **Definition**: The part of the prediction error that can be decreased by improving the model's accuracy. It stems from the difference between the true model (g) and the estimated model ($\hat{g}$).
- **Formula**: $[g(X) - \hat{g}(X)]^2$
- **Characteristics**: Can be minimized through better model selection, more data, or improved learning techniques.

Irreducible Error

- **Definition**: The error inherent in the prediction process, unaffected by the model used. It arises from randomness or unknown variables that cannot be modeled or predicted.
- **Formula**: $\mathrm{Var}(\epsilon)$
- **Characteristics**: Sets the lower bound for the prediction error, representing noise in the data that cannot be eliminated.

Decomposing Generalization Risk

Generalization Risk Components

- **Irreducible Risk (l^*)**: Lowest possible risk, inherent to the problem.
- **Approximation Error**: Difference between l^* and the best possible learner within G.
- **Statistical Error**: Variability in learner prediction accuracy for new data.

Equation Representation

$$l(g_{G\tau}) = l^* + (l(g_G) - l^*) + (l(g_{G\tau}) - l(g_G))$$

- l^*: Irreducible risk
- g_G: Best learner within G
- $l(g_{G\tau})$: Actual risk of a learner

Understanding the Tradeoffs

- **Approximation–Estimation Tradeoff**: Balancing model complexity with the accuracy of estimating g^*.
- **Bias–Variance Tradeoff**: Managing the tradeoff between model bias and variance to optimize prediction accuracy.

Understanding Irreducible Error

- *Irreducible Risk (l):** Minimum error inherent in data, unimprovable by any model.
  - $l^* := l(g^*)$
- **Best Learner (g_G):** Optimal model in class G, minimizing loss.
  - $g_G := \underset{g \in G}{\operatorname{argmin}}\, l(g)$
- **Key Insight:** No model in G can surpass l^* in prediction accuracy.
- **Implication:** Sets the benchmark for model performance, guiding realistic expectations.
- **Example:** In finance, l^* defines the predictability limit in stock market forecasting.

Understanding Approximation Error

Approximation Error

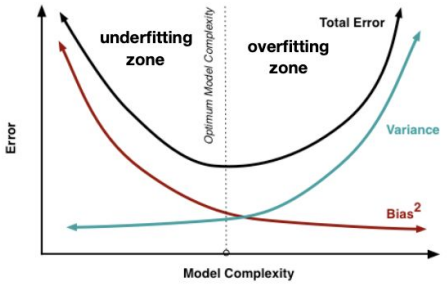
- **Definition**: Measures the gap between the irreducible risk (l^*) and the lowest risk achievable within a chosen function class G, denoted by $l(g_G)$, where $g_G = \text{argmin}_{g \in G} l(g)$.
- **Formula**: $l(g_G) - l^*$
- **Key Points**:
  - **Influence of Class G**: The choice of G is crucial. A limited G may not contain the optimal function g^*, leading to a higher approximation error.
  - **Numerical and Functional Analysis Role**: Identifying the best g within G and minimizing $l(g)$ over G involves numerical and functional analysis, independent of the training data τ.
  - **Reducing Approximation Error**: To lower this error, one needs to expand G to encompass a broader set of functions, potentially including g^*.
- **Implications**:
  - A fixed, restricted G might result in an approximation error that significantly contributes to the generalization risk.
  - Expanding G can reduce approximation error but may increase complexity and the risk of overfitting.

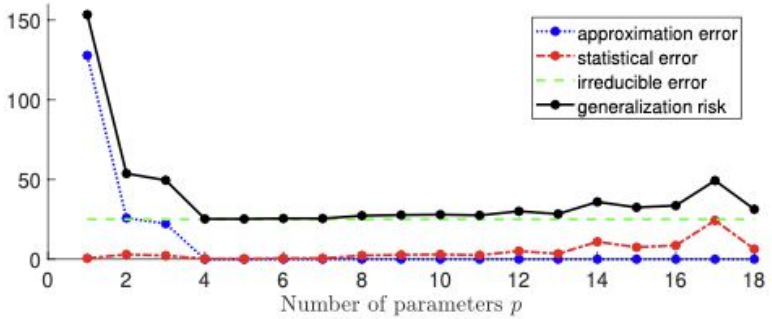
Understanding Statistical Estimation Error

- **Statistical Error Definition:**
  - Discrepancy in predictions between g_{G_τ} and the optimal g_G.
- **Dependency on Training Set (τ):**
  - Error $\epsilon = l(g_{G_\tau}) - l(g_G)$.
- **Convergence to Zero:**
  - $\lim_{|\tau| \to \infty} \epsilon(\tau) = 0$ in probability or expectation.
- **Importance of Training Size:**
  - Larger τ enhances estimation accuracy of g_{G_τ}.
- **Practical Insight:**
  - Ensures effective learning with increased data, guiding towards optimal prediction.

Approximation-Estimation Tradeoff & Error Decomposition

- **Approximation-Estimation Tradeoff:**
  - Balancing simplicity (small statistical error) and richness (small approximation error) in class G.
- **Squared-Error Loss Context:**
  - Generalization risk: $l(g_{G_\tau}) = E(Y - g_{G_\tau}(X))^2$.
  - Optimal prediction: $g^*(x) = E[Y|X = x]$.
- **Error Components:**
  1. **Irreducible Error:** $l^* = E(Y - g^*(X))^2$.
  2. **Approximation Error:** $E(g_G(X) - g^*(X))^2$.
  3. **Statistical Error (Linear Class G):** $E(g_{G_\tau}(X) - g_G(X))^2$.
- **Error Decomposition (Equation 2.17):**
  - $l(g_{G_\tau}) = l^* + E(g_G(X) - g^*(X))^2 + E(g_{G_\tau}(X) - g_G(X))^2$.
    - Illustrates the distinct contributions of each error type to overall risk.





Approximation-Estimation Tradeoff & Error Decomposition

- **Polynomial Regression Context:**
  - Class G_p: Linear functions of $x = [1, u, u^2, \ldots, u^{p-1}]^\top$.
  - Optimal function: $g^*(x) = x^\top \beta^*$.
- **Approximation Error Analysis:**
  - For any $g \in G_p$: $g(x) = h(u) = \beta_1 + \beta_2 u + \ldots + \beta_p u^{p-1}$.
  - Approximation error formula: $\int_0^1 \left([1, u, \ldots, u^{p-1}]\beta - [1, u, u^2, u^3]\beta^*\right)^2 du$.
- **Error Minimization:**
  - Minimize by setting gradient with respect to β to zero, leading to linear equations.
- **Error Dynamics:**
  - Approximation error decreases as p increases, reaching zero for $p > 4$.

Polynomial Regression & Approximation Error

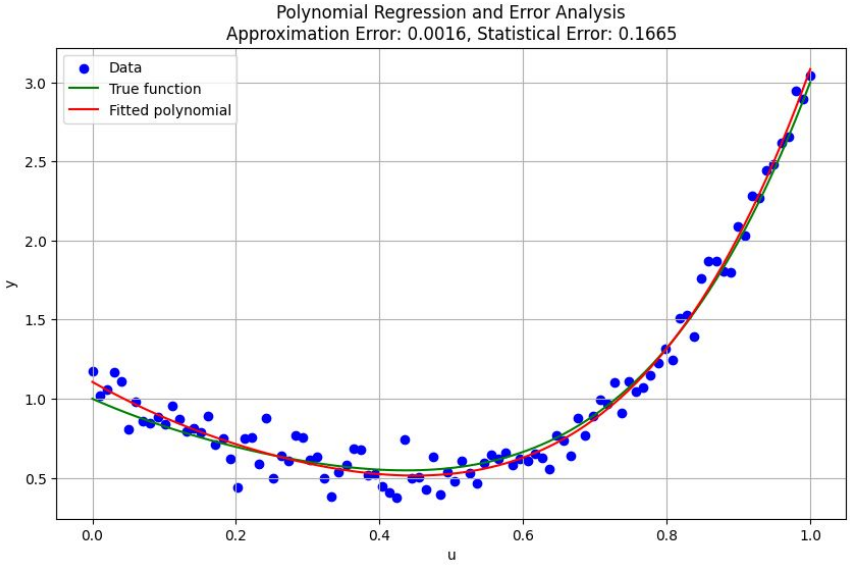
- **Polynomial Regression Context:**
  - Class G_p: Linear functions of $x = [1, u, u^2, \ldots, u^{p-1}]^\top$.
  - Optimal function: $g^*(x) = x^\top \beta^*$.
- **Approximation Error Analysis:**
  - For any $g \in G_p$: $g(x) = h(u) = \beta_1 + \beta_2 u + \ldots + \beta_p u^{p-1}$.
  - Approximation error formula: $\int_0^1 \left([1, u, \ldots, u^{p-1}]\beta - [1, u, u^2, u^3]\beta^*\right)^2 du$.
- **Error Minimization:**
  - Minimize by setting gradient with respect to β to zero, leading to linear equations.
- **Error Dynamics:**
  - Approximation error decreases as p increases, reaching zero for $p > 4$.

Polynomial Regression & Estimation Error

- **Statistical Error Computation:**
  - Given $g_\tau(x) = x^\top b\beta$, statistical error: $\int_0^1 ([1, \ldots, u^{p-1}](b\beta - \beta_p))^2 du$.
- **Statistical Error Expression:**
  - Written as $(b\beta - \beta_p)^\top H_p (b\beta - \beta_p)$, where H_p is the Hilbert matrix.
- **Error Behavior with Complexity:**
  - Statistical error has a global minimum at $p = 4$; increases for $p > 4$.
- **Optimal Complexity Insight:**
  - Considering both errors, $p = 4$ minimizes the generalization risk, balancing approximation and statistical errors.

Polynomial Regression & Estimation Error

```python
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import inv
from numpy.polynomial.polynomial import Polynomial

# Generate some sample data
np.random.seed(0)
u = np.linspace(0, 1, 100)
true_beta = np.array([1, -2, 3, -4, 5])
y_true = np.polyval(true_beta[::-1], u)
y = y_true + np.random.normal(0, 0.1, len(u))

# Polynomial regression for degree 4
p = 5  # Degree is 4, but we have 5 coefficients
X = np.vander(u, p, increasing=True)
beta_hat = inv(X.T @ X) @ X.T @ y

# Approximation error calculation
approximation_error = np.trapz((X @ beta_hat - X @ true_beta) ** 2, u)
#approximation_error = np.sum((X @ beta_hat - X @ true_beta) ** 2) * (u[1] - u[0])


# Statistical error calculation
# Assuming b_beta is beta_hat and beta_p is true_beta for demonstration
H_p = np.vander(u, p, increasing=True).T @ np.vander(u, p, increasing=True)
statistical_error = (beta_hat - true_beta).T @ H_p @ (beta_hat - true_beta)

# Visualization
plt.figure(figsize=(10, 6))
plt.scatter(u, y, label='Data', color='blue')
plt.plot(u, y_true, label='True function', color='green')
plt.plot(u, X @ beta_hat, label='Fitted polynomial', color='red')
plt.title(f'Polynomial Regression and Error Analysis\nApproximation Error: {approx
plt.legend()
plt.xlabel('u')
plt.ylabel('y')
plt.grid(True)
plt.show()
```



$$\text{Approximation Error} = \int_0^1 (\hat{y} - y_{\text{true}})^2 du$$

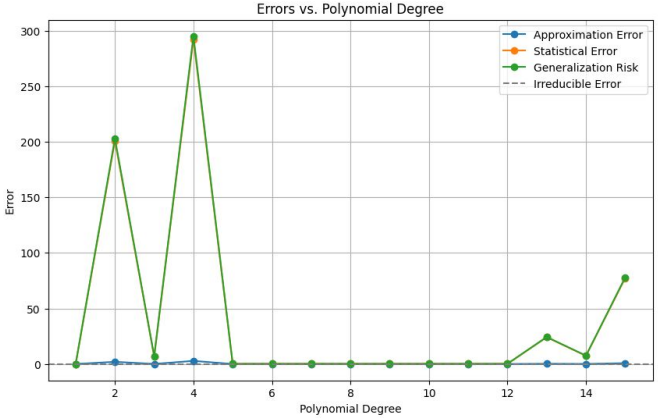
$$: \int_0^1 \left([1, u, \ldots, u^{p-1}]\beta - [1, u, u^2, u^3]\beta^*\right)^2 du.$$

Statistical Error Expression:

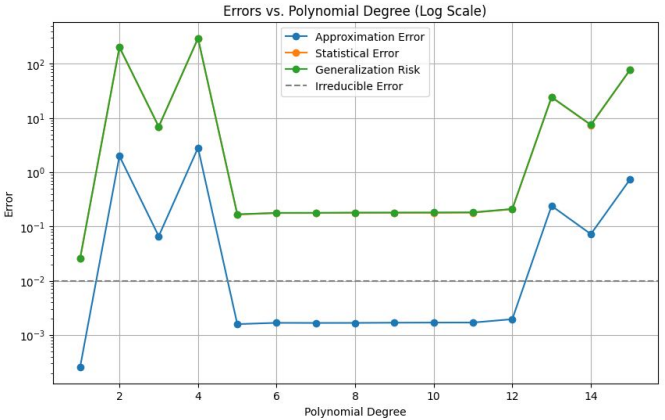
- Written as $(b\beta - \beta_p)^\top H_p (b\beta - \beta_p)$, where H_p is the Hilbert matrix.

Polynomial Regression & Estimation Error

```python
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import inv

# Generate some sample data
np.random.seed(0)
u = np.linspace(0, 1, 100)
true_beta_extended = true_beta = np.array([1, -2, 3, -4, 5,1, -2, 3, 0, 5,1, -2, 0, -4, 5])
true_beta_extended = np.concatenate([np.array([1, -2, 3, -4, 5]), np.zeros(10)])
y_true = np.polyval(true_beta_extended[::-1], u)
y = y_true + np.random.normal(0, 0.1, len(u))

# Irreducible error (noise variance)
irreducible_error = 0.1**2

# Initialize lists to store the errors
approximation_errors = []
statistical_errors = []
generalization_risks = []

# Iterate over a range of polynomial degrees from 1 to 15
for p in range(1, 16):
    X = np.vander(u, p, increasing=True)
    beta_hat = inv(X.T @ X) @ X.T @ y

    # Approximation error calculation
    approximation_error = np.trapz((X @ beta_hat - X @ true_beta_extended[:p]) ** 2, u)
    approximation_errors.append(approximation_error)

    # Statistical error calculation
    H_p = X.T @ X
    adjusted_true_beta = true_beta_extended[:p]
    statistical_error = (beta_hat - adjusted_true_beta).T @ H_p @ (beta_hat - adjusted_true_beta)
    statistical_errors.append(statistical_error)

    # Generalization risk calculation (sum of approximation and statistical errors)
    generalization_risks.append(approximation_error + statistical_error)
```



Log Scale



CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

Estimating Generalization Risk with In-Sample Risk Estimation + Optimism Concept

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Estimating Generalization Risk

- **Objective:**
  - Develop methods to estimate generalization risk without the drawbacks of test loss.
- **Generalization Risk Estimation via Test Loss:**
  - Test loss can provide inconsistent risk estimates across different training sets.
  - Inefficient data use when limited data is reserved for testing.

In-Sample Risk Estimation Methodology

In-Sample Risk Estimation

- **Objective**: Estimate the risk of a prediction function g within a training dataset τ.
- **Definition**: Measures the average loss of g over the training samples.
- **Formula**:
  $\hat{l}(g) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, g(x_i))$
  Where:
  - N is the number of training samples,
  - L is the loss function,
  - y_i and x_i are the observed output and input for the i-th sample,
  - $g(x_i)$ is the prediction of g for x_i.
- **Key Points**:
  - **Purpose**: Provides an estimate of how well g performs on the training data.
  - **Limitation**: May not accurately reflect out-of-sample risk due to overfitting or underfitting.
- **Significance**:
  - Critical for model selection and tuning within the training phase.
  - Serves as a basis for adjustments before evaluating the model's generalization capabilities.

Optimism in Statistical Learning and Data Science

Optimism in Model Evaluation

- **Definition**: The tendency for in-sample risk estimates to underestimate the true out-of-sample generalization risk due to overfitting.
- **Mathematical Context**:
  - In-sample Risk: $l_{\text{in}}(g)$
  - True Generalization Risk: $l_{\text{out}}(g)$
  - Optimism: $l_{\text{opt}} = l_{\text{out}}(g) - l_{\text{in}}(g)$
- **Key Points**:
  - **Source of Optimism**: Arises when a model is too complex relative to the amount of training data, capturing noise as if it were a real pattern.
  - **Impact**: Leads to overly optimistic performance estimates during training, which do not hold up on new, unseen data.

- **Mitigation Strategies**:
  - Cross-validation
  - Regularization techniques
  - Choosing simpler models when appropriate

Expected Optimism and Covariance

- **Expected Optimism (EXopT):**
  - Captures the bias in training loss as an estimate of in-sample risk.
  - $E[\mathrm{op}_T | X_1 = x_1, ..., X_n = x_n] =: E_X \mathrm{op}_T$.
- **Covariance Relationship:**
  - Expected optimism is related to the covariance between predicted and observed responses.
  - For squared-error and 0–1 loss: $E_X \mathrm{op}_T = \frac{2}{n} \sum_{i=1}^{n} \mathrm{Cov}_X(g_T(x_i), Y_i)$.
- **Interpretation:**
  - Covariance measures the degree to which the model's predictions are in line with actual outcomes.
  - A high covariance indicates that the model's training loss is overly optimistic about its performance.
- **Implications for Model Assessment:**
  - Understanding this covariance helps in adjusting for optimism, leading to more accurate risk estimates.
  - Provides a mathematical foundation for correcting overfitting in model evaluations.

Expected Optimism and Covariance

- **Expected Optimism (EXopT):**
  - Quantifies training loss's underestimation of in-sample risk.
  - $E_X \text{op}_T = \frac{2}{n} \sum \text{Cov}_X(g_T(x_i), Y_i)$.
- **Covariance Role:**
  - Links expected optimism to the alignment between model predictions and actual outcomes.
  - Higher covariance implies greater optimism, indicating potential overfitting.
- **Model Evaluation:**
  - Covariance analysis aids in adjusting risk estimates, enhancing model assessment accuracy.

Proof Outline for Expected Optimism

- **Conditional Expectation:**
  - Expectations are conditional on feature vectors $X_1 = x_1, ..., X_n = x_n$.
- **Independent Response Generation:**
  - Generate new responses $Y_i' \sim f(y|x_i)$ independent of training responses $Y_1, ..., Y_n$.
- **In-Sample Risk Estimation:**
  - Calculate $l_{\text{in}}(g_\tau) = \frac{1}{n} \sum_{i=1}^{n} \text{Loss}(Y_i', g_\tau(x_i))$.
- **Predicted vs. Actual Response:**
  - Let Y_i be the actual and $g_T(x_i)$ the predicted value. Introduce Y_i' as an independent copy of Y_i.
- **Expected Optimism Computation:**
  - $E_X \text{op}_T = \frac{1}{n} \sum_{i=1}^{n} (\text{Loss}(Y_i', g_\tau(x_i)) - \text{Loss}(Y_i, g_\tau(x_i)))$.
- **Role of Covariance:**
  - Link expected optimism to the covariance between $g_T(x_i)$ and Y_i for squared-error and 0–1 loss.
- **Implication:**
  - This framework provides a methodical approach to quantify and correct for the optimism in training loss, enhancing the reliability of model risk assessments.

Calculation in Expected Optimism Proof (1)

- **Setting the Scene:**
  - Consider the squared-error loss: $\text{Loss}(Y, g_T(x)) = (Y - g_T(x))^2$.
- **Key Variables:**
  - Actual response: Y_i.
  - Predicted response: $g_T(x_i)$.
  - Independent copy: Y_i'.
- **In-Sample Risk Estimation:**
  - $l_{\text{in}}(g_\tau) = \frac{1}{n}\sum_{i=1}^{n}(Y_i' - g_\tau(x_i))^2$.
- **Training Loss:**
  - $l_\tau(g_\tau) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - g_\tau(x_i))^2$.

Calculation in Expected Optimism Proof (1)

- **Expected Optimism (EXopT):**
  - $E_X \text{op}_T = \frac{1}{n} \sum_{i=1}^{n} \left[E((Y_i' - g_T(x_i))^2) - E((Y_i - g_T(x_i))^2) \right]$.
- **Expanding the Expectation:**
  - Use linearity of expectation and properties of covariance:
    - $E((Y_i' - g_T(x_i))^2) = \text{Var}(Y_i') + \text{Var}(g_T(x_i)) - 2\text{Cov}(Y_i', g_T(x_i))$.
    - Since Y_i' and $g_T(x_i)$ are independent, $\text{Cov}(Y_i', g_T(x_i)) = 0$.
- **Combining Terms:**
  - Simplify the expression by canceling out common terms in the optimism calculation.
- **Final Expression:**
  - Derive the expression for $E_X \text{op}_T$ that relates it to the covariance between the observed and predicted responses for the squared-error loss.

Calculation in Expected Optimism Proof (2)

- **Expected Optimism (EXopT):**
  - $E_X \text{op}_T = \frac{1}{n} \sum_{i=1}^{n} \left[E((Y_i' - g_T(x_i))^2) - E((Y_i - g_T(x_i))^2) \right]$.
- **Expanding the Expectation:**
  - Use linearity of expectation and properties of covariance:
    - $E((Y_i' - g_T(x_i))^2) = \text{Var}(Y_i') + \text{Var}(g_T(x_i)) - 2\text{Cov}(Y_i', g_T(x_i))$.
    - Since Y_i' and $g_T(x_i)$ are independent, $\text{Cov}(Y_i', g_T(x_i)) = 0$.
- **Combining Terms:**
  - Simplify the expression by canceling out common terms in the optimism calculation.
- **Final Expression:**
  - Derive the expression for $E_X \text{op}_T$ that relates it to the covariance between the observed and predicted responses for the squared-error loss.

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

K-Fold Cross Validation

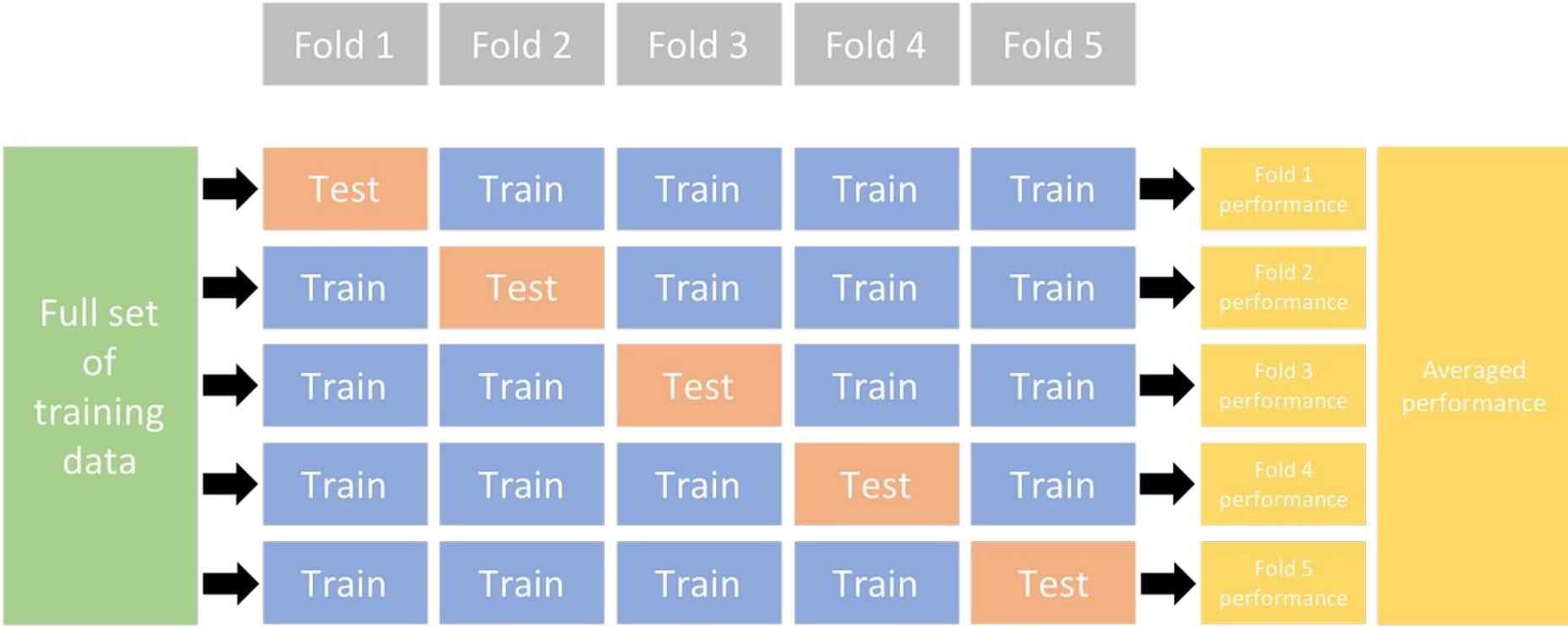
CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Introduction to K-Fold Cross-Validation

1. **What is Cross-Validation?**
   - A robust statistical method to estimate the generalization capability of a predictive model.
2. **Need for Cross-Validation:**
   - Essential when a large, separate test dataset is unavailable.
   - Helps in effectively using limited data for both training and validation.
3. **K-Fold Cross-Validation Overview:**
   - Involves dividing the dataset into K distinct subsets or 'folds'.
   - Each fold gets a turn to serve as a test set with the remaining $K - 1$ folds used for training.
4. **Purpose of Cross-Validation:**
   - Aims to provide a reliable estimate of the model's performance on unseen data.
   - Balances the model's ability to generalize beyond the training data while avoiding overfitting.
5. **Practical Relevance:**
   - Facilitates model selection, tuning, and validation, ensuring the model's effectiveness and robustness.

Introduction to K-Fold Cross-Validation



https://docs.ultralytics.com/guides/kfold-cross-validation/

Introduction to K-Fold Cross-Validation

1. **Partitioning the Dataset:**
   - Split dataset T of size n into K folds: $C_1, C_2, \ldots, C_K$.
2. **Iterative Training and Testing Process:**
   - For each iteration k, train the model on T_{-k} and test on C_k.
3. **Calculation of Test Loss:**
   - Compute the test loss l_{C_k} for each fold, serving as an estimator for the generalization risk.
4. **Cross-Validation Loss Computation:**
   - $CV_K = \frac{1}{n} \sum_{k=1}^{K} n_k l_{C_k}(g_{T_{-k}})$.
   - Represents the weighted average of the test losses from each fold.
5. **Estimating Expected Generalization Risk:**
   - CV_K estimates the expected generalization risk $E[l(g_T)]$, offering a comprehensive measure of the model's predictive power.
6. **Conclusion:**
   - The mathematical rigor of K-fold cross-validation provides a systematic and unbiased approach to evaluating model performance, crucial for data-driven decision-making in model development.

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

Data Modeling in Statistical Learning

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Introduction to Data Modeling in Statistical Learning

- **Purpose**: Create mathematical representations of real-world phenomena to analyze, predict, and make decisions.

Importance in Statistical Learning

- Facilitates understanding complex systems.
- Enables prediction and inference from data.

Modeling with Random Vectors

- **Data as Random Vectors**:
  - $x = [x_1, ..., x_p]^\top$
- **Outcome of Random Vector**:
  - $X = [X_1, ..., X_p]^\top$

Unknown Probability Distribution Functions

- **Assumption**:
  - x arises from an unknown pdf f.
- **Initial Model**:
  - Assume x is the outcome of X with pdf f.

The Concept of IID in Data Analysis

IID: Independent and Identically Distributed

- **Assumption**:
  - $X_1, ..., X_n \sim \text{iid from } f \text{ or Dist.}$

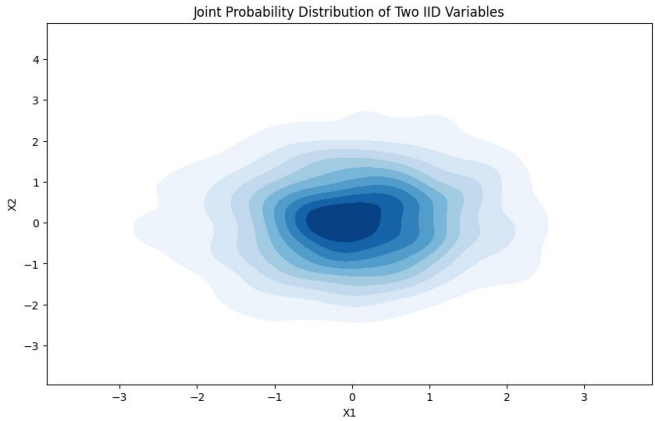
Simplification in Modeling

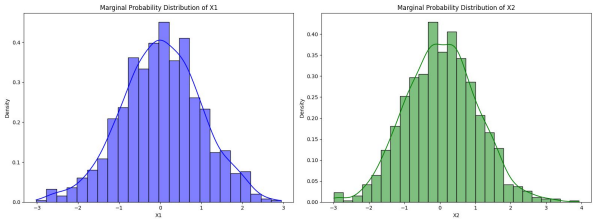
- **Joint Density**:
  - $f_{X_1,...,X_n}(x_1, ..., x_n) = f(x_1) \cdot ... \cdot f(x_n)$

Implications for Data Analysis

- **Knowledge Independence**:
  - Knowing one variable (X_i) gives no information about another (X_j).
- **Symmetry in Joint PDF**:
  - $f_{X_1,...,X_n}(x_1, ..., x_n) = f_{X_{\pi_1},...,X_{\pi_n}}(x_{\pi_1}, ..., x_{\pi_n})$ for any permutation π.
- **Exchangeability**:
  - A sequence $X_1, X_2, ...$ is exchangeable if permutational invariance holds for any finite subset.

```python
from scipy.stats import multivariate_normal

# Generate data from a bivariate normal distribution
mean = [0, 0]  # Mean of both variables
cov = [[1, 0], [0, 1]]  # Covariance matrix (diagonal, since variab
data = np.random.multivariate_normal(mean, cov, 1000)

x, y = data.T

# Plot the joint probability distribution
plt.figure(figsize=(10, 6))
sns.kdeplot(x=x, y=y, cmap="Blues", fill=True)
plt.title('Joint Probability Distribution of Two IID Variables')
plt.xlabel('X1')
plt.ylabel('X2')
plt.show()

# Plot the marginal probability distributions
fig, ax = plt.subplots(1, 2, figsize=(16, 6))

# X1 marginal
sns.histplot(x, kde=True, ax=ax[0], color="blue", stat="density")
ax[0].set_title('Marginal Probability Distribution of X1')
ax[0].set_xlabel('X1')
ax[0].set_ylabel('Density')

# X2 marginal
sns.histplot(y, kde=True, ax=ax[1], color="green", stat="density")
ax[1].set_title('Marginal Probability Distribution of X2')
ax[1].set_xlabel('X2')
ax[1].set_ylabel('Density')

plt.tight_layout()
plt.show()
```





The Concept of IID in Data Analysis

IID: Independent and Identically Distributed

- **Assumption**:
  - $X_1, ..., X_n \sim \text{iid from } f \text{ or Dist.}$

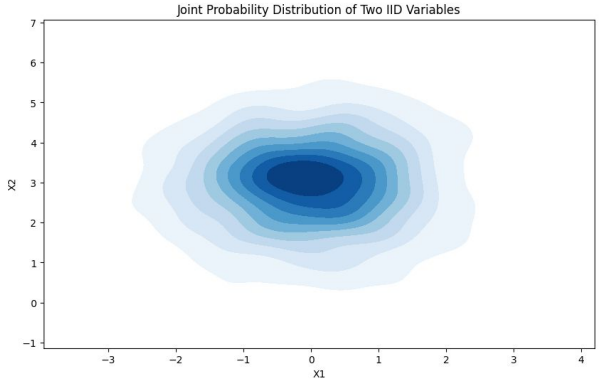
Simplification in Modeling

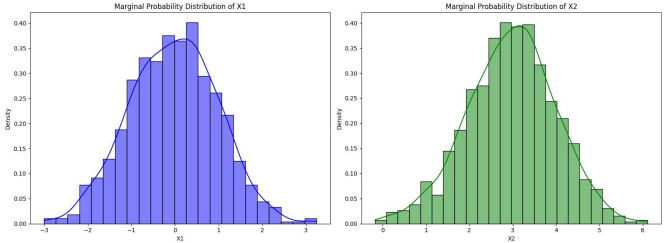
- **Joint Density**:
  - $f_{X_1,...,X_n}(x_1, ..., x_n) = f(x_1) \cdot ... \cdot f(x_n)$

Implications for Data Analysis

- **Knowledge Independence**:
  - Knowing one variable (X_i) gives no information about another (X_j).
- **Symmetry in Joint PDF**:
  - $f_{X_1,...,X_n}(x_1, ..., x_n) = f_{X_{\pi_1},...,X_{\pi_n}}(x_{\pi_1}, ..., x_{\pi_n})$ for any permutation π.
- **Exchangeability**:
  - A sequence $X_1, X_2, ...$ is exchangeable if permutational invariance holds for any finite subset.

```python

# Generate data from a bivariate normal distribution
mean = [0, 3]  # Mean of both variables
cov = [[1, 0], [0, 1]]  # Covariance matrix (diagonal, since variables
data = np.random.multivariate_normal(mean, cov, 1000)

x, y = data.T

# Plot the joint probability distribution
plt.figure(figsize=(10, 6))
sns.kdeplot(x=x, y=y, cmap="Blues", fill=True)
plt.title('Joint Probability Distribution of Two IID Variables')
plt.xlabel('X1')
plt.ylabel('X2')
plt.show()

# Plot the marginal probability distributions
fig, ax = plt.subplots(1, 2, figsize=(16, 6))

# X1 marginal
sns.histplot(x, kde=True, ax=ax[0], color="blue", stat="density")
ax[0].set_title('Marginal Probability Distribution of X1')
ax[0].set_xlabel('X1')
ax[0].set_ylabel('Density')

# X2 marginal
sns.histplot(y, kde=True, ax=ax[1], color="green", stat="density")
ax[1].set_title('Marginal Probability Distribution of X2')
ax[1].set_xlabel('X2')
ax[1].set_ylabel('Density')

plt.tight_layout()
plt.show()

```





Understanding Sampling Distributions

Overview of Sampling Distributions

- **Definition**: Describes the probability of observing a statistic (e.g., mean, variance) within a sample drawn from a larger population.

Common Sampling Distributions

- **Normal Distribution (N)**:
  - $N(\mu, \sigma^2)$
- **Binomial Distribution (Bin)**:
  - $\text{Bin}(n, p)$
- **Exponential Distribution (Exp)**:
  - $\text{Exp}(\lambda)$

Parameter Estimation in Parametric Models

Parameter Estimation Overview

- **Objective**: Estimate the parameters of a model's sampling distribution based on observed data.

Parametric vs. Non-Parametric Settings

- **Parametric Models**: Assume a specific form for the sampling distribution, defined by a set of parameters.
  - Example: Estimating μ and σ^2 in a normal distribution.
- **Non-Parametric Models**: Do not assume a specific form for the sampling distribution.
  - Focus on estimating the distribution directly from data.

Parametric Models in Statistical Learning

Parametric Models Defined

- **Assumption**: The model assumes a specific form for the sampling distribution, characterized by a finite set of parameters.

Examples and Parameter Estimation

- **Normal Distribution**:
  - Model: $N(\mu, \sigma^2)$
  - Parameters: Mean (μ) and variance (σ^2)
  - Estimation: Use data to estimate μ and σ^2.

Advantages

- **Simplicity**: Fewer parameters to estimate simplifies analysis.
- **Efficiency**: More data-efficient when the model form is correct.

Non-Parametric Models in Statistical Learning

Non-Parametric Models Defined

- **Approach**: Do not assume a specific form for the sampling distribution. Focus on estimating the distribution directly from the data.

Example: Kernel Density Estimation

- **Concept**: Estimate the probability density function of a random variable without assuming its shape.
- **Application**: Use kernels to smooth data points into a continuous density curve.

Advantages

- **Flexibility**: Can model complex distributions that parametric models cannot.
- **Adaptability**: More robust to data form assumptions, ideal for exploratory data analysis.

These slides differentiate between parametric and non-parametric models in statistical learning, providing clear examples for each and highlighting their respective advantages and use cases.

Example. Income vs. Seniority vs. Year of Education

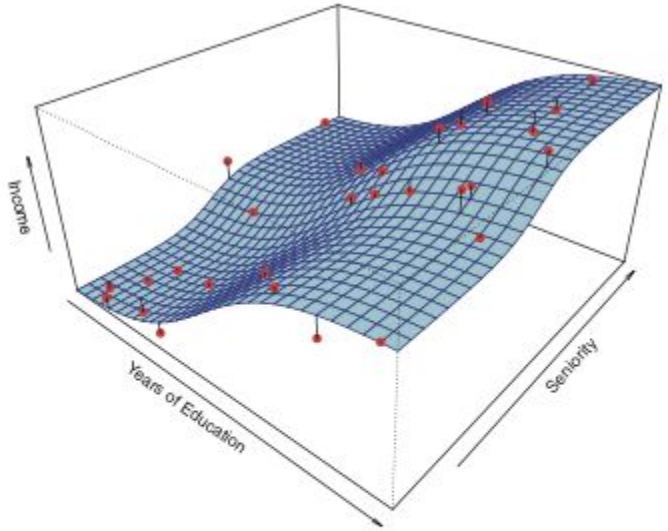


FIGURE 2.3. *The plot displays* `income` *as a function of* `years of education` *and* `seniority` *in the* `Income` *data set. The blue surface represents the true underlying relationship between* `income` *and* `years of education` *and* `seniority`*, which is known since the data are simulated. The red dots indicate the observed values of these quantities for* 30 *individuals.*

Parametric Estimation

Two-Step Approach

1. **Assumption on Functional Form**:
   - Assume f is linear: $f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p$.
   - Simplifies estimating $f(X)$ to estimating $\beta_0, \beta_1, ..., \beta_p$.
2. **Model Fitting**:
   - Use training data to estimate parameters (β).
   - Common approach: (Ordinary) Least Squares.

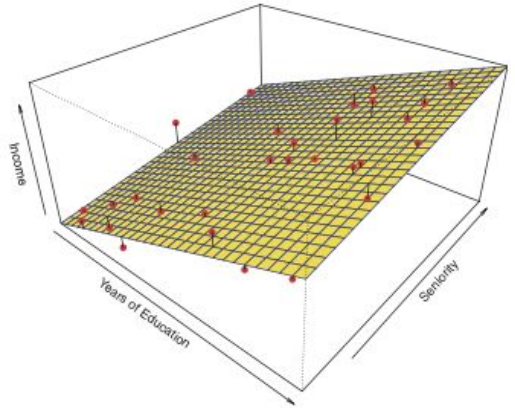


FIGURE 2.4. *A linear model fit by least squares to the* Income *data from Figure 2.3. The observations are shown in red, and the yellow plane indicates the least squares fit to the data.*

Key Concept

- **Parametric Approach**: Reduces estimating a complex function to estimating a set of parameters.
- **Advantages**: Simplicity and efficiency in estimation.
- **Challenges**: Risk of model mismatch with the true f.

Non-Parametric Estimation

Approach Without Explicit Assumptions

- **Objective**: Estimate f closely to data points without assuming a specific form.

Example: Thin-Plate Spline

- **Application**: Fit data with a flexible surface to capture complex patterns.
- **Smoothness Selection**: Analyst must decide the level of smoothness, balancing detail and overfitting.

Key Concept

- **Non-Parametric Approach**: Offers flexibility to fit a wide range of shapes for f.
- **Advantages**: Can accurately fit more complex models.
- **Challenges**: Requires a larger number of observations; risk of overfitting.

Comparison

- **Parametric vs. Non-Parametric**:
  - Parametric methods are efficient with fewer observations but risk model mismatch.
  - Non-Parametric methods offer flexibility at the cost of needing more data and increased risk of overfitting.

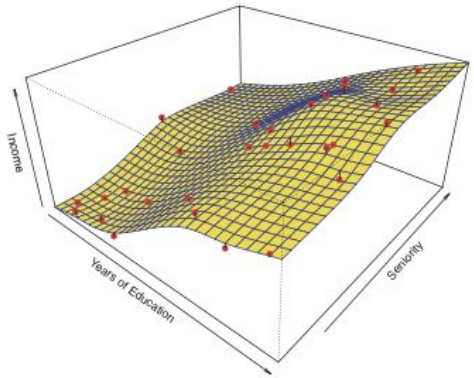


FIGURE 2.5. *A smooth thin-plate spline fit to the* Income *data from Figure 2.3 is shown in yellow; the observations are displayed in red. Splines are discussed in Chapter 7.*

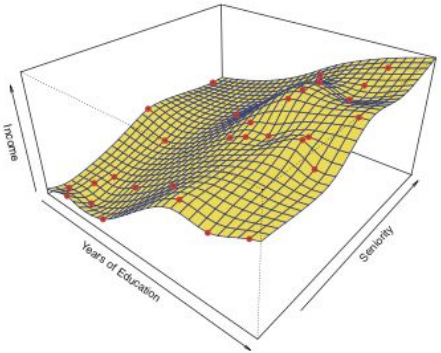


FIGURE 2.6. *A rough thin-plate spline fit to the* Income *data from Figure 2.3. This fit makes zero errors on the training data.*

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

Understanding Sampling Distributions and Exchangeability

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Non-Parametric Estimation

Approach Without Explicit Assumptions

- **Objective**: Estimate f closely to data points without assuming a specific form.

Example: Thin-Plate Spline

- **Application**: Fit data with a flexible surface to capture complex patterns.
- **Smoothness Selection**: Analyst must decide the level of smoothness, balancing detail and overfitting.

Key Concept

- **Non-Parametric Approach**: Offers flexibility to fit a wide range of shapes for f.
- **Advantages**: Can accurately fit more complex models.
- **Challenges**: Requires a larger number of observations; risk of overfitting.

Comparison

- **Parametric vs. Non-Parametric**:
  - Parametric methods are efficient with fewer observations but risk model mismatch.
  - Non-Parametric methods offer flexibility at the cost of needing more data and increased risk of overfitting.

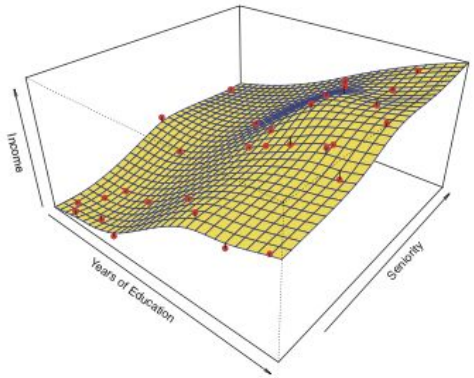


FIGURE 2.5. *A smooth thin-plate spline fit to the* Income *data from Figure 2.3 is shown in yellow; the observations are displayed in red. Splines are discussed in Chapter 7.*

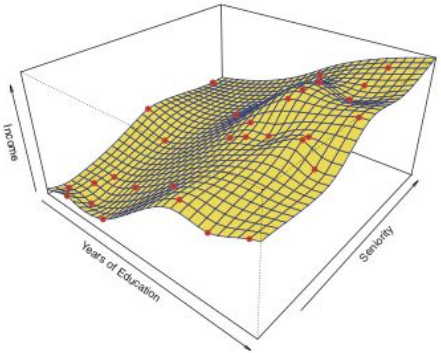


FIGURE 2.6. *A rough thin-plate spline fit to the* Income *data from Figure 2.3. This fit makes zero errors on the training data.*

Robotics & Internet-of-Things

جامعة الأمير سلطان PRINCE SULTAN UNIVERSITY

CS316: INTRODUCTION TO AI AND DATA SCIENCE

CHAPTER 7
STATISTICAL LEARNING

LECTURE 3
DATA ENTROPY IN DATA SCIENCE

Prof. Anis Koubaa

February 2024

www.riotu-lab.org

Information in Statistical Learning

- **In Statistical Learning**: Information quantifies the insight or knowledge gained from data regarding underlying patterns or predictive relationships.

Link to Probability and Entropy

- **Entropy**: Measures system uncertainty or randomness.
  - **Equation**: $H(X) = -\sum P(x_i) \log_2 P(x_i)$
- **Probability**: Basis for calculating entropy, reflecting event likelihoods.

Key Concepts

- **Higher Entropy**: Indicates more unpredictability, equating to greater information content.
- **Message Source**: The average information output is gauged by its entropy.

Statistical Learning Implication

- **Predictive Modeling**: Entropy helps in understanding and optimizing models by measuring the amount of uncertainty or information a model can handle or reduce.

Essentials of Information Theory

- **Definition of Information Theory**:
  - A mathematical framework for quantifying information, communication, and uncertainty. Fundamental to data compression, error detection and correction, and cryptography within data science.
- **Entropy as a Core Concept**:
  - Entropy (H) measures the amount of uncertainty or randomness in a system or dataset. It quantifies the expected value of the information contained in a message.
  - Equation: $H(X) = -\sum_i P(x_i) \log_2 P(x_i)$, where $P(x_i)$ is the probability of event x_i.
- **Historical Context and Contributions by Claude Shannon**:
  - Claude Shannon, known as the "father of information theory," introduced the concept of entropy in his landmark 1948 paper, "A Mathematical Theory of Communication."
  - Shannon's work laid the foundation for modern digital communication and data encoding, emphasizing the importance of entropy in measuring information efficiency and transmission capacity.

Relevance to Data Science:

- Information theory principles guide data encoding, compression, and transmission processes, ensuring efficient and secure data handling.
- **Entropy** is utilized in **machine learning** for **feature selection**, **model complexity assessment,** and **understanding data distributions**, impacting **decision-making and predictive modeling**.

Probability, Entropy, and Information Content

Core Concepts

- **Entropy**: Quantifies uncertainty or randomness in a system.
- **Information Theory**: Entropy measures the average information produced by a source.

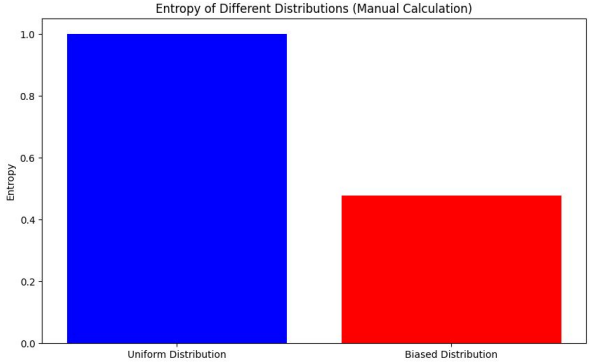
Key Insights

- **Higher Entropy**: Equals more unpredictability and information.
- **Function**: $H(X) = -\sum P(x_i) \log_2 P(x_i)$.
- **Interpretation**: Entropy increases with the diversity of possible outcomes.

Conclusion

- Entropy links unpredictability to information content, serving as a fundamental measure in information theory and data science.

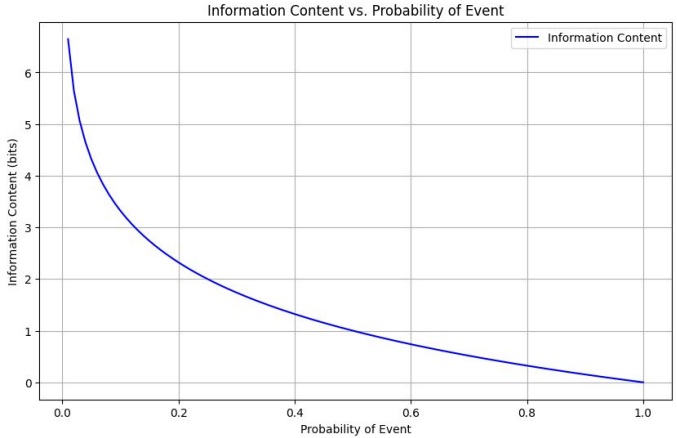
$$-\sum P(x_i) \log_2 P(x_i)$$



Understanding Logarithms in Entropy

- **Logarithmic Function Purpose**: Quantifies information content; base 2 reflects binary encoding (bits).
- **Key Insight**:
  - Log function transforms event probabilities to information measures.
  - Common events ($P(x_i)$ high) → Less information ($\log_2 P(x_i)$ small).
- **Role of Negative Sign**:
  - Ensures entropy is positive, mirroring information's additive nature.
- **Interpretation**:
  - Logarithm inversely relates event probability with its 'surprise' level.
  - Certain event ($P(x) = 1$) yields no information ($\log_2 P(x) = 0$).
- **Summary**:
  - Logarithms in entropy articulate how unpredictability translates to information content, highlighting the inverse relationship between probability and information gain.

```python
import numpy as np
import matplotlib.pyplot as plt

# Define probabilities from 0.01 to 1 (avoiding 0 to prevent log(0))
probabilities = np.linspace(0.01, 1, 100)
# Calculate the information content for each probability
information_content = -np.log2(probabilities)

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(probabilities, information_content, color='blue', label='Information Content')
plt.xlabel('Probability of Event')
plt.ylabel('Information Content (bits)')
plt.title('Information Content vs. Probability of Event')
plt.grid(True)
plt.legend()
plt.show()

```



CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
DATA ENTROPY IN DATA SCIENCE

Why Using Log2?

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Why Log Base 2 for Measuring Information

Communication Context

Why Log Base 2?

1. **Binary System Alignment**
   - Matches the binary (0 or 1) nature of digital information processing.
   - $\log_2(n)$ calculates the bits needed for n distinct messages.
2. **Communication Efficiency**
   - Minimizes bits required for encoding, aligning with information theory's goals.
   - Example: 4 messages need 2 bits ($\log_2(4) = 2$).

```python
import math

# Define a function to calculate bits needed for a given number of outcomes
def calculate_bits(outcomes):
    return math.log2(outcomes)

# Calculate bits for 1, 2, 4, 8, 16, 32 outcomes
outcomes_list = [1, 2, 4, 8, 16, 32, 64, 128, 256]
bits_needed = [calculate_bits(outcomes) for outcomes in outcomes_list]

# Pair each number of outcomes with the corresponding bits needed
outcome_bits_pairs = list(zip(outcomes_list, bits_needed))

outcome_bits_pairs

```

```
[(1, 0.0),
 (2, 1.0),
 (4, 2.0),
 (8, 3.0),
 (16, 4.0),
 (32, 5.0),
 (64, 6.0),
 (128, 7.0),
 (256, 8.0)]
```

Why Log Base 2 for Measuring Information

3. **Uncertainty Reduction Quantification**
   - Information reduces uncertainty about the sent message.
   - Log base 2 measures this reduction, representing information as binary questions.
4. **Universality and Simplicity**
   - Offers a simple, universal measure for comparing information across systems.
   - Facilitates straightforward calculations and comparisons in bits.

Communication Context

Essentials of Information Theory

```python
import math

# Define a function to calculate bits needed for a given number of outcomes
def calculate_bits(outcomes):
    return math.log2(outcomes)

# Calculate bits for 1, 2, 4, 8, 16, 32 outcomes
outcomes_list = [1, 2, 4, 8, 16, 32, 64, 128, 256]
bits_needed = [calculate_bits(outcomes) for outcomes in outcomes_list]

# Pair each number of outcomes with the corresponding bits needed
outcome_bits_pairs = list(zip(outcomes_list, bits_needed))

outcome_bits_pairs

```

```
[(1, 0.0),
 (2, 1.0),
 (4, 2.0),
 (8, 3.0),
 (16, 4.0),
 (32, 5.0),
 (64, 6.0),
 (128, 7.0),
 (256, 8.0)]
```

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
DATA ENTROPY IN DATA SCIENCE

Essentials of Information Theory

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Essentials of Information Theory

Mathematical Formulation of Entropy

- **Equation for Discrete Variables**:

$$H(X) = -\sum_i P(x_i) \log_2 P(x_i)$$

- **Equation for Continuous Variables**:

$$H(X) = -\int p(x) \log_2 p(x) dx$$

Entropy in the Context of Information Content

- Entropy measures the average amount of information produced by a stochastic source of data.
- Higher entropy indicates more unpredictability in information content.

Examples of Entropy Calculations

- **Coin Toss**:
  - A fair coin has two outcomes (Heads or Tails) with equal probability.
  - Entropy:

$$H(X) = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1 \text{ bit}$$

- **Dice Roll**:
  - A fair six-sided die has six outcomes, each with an equal probability.
  - Entropy:

$$H(X) = -\sum_{i=1}^{6} \frac{1}{6} \log_2 \frac{1}{6} \approx 2.58 \text{ bits}$$

The entropy of a fair coin toss is 1 bit, indicating predictability among two outcomes, whereas a fair six-sided die has an entropy of about 2.58 bits, showing greater unpredictability due to more outcomes.

Conditional Entropy, Mutual Information

X and Y independent

$$P(X = x, Y = y) = P(X = x) \times P(Y = y)$$

```
# Marginal probability distributions of X and
p_x = np.array([0.5, 0.5])  # P(X)
p_y = np.array([0.5, 0.5])  # P(Y)

# Joint probability distribution of X and Y
p_xy = np.array([[0.25, 0.25], [0.25, 0.25]])
```

1. **Conditional Entropy:**
   - **Definition:** Measure of uncertainty in X given Y.
   - **Mathematical Expression:**

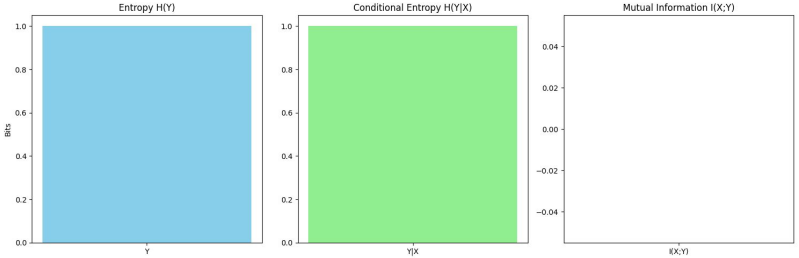
$$H(X|Y) = -\sum_{x \in X, y \in Y} p(x,y) \log_2 \frac{p(x,y)}{p(y)}$$

   - **Interpretation:** Amount of information required to describe X when Y is known.
2. **Mutual Information:**
   - **Definition:** Quantifies the amount of information shared between X and Y.
   - **Mathematical Expression:**

$$I(X;Y) = \sum_{x \in X, y \in Y} p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)}$$

   - **Relationship:**

$$I(X;Y) = H(X) - H(X|Y)$$



1. **Entropy of Y (H(Y))**: This bar shows the entropy of Y, calculated as 1.0 bits. Entropy here measures the uncertainty or variability in Y, with our simple scenario assuming a uniform distribution (hence the maximum entropy for a binary variable).
2. **Conditional Entropy of Y given X (H(Y|X))**: Also shown as 1.0 bits, which in our simplified, independent scenario, equals the entropy of Y. This reflects that knowing X does not reduce uncertainty about Y due to the assumed independence. In scenarios where X and Y are not independent, $H(Y|X)$ would typically be lower than $H(Y)$, indicating that X provides information about Y.
3. **Mutual Information (I(X;Y))**: Illustrated as 0.0 bits, highlighting that in our scenario, knowing X provides no additional information about Y (again, under the assumption of independence). Mutual information measures the amount of information that one variable reveals about the other; a value of 0 indicates no shared information.

Conditional Entropy, Mutual Information

X and Y dependent

```
# Adjusted joint probability distribution
p_xy = np.array([[0.3, 0.2], [0.2, 0.3]])

# Calculate marginal probabilities of Y by
p_y = np.sum(p_xy, axis=0)
```

1. **Conditional Entropy:**
   - **Definition:** Measure of uncertainty in X given Y.
   - **Mathematical Expression:**

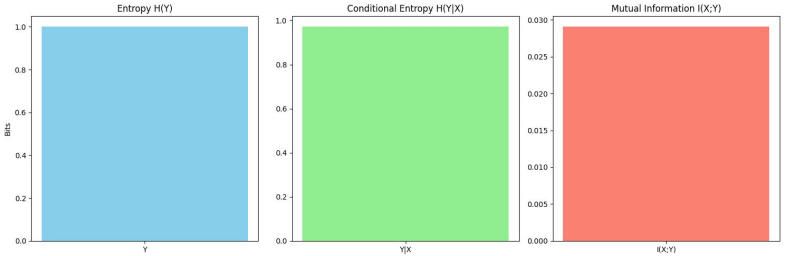
$$H(X|Y) = -\sum_{x \in X, y \in Y} p(x,y) \log_2 \frac{p(x,y)}{p(y)}$$

   - **Interpretation:** Amount of information required to describe X when Y is known.
2. **Mutual Information:**
   - **Definition:** Quantifies the amount of information shared between X and Y.
   - **Mathematical Expression:**

$$I(X;Y) = \sum_{x \in X, y \in Y} p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)}$$

   - **Relationship:**

$$I(X;Y) = H(X) - H(X|Y)$$



- The **Entropy of** Y ($H(Y)$) is approximately 1.0 bit. This entropy value indicates the level of uncertainty or variability in Y without considering any other variables.
- The **Conditional Entropy of** Y **given** X ($H(Y|X)$) is approximately 0.971 bits. This is slightly less than the entropy of Y, reflecting that knowing the value of X reduces the uncertainty about Y slightly due to their dependency.
- The **Mutual Information between** X **and** Y ($I(X;Y)$) is approximately 0.029 bits. This value quantifies the amount of information that X and Y share; it's the reduction in uncertainty about Y that knowing X provides (and vice versa). Even though X and Y are not strongly dependent in this example (as seen by the relatively small mutual information), there's still a non-zero mutual information indicating some level of dependency.

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
DATA ENTROPY IN DATA SCIENCE

Entropy Rate in Stochastic Processes

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Entropy Rate in Stochastic Processes (E.g. Markov Chain)

1. **Context:**
   - Utilized in examining sequences or stochastic processes to ascertain their information characteristics.
2. **Definition:**
   - Mathematical Expression for Entropy Rate ($H'(X)$):

$$H'(X) = \lim_{n \to \infty} \frac{1}{n} H(X_1, X_2, \ldots, X_n)$$

   - Denotes the average entropy per symbol as the sequence length approaches infinity.
3. **Significance:**
   - Essential for deducing the predictability of a process, indicating how much history influences future behavior.
   - Provides insights into the memory depth of the process, revealing the extent to which past events inform future outcomes.

Example: Entropy Rate in Markov Chains - Concept

Markov Chains and Stationary Distribution

- **Markov Chain**: Sequence of random states with transition probabilities dependent only on the current state.
- **Stationary Distribution** π: Long-term state probabilities, satisfying $\pi P = \pi$ and $\sum \pi_i = 1$.

Transition Entropy

- **Definition**: Measures unpredictability of state transitions.
- **Calculation**: $H_{\text{transition}} = -\sum P_{ij} \log_2 P_{ij}$ for transitions from state i to j.

Entropy Rate

- **Purpose**: Quantifies average rate of new information production in a stochastic process.
- **Formula**: Weighted sum of transition entropies, $\sum \pi_i H_{\text{transition},i}$.

Insights

- **Interpretation**: Reflects process predictability; higher entropy rate implies greater randomness.

Example: Example Analysis with Python

Stationary Distribution Calculation

- **Equation:** Solve $\pi \mathbf{P} = \pi$ and $\sum \pi_i = 1$
- **Result:**
  - $\pi_A \approx 0.833$
  - $\pi_B \approx 0.167$

Transition Entropies

- **Equation:** $H_i = -\sum_j P_{ij} \log_2 P_{ij}$
- **Results:**
  - $H_A \approx 0.469$ bits
  - $H_B \approx 1.0$ bit

Entropy Rate of Markov Chain

- **Equation:** $H = \sum_i \pi_i H_i$
- **Result:**
  - $H \approx 0.557$ bits

Markov Chain Example

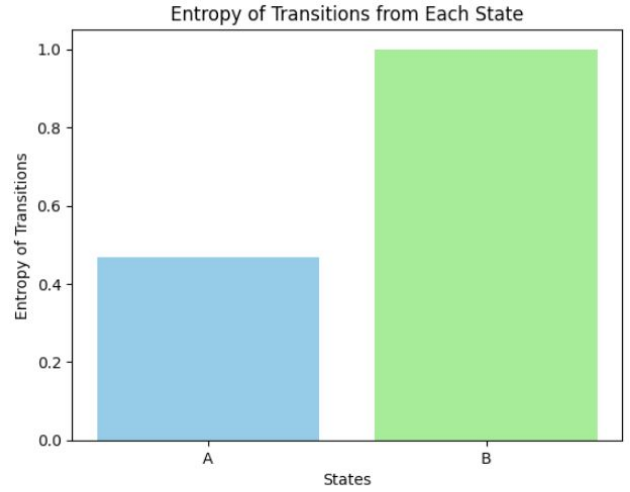
- States: A and B with transitions $P(A \to A) = 0.9, P(A \to B) = 0.1, P(B \to A) = 0.5, P(B \to B) = 0.5$.

Calculations and Visualization

- **Stationary Distribution**: $\pi_A \approx 0.833, \pi_B \approx 0.167$.
- **Transition Entropies**: $H_A \approx 0.469$ bits, $H_B \approx 1.0$ bit.
- **Entropy Rate**: ≈ 0.557 bits, indicating the chain's average new information rate.

Interpretation

- **Predictability**: State A shows lower transition entropy, indicating more predictability compared to B.
- **Overall Process**: With an entropy rate of 0.557 bits, the Markov chain exhibits moderate unpredictability over time.




```
(array([0.83333333, 0.16666667]),
 array([0.46899559, 1.        ]),
0.5574963279910677)
```

Applications of Entropy in Data Science

1. **Entropy in Machine Learning Models:**
   - Application: Utilized in decision tree algorithms (e.g., ID3, C4.5) for attribute selection.
   - Mechanism: Chooses the attribute that provides the highest information gain, effectively reducing entropy at each node split.
2. **Entropy as a Feature Selection Mechanism:**
   - Role: Identifies relevant features by measuring the amount of information each feature contributes to the target variable.
   - Benefit: Enhances model accuracy and efficiency by eliminating redundant or irrelevant features.

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
DATA ENTROPY IN DATA SCIENCE

Applications of Entropy in Data Science

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Applications of Entropy in Data Science

3. **Using Entropy for Anomaly Detection:**
   - Concept: Anomalies can cause significant changes in the entropy of a dataset.
   - Approach: Monitor entropy variations to detect anomalies, as they typically introduce disorder and increase entropy.
4. **Entropy in Data Analysis:**
   - Usage: Assess the complexity and predictability of datasets.
   - Application: In clustering, entropy can evaluate the homogeneity of clusters, with lower entropy indicating more coherent groupings.

Entropy in Cryptography and Security

1. **Introduction to Cryptographic Systems:**
   - Fundamental Concept: Entropy is pivotal in ensuring the unpredictability and security of cryptographic systems.
   - Role of Entropy: Enhances the robustness against attacks by ensuring that cryptographic keys and protocols are not easily predictable.
2. **Random Number Generation and Entropy Sources:**
   - Importance: Secure random number generation is crucial for encryption, key generation, and other cryptographic operations.
   - Entropy Sources: Utilizes various unpredictable sources (e.g., mouse movements, keystroke timings) to generate high-entropy random numbers.

Entropy in Cryptography and Security

3. **Entropy Measures in Password Strength:**
   - Concept: Entropy is a key metric in evaluating password strength, with higher entropy indicating a stronger password.
   - Application: Password policies enforce complexity requirements to increase entropy, reducing the chances of successful brute-force attacks.
4. **Cryptographic Key Generation:**
   - Relationship: The strength of a cryptographic key is directly tied to its entropy.
   - Best Practice: High-entropy keys are less susceptible to guessing or brute-force attacks, underscoring the importance of entropy in secure key generation.

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

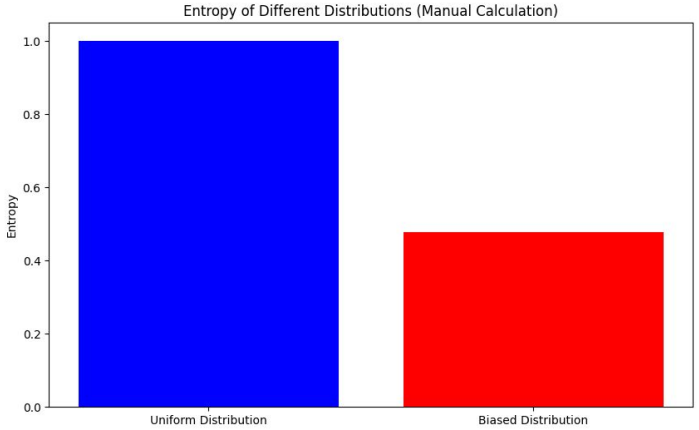
Entropy in Data Science
+
Examples

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Entropy in Data Science

- **Definition of Entropy**:
  - A measure of the uncertainty or randomness in a probability distribution, denoted as $H(P)$ for a distribution P.
- **Mathematical Expression**:
  - For discrete variables: $H(P) = -\sum_x P(x) \log P(x)$
  - For continuous variables: Analogously defined through differential entropy.
- **Relevance to Data Science**:
  - **Information Theory**: Fundamental for measuring information content and data compression efficiency.
  - **Machine Learning**: Utilized in decision tree algorithms (e.g., ID3, C4.5) for feature selection based on information gain.
  - **Model Evaluation**: In conjunction with KL divergence, entropy helps assess model performance by measuring the spread or predictability of the model's predictions.
- **Application Cases**:
  - **Feature Selection**: Identifying the most informative features that reduce uncertainty in predictions.
  - **Clustering Evaluation**: Measuring cluster purity or the effectiveness of segmentation.
  - **Communications**: Optimizing data encoding for efficient transmission or storage.

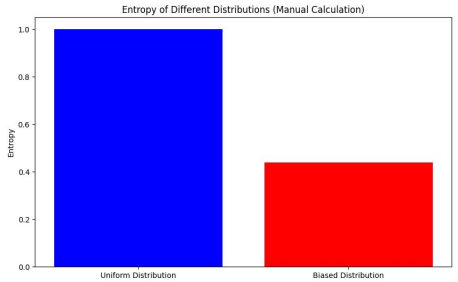


Understanding Entropy Through Python Example

- **Objective**: Illustrate entropy calculation and its implications using Python.

Entropy Equation:

- **Definition**: $H(P) = -\sum_x P(x) \log_2 P(x)$
- **Components**:
  - $P(x)$: Probability of observing class x.
  - $\log_2 P(x)$: Logarithm base 2 of $P(x)$, reflecting information content.

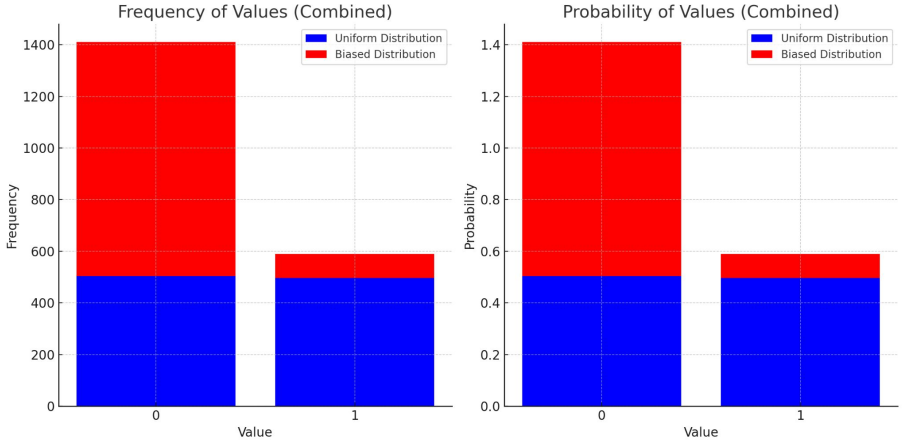



```python
import numpy as np
import matplotlib.pyplot as plt

# Define a function to calculate entropy of a dataset manually
def calculate_entropy_manual(data):
    # Count the frequency of each value in the dataset
    value_counts = np.bincount(data)
    probabilities = value_counts[np.nonzero(value_counts)] / len(data)
    # Calculate entropy manually
    entropy = -np.sum(probabilities * np.log2(probabilities))
    return entropy

# Generate a sample dataset with varying degrees of randomness
dataset_uniform = np.random.randint(0, 2, 1000)  # High entropy
dataset_biased = np.random.choice([0, 1], size=1000, p=[0.9, 0.1])  # Low

# Calculate entropy for each dataset using the manual method
entropy_uniform_manual = calculate_entropy_manual(dataset_uniform)
entropy_biased_manual = calculate_entropy_manual(dataset_biased)

# Plot the results
plt.figure(figsize=(10, 6))
plt.bar(['Uniform Distribution', 'Biased Distribution'],
    [entropy_uniform_manual, entropy_biased_manual], color=['blue', 'red'])
plt.ylabel('Entropy')
plt.title('Entropy of Different Distributions (Manual Calculation)')
plt.show()
```

Understanding Entropy Through Python Example

Python Implementation:

1. **Frequency Calculation**: `value_counts = np.bincount(data)`
   - Counts occurrences of each value in the dataset.
2. **Probability Calculation**: `probabilities = value_counts[np.nonzero(value_counts)] / len(data)`
   - Computes the probability of each value.
3. **Entropy Calculation**: `entropy = -np.sum(probabilities * np.log2(probabilities))`
   - Applies the entropy formula to calculate dataset entropy.




```python
import numpy as np
import matplotlib.pyplot as plt

# Define a function to calculate entropy of a dataset manually
def calculate_entropy_manual(data):
    # Count the frequency of each value in the dataset
    value_counts = np.bincount(data)
    probabilities = value_counts[np.nonzero(value_counts)] / len(data)
    # Calculate entropy manually
    entropy = -np.sum(probabilities * np.log2(probabilities))
    return entropy

# Generate a sample dataset with varying degrees of randomness
dataset_uniform = np.random.randint(0, 2, 1000)  # High entropy
dataset_biased = np.random.choice([0, 1], size=1000, p=[0.9, 0.1])  # Low

# Calculate entropy for each dataset using the manual method
entropy_uniform_manual = calculate_entropy_manual(dataset_uniform)
entropy_biased_manual = calculate_entropy_manual(dataset_biased)

# Plot the results
plt.figure(figsize=(10, 6))
plt.bar(['Uniform Distribution', 'Biased Distribution'],
  [entropy_uniform_manual, entropy_biased_manual], color=['blue', 'red'])
plt.ylabel('Entropy')
plt.title('Entropy of Different Distributions (Manual Calculation)')
plt.show()
```

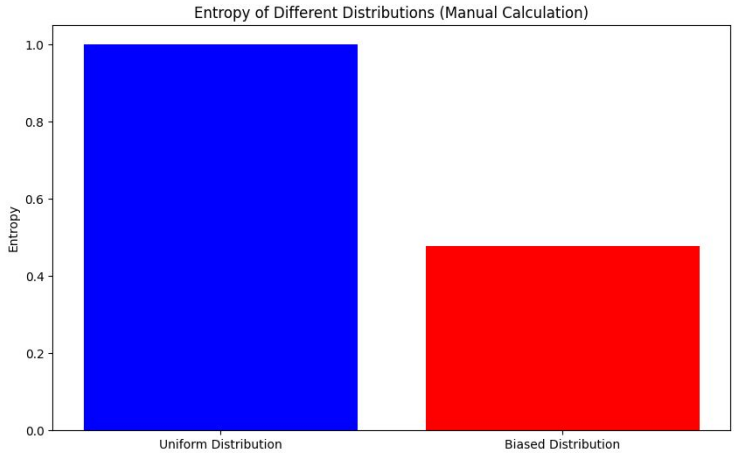
- **Uniform Distribution**: The variance is approximately 0.25.
- **Biased Distribution**: The variance is approximately 0.085.

Understanding Entropy Through Python Example

- **High Entropy (Uniform Distribution)**: Indicates greater unpredictability or randomness in data distribution.
- **Lower Entropy (Biased Distribution)**: Indicates less unpredictability or a more predictable data distribution.

Conclusion:

- **Entropy as a Measure**: Entropy quantifies the uncertainty in a dataset. High entropy suggests a uniform distribution of outcomes, implying higher unpredictability. Conversely, low entropy indicates a skewed distribution, suggesting predictability.




```python
import numpy as np
import matplotlib.pyplot as plt

# Define a function to calculate entropy of a dataset manually
def calculate_entropy_manual(data):
    # Count the frequency of each value in the dataset
    value_counts = np.bincount(data)
    probabilities = value_counts[np.nonzero(value_counts)] / len(data)
    # Calculate entropy manually
    entropy = -np.sum(probabilities * np.log2(probabilities))
    return entropy

# Generate a sample dataset with varying degrees of randomness
dataset_uniform = np.random.randint(0, 2, 1000)  # High entropy
dataset_biased = np.random.choice([0, 1], size=1000, p=[0.9, 0.1])  # Low

# Calculate entropy for each dataset using the manual method
entropy_uniform_manual = calculate_entropy_manual(dataset_uniform)
entropy_biased_manual = calculate_entropy_manual(dataset_biased)

# Plot the results
plt.figure(figsize=(10, 6))
plt.bar(['Uniform Distribution', 'Biased Distribution'],
  [entropy_uniform_manual, entropy_biased_manual], color=['blue', 'red'])
plt.ylabel('Entropy')
plt.title('Entropy of Different Distributions (Manual Calculation)')
plt.show()
```

- **Uniform Distribution**: The variance is approximately 0.25.
- **Biased Distribution**: The variance is approximately 0.085.

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

Advanced Topics in Entropy

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Cross-Entropy in Machine Learning

1. **Definition:**
   - Cross-Entropy: Quantifies the discrepancy between two probability distributions, often between the ground truth (P) and the predictions of a model (Q).
   - Mathematical Expression:

$$H(P, Q) = -\sum_{x} P(x) \log Q(x)$$

2. **Applications:**
   - Integral in optimizing classification models and neural network training.
   - Aims to minimize cross-entropy, enhancing model accuracy by adjusting parameters.
3. **Significance:**
   - Acts as a crucial loss function metric in classification, steering iterative model improvements.
   - Targets enhancing model precision over successive training iterations.
4. **Illustrative Example:**
   - Binary Classification Context: True label $y = 1$ with a predicted probability $p = 0.9$.
   - Cross-Entropy Loss Calculation:

$$-[y \log(p) + (1 - y) \log(1 - p)] = -(1 \log(0.9) + 0 \log(0.1))$$

   - Objective: Minimize this loss to align predicted probabilities closely with actual class labels.

Cross-Entropy in Binary Classification

Concept Overview

- **Cross-Entropy**: Measures the difference between true label distribution and model's predicted probabilities.
- **Usage**: Acts as a loss function in classification, guiding model optimization.

Mathematical Formulation

- For true label y and predicted probability p, cross-entropy H is:

$$H(y, p) = -(y \log(p) + (1 - y) \log(1 - p))$$

Predictive Accuracy Indicator

- **Lower Cross-Entropy**: Indicates predictions close to true labels.
- **Higher Cross-Entropy**: Signals predictions far from true labels.

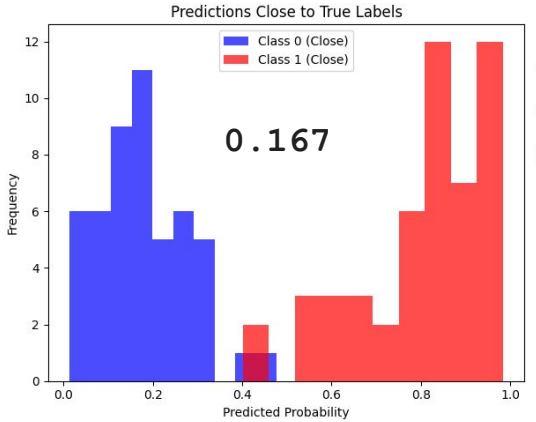
Cross-Entropy Visualization & Interpretation

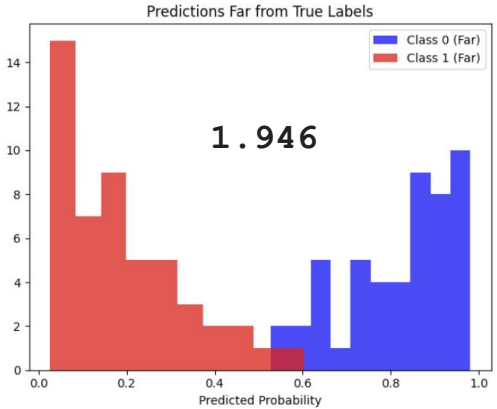
Scenario Analysis

- **Scenario 1**: Predictions near true labels ($H \approx 0.167$) → High accuracy.
- **Scenario 2**: Predictions diverge from true labels ($H \approx 1.946$) → Low accuracy.

Visualization Insights

- Histograms compare predicted probabilities for class 0 (Blue) and class 1 (Red) against true labels.
- **Close Predictions**: Probabilities align closely with actual class labels.
- **Distant Predictions**: Misalignment between predicted probabilities and true class labels.





Kullback-Leibler Divergence

1. **Concept:**
   - Kullback-Leibler (KL) Divergence: Quantifies the difference between two probability distributions, highlighting the extent to which one distribution deviates from a reference distribution.
2. **Mathematical Expression:**
   - Formula for KL Divergence:

$$D_{KL}(P||Q) = \sum_x P(x) \log \left(\frac{P(x)}{Q(x)} \right)$$

   - P and Q are the two probability distributions being compared, with P typically representing the true distribution and Q the approximation.

Kullback-Leibler Divergence

3. **Applications:**
   - Crucial for assessing model performance by quantifying the deviation of predicted distributions from actual data distributions.
   - Aids in selecting optimal models or parameters that minimize this divergence, enhancing predictive accuracy.
4. **Illustrative Example:**
   - Consider two simple probability distributions over a dataset with three outcomes: $P = \{0.2, 0.5, 0.3\}$ and $Q = \{0.3, 0.4, 0.3\}$.
   - KL Divergence Calculation:

$$D_{KL}(P||Q) = 0.2 \log\left(\frac{0.2}{0.3}\right) + 0.5 \log\left(\frac{0.5}{0.4}\right) + 0.3 \log\left(\frac{0.3}{0.3}\right)$$

   - This result quantifies the information loss when Q is used to approximate P, guiding decisions in model selection and evaluation.

CHAPTER 7
STATISTICAL LEARNING

LECTURE 3
DATA ENTROPY IN DATA SCIENCE

Summary

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Introduction to Statistical Learning

- **Objective of Statistical Learning**: To understand and model the relationship between inputs and outputs based on observed data.
- **Key Concepts**:
  - **Sampling Distribution Approximation**: Models like $g(x|\beta)$ use parameter vectors β to approximate distributions (e.g., Normal, Binomial, Exponential).
  - **Parameter Estimation**: Unknown parameters β are estimated from data, contrasting with non-parametric approaches where the distribution form is not assumed.
  - **Graphical Representation**: Histograms, density plots, and empirical CDFs visualize data distributions, aiding in understanding underlying patterns.

Data Collection and Exchangeability

- **Objective of Statistical Learning**: To understand and model the relationship between inputs and outputs based on observed data.
- **Key Concepts**:
  - **Sampling Distribution Approximation**: Models like $g(x|\beta)$ use parameter vectors β to approximate distributions (e.g., Normal, Binomial, Exponential).
  - **Parameter Estimation**: Unknown parameters β are estimated from data, contrasting with non-parametric approaches where the distribution form is not assumed.
  - **Graphical Representation**: Histograms, density plots, and empirical CDFs visualize data distributions, aiding in understanding underlying patterns.

Modeling Trade-offs and Mathematical Formulation

- **Model Complexity vs. Tractability**:
  - Simple Models: High interpretability, potentially low fidelity.
    - $\text{Tractability} \propto \frac{1}{\text{Complexity}}$
  - Complex Models: Lower interpretability, potentially high fidelity.
    - $\text{Fidelity} \propto \text{Complexity}$
- **Trade-off Visualization**:
  - A triangle, vertices representing simplicity, complexity, and generality.
    - $\text{Simplicity} + \text{Complexity} + \text{Generality} = \text{Constant}$
- **Application vs. Analysis Balance**:
  - Optimal Model Selection: $\min_{\text{model}} \left(\text{Bias}^2 + \text{Variance} + \text{Irreducible Error}\right)$
    - Balances bias (from simplicity) against variance (from complexity).

Mathematical Framework for Unsupervised Learning

- **Objective**: Learn f from data $\{x_1, \ldots, x_n\}$ without outputs.
  - f: Unknown probability density function (pdf).
- **Model Class Specification**:
  - $G_p = \{g(\cdot|\theta)|\theta \in \Theta\}$
  - $\Theta \subseteq \mathbb{R}^p$: Parameter space.
- **Risk Minimization**:
  - Goal: Identify $g^* \in G_p$ minimizing risk.
  - $g^* = \underset{g \in G_p}{\operatorname{argmin}}\, L(g, f)$
  - Risk $L(g, f)$: Typically quantified by Kullback-Leibler (KL) divergence, $D_{KL}(f||g)$.
- **Challenges**:
  - True f may not be in G_p, particularly as p grows.
  - Emphasizes the need for model selection criteria, such as Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC), to balance model complexity and fit.

Understanding Kullback-Leibler (KL) Divergence

- **Definition**:
  - A measure of how one probability distribution P diverges from a second, reference distribution Q.
- **Mathematical Expression**:
  - For discrete variables: $D_{KL}(P||Q) = \sum_x P(x) \log \left(\frac{P(x)}{Q(x)} \right)$
  - For continuous variables: $D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) \log \left(\frac{p(x)}{q(x)} \right) dx$
- **Properties**:
  - **Non-negative**: $D_{KL}(P||Q) \geq 0$ (with equality iff $P = Q$).
  - **Not Symmetric**: $D_{KL}(P||Q) \neq D_{KL}(Q||P)$.
- **Interpretation**:
  - Represents the "information lost" when Q is used to approximate P.
- **Application in Machine Learning**:
  - Used to measure the difference between the true data distribution and the model's distribution, guiding model optimization.

```python
import numpy as np
from scipy.stats import entropy

# Example discrete distributions
P = np.array([0.2, 0.5, 0.3])
Q = np.array([0.1, 0.4, 0.5])

# Compute KL divergence
D_KL_P_Q = entropy(P, Q, base=np.e)
print(f"KL Divergence D_KL(P || Q): {D_KL_P_Q}")

D_KL_Q_P = entropy(Q, P, base=np.e)
print(f"KL Divergence D_KL(Q || P): {D_KL_Q_P}")

```

```
KL Divergence D_KL(P || Q): 0.09695352463929671
KL Divergence D_KL(Q || P): 0.09684067330131696
```

END OF LECTURE

CHAPTER 7
STATISTICAL LEARNING

LECTURE 2
FOUNDATIONS OF PREDICTIVE MODELING IN DATA SCIENCE

Ordinary Least Squares (OLS) Regression & Projection Matrix Overview

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

MATH TIP: Matrix Pseudo-Inverse

Matrix Pseudo-Inverse

- **Definition**: For matrix A, its pseudo-inverse A^+ provides a generalized solution where A is not invertible or A is not square.
- **Formula**: $A^+ = (A^T A)^{-1} A^T$ for A of full column rank.

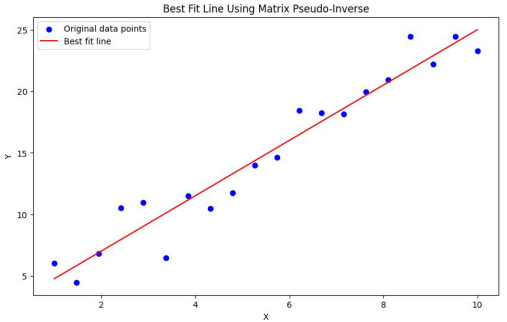
Best Fit Solution to $Ax = b$

- **Objective**: Find x that minimizes $||Ax - b||^2$, the squared error.
- **Solution**: $x_{best} = A^+ b$.
- **Application**: $A^+ b$ yields the least squares solution, offering the "best fit" for overdetermined systems.

Properties

- **Minimization**: $A^+ b$ minimizes the Euclidean norm of the error vector.
- **Generalization**: Applicable to any linear system, providing a solution even in the absence of an exact solution.

```python
import numpy as np
import matplotlib.pyplot as plt

# Generate a 2D matrix A (for simplicity, let's assume it represents 2D p
np.random.seed(0) # For reproducibility
x = np.linspace(1, 10, 20)
y = 2.5 * x + np.random.randn(20) * 2 # Linear relation with some noise

A = np.vstack([x, np.ones(len(x))]).T
b = y

# Calculate the pseudo-inverse of A
A_pinv = np.linalg.pinv(A)

# Find the best fit solution x_best
x_best = A_pinv @ b

# Plot the original data points and the best fit line
plt.figure(figsize=(10, 6))
plt.scatter(x, y, color='blue', label='Original data points')
plt.plot(x, x_best[0]*x + x_best[1], color='red', label='Best fit line')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Best Fit Line Using Matrix Pseudo-Inverse')
plt.legend()
plt.show()

x_best
```



Introduction to Ordinary Least Squares (OLS) Regression

Objective of OLS Regression

- **Purpose**: To find the best-fitting line through a set of data points by minimizing the sum of the squares of the vertical distances (residuals) of the points from the line.

Mathematical Formulation

- **Model Equation**: $y = X\beta + \epsilon$
  - y: Vector of observed values.
  - X: Design matrix containing the independent variables.
  - β: Vector of coefficients to be estimated.
  - ϵ: Vector of error terms.

Minimization Criterion

- **Sum of Squared Residuals (SSR)**: $SSR = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- **Objective**: Minimize SSR to find optimal β.

Minimization Criterion

- **Sum of Squared Residuals (SSR)**: $SSR = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- **Objective**: Minimize SSR to find optimal β.

Solving for Coefficients

- **Normal Equations**: $X^T X\beta = X^T y$
- **Solution**: $\beta = (X^T X)^{-1} X^T y$
  - Provides the estimates of β that minimize the SSR.

$$\hat{\beta} = \arg\min_{\beta} \|y - X\beta\|^2$$

PROOF. Ordinary Least Squares (OLS) Regression

Ordinary Least Squares (OLS): Overview

- **Objective**: Minimize the sum of squared residuals between observed and predicted values in a linear regression model.
- **Model**: $y = X\beta + \epsilon$, where y is the vector of observed values, X is the design matrix, β represents model parameters, and ϵ is the error term.

Derivation of Normal Equations

1. **Objective Function**: The goal of OLS is to find β that minimizes the sum of squared residuals (SSR), given by $SSR = (y - X\beta)^T(y - X\beta)$.
2. **Differentiation**: To minimize SSR, we take its derivative with respect to β and set it equal to zero. This yields:
   $\frac{\partial SSR}{\partial \beta} = \frac{\partial}{\partial \beta}[(y - X\beta)^T(y - X\beta)] = 0$
3. **Expanding and Simplifying**: The derivative simplifies to:
   $-2X^T(y - X\beta) = 0$
   Further simplifying, we get:
   $X^T y = X^T X\beta$
4. **Resulting in Normal Equations**: The final form is the set of equations known as the Normal Equations:
   $X^T X\beta = X^T y$

Minimization Criterion

- **Sum of Squared Residuals (SSR)**: $SSR = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- **Objective**: Minimize SSR to find optimal β.

Polynomial Regression with First Order Polynomial

- **Objective**: Model relationship between a dependent variable y and an independent variable x using a first-order polynomial (linear relationship).
- **Mathematical Form**: $y = \beta_0 + \beta_1 x + \epsilon$, where:
  - β_0 is the y-intercept,
  - β_1 is the slope,
  - ϵ represents error terms.

Design Matrix for First Order Polynomial

- For n observations, the design matrix X and vector y are represented as:
  - $$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$
  - X incorporates both the intercept and the linear term.

Projection Matrix in OLS Regression

- **Projection Matrix (P)**: $P = X(X^T X)^{-1} X^T$
- Projects y onto the space spanned by X, resulting in fitted values $\hat{y}$.

Example Setup

- Observations: $y = [2, 3, 5], x = [1, 2, 3]$
- Design Matrix X: $\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}, y = \begin{bmatrix} 2 \\ 3 \\ 5 \end{bmatrix}$

Calculating the Projection Matrix

1. Compute $X^T X$: $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix}$
2. Find $(X^T X)^{-1}$: $\begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix}^{-1} = \frac{1}{10} \begin{bmatrix} 14 & -6 \\ -6 & 3 \end{bmatrix}$
3. Calculate P: $P = X \left(\frac{1}{10} \begin{bmatrix} 14 & -6 \\ -6 & 3 \end{bmatrix} \right) X^T$

Fitted Values ($\hat{y}$)

- $\hat{y} = PXy$
- Substituting P and y into the formula gives us the fitted values, demonstrating how observations are projected onto the model's linear space.

Polynomial Regression with First Order Polynomial

Example Setup

- Observations: $y = [2, 3, 5], x = [1, 2, 3]$
- Design Matrix X: $\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}, y = \begin{bmatrix} 2 \\ 3 \\ 5 \end{bmatrix}$

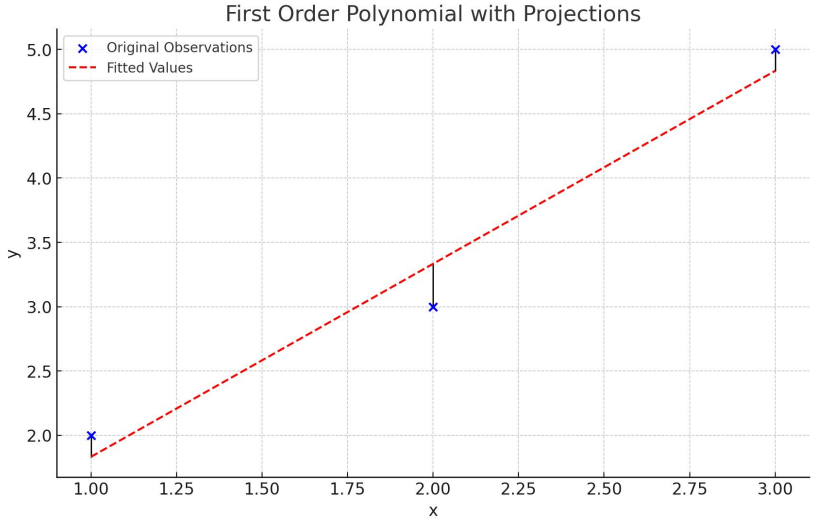
Calculating the Projection Matrix

1. Compute $X^T X$: $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix}$
2. Find $(X^T X)^{-1}$: $\begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix}^{-1} = \frac{1}{10} \begin{bmatrix} 14 & -6 \\ -6 & 3 \end{bmatrix}$
3. Calculate P: $P = X \left(\frac{1}{10} \begin{bmatrix} 14 & -6 \\ -6 & 3 \end{bmatrix} \right) X^T$

Fitted Values ($\hat{y}$)

- $\hat{y} = PXy$
- Substituting P and y into the formula gives us the fitted values, demonstrating how observations are projected onto the model's linear space.

$$P = \begin{bmatrix} 0.833 & 0.333 & -0.167 \\ 0.333 & 0.333 & 0.333 \\ -0.167 & 0.333 & 0.833 \end{bmatrix}$$



- **Blue dots**: Represent the original observations ($y = [2, 3, 5]$ corresponding to $x = [1, 2, 3]$).
- **Red dashed line**: Shows the fitted values ($\hat{y}$) calculated from the projection of y onto the linear space defined by the design matrix X. The fitted values are $\hat{y} = [1.833, 3.333, 4.833]$.

Polynomial Regression with First Order Polynomial

Example Setup

- Observations: $y = [2, 3, 5], x = [1, 2, 3]$
- Design Matrix X: $\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}, y = \begin{bmatrix} 2 \\ 3 \\ 5 \end{bmatrix}$

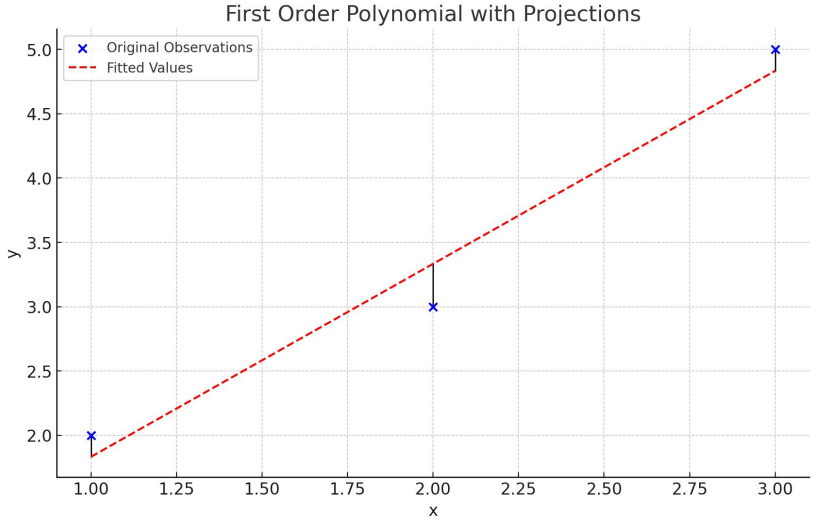
Calculating the Projection Matrix

1. Compute $X^T X$: $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix}$
2. Find $(X^T X)^{-1}$: $\begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix}^{-1} = \frac{1}{10} \begin{bmatrix} 14 & -6 \\ -6 & 3 \end{bmatrix}$
3. Calculate P: $P = X \left(\frac{1}{10} \begin{bmatrix} 14 & -6 \\ -6 & 3 \end{bmatrix} \right) X^T$

Fitted Values ($\hat{y}$)

- $\hat{y} = PXy$
- Substituting P and y into the formula gives us the fitted values, demonstrating how observations are projected onto the model's linear space.

$$P = \begin{bmatrix} 0.833 & 0.333 & -0.167 \\ 0.333 & 0.333 & 0.333 \\ -0.167 & 0.333 & 0.833 \end{bmatrix}$$



- **Blue dots**: Represent the original observations ($y = [2, 3, 5]$ corresponding to $x = [1, 2, 3]$).
- **Red dashed line**: Shows the fitted values ($\hat{y}$) calculated from the projection of y onto the linear space defined by the design matrix X. The fitted values are $\hat{y} = [1.833, 3.333, 4.833]$.

MGF In Summary

Definitions and Preliminaries:

- **Moment Generating Function (MGF)**: For a random variable X, the MGF is defined as $M_X(t) = E[e^{tX}]$, where $E[\cdot]$ denotes the expected value.
- **Properties of MGFs**:
  - The MGF of the sum of independent random variables is the product of their MGFs.
  - If X has a mean μ and variance σ^2, then the MGF of X can be approximated (for small t) as $M_X(t) \approx 1 + \mu t + \frac{\sigma^2 t^2}{2}$, using the Taylor series expansion of e^{tX} about $t = 0$.

CLT Theorem Statement

Theorem Statement:

Let $X_1, X_2, \ldots, X_n$ be a sequence of i.i.d. random variables with mean μ and variance σ^2. Define the sample mean $\bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$. Then, as $n \to \infty$, the standardized sample mean $\sqrt{n}(\bar{X}_n - \mu)$ converges in distribution to a standard normal distribution $N(0, \sigma^2)$.

CLT Proof using MGF (1)

MGF of a Single Observation: For each X_i, the MGF $M_{X_i}(t)$ exists and can be expanded as $1 + \mu t + \frac{\sigma^2 t^2}{2} + o(t^2)$, where $o(t^2)$ represents higher-order terms of t.

MGF of the Sum $\sum_{i=1}^{n} X_i$: Since the X_i are independent, the MGF of their sum is the product of their MGFs:

$$M_{\sum_{i=1}^{n} X_i}(t) = (M_{X_1}(t))^n = \left(1 + \mu t + \frac{\sigma^2 t^2}{2} + o(t^2)\right)^n$$

CLT Proof using MGF (2)

Standardized Sample Mean: We are interested in the MGF of $\sqrt{n}(\bar{X}_n - \mu)$. Let $Y_n = \sqrt{n}(\bar{X}_n - \mu)$, which is equivalent to $\frac{\sum_{i=1}^{n} X_i - n\mu}{\sqrt{n}}$. The MGF of Y_n is $M_{Y_n}(t) = E[e^{tY_n}]$.

. **Applying the MGF of the Sum**: The MGF of Y_n can be related to the MGF of the sum $\sum_{i=1}^{n} X_i$ by considering the transformation $t \to \frac{t}{\sqrt{n}}$ and centering by μ. Substituting and simplifying, we aim to show that:

$M_{Y_n}(t) \to e^{\frac{\sigma^2 t^2}{2}}$ as $n \to \infty$.