18-661: Introduction to ML for Engineers

HW 1 Review

Spring 2025

ECE – Carnegie Mellon University

In-Class Quiz 1

Question: When we set $\alpha = 1$ and $\beta = 1$ in the Beta distribution Beta(α, β), it reduces to which distribution?

Choices:

1. Uniform
2. Bernoulli
3. Gaussian
4. Exponential

In-Class Quiz 1

The Beta distribution is a continuous probability distribution defined on the interval [0, 1], making it ideal for modeling random variables that represent proportions or probabilities.

The probability density function (PDF) of the Beta distribution is:

$$f(x; \alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)} \quad \text{for } 0 \leq x \leq 1,$$

where $B(\alpha, \beta)$ is the Beta function.

When $\alpha = 1$ and $\beta = 1$, the PDF simplifies to:

$$f(x; 1, 1) = \frac{x^0(1-x)^0}{B(1, 1)} = \frac{1}{B(1, 1)}.$$

Since $B(1, 1) = 1$, we have:

$$f(x; 1, 1) = 1 \quad \text{for } 0 \leq x \leq 1.$$

This is the PDF of a Uniform distribution over the interval [0, 1].

Answer: 1. Uniform

In-Class Quiz 2

Question: For which of these prior distributions does the Maximum a Posteriori (MAP) estimation reduce to the Maximum Likelihood Estimation (MLE)?

Choices:

1. Uniform
2. Bernoulli
3. Gaussian
4. Exponential

In-Class Quiz 2

The MAP estimate is given by:

$$\theta_{\text{MAP}} = \arg\max_{\theta} \left[\log P(X \mid \theta) + \log P(\theta)\right],$$

where $P(X \mid \theta)$ is the likelihood and $P(\theta)$ is the prior distribution.

If the prior $P(\theta)$ is uniform, $\log P(\theta)$ is constant and does not affect the maximization. Therefore, the MAP estimate simplifies to:

$$\theta_{\text{MAP}} = \arg\max_{\theta} \log P(X \mid \theta),$$

which is exactly the MLE.

Answer: 1. Uniform

Linear Algebra: Key Concepts

Column Space (col(A)):

- The set of all linear combinations of the columns of A.
- Represents the subspace of $\mathbb{R}^m$ spanned by the columns of A.
- $b \in \text{col}(A)$ implies b can be expressed as Ax for some x.

Rank of a Matrix:

- The *rank* of A is the dimension of $\text{col}(A)$.
- Equal to the number of linearly independent columns in A.
- A full-rank matrix ($\text{rank}(A) = n$) means the columns of A are linearly independent.

- Let $B_k \in \mathbb{R}^{3\times 3}$, $k = 1, 2, 3$, where $B_k = B_k^\top$ (symmetric).
- Eigenvectors: $v_1 = [1, 0, 0]^\top$, $v_2 = [0, 1, 0]^\top$, $v_3 = [0, 0, 1]^\top$.
- Eigenvalues:

$$B_1 : \beta_{11} = 5,\ \beta_{12} = 2,\ \beta_{13} = -1,$$

$$B_2 : \beta_{21} = 3,\ \beta_{22} = -2,\ \beta_{23} = 4$$

Tasks:

1. Verify if B_1 is positive definite.
2. Determine if B_1 and B_2 commute, i.e., $B_1 B_2 = B_2 B_1$.

- Let $B_k \in \mathbb{R}^{3\times 3}$, $k = 1, 2, 3$, where $B_k = B_k^\top$ (symmetric).
- Eigenvectors: $v_1 = [1, 0, 0]^\top$, $v_2 = [0, 1, 0]^\top$, $v_3 = [0, 0, 1]^\top$.
- Eigenvalues:

$$B_1 : \beta_{11} = 5,\ \beta_{12} = 2,\ \beta_{13} = -1,$$
$$B_2 : \beta_{21} = 3,\ \beta_{22} = -2,\ \beta_{23} = 4$$

Verifying if B_1 is Positive Definite:

- A symmetric matrix is positive definite if all its eigenvalues are positive.
- Given $B_1 = \mathrm{diag}(5, 2, -1)$, its eigenvalues are $5, 2, -1$.
- Since B_1 has a negative eigenvalue (-1), it is not positive definite.

Linear Algebra: Examples

- Let $B_k \in \mathbb{R}^{3\times 3}$, $k = 1, 2, 3$, where $B_k = B_k^\top$ (symmetric).
- Eigenvectors: $v_1 = [1, 0, 0]^\top$, $v_2 = [0, 1, 0]^\top$, $v_3 = [0, 0, 1]^\top$.
- Eigenvalues:

$$B_1 : \beta_{11} = 5,\ \beta_{12} = 2,\ \beta_{13} = -1,$$

$$B_2 : \beta_{21} = 3,\ \beta_{22} = -2,\ \beta_{23} = 4$$

Determining if B_1 and B_2 Commute:

- Compute $B_1 B_2$:

$$B_1 B_2 = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 3 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 4 \end{bmatrix} = \begin{bmatrix} 15 & 0 & 0 \\ 0 & -4 & 0 \\ 0 & 0 & -4 \end{bmatrix}.$$

- Compute $B_2 B_1$:

$$B_2 B_1 = \begin{bmatrix} 3 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} 5 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 15 & 0 & 0 \\ 0 & -4 & 0 \\ 0 & 0 & -4 \end{bmatrix}.$$

- Since $B_1 B_2 = B_2 B_1$, B_1 and B_2 commute.

Matrix Calculus: Examples

- Find the derivative of:

$$f(X) = c^\top X X^\top d, \quad X \in \mathbb{R}^{m \times n},\ c, d \in \mathbb{R}^m.$$

Matrix Calculus: Examples

- Find the derivative of:

$$f(X) = c^\top X X^\top d, \quad X \in \mathbb{R}^{m\times n},\ c, d \in \mathbb{R}^m.$$

- The scalar function $f(X)$ can be written as:

$$f(X) = \mathrm{tr}(c^\top X X^\top d) = \mathrm{tr}(dc^\top X X^\top) = \mathrm{tr}(X^\top dc^\top X).$$

- This uses the cyclic property of the trace: $\mathrm{tr}(ABC) = \mathrm{tr}(CAB)$.
- For $\mathrm{tr}(X^\top AX)$, the derivative is:

$$\nabla_X \mathrm{tr}(X^\top AX) = AX + A^\top X.$$

- Here, $A = dc^\top$, so:

$$\nabla_X \mathrm{tr}(dc^\top X X^\top) = (dc^\top)X + (dc^\top)^\top X.$$

- Simplify the transpose and substituting back:

$$\nabla_X f(X) = dc^\top X + cd^\top X.$$

MLE: Examples

- Dataset: $\mathcal{D} = \{x_1, \ldots, x_n\}$ i.i.d. from $\mathcal{N}(\mu, \sigma^2)$.

MLE: Examples

- Dataset: $\mathcal{D} = \{x_1, \ldots, x_n\}$ i.i.d. from $\mathcal{N}(\mu, \sigma^2)$.
- Likelihood:

$$L(\mu|\mathcal{D}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right).$$

- Log-likelihood:

$$\ell(\mu|\mathcal{D}) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2.$$

- Solve:

$$\frac{\partial \ell(\mu|\mathcal{D})}{\partial \mu} = 0 \implies \hat{\mu}_{\mathsf{MLE}} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

MAP: Examples

- Dataset: $\mathcal{D} = \{x_1, \ldots, x_n\}$ i.i.d. from $\mathcal{N}(\mu, \sigma^2)$.
- Prior: $\mu \sim \mathcal{N}(\mu_0, \tau^2)$.

MAP: Examples

- Dataset: $\mathcal{D} = \{x_1, \ldots, x_n\}$ i.i.d. from $\mathcal{N}(\mu, \sigma^2)$.
- Prior: $\mu \sim \mathcal{N}(\mu_0, \tau^2)$.
- Posterior:
$$p(\mu|\mathcal{D}) \propto L(\mu|\mathcal{D}) \cdot p(\mu).$$
- Log-posterior:
$$\log p(\mu|\mathcal{D}) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2$$
$$-\frac{n}{2}\log(2\pi\tau^2) - \frac{1}{2\tau^2}(\mu - \mu_0)^2.$$
- Solve:
$$\hat{\mu}_{\mathrm{MAP}} = \frac{\frac{1}{\sigma^2}\sum_{i=1}^{n} x_i + \frac{\mu_0}{\tau^2}}{\frac{n}{\sigma^2} + \frac{1}{\tau^2}}.$$

Big-O Notation: Definition and Purpose

What is Big-O Notation?

- Big-O notation describes the upper bound of an algorithm's time or space complexity.
- It measures the worst-case growth rate of the runtime or memory usage as a function of the input size (n).

Key Idea:

- Ignore lower-order terms and constants to focus on dominant behavior.
- Example: For $T(n) = 5n^2 + 3n + 2$,

$$T(n) = O(n^2),$$

  because the quadratic term n^2 dominates for large n.

Big-O Notation: Formal Definition

Formal Definition:

- A function $f(n)$ is $O(g(n))$ if there exist constants $c > 0$ and $n_0 > 0$ such that:

$$f(n) \leq c \cdot g(n) \quad \text{for all } n \geq n_0.$$

Example:

- Suppose $f(n) = 3n + 2$ and $g(n) = n$.
- Find constants c and n_0 such that $f(n) \leq c \cdot g(n)$:

$$3n + 2 \leq 4n \quad \text{for } n \geq 2 \qquad \implies c = 4, n_0 = 2.$$

- Conclusion: $f(n) = O(n)$.

Big-O Notation: Common Complexities

Common Time Complexities:

- Constant Time: $O(1)$
- Logarithmic Time: $O(\log n)$
- Linear Time: $O(n)$
- Linearithmic Time: $O(n \log n)$
- Quadratic Time: $O(n^2)$
- Exponential Time: $O(2^n)$

Graphical Intuition:

$$\text{As } n \to \infty, \quad O(n) \ll O(n^2) \ll O(2^n).$$

Big-O in Online Updates

What are Online Updates?

- Online updates process data sequentially, updating the model incrementally with each new data point.
- Example: Stochastic Gradient Descent (SGD) processes one sample at a time.

Analyzing Complexity:

- For each data point x_i:

$$w^{(t+1)} = w^{(t)} - \eta \nabla \ell(x_i, w^{(t)}),$$

  where $\nabla \ell$ is the gradient of the loss.
- Cost of one update: $O(d)$, where d is the dimensionality of x_i.
- Total cost for n updates: $O(nd)$.

Big-O Example: Online Learning

Example: Linear Regression with SGD

- Model: $y = Xw + \epsilon$.
- Loss: Mean squared error (MSE):

$$\ell(w) = \frac{1}{2}\|Xw - y\|^2.$$

- Gradient for one sample:

$$\nabla\ell(w) = x_i(x_i^\top w - y_i).$$

- Complexity for one update: $O(d)$.
- Total cost for n samples: $O(nd)$.

Why Online Updates Are Efficient:

- Incremental updates reduce memory overhead.
- Avoids computing over the entire dataset at once.

Python: What is Python?

- Python is a high-level, interpreted programming language known for its readability and simplicity.
- It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.
- Widely used in data analysis, machine learning, web development, automation, and more.

Python: Basic Python Syntax

- **Variables and Data Types:**
  # Integer count = 10
  # Floating-point number price = 19.99
  # String name = "Alice"
  # Boolean is_valid = True
- **Printing Output:**
  print("Hello, World!")
- **Comments:**
  # This is a single-line comment
  ' ' '
  This is a
  multi-line comment
  ' ' '

Python: Introduction to NumPy

- NumPy is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, Fourier transform, and matrices. :contentReference[oaicite:0]index=0
- NumPy provides a high-performance multidimensional array object and tools for working with these arrays.
- To use NumPy, it must first be installed and imported:

Python: NumPy Arrays

- NumPy's main object is the homogeneous multidimensional array.
- Creating a NumPy array:
```
import numpy as np
# Creating a 1D array
arr = np.array([1, 2, 3, 4, 5])
# Creating a 2D array
arr_2d = np.array([[1, 2, 3], [4, 5, 6]])
```
- Accessing elements:
```
# Accessing the first element
print(arr[0]) # Output: 1
# Accessing an element in 2D array
print(arr_2d[0, 1]) # Output: 2
```

Python: Differences Between Python Lists and NumPy Arrays

- **Data Types:**
  - **Python Lists:** Can store elements of different data types (e.g., integers, strings, objects).
  - **NumPy Arrays:** Require all elements to be of the same data type, enabling efficient storage and operations.
- **Performance:**
  - **Python Lists:** Less efficient for numerical computations due to the lack of optimized operations.
  - **NumPy Arrays:** Implemented in C, providing faster computation speeds and better memory management for numerical operations.
- **Functionality:**
  - **Python Lists:** General-purpose; limited functionality for mathematical operations.
  - **NumPy Arrays:** Offer extensive mathematical and scientific computing functionality, including element-wise operations and broadcasting.

Python: Differences Between Python Lists and NumPy Arrays

- **Memory Usage:**
  - **Python Lists:** Store references to objects, which can lead to higher memory consumption, especially with large datasets.
  - **NumPy Arrays:** Store data in contiguous memory blocks, resulting in more efficient memory usage.
- **Mutability:**
  - **Python Lists:** Elements can be changed, added, or removed; lists can grow or shrink dynamically.
  - **NumPy Arrays:** Elements can be changed, but the size of the array is fixed upon creation; resizing requires creating a new array.

Python: Introduction to Matplotlib

- Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.
- Commonly used for plotting data to visualize relationships and trends.
- Basic usage example:
  import matplotlib.pyplot as plt
  # Data
  x = [1, 2, 3, 4, 5]
  y = [2, 4, 6, 8, 10]
  # Create a plot
  plt.plot(x, y)
  # Add title and labels
  plt.title('Simple Linear Plot')
  plt.xlabel('X-axis')
  plt.ylabel('Y-axis')
  # Show the plot
  plt.show()