Estimation, Mixture Models and Random Processes with Python Simulations

Applied Stochastic Processes: HW 4

Due Date: November 10, 2024

Objectives

This assignment is designed to help students develop a deep understanding of probability and random variables. Students will enhance their problem-solving skills and gain practical experience through Python-based simulations.

Policy

- You can discuss HW problems with other students, but the work you submit must be written in your own words and not copied from anywhere else. This includes codes.
- However, do write down (at the top of the first page of your HW solutions) the names of all the people with whom you discussed this HW assignment.
- You may decide to write out your solution with pen and paper and use a scanning app to turn in a PDF submission or may choose to type out your solution with LATEX. We strongly encourage the latter.

Expected Topics Covered

1. Methods of Estimation
2. Mixture Models and EM Algorithm
3. Random Processes and Markov Chains

Submission Guidelines

- Submit your code solutions as a Jupyter notebook file (.ipynb) or as Python scripts (.py). You can also convert them to pdf and attach them to your final submission document. Be sure to indicate which code belongs to what question.
- For those using **LaTeX**, you can paste your code by importing the python environment `pythonhiglights`.

```
\usepackage{pythonhiglights}

\begin{python}
# import necessary libraries
import math
import random

# example arithmetic evaluation
a, b = 2, 3
c = a + b
print(f"The sum of the numbers {a} and {b} is {c}")
\end{python}
```

- Ensure all code is well-documented and includes comments explaining each step.
- Provide a brief report summarizing your findings and the results of your simulations.

Question 1: MoM, MLE, Bias and Consistency (20 points)

Consider a normal distribution defined by the probability density function (PDF):

$$f(y;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu)^2}{2\sigma^2}}, \quad -\infty < y < \infty,$$

where μ is the mean and σ^2 is the variance. Given a random sample $Y = \{y_1, y_2, \ldots, y_n\}$ drawn from this normal distribution, perform the following tasks:

(a) **(5 points)** Use the method of moments to derive the estimators for μ and σ^2.

(b) **(5 points)** Derive the Maximum Likelihood Estimators (MLE) for μ and σ^2.

(c) **(3 points)** Calculate the bias of the MoM estimators $\hat{\mu}_{\text{MoM}}$ and $\hat{\sigma}^2_{\text{MoM}}$.

(d) **(3 points)** Calculate the bias of the MLE estimators $\hat{\mu}_{\text{MLE}}$ and $\hat{\sigma}^2_{\text{MLE}}$.

(e) **(4 points)** Show that both the MoM and MLE estimators are consistent, meaning that as $n \to \infty$, $\hat{\mu}_{\text{MoM}} \to \mu$, $\hat{\sigma}^2_{\text{MoM}} \to \sigma^2$, $\hat{\mu}_{\text{MLE}} \to \mu$, and $\hat{\sigma}^2_{\text{MLE}} \to \sigma^2$ in probability.

Question 2: Spam-Ham Detection Using MLE and MAP (30 points)

In digital communication, distinguishing spam from ham (non-spam) is crucial for email security. Statistical techniques such as Maximum Likelihood Estimation (MLE) and Maximum A Posteriori (MAP) estimation are effective for classification. This section aims to build a spam-ham classifier using both MLE and MAP methods.

Part 1: MLE/MAP on Toy Dataset (10 points)

You are provided with a mini dataset containing six SMS messages labeled as either spam or ham. A single feature, "offer," indicates the presence (1) or absence (0) of the word "offer" in each message.

Message ID	Message Content	"Offer" (X)	Class (Y)
1	"Special offer now!"	1	1 (Spam)
2	"Meeting at 10 AM"	0	0 (Ham)
3	"Claim your offer"	1	1 (Spam)
4	"Lunch tomorrow?"	0	0 (Ham)
5	"Free offer available!"	1	1 (Spam)
6	"Hello, how are you?"	0	0 (Ham)

Table 1: Mini Dataset for Spam-Ham Detection

Calculate the following MLE estimates:

(a) **(1 point)** $\pi = P(Y = 1)$: Probability that a message is spam.

(b) **(1 point)** $\theta_{\text{spam}} = P(X = 1 \mid Y = 1)$: Probability that "offer" appears in a spam message.

(c) **(1 point)** $\theta_{\text{ham}} = P(X = 1 \mid Y = 0)$: Probability that "offer" appears in a ham message.

(d) **(2 points)** Derive the likelihood function and maximize it to find the parameter estimates.

Assume Beta priors:

(e) **(2 points)** $\pi \sim \text{Beta}(2, 2)$

(f) **(2 points)** $\theta_{\text{spam}} \sim \text{Beta}(2, 1)$

(g) **(2 points)** $\theta_{\text{ham}} \sim \text{Beta}(1, 2)$

Calculate the MAP estimates for π, θ_{spam}, and θ_{ham} using prior information.

Part 2: Practical Implications (4 points)

Discuss the following:

(a) **(1 point)** How do different prior choices affect MAP estimates?

(b) **(1 point)** Why might MLE overfit with small datasets?

(c) **(1 point)** In what scenarios would MLE or MAP perform better?

(d) **(1 point)** What is the bias-variance trade-off between MLE and MAP?

Part 3: Real-World Implementation (10 points)

In this exercise, you will classify messages as either "spam" or "ham" (not spam) using a Naive Bayes classifier. You will implement two different estimation methods. Use the "SMS Spam Collection" dataset, available at this link, to implement a spam-ham detection classifier using:

- **Maximum Likelihood Estimation (MLE)**: Estimates the parameters based solely on the training data without prior beliefs about the parameters.
- **Maximum A Posteriori (MAP)**: Incorporates prior beliefs about the parameters into the estimation, using Laplace smoothing to handle zero probabilities.

"

Tasks

Step 1: Data Loading and Preprocessing

1. **Load the Dataset**:
   - Download the SMS Spam Collection dataset and load it into your environment using pandas.
   - Ensure the dataset is read correctly, with columns labeled "label" for spam/ham and "message" for the text content.
2. **Preprocess the Text Messages**:
   - Convert all text to lowercase.
   - Remove punctuation and special characters.
   - Tokenize the messages (split the messages into words).

   Example Preprocessing Steps:
   - Lowercasing: Convert "Hello!" to "hello".
   - Remove punctuation: Convert "This is spam!!!" to "this is spam".
3. **Split the Dataset**:
   - Divide the dataset into training and test sets (e.g., 80% training, 20% testing).
   - Ensure that both sets maintain the same class distribution.

Step 2: Implement Maximum Likelihood Estimator (MLE)

1. **Calculate Probabilities**:
   - For each class (spam and ham), calculate the probability of each word appearing in that class based on the training data.
2. **Implement Prediction Function**:
   - Create a function to classify messages using the calculated probabilities and the prior probabilities of each class.
3. **Evaluate the Classifier**:
   - Use metrics such as accuracy, precision, recall, and F1 score to evaluate the performance of the MLE classifier on the test set.

Step 3: Implement Maximum A Posteriori (MAP)

1. **Implement MAP Estimator**:
   - Calculate the same probabilities as in MLE but include Laplace smoothing to avoid zero probabilities.
2. **Implement Prediction Function**:
   - Create a prediction function similar to MLE but using the MAP probabilities.
3. **Evaluate the Classifier**:
   - Again, use accuracy, precision, recall, and F1 score to evaluate the MAP classifier's performance on the test set.

Step 4: Compare Results

1. **Performance Comparison**:
   - Create a comparison table that summarizes the accuracy, precision, recall, and F1 score for both classifiers.
2. **Discussion**:
   - Reflect on the differences in performance:
     - How did incorporating prior knowledge in MAP affect the predictions?
     - Were there any significant changes in the classification of messages between MLE and MAP?
     - What factors might account for any differences in the performance metrics?

Step 5: Vary the Prior (MAP)

1. **Experiment with Different Alpha Values**:
   - Run the MAP classifier with varying values of the Laplace smoothing parameter (alpha) such as 0.1, 0.5, 1, and 5.
   - Observe how these variations affect the results and the evaluation metrics.
2. **Discussion of Findings**:
   - Summarize your observations regarding the impact of varying the prior on the classification performance.

Deliverables

Prepare a Jupyter Notebook containing the following:

- Data loading and preprocessing steps.
- MLE and MAP classifier implementations with detailed comments.
- Evaluation metrics for both classifiers.
- A comparison table summarizing the performance metrics.
- A discussion section reflecting on your observations and insights gained from the exercise.

Reflection

Conclude the exercise by reflecting on the differences between MLE and MAP:

- Discuss scenarios where one might prefer MLE over MAP and vice versa.
- Reflect on the impact of incorporating prior information into the estimation process.
- Consider how understanding these differences might be beneficial in real-world applications.

Question 3: Blind Estimation of a Corrupted Variable (20 points)

In signal processing, estimating a hidden signal corrupted by noise is a common challenge. This problem focuses on estimating a random variable X, which is corrupted by Gaussian noise N. The observed variable Y is given by:

$$Y = X + N,$$

where $N \sim N(0, \sigma^2)$ is Gaussian noise with zero mean and variance σ^2, and is uncorrelated with X.

Part 1: Analytical Derivations (10 points)

(a) **(2 points)** Derive the mean and variance of Y.

(b) **(3 points)** Derive the Best Linear Estimator of X

(c) **(2 points)** The orthogonality principle states that the error $\epsilon = X - \hat{X}$ must be uncorrelated with Y, i.e.:

$$E[(X - \hat{X})Y] = 0.$$

Substitute $\hat{X}$ into the expression and derive equations for a and b that minimize the Mean Squared Error (MSE).

(d) **(1 point)** Define bias as:

$$\text{Bias}(\hat{X}) = E[\hat{X}] - E[X].$$

Show that the estimator $\hat{X}$ derived above is unbiased.

(e) **2 points** Define the Mean Squared Error (MSE) as:

$$\text{MSE}(\hat{X}) = E[(X - \hat{X})^2].$$

Using the derived values of a and b, calculate the MSE and verify that it is minimized for the derived linear estimator.

Part 2: Practical Implementation with the California Housing Dataset (10 points)

Dataset Overview

The California Housing Dataset is a widely used dataset that contains information about various attributes of houses in California, including median house values, median income, housing age, and more. We will focus on estimating the original value of a key variable that has been corrupted by artificial noise.

Step-by-Step Implementation

Load the Dataset:

- Use the `sklearn.datasets` module to load the California Housing Dataset.

Introduce Noise:

- Select a variable, such as the "Average Number of Rooms" (AveRooms), and add Gaussian noise to simulate corruption.
- Use a Gaussian noise distribution with mean $\mu = 0$ and standard deviation $\sigma = 0.5$.
- The observed variable Y can be represented as:

$$Y = X + N, \quad \text{where } N \sim N(0, 0.5^2)$$

Downsample the Data:

- After splitting the dataset, randomly select 200 points from the test set to reduce clutter in visualizations and focus on key trends.

Train-Test Split:

- Split the data into training and testing sets using an 80-20 split ratio. This means 80% of the data will be used for training and 20% for testing.

Apply the Best Linear Estimator:

- Implement the best linear estimator (e.g., linear regression) to estimate the original values of the corrupted variable.

Evaluate the Estimator:

- Calculate the bias and Mean Squared Error (MSE) of the estimator. Analyze how close the estimated values are to the original values.

Visualize the Results:

- Plot the original, noise-corrupted, and estimated values of the selected variable. The plots should clearly distinguish between these values to facilitate comparison.

Discussion

In this implementation, consider the following questions to guide your discussion:

- **Bias and MSE:** What do the calculated bias and Mean Squared Error (MSE) reveal about the accuracy of our estimates? How might these metrics inform our understanding of model performance?
- **Impact of Noise:** In what ways does the introduction of Gaussian noise reflect real-world measurement errors? How do the estimates change as a result of this noise, and what implications does this have for data integrity?
- **Model Limitations:** What are the limitations of using linear regression in this context? Are there scenarios where a more complex model might provide better estimates? What factors should be considered when choosing a modeling approach?

Question 4: Mixture Models and the EM Algorithm (30 points)

1. Consider a dataset that is generated from a mixture of two Gaussian distributions. The first component has mean $\mu_1 = 0$ and variance $\sigma_1^2 = 1$, and the second component has mean $\mu_2 = 5$ and variance $\sigma_2^2 = 2$. The mixing proportions are $\pi_1 = 0.3$ and $\pi_2 = 0.7$.
   (a) **(2 points)** Write down the probability density function of the mixture model.
   (b) **(4 points** Derive the Expectation-Maximization (EM) algorithm steps for estimating the parameters of this mixture model.
   (c) **(6 points)** Generate a synthetic dataset of 1000 samples from this mixture model using Python and implement the EM algorithm in Python and estimate the parameters of the mixture model from the synthetic dataset.
   (d) **(2 points)**(2 points) Compare the estimated parameters with the true parameters and discuss the results.
2. **(6 points)** Consider a set of data points $x = \{1, 2, 3, 6, 7, 8\}$ which we assume come from a mixture of two Gaussian distributions. Use the Expectation-Maximization (EM) algorithm to fit a Gaussian Mixture Model (GMM) to this data. Start with the following initial parameters:
   - Means: $\mu_1 = 2$, $\mu_2 = 7$
   - Variances: $\sigma_1^2 = 1$, $\sigma_2^2 = 1$
   - Mixing coefficients: $\pi_1 = 0.5$, $\pi_2 = 0.5$

Perform the EM algorithm manually for three iterations and report the updated parameters μ_1, μ_2, σ_1^2, σ_2^2, π_1, and π_2 after each iteration. Use the following Gaussian probability density function for calculations:

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

3. **(10 points)** The Iris dataset contains 150 samples of iris flowers, each with four features: sepal length, sepal width, petal length, and petal width. The dataset also includes the species of the iris flowers, but we'll ignore this information for clustering.

   Use the Expectation-Maximization (EM) algorithm to fit a Gaussian Mixture Model (GMM) to this data. Assume the data comes from a mixture of three Gaussian distributions. Perform the following steps:

   (a) Preprocess the data by standardizing each feature.
   (b) Initialize the parameters of the GMM (means, variances, and mixing coefficients) randomly.
   (c) Implement the EM algorithm to fit the GMM to the data.
   (d) Perform the EM algorithm for a maximum of 100 iterations or until convergence.
   (e) Report the final parameters (means, variances, and mixing coefficients).
   (f) Visualize the clustering results on the first two principal components of the data.

Question 5: Random Processes and Markov Chains (40 points)

Problem 1: Simple Random Walk Process (8 points)

Imagine a scenario where a delivery robot moves along a straight path, starting from the origin (0 meters). At each time step, the robot has an equal chance of moving one meter to the left (backwards) or one meter to the right (forwards). This model can help us understand how the robot's position changes over time, which can be useful in logistics and supply chain management. Let X_n denote the position of the robot at time n.

(a) **(2 points)** Derive the first-order probability mass function for the position of the robot at time n, i.e., $P(X_n = x)$, where x is an integer representing the robot's position.

(b) **(2 points)** Calculate the probability that the robot, starting at the origin $X_0 = 0$, is located at $X_4 = -2$ after 4 steps. ***Hint:*** Use the binomial distribution, considering the number of steps to the left and right.

(c) **(2 points)** Derive the expressions for the mean $E[X_n]$ and variance $\text{Var}(X_n)$ of the random walk at any time $n \in \mathbb{N}$.

(d) **(2 points)** Discuss whether the mean and variance depend on the initial position X_0.

Problem 2: Wide Sense Stationarity (WSS) (8 points)

Consider a discrete-time process $\{X_n\}$ with mean μ and autocovariance function $\gamma(k)$, where k is the lag.

(a) **(2 points)** Derive the mathematical conditions that must be satisfied for the process $\{X_n\}$ to be classified as Wide Sense Stationary (WSS). Clearly define what it means for the mean and autocovariance function to be time-invariant.

(b) **(2 points)** Let $X_n = aX_{n-1} + W_n$, where W_n is white noise with mean 0 and variance σ^2, and $|a| < 1$. Show that the process $\{X_n\}$ is WSS by deriving its mean, variance, and autocovariance function, ensuring that the conditions of WSS are satisfied.

(c) **(2 points)** Discuss how the parameter a influences the behavior of the process, particularly in terms of its stability and variance over time.

(d) **(2 points)** Consider the case where $|a| \geq 1$ and explain why the process may no longer be WSS under such conditions.

Problem 3: Gambler's Ruin Problem as a Markov Chain (24 points)

Consider a gambling game where, at each turn, a gambler either wins 1 dollar with probability 0.4 or loses 1 dollar with probability 0.6. The gambler starts with an initial wealth of 1 dollar and stops playing when the total wealth reaches $N = 5$ dollars or falls to 0 dollars (ruin). Let X_n denote the wealth of the gambler after the n-th play.

(a) **(2 points)** Show that the process X_n is a Markov chain, explaining how it satisfies the Markov property

(b) **(4 points)** Construct the transition probability matrix for the Markov chain, ensuring that it captures both the winning and losing probabilities. Explicitly show the entries corresponding to the absorbing states and transient states.

(c) **(5 points)** Derive the probability that the gambler reaches the absorbing state at $N = 5$ dollars before reaching 0 dollars (ruin).

(d) **(3 points)** Discuss how the probabilities change as p varies, particularly focusing on the expected duration of the game and the likelihood of reaching 5 dollars versus 0 dollars.

(e) **(10 points)** Simulate a gambling game where you either win or lose money based on specific probabilities based on the above parameters.

Simulation Steps

1. **Initialize Simulation:**
   - Define the number of trials (`num_trials = 10`) to run, starting wealth for the player, probability of winning each round, and the target wealth to reach.
2. **Conduct Trials:**
   - For each trial:
     * Start with the initial wealth.
     * Track the wealth changes throughout the game.
     * Simulate rounds of play:
       · Continue playing until either the target wealth is reached or the player loses all their money.
       · In each round, determine the outcome based on the defined probability, updating the wealth accordingly.
3. **Store Results:**
   - Keep a record of the wealth progression for each trial.
4. **Visualize Outcomes:**
   - Create plots to illustrate the wealth changes over the course of several trials.
   - Mark significant states, such as the target wealth and the point of ruin.
5. **Analyze Results:**
   - Calculate the number of times the player wins or loses across all trials.
   - Determine the overall winning rate and the average number of rounds played per trial.
   - Present these statistics for review.

Discussion Points

- Reflect on how the probabilities affect the game's dynamics.
- Consider the implications of reaching either absorbing state and the significance of randomness in this process.
- Discuss how this simulation relates to concepts of Markov chains and random walks.