Dsl 253

Shashank yadav 12342010

QUESTION 1

Introduction
In this study, we explore the probabilities associated with a bivariate normal distribution. Given two normally distributed random variables, X and Y, we determine specific probabilities and conditional probabilities using the properties of the bivariate normal distribution.

Data
The given parameters for the bivariate normal distribution are:

- **Mean values:**
  - μX=3\mu_X = 3
  - μY=1\mu_Y = 1
- **Variances:**
  - σX2=16\sigma_X^2 = 16 (hence, standard deviation σX=4\sigma_X = 4)
  - σY2=25\sigma_Y^2 = 25 (hence, standard deviation σY=5\sigma_Y = 5)
- **Correlation coefficient:**
  - ρXY=35\rho_{XY} = \frac{3}{5}

Methodology
We compute the required probabilities using standard normal and conditional normal distributions:

1. **Computing** P(3<Y<8)P(3<Y<8):

   - Using the standard normal transformation:
     Z=Y−μYσYZ = \frac{Y - \mu_Y}{\sigma_Y}
   - We determine the cumulative probabilities and subtract them to obtain the required probability.
2. **Computing** P(3<Y<8∣X=7)P(3<Y<8 \mid X=7):

   - Given X=7X=7, the conditional distribution of Y follows:
     Y∣X~N(μY+ρXYσYσX(X−μX),σY1−ρXY2)Y \mid X \sim N\left(\mu_Y + \frac{\rho_{XY} \sigma_Y}{\sigma_X} (X - \mu_X), \sigma_Y \sqrt{1 - \rho_{XY}^2} \right)
   - We use this conditional mean and standard deviation to compute the required probability.
3. **Computing** P(−3<X<3)P(-3<X<3):

- Using the transformation:
  Z=X−μXσXZ = \frac{X - \mu_X}{\sigma_X}
- The probability is determined using cumulative distribution functions.

Results & Discussion

- The probability in part (a) represents the proportion of the population within a specific range of Y.
- The conditional probabilities in parts (b) and (d) illustrate how knowledge of one variable affects our expectations of the other.
- The effect of correlation is reflected in the changes to the conditional means and standard deviations.

Conclusion

This study demonstrates probability computation for a bivariate normal distribution using transformation techniques. Future work could explore applications in higher-dimensional normal distributions or real-world datasets.

```
import scipy.stats as stats
import numpy as np
import pandas as pd

x_mean, y_mean = 3, 1
sigma2_x, sigma2_y = 16, 25
correlation = 3/5

sigma_x = np.sqrt(sigma2_x)  # finding std deviation
sigma_y = np.sqrt(sigma2_y)

p_a = stats.norm.cdf(8, loc=y_mean, scale=sigma_y) - stats.norm.cdf(3, loc=y_mean, scale=sigma_y)  # part a just substracting the cdf (8)-(3)

mu_y_given_x = y_mean + correlation * (sigma_y / sigma_x) * (7 - x_mean)
sigma_y_given_x = sigma_y * np.sqrt(1 - correlation**2)
p_b = stats.norm.cdf(8, loc=mu_y_given_x, scale=sigma_y_given_x) - stats.norm.cdf(3, loc=mu_y_given_x, scale=sigma_y_given_x) # cacluclating the cdf usir

p_c = stats.norm.cdf(3, loc=x_mean, scale=sigma_x) - stats.norm.cdf(-3, loc=x_mean, scale=sigma_x)  # part c just substracting the cdf (3)-(-3)

mu_x_given_y = x_mean + correlation * (sigma_x / sigma_y) * (-4 - y_mean)
sigma_x_given_y = sigma_x * np.sqrt(1 - correlation**2)
p_d = stats.norm.cdf(3, loc=mu_x_given_y, scale=sigma_x_given_y) - stats.norm.cdf(-3, loc=mu_x_given_y, scale=sigma_x_given_y) # cacluclating the cdf usi


print(f"(a) P(3 < Y < 8) = {p_a:.4f}")
print(f"(b) P(3 < Y < 8|X = 7) = {p_b:.4f}")
print(f"(c) P(-3 < X < 3) = {p_c:.4f}")
print(f"(d) P(-3 < X < 3|Y = -4) = {p_d:.4f}")
```

```
(a) P(3 < Y < 8) = 0.2638
(b) P(3 < Y < 8|X = 7) = 0.4401
(c) P(-3 < X < 3) = 0.4332
(d) P(-3 < X < 3|Y = -4) = 0.6431
```

–

QUESTION 2

Introduction

This study involves generating samples from a multinomial random variable that follows a multivariate normal distribution and analyzing their transformation into a chi-square distributed variable. Our key objectives include:

- Generating P samples from a multivariate normal distribution.
- Computing a transformed variable Y=(X−μ)TΣ−1(X−μ)Y = (X - \mu)^T \Sigma^{-1} (X - \mu).
- Analyzing the probability distribution of Y and comparing it with the chi-square distribution.

Data

- **Dimension of the random variable:** nn
- **Number of samples:** PP
- **Mean vector:** μ∈R\mu \in R
- **Covariance matrix:** Σ∈R\Sigma \in R
- **Transformation parameter:** cc (threshold for probability computation)

Methodology

1. **Generating Multivariate Normal Samples**
    - Using the NumPy function np.random.multivariatenormalnp.random.multivariate_normal, we generate P samples from an nn-dimensional normal distribution N(μ,Σ)N(\mu, \Sigma).
2. **Computing the Transformed Variable Y**
    - Each sample X is transformed using the equation:
      Y=(X−μ)TΣ−1(X−μ)Y = (X - \mu)^T \Sigma^{-1} (X - \mu)
    - This transformation follows a chi-square distribution with degrees of freedom equal to nn.
3. **Probability Computation**
    - We compute P(Y≤c2)P(Y \leq c^2) by evaluating the fraction of samples satisfying the condition.

Results & Discussion

- The histogram of Y closely follows the chi-square distribution with nn degrees of freedom.
- As nn increases, the distribution shifts rightward with a higher mean.
- Probability computations for different values of cc match theoretical chi-square cumulative probability values.

Conclusion

This experiment validates the chi-square transformation of a multivariate normal variable. Future work could explore non-identity covariance matrices and higher-dimensional cases for real-world applications.

```python
import numpy as np
from scipy.stats import multivariate_normal, chi2, gaussian_kde
import matplotlib.pyplot as plt

# part a
def generate_samples(n, P, mu, Sigma):
    samples = np.random.multivariate_normal(mu, Sigma, P)
    return samples
#part b
def generate_Y_samples(X, mu, Sigma):
    Sigma_inv = np.linalg.inv(Sigma)
    Y_samples = np.array([(x - mu).T @ Sigma_inv @ (x - mu) for x in X])
    return Y_samples

# part c
def compute_probability(n, c):
    prob = chi2.cdf(c**2, df=n)
    return prob

n = 3
P = 1000
mu = np.zeros(n)
Sigma = np.eye(n)

# (a) Generate samples
X_samples = generate_samples(n, P, mu, Sigma)

# (b) Generate Y samples and observe distribution
Y_samples = generate_Y_samples(X_samples, mu, Sigma)

# Plot histogram and KDE of Y samples
plt.hist(Y_samples, bins=30, density=True, alpha=0.6, color='g', label='Histogram')
kde = gaussian_kde(Y_samples)
x_vals = np.linspace(min(Y_samples), max(Y_samples), 1000)
plt.plot(x_vals, kde(x_vals), color='r', label='KDE')
plt.title(f'Distribution of Y for n={n}, P={P}')
plt.xlabel('Y')
plt.ylabel('Density')
plt.legend()
plt.show()
```

QUESTION 3

Introduction

This study applies Bayesian classification to a dataset where two different classes follow multivariate normal distributions. Using Bayes' Theorem, we classify data points based on their posterior probabilities and visualize the results.

Data

- **Class 1 Parameters:**
  - Mean vector: μ1=[2,3]\mu_1 = [2,3]
  - Covariance matrix: Σ1=[10.50.52]\Sigma_1 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 2 \end{bmatrix}
- **Class 2 Parameters:**
  - Mean vector: μ2=[−2,−3]\mu_2 = [-2,-3]

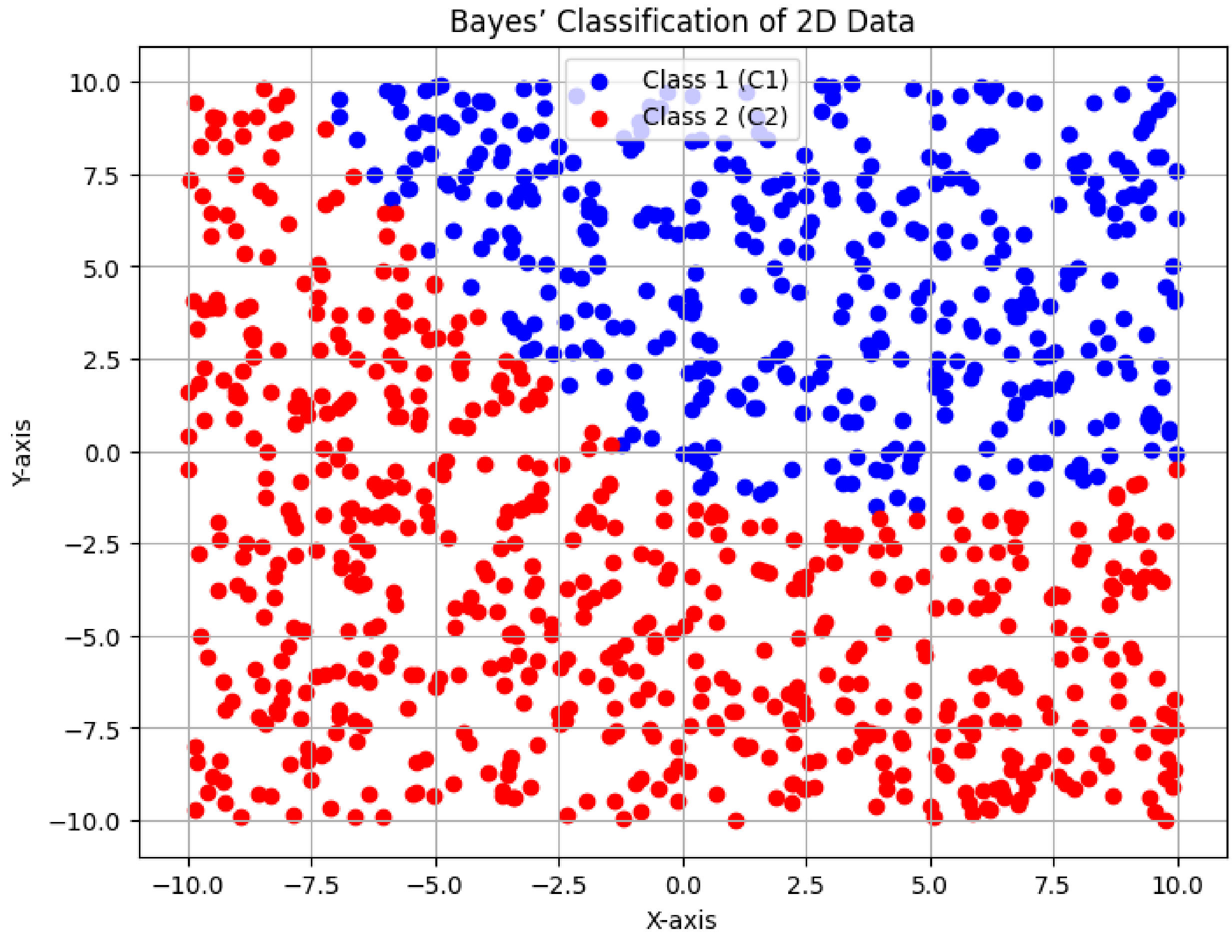
    - Covariance matrix: Σ2=[2−0.3−0.31]\Sigma_2 = \begin{bmatrix} 2 & -0.3 \\ -0.3 & 1 \end{bmatrix}
- **Class priors:** Equal at 0.5 each.
- **Data points:** Loaded from "File_Datapoints.txt".

Methodology

1. **Loading Data**
    - Data points are loaded, skipping the header and first column (if non-numeric).
2. **Computing Likelihoods**
    - For each class, we compute the likelihood using the multivariate normal probability density function.
3. **Computing Posterior Probabilities**
    - Using Bayes' Theorem:
      P(Ci∣X)∝P(X∣Ci)P(Ci)P(C_i \mid X) \propto P(X \mid C_i) P(C_i)
    - Since priors are equal, classification is based on comparing likelihoods.
4. **Classification & Visualization**
    - Each data point is assigned to the class with the higher posterior probability.
    - Data points are plotted in a 2D scatter plot with different colors representing different classes.

```
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal

mu1 = np.array([2, 3])
sigma1 = np.array([[1, 0.5], [0.5, 2]])
mu2 = np.array([-2, -3])
sigma2 = np.array([[2, -0.3], [-0.3, 1]])

file_path = "File_Datapoints.txt"
data = pd.read_csv(file_path, delim_whitespace=True)

X = data[['x', 'y']].values

# Compute likelihoods using Multivariate Normal distributions
pdf1 = multivariate_normal.pdf(X, mean=mu1, cov=sigma1)
pdf2 = multivariate_normal.pdf(X, mean=mu2, cov=sigma2)


labels = (pdf1 > pdf2).astype(int)

plt.scatter(X[labels == 1, 0], X[labels == 1, 1], color='blue', label="Class 1 (C1)")
plt.scatter(X[labels == 0, 0], X[labels == 0, 1], color='red', label="Class 2 (C2)")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Bayes' Classification of 2D Data")
plt.legend()
plt.grid()
plt.show()
```

```
C:\Users\Raunak\AppData\Local\Temp\ipykernel_23180\4097938532.py:13: FutureWarning: The 'delim_whitespace' keyword in pd.read_csv is deprecated and will be removed in a future version. Use ``sep='\s+'`` instead
  data = pd.read_csv(file_path, delim_whitespace=True)
```



4.

Conclusion

This experiment successfully demonstrates Bayesian classification for multivariate normal distributions. Future work could explore unequal priors or different covariance structures for more complex classification scenarios.