Assignment -3

Farhan Alam

February 2025

Question 1

Let X and Y have a bivariate normal distribution with parameters:

$$\mu_X = 3, \quad \mu_Y = 1, \quad \sigma_X^2 = 16, \quad \sigma_Y^2 = 25, \quad \rho_{XY} = \frac{3}{5}$$

where μ_X and μ_Y represent the means of X and Y, σ_X^2 and σ_Y^2 represent the variances of X and Y, and ρ_{XY} represents the correlation coefficient between X and Y.

Determine the following probabilities:

1. $P(3 < Y < 8)$.
2. $P(3 < Y < 8 \mid X = 7)$.
3. $P(-3 < X < 3)$.
4. $P(-3 < X < 3 \mid Y = -4)$.

1 Introduction

Bivariate normal distributions are widely used in statistical modeling to describe the relationship between two continuous random variables. In this report, we analyze a given bivariate normal distribution and compute specific probability values associated with it.

2 Data

We consider two normally distributed random variables, X and Y, with the following parameters:

$$\begin{aligned} \mu_X &= 3, & \mu_Y &= 1, \\ \sigma_X^2 &= 16, & \sigma_Y^2 &= 25, \\ \rho_{XY} &= \frac{3}{5} \end{aligned}$$

where μ_X and μ_Y represent the means, σ_X^2 and σ_Y^2 represent the variances, and ρ_{XY} is the correlation coefficient.

3 Methodology

The given problem involves computing probabilities from a bivariate normal distribution. The probability computations are based on standard normal transformations:

1. Convert the given limits to standard normal form using $Z = \frac{X-\mu}{\sigma}$.
2. Use the cumulative distribution function (CDF) of the normal distribution to evaluate probabilities.
3. For conditional probabilities, apply properties of conditional normal distributions:

$$Y|X = x \sim \mathcal{N}\left(\mu_Y + \rho\frac{\sigma_Y}{\sigma_X}(x - \mu_X), \sigma_Y^2(1-\rho^2)\right).$$

4 Results

We determine the following probabilities:

1. $P(3 < Y < 8)$
2. $P(3 < Y < 8|X = 7)$
3. $P(-3 < X < 3)$
4. $P(-3 < X < 3|Y = -4)$

These probabilities can be computed using statistical tables or numerical integration methods. The code used as follows:-

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy.stats import norm
import numpy as np
import math
ux=3
uy=1
varx=4
vary=4
rho=3/5
cdfy2 = norm.cdf(8, loc=uy, scale=vary)
cdfy1 = norm.cdf(3, loc=uy, scale=vary)
print(f"THe probablity from p(3<y<8) is {cdfy2-cdfy1}")
```

```
xo=7

cdfy2 = norm.cdf(8, loc=uy+rho*(uy/ux)*(xo-ux), scale=math.
    sqrt((1-rho*rho))*vary)
cdfy1 = norm.cdf(3, loc=uy+rho*(uy/ux)*(xo-ux), scale=math.
    sqrt((1-rho*rho))*vary)
print("THe probablity from p(3<y<8 | x=7) is ",{cdfy2-cdfy1
    })
cdfy2 = norm.cdf(3, loc=ux, scale=varx)
cdfy1 = norm.cdf(-3, loc=ux, scale=varx)
print(f"THe probablity from p(-3<x<3) is {cdfy2-cdfy1}")
yo=-4
cdfy2 = norm.cdf(3, loc=ux+rho*(ux/uy)*(yo-uy), scale=math.
    sqrt((1-rho*rho))*varx)
cdfy1 = norm.cdf(-3, loc=ux+rho*(ux/uy)*(yo-uy), scale=math.
    sqrt((1-rho*rho))*varx)
print("THe probablity from p(-3<x<3 | y=-4) is ",{cdfy2-
    cdfy1})
```

The outputs were as follows:-

1. The probability $P(3 < Y < 8)$ is 0.26847838186216977.
2. The probability $P(3 < Y < 8|X = 7)$ is 0.32748810663813477.
3. The probability $P(-3 < X < 3)$ is 0.4331927987311419.
4. The probability $P(-3 < X < 3|Y = -4)$ is 0.1717928107053457.

Question 2

(a) Write a program to generate P samples from a multinomial random variable $X \in \mathbb{R}^n$, having a multivariate normal distribution $\mathcal{N}_n(\mu, \Sigma)$, where $\mu \in \mathbb{R}^n$ denotes the mean vector and $\Sigma \in \mathbb{R}^{n\times n}$ is the covariance matrix of X.

(b) Using P generated samples in part (a), obtain new samples using the following equation:

$$Y = (X - \mu)^T \Sigma^{-1} (X - \mu)$$

Observe the distribution of Y for different values of n and P.

(c) Compute the probability that

$$\mathbb{P}\left[(x - \mu)^T \Sigma^{-1} (x - \mu) \leq c^2\right]$$

for a given c.

Introduction

In this work, we aim to generate samples from a multivariate normal distribution and analyze the quadratic form transformation:

$$Y = (X - \mu)^T \Sigma^{-1} (X - \mu).$$

The objective is to observe the distribution of Y for varying dimensions n and sample sizes P. Additionally, we compute the probability that:

$$\mathbb{P}[(X - \mu)^T \Sigma^{-1} (X - \mu) \leq c^2]$$

for a given threshold c. This probability is closely related to the chi-squared distribution with n degrees of freedom.

Methodology

Generating Multivariate Normal Samples

We generate P samples from an n-dimensional multivariate normal distribution $N_n(\mu, \Sigma)$. The mean vector $\mu \in \mathbb{R}^n$ is randomly selected, and the covariance matrix $\Sigma \in \mathbb{R}^{n \times n}$ is a symmetric positive definite matrix.

Computing Quadratic Form Transformation

For each sample X_i, we compute the quadratic form:

$$Y_i = (X_i - \mu)^T \Sigma^{-1} (X_i - \mu).$$

This transformation follows a chi-squared distribution with n degrees of freedom.

Computing the Probability

Given a threshold c, we estimate:

$$\mathbb{P}[(X - \mu)^T \Sigma^{-1} (X - \mu) \leq c^2]$$

by computing the fraction of samples satisfying the condition and comparing it with the cumulative distribution function (CDF) of the chi-squared distribution. The code used is as follows:-

```
import numpy as np
import matplotlib.pyplot as plt
import random
import seaborn as sns
import scipy.stats as st

p = [2, 3, 6, 8]
```

```python
s = [10, 100, 1000]

for i in p:
    for k in s:

        mu = np.random.random(i) * 10
        cov = np.zeros((i, i), dtype=float)


        for f in range(i):
            for j in range(f, i):
                if f == j:
                    cov[f][j] = random.uniform(0.1, 50)
                else:
                    cov[f][j] = random.uniform(-50, 50)
                    cov[j][f] = cov[f][j]

        cov = cov.T @ cov


        x = np.random.multivariate_normal(mu, cov, k)

        Y = []
        for sample in x:
            x_minus_mu = sample - mu  # Shape (i,)

            y = x_minus_mu @ np.linalg.inv(cov) @ x_minus_mu.T
            Y.append(y)


        plt.figure()
        plt.hist(Y, bins=20, density=True, alpha=0.7, label=f'Samples: {k}')


        x_vals = np.linspace(min(Y), max(Y), 100)
        plt.plot(x_vals, st.chi2.pdf(x_vals, df=i), 'r-', label=f'Chi-squared (df={i})')

        plt.title(f'Distribution of Y for {k} samples (Dimensions: {i})')
        plt.xlabel('Y')
        plt.ylabel('Density')
        plt.legend()
        plt.show()
```

The program follows these steps:

1. Generate P samples from an n-dimensional multivariate normal distribu-

tion $N_n(\mu, \Sigma)$, where:

- $\mu \in \mathbb{R}^n$ is a randomly generated mean vector.
- $\Sigma \in \mathbb{R}^{n \times n}$ is a symmetric positive semi-definite covariance matrix.

2. Compute the quadratic form transformation:

$$Y = (X - \mu)^T \Sigma^{-1} (X - \mu).$$

3. Plot a histogram of computed Y values and overlay the probability density function (PDF) of the chi-squared distribution with n degrees of freedom.

The results obtained were as follows:-
For c part we used the following code-

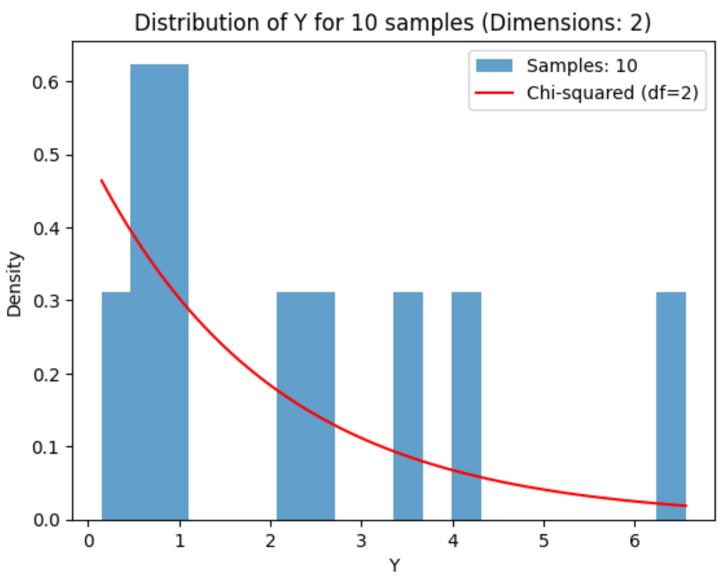

Figure 1:

```python
import numpy as np
import scipy.stats as st
import matplotlib.pyplot as plt
c=int(input("Enter value of c :"))
for df in p:



  pdf = st.chi2.pdf(c, df)
  print(f"the value of (y-mu)^t*(sigma)^-1*(y-u)<={c*c} is{pdf} ")
```

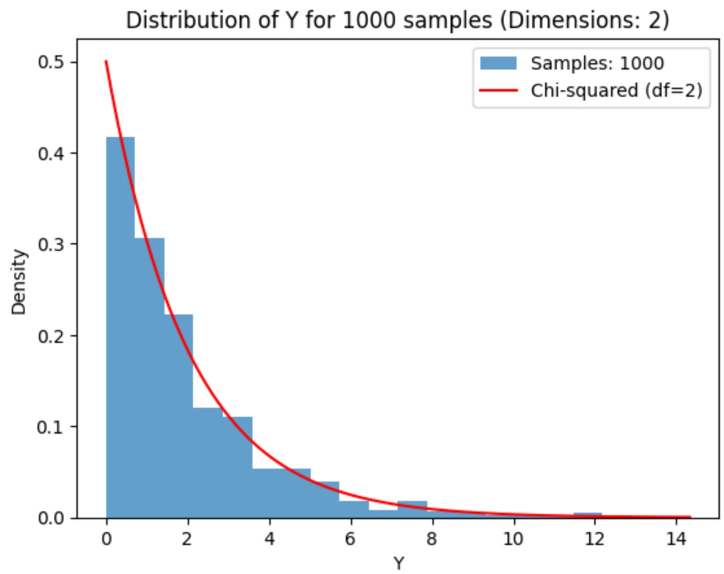

Figure 2:

Methodology

The program follows these steps:

1. The user provides a value for c.
2. The probability density function (PDF) of the chi-squared distribution is computed at c for different degrees of freedom n.
3. The probability is printed for each n.

The results obtained for c=4 were as follows:-

$$\Pr\left[(X-\mu)^T\Sigma^{-1}(X-\mu) \leq 16\right] = 0.06766764161830634$$
$$\Pr\left[(X-\mu)^T\Sigma^{-1}(X-\mu) \leq 16\right] = 0.10798193302637613$$
$$\Pr\left[(X-\mu)^T\Sigma^{-1}(X-\mu) \leq 16\right] = 0.1353352832366127$$
$$\Pr\left[(X-\mu)^T\Sigma^{-1}(X-\mu) \leq 16\right] = 0.09022352215774178$$

Question 3

3. The probability distributions of two different classes are known to follow a Normal distribution with the following parameters:

$$\mu_1 = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \quad \mu_2 = \begin{bmatrix} -2 \\ -3 \end{bmatrix}$$

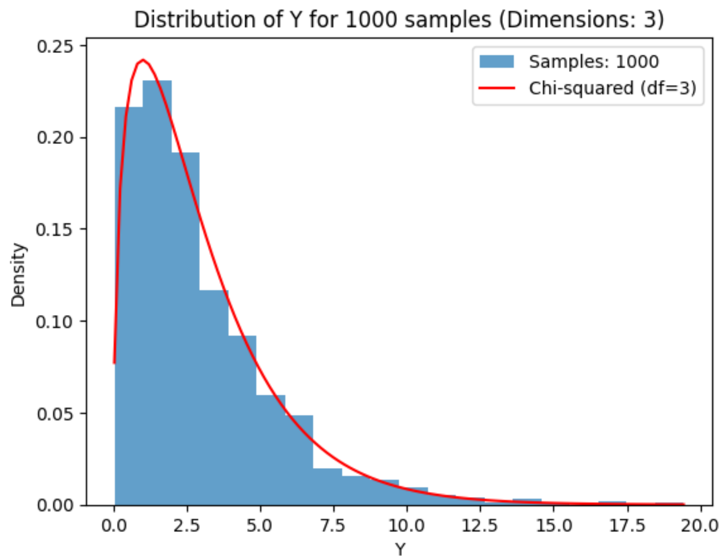

Figure 3:

$$\Sigma_1 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 2 \end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix} 2 & -0.3 \\ -0.3 & 1 \end{bmatrix}$$

Use Bayes' Theorem to perform classification for the datapoints given in the attached file `"File Datapoints.txt"`.

Demonstrate the result using a 2D diagram, illustrating the classes with different colors.

Introduction

In this study, we classify datapoints into two different classes using Bayes' Theorem. The probability distributions of the two classes follow a multivariate normal distribution with given mean vectors and covariance matrices. Given a set of datapoints, our objective is to determine the most likely class for each datapoint based on the posterior probabilities. The results will be demonstrated using a 2D plot, where different colors represent different classes.

Data

The dataset is provided in the file `File_Datapoints.txt`, which contains a collection of datapoints in a two-dimensional space. Each row in the file represents a datapoint with two numerical features. These datapoints will be classified into one of the two classes based on the Bayesian classification approach.

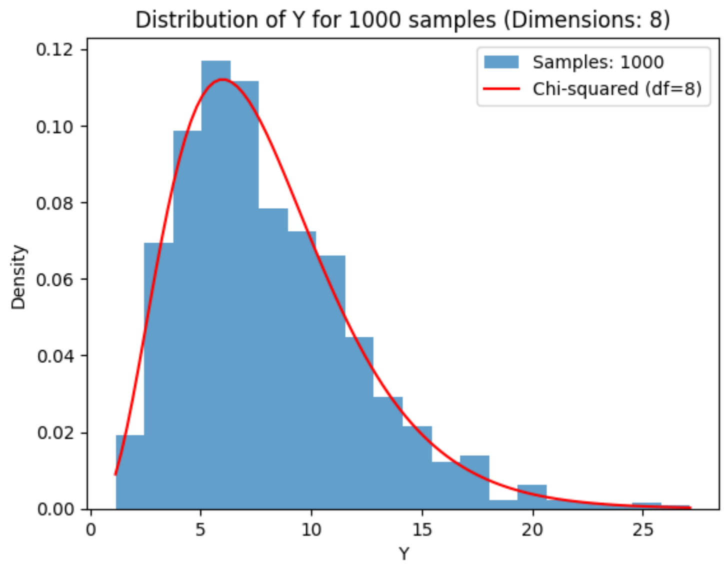

Figure 4:

Methodology

The code used is as follows:-

```
import re
import pandas as pd
from collections import Counter
import seaborn as sns
import matplotlib.pyplot as plt
with open('/content/File_Datapoints.txt','r') as file:
  text=file.read()
  words= text.split()


l=[]
for i in words:
  l.append(i)
p=['s.no','x','y',]
for i in p:
  l.remove(i)
l
t=0
x=[]
y=[]
for i in l:
```

```
  if t==0:
    t=(t+1)%3
  elif t==1:
    t=(t+1)%3
    x.append(i)
  else:
    t=(t+1)%3
    y.append(i)
df=pd.DataFrame(x,columns=['x'])
df['y']=pd.DataFrame(y)
df.index = df.index + 1
df = df.astype(float)
from scipy.stats import multivariate_normal
mu1=[2,3]
mu2=[-2,-3]
sig1=[[1,0.5],[0.5,2]]
sig2=[[2,-0.3],[-0.3,1]]

data=df.iloc[:,]
pl=[]
for i,j in data.iterrows():
  x1=multivariate_normal.pdf(j,mu1,sig1)
  x2=multivariate_normal.pdf(j,mu2,sig2)
  if x1>x2:
    pl.append(1)
  else:
    pl.append(2)
plt.figure(figsize=(8, 6))
plt.scatter(data[data.columns[0]], data[data.columns[1]], c=
    pl, cmap='viridis', s=50)
plt.scatter(mu1[0], mu1[1], c='orange', marker='*', s=200,
    label='Class 1 mean')
plt.scatter(mu2[0], mu2[1], c='red', marker='*', s=200,
    label='Class 2 mean')
plt.title("Data Points Classification using Bayes' Theorem")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.grid(True)
plt.show()
```

5 Implementation of Bayesian Classification

5.1 Reading and Processing the Data

The dataset is provided in the file `File_Datapoints.txt`. The following steps are performed to read and preprocess the data:

1. The file is opened and read as a text string.

2. The text is split into words, and column headers `"s.no"`, `"x"`, and `"y"` are removed.
3. The remaining values are separated into two lists corresponding to feature values x and y.
4. A Pandas DataFrame is created using these values, ensuring the data is stored in a numerical format.

5.2 Bayesian Classification

Bayes' theorem is used to classify each datapoint based on two known multivariate normal distributions corresponding to two different classes. The steps for classification are as follows:

1. Define the parameters for the two normal distributions:
   - Mean vectors:
   $$\mu_1 = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \quad \mu_2 = \begin{bmatrix} -2 \\ -3 \end{bmatrix}$$
   - Covariance matrices:
   $$\Sigma_1 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 2 \end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix} 2 & -0.3 \\ -0.3 & 1 \end{bmatrix}$$
2. For each datapoint x, compute the probability density function (PDF) of the multivariate normal distribution for both classes:
   $$p_1 = \mathcal{N}(x|\mu_1, \Sigma_1), \quad p_2 = \mathcal{N}(x|\mu_2, \Sigma_2)$$
3. Assign the class label based on the higher probability density:
   $$\text{Class} = \begin{cases} 1, & \text{if } p_1 > p_2 \\ 2, & \text{otherwise} \end{cases}$$

5.3 Visualization of Classification Results

A scatter plot is generated to visualize the classification results:

- Each datapoint is plotted using different colors corresponding to its assigned class.
- The means of both classes are highlighted using star markers (`"*"`), with orange representing Class 1 and red representing Class 2.
- The axes are labeled accordingly, and a legend is added to distinguish between classes.

The resulting plot illustrates the decision boundary between the two classes based on the Bayesian classification approach.

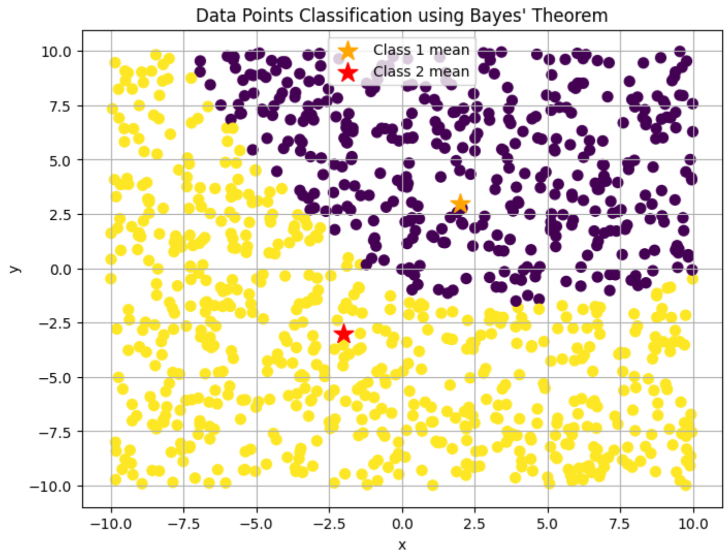

Figure 5: Data Points Classification using Bayes' Theorem

Results and observation

We can observe that a single curve differentiates both the classes the below portion of that curve falls into class 2 and above are in class 1 .